On the Optimal Encoding Ladder of Tiled 360° Videos for Head-Mounted Virtual Reality

Ching-Ling Fan, Shou-Cheng Yen, Chun-Ying Huang, *Member, IEEE*, and Cheng-Hsin Hsu, *Senior Member, IEEE*

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) has been widely used by several popular streaming services, such as YouTube, Netflix, and Facebook. Adopting DASH requires to pre-determine a set of encoding configurations, called encoding ladder, to generate a set of representations stored on the streaming server. These representations are adaptively requested by clients according to their network conditions during streaming sessions. In this article, we aim to solve the optimal laddering problem that determines the optimal encoding ladder to maximize the client viewing quality. In particular, we consider video models, viewing probability, and client distribution to formulate the mathematical problem. We use divide-and-conquer approach to decompose the problem into two subproblems: (i) per-class optimization for clients with different bandwidths and (ii) global optimization to maximize the overall viewing quality under the storage limit of the streaming server. We propose two algorithms for each of the per-class optimization and global optimization problems. Analytical analysis and real experiments are conducted to evaluate the performance of our proposed algorithms, compared to other state-of-the-art algorithms. Based on the results, we recommend a combination of the proposed algorithms to solve the optimal laddering problem. The evaluation results show the merits of our recommended algorithms, which: (i) outperform the state-of-the-art algorithms by up to 52.17 and 26.35 in Viewport Video Multi-Method Assessment Fusion (V-VMAF) in per-class optimization, (ii) outperform the state-of-the-art algorithms by up to 43.14 in V-VMAF for optimal laddering in global optimization, (iii) achieve good scalability under different storage limits and number of bandwidth classes, and (iv) run faster than the state-of-the-art algorithms.**

***Keywords*-360° videos, encoding ladders, encoder configurations, video streaming, adaptive streaming, virtual reality, augmented reality, mixed reality, extended reality, optimization**

I. Introduction

Global IP video traffic has been forecast to account for 82% of all business Internet traffic by 2022 [1], which can be attributed to the fast development of too many multimedia network services. In particular, Dynamic Adaptive Streaming over HTTP (DASH) [2], [3] has been widely used by several popular multimedia streaming services, such as YouTube, Facebook, and Netflix. Videos on DASH servers are encoded with different encoding *configurations*, such as resolutions and Quantization Parameters (QPs), where the resulting encoded videos are referred to as *representations*. To accommodate network dynamics and client heterogeneity, the encoded videos are split into a series of short segments. Streaming different representations of each segment provides different quality levels while consuming different amounts of network bandwidth.

C. Fan, S. Yen, and C. Hsu are with the Department of Computer Science, National Tsing Hua University. C. Huang is with the Department of Computer Science, National Chiao Tung University.

Fig. 1 illustrates a sample DASH streaming service *offered by a content distributor, such as YouTube and Netflix.* Content distributors receive production-quality videos from *content providers*, such as Walt Disney and Warner Bros., prepare multiple representations of segments, and distribute the videos over the Internet to clients. The DASH streaming service contains three entities: (i) *production server*, which produces the encoded video segments, (ii) *streaming server*, which stores the encoded video segments and sends the video streams, and (iii) *clients*, which request, decode, and render the video segments to viewers. The operations among these three entities can be divided into two phases: (i) preparation phase, which is triggered only when new videos are added to the streaming servers and (ii) streaming phase, which is triggered at individual streaming sessions when the clients request video segments from the streaming servers. The Adaptive BitRate (ABR) algorithm [4], [5] is a key client-side component in the streaming phase. The goal of the ABR algorithm is to adaptively select the representations that minimize the video distortion without congesting the networks. However, the ABR algorithms can *only* request for the representations that are generated in the preparation phase. Hence, content distributors must carefully choose a set of representations to cover a broad range of network bandwidth. The set of configurations used for generating the above mentioned set of representations is referred to as the *encoding ladder*[1].

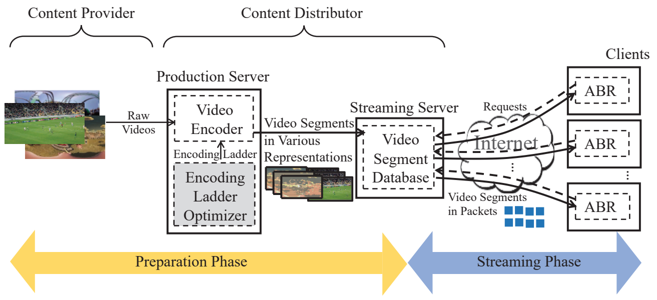


Fig. 1: Two phases of DASH services: preparation and streaming.

At first glance, to support clients with diverse network bandwidth and computational power, the encoding ladder shall

[1]By varying different encoding parameters of the configurations, multiple Rate-Distortion (R-D) curves can be generated. However, at any given bitrate, a configuration is selected from one of the R-D curves following a selection criterion. Combining all the selected encoding configurations at different bitrates results in the encoding ladder. More details on the encoding ladders can be found in the literature [6], [7].

contain *one* representation for *every single* client, so as to capitalize all the available bandwidth. Doing so, however, requires long encoding time and huge storage usage, and thus is not scalable as the server resources are bounded. Hence, the production server should optimally determine the encoding ladder to maximize the overall viewing quality of clients without exceeding the storage limit on the server. We call this problem as the *optimal laddering* problem. Through solving the optimal laddering problem, content distributors store the segment representations that better fit the client bandwidth distribution on the streaming server. Therefore, the ABR algorithms on the clients will have better chance to switch to the representations with smaller video distortion. In this article, we focus on on-demand videos, which can be carefully analyzed offline for the optimal encoding ladders.

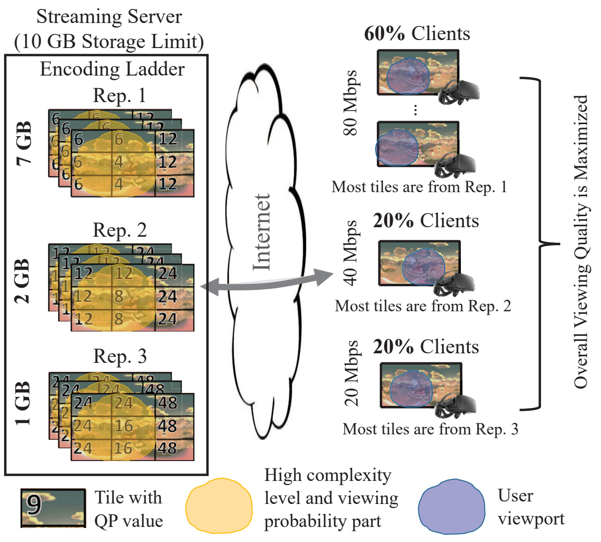


Fig. 2: An illustrative example of the optimal laddering problem.

Although the optimal laddering problem has been studied for conventional videos [6], [7], solving the same problem for tiled[2] 360° videos [9] to Head-Mounted Displays (HMDs) is much more challenging for a couple of reasons. First, the extremely high resolution of 360° videos consumes much more storage space than conventional videos. Second, the optimal ladders may be different among tiles of the same video. This is because each tile may have different complexity levels and viewing probabilities. For example, the tiles that are frequently viewed (e.g., the main foreground objects) may need to have higher quality levels than those tiles that are rarely viewed (e.g., straight up/down).

In this paper, we study the optimal laddering problem for tiled 360° videos to HMDs in the preparation phase. This is different from the majority of prior work on designing ABR algorithms for tiled 360° videos [10], [11], [12], [13], [14], [15], [16], where the viewing quality of each client in the streaming phase is maximized under a *given* encoding ladder. To the best of our knowledge, Ozcinar et al. [17] is the only work that also computes the encoding ladder for 360° videos. However, their solution allocates even bitrate to all tiles and assumes all tiles have the same complexity level and viewing probability. In contrast, we consider a more general setup, which consists of diverse:

- *Video model*, which maps the encoding configurations (i.e., QP in this article) to expected video distortion levels and bitrates. The video model captures the complexity levels of individual tiles of each video.
- *Viewing probability*, which quantifies the chance of each tile being viewed by HMD viewers. The viewing probability can be estimated from historical data of other HMD viewers or computed using fixation prediction algorithms [18], [19]. We take the former approach in this article if not otherwise specified.
- *Client distribution*, which represents the fraction of clients with different amounts of available bandwidth. The clients with the same available bandwidth is considered in the same bandwidth *class* in our problem.

Fig. 2 shows an illustrative example of our optimal laddering problem, in which we allocate more resources to the tiles with higher complexity levels and viewing probabilities (e.g., with smaller QP values) and to the representations being requested by more clients (e.g., with larger storage space). The eventual goal is to maximize the overall viewing quality in the streaming phase.

To solve the optimal laddering problem, we leverage the divide-and-conquer approach to decompose the problem into two subproblems: (i) per-class optimization and (ii) global optimization. The per-class optimization problem focuses on the optimization for each class with a given available bandwidth. We formulate this problem into a convex optimization problem [20] considering video models and viewing probability. We solve it using a suite of mathematics tools, such as Lagrangian multiplier and rounding, in the Rate-Distortion Optimization (RDO) fashion [21]. However, the complex video models result in non-negligible computation overhead. Thus, we also propose a more efficient greedy algorithm that iteratively sets the encoding configurations of individual tiles. For the global optimization, the goal is to adjust the solution from the per-class optimization to meet the overall storage limit. In this problem, the client bandwidth distribution is considered when optimizing the overall viewing quality across all clients at the given storage limit. We have also proposed two algorithms for global optimization. One of them runs more efficiently and the other one offers better video quality. We have conducted experiments on a real testbed to evaluate our proposed algorithms, compared to state-of-the-art algorithms. We then recommend a combination of our proposed algorithms to efficiently solve the optimal laddering problem while achieving high viewing quality.

We make the following contributions in this article.

- We formulate the optimal laddering problem for 360° videos into a mathematical optimization problem con-

[2]The modern HEVC codecs [8] spatially divide a video into smaller rectangular subvideos, referred to as tiles, which can be independently streamed and decoded. Tiles are essential for 360° video streaming, in which HMD viewers only watch small portions of the whole 360° videos at any moment.

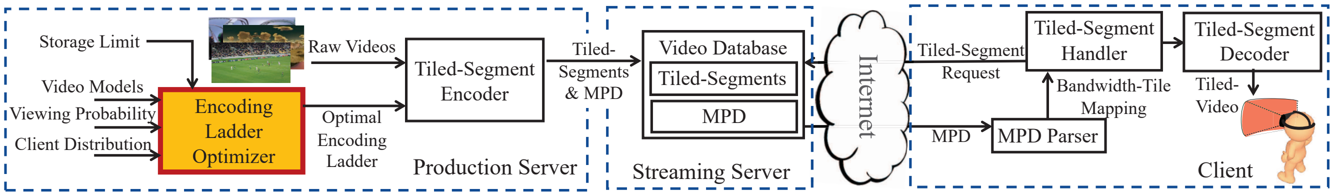


Fig. 3: System overview of our considered 360° video streaming system.

sidering video models, viewing probability, and client distribution.

- We solve the problem using the divide-and-conquer approach with a diverse suite of mathematical tools. We also analytically analyze the performance of our proposed algorithms.
- We conduct extensive experiments to show the performance and practicality of our proposed algorithms. Our evaluation results reveal that: (i) our algorithms outperform state-of-the-art algorithms by up to 52.17 and 26.35 in Viewport Video Multi-method Assessment Fusion (V-VMAF) [22] in per-class optimization, (ii) our algorithms outperform the state-of-the-art algorithms by up to 43.14 in V-VMAF in global optimization problem for the optimal ladders, (iii) our algorithms scale well under different storage limits and different number of bandwidth classes, and (iv) our algorithms run faster than the state-of-the-art algorithms.

II. System Overview

Fig. 3 details our considered streaming system for tiled 360° videos. The components are introduced below.

- **Encoding ladder optimizer** determines the optimal encoding ladder under the storage limit. The resulting ladder is then used for encoding the tiled-segments.
- **Tiled-segment encoder** on the production server compresses and splits the videos into tiled-segments. Each tiled-segment is a tile across multiple consecutive video frames. Tiled-segments are the basic streaming units that can be independently encoded and decoded.
- **Video database** on the streaming server stores the encoded tiled-segments following the encoding ladder. The video database also stores the MPD (Media Presentation Description) files, which provide the meta-data of the representations to the clients.
- **MPD parser** on the client parses the MPD files from the server to get the mapping between representations and URLs.
- **Tiled-segment handler** on the client sends the requests and receives the incoming tiled-segments. It also implements ABR algorithms for selecting the tiled-segments.
- **Tiled-segment decoder** on the client decodes the received tiled-segments for the HMD viewers.

The interactions among these components are as follows. At the production server, raw 360° videos are encoded and segmented into tiled-segments according to the encoding ladder computed by the encoding ladder optimizer. These tiled-segments and the MPD files are stored in the video database on the streaming server. At each streaming session, the MPD parser parses the MPD file for the meta-data of the tiled-segments. The tiled-segment handler then adaptively requests videos from the streaming server. The streamed tiled-segments are decoded by the tiled-segment decoder for the HMD viewers. *Among the above components, the encoding ladder optimizer (shaded block in Fig. 3) is the core component studied in this article, which will be detailed in the remaining sections.*

III. Optimal Laddering Problem

In this section, we first describe our research problem, which is followed by the system models and problem formulation. Table I summarizes the symbols used in this article.

TABLE I: Symbol Table

Symbol	Description
V	Number of videos
C	Number of bandwidth classes
T	Number of segments
N	Number of tiles
Q	Maximum of available QP values
S	Storage limit on server
$f_{v,c}$	Probability of client in bandwidth class c watching video v (client distribution)
$p_{v,t,n}$ $= p_\phi$	Viewing probability of tile n at segment t for video v
a_n	Area scaling factor of tile n
$x_{v,t,n,c,q}$ $= x_{\phi,c,q}$	Whether tile n with QP q at segment t is selected to be transmitted in bandwidth class c watching video v
$y_{v,t,n,q}$ $= y_{\phi,q}$	Whether tile n with QP q at segment t watching video v is selected to be stored on the server
$D_{v,t,n}$ $= D_\phi$	Maximum distortion of tile n at segment t of video v
$d_{v,t,n}(q)$ $= d_\phi(q)$	Distortion of tile n with QP q at segment t of video v (distortion model)
$r_{v,t,n}(q)$ $= r_\phi(q)$	Bitrate of tile n with QP q at segment t of video v (bitrate model)
b_c	Available bandwidth of bandwidth class c

A. Problem Statement

Our research problem can be described as follows. Given a 360° video server with a storage limit of S, each 360° video is divided into T segments, where each segment is further divided into N tiles. We classify the clients into C classes based on their available bandwidth b_c. A class c client has a probability of $f_{v,c}$ to watch video v. In particular, $f_{v,c} = w_v^v \times w_c^b$. w_v^v denotes the video popularity and w_c^b denotes the fraction of clients at bandwidth class c, where $\sum_{v=1}^{V} w_v^v = 1$ and $\sum_{c=1}^{C} w_c^b = 1$. Let n denote the tile number and q denote the encoding QP[3]. The goal of the

[3]We focus on controlling QP for rate control, while other parameters may be used as well. For example, studies in the literature propose to employ Lagrangian multiplier λ [23], [24], [25] instead of QP to control the video quality for higher rate control accuracy.

encoding ladder optimizer is to make two sets of decisions to minimize the overall viewing distortion. First, the tiles and their representations stored on the streaming server need to be determined. These are captured by the boolean variables $y_{v,t,n,q} \in \{0,1\}$, where v denotes the video, t denotes the segment number, n denotes the tile number, and q denotes the encoding QP. $y_{v,t,n,q} = 1$ if and only if video v's tiled-segment (t,n) has a representation with QP q stored on the streaming server. Second, the tiles and their representations that are planned to be streamed to the clients need to be determined. These are captured by the boolean variables $x_{v,t,n,c,q} \in \{0,1\}$, where $x_{v,t,n,c,q} = 1$ if and only if the representation with QP q of tiled-segment (t,n) is streamed to class c clients who watch video v. We use ϕ to represent (v,t,n) and $\Phi = \{(v,t,n)|v \in [1,V], t \in [1,T], n \in [1,N]\}$ to denote all possible 3-tuples in the remaining article for brevity. For example, we interchangeably write $y_{v,t,n,q}$ and $y_{\phi,q}$ as well as $x_{v,t,n,c,q}$ and $x_{\phi,c,q}$ if they do not cause any ambiguity.

TABLE II: The Adj. R^2 of the Video Models for the Considered Videos

Video	**Mega Coaster**	**Roller Coaster**	**Shark Shipwreck**	**Hog Rider**	**Chariot Race**	**SFR Sport**
MSE	0.99	0.99	0.99	0.99	0.99	0.93
Bitrate	0.99	0.99	0.99	0.99	0.99	0.99

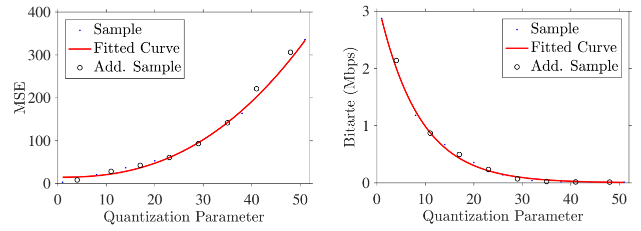
B. Video Models

We let $d_\phi(q)$ and $r_\phi(q)$ be the distortion and bitrate models, which are functions of QPs. The video models allow us to estimate the viewing quality and consumed bandwidth when solving the optimal laddering problem. To understand the properties of the models, we divide videos from a public dataset [26] into 6×4 tiles and encode them multiple times with different QPs in $\{1,8,14,20,26,32,38,44,51\}$ using Kvazaar [27]. The Mean Square Error (MSE)[4] and bitrate of each tile under these QPs are measured. We then use the measured results to estimate the model parameters. Several possible functions can be adopted for the models, such as linear, power, and exponential functions. Our pilot tests indicate that the linear function has the worst modeling performance. In contrast, the distortion and bitrate functions can be well modeled by the power and exponential functions, respectively. Our findings are inline with other empirical models [28] with only minor differences. Specifically, we write these two models as:

$$d_\phi(q) = \alpha^d_\phi q^{\beta^d_\phi} + \gamma^d_\phi; \tag{1}$$

$$r_\phi(q) = \alpha^r_\phi e^{\beta^r_\phi q}. \tag{2}$$

In the models, α^d_ϕ, β^d_ϕ, γ^d_ϕ, α^r_ϕ, and β^r_ϕ are model parameters. We fit the distortion and bitrate models for individual tiled-segments. We plot the MSE and bitrate of a sample tiled-segment from *Mega Coaster* in Figs. 4(a) and 4(b) under different QPs. These figures reveal that Eqs. (1) and (2) are reasonably accurate, as the curves closely follow the samples. We also encode the tiled-segment with several additional QPs to evaluate the accuracy of the resulting models. We mark these *additional samples* in the figures with circles, which are also close to the model curves. We notice that figures from video models of other tiled-segments are similar, and are left out due to the limited space. The average adjusted R^2 of the distortion and the bitrate models are reported in Table II. This table confirms the accuracy of our video models. The power and exponential models are both convex functions. This property is utilized by our solution proposed later.



(a) (b)

Fig. 4: Sample video models: (a) MSE over QP ($d_\phi(q)$) and (b) bitrate over QP ($r_\phi(q)$).

C. Problem Formulation

The optimal laddering problem is quite hard to solve. We give the proof of the following lemma in Appendix B.

Lemma 1: The optimal laddering problem is NP-hard.

We formulate the optimal laddering problem into an Integer Linear Programming (ILP) problem. The decision variables of our formulation are $x_{\phi,c,q}$ and $y_{\phi,q}$. When projecting tiles of the 2D reconstructed videos to the 3D sphere for generating the viewports, different sphere areas are covered by each tile (mainly due to different latitudes). To account for this, we let a_n be the area scaling factor of tile n, which is defined as the ratio of the area of tile n to that of the whole 3D sphere. Concretely, we let A_n be the area of tile n on the sphere, where $1 \leq n \leq N$. We then write scaling factor a_n as $\frac{A_n}{\sum_{i=1}^{N} A_i}$. A larger a_n value indicates that tile n affects the resulting viewports more. Note that without incurring ambiguity, we interchangeably write a_ϕ and a_n; we define $a_\phi = a_n$, where $\phi = (v,t,n)$ as a is different only when n changes. Similarly, we let $f_{\phi,c} = f_{v,c}$, where $\phi = (v,t,n)$. With the notations defined so far, we write our problem as:

$$\min \sum_{c=1}^{C} \sum_{\phi \in \Phi} f_{\phi,c} p_\phi a_\phi \sum_{q=1}^{Q} d_\phi(q) x_{\phi,c,q} \tag{3a}$$

$$st: \sum_{n=1}^{N} \sum_{q=1}^{Q} r_{v,t,n}(q) x_{v,t,n,c,q} \leq b_c \quad c \in [1,C], v \in [1,V], t \in [1,T]; \tag{3b}$$

$$\sum_{\phi \in \Phi} \sum_{q=1}^{Q} r_\phi(q) y_{\phi,q} \leq S; \tag{3c}$$

$$x_{\phi,c,q} \leq y_{\phi,q} \quad c \in [1,C], q \in [1,Q], \phi \in \mathbf{\Phi}; \tag{3d}$$

$$\sum_{q=1}^{Q} x_{\phi,c,q} = 1 \quad c \in [1,C], \phi \in \mathbf{\Phi}; \tag{3e}$$

$$x_{\phi,c,q} \in \{0,1\} \quad c \in [1,C], q \in [1,Q], \phi \in \mathbf{\Phi}; \tag{3f}$$

$$y_{\phi,q} \in \{0,1\} \quad q \in [1,Q], \phi \in \mathbf{\Phi}. \tag{3g}$$

The objective function in Eq. (3a) minimizes the expected

[4]Instead of 360° video quality metrics, we use MSE of individual tiles to quantify the distortion of the video. This is because our problem is to determine the QP values of individual tiles, i.e., at the tile level, while 360° video quality metrics are at the *video level*. Nonetheless, we employ 360° specific metrics, e.g., V-VMAF, to quantify the overall video quality in our evaluations. We note that, in addition to MSE, other quality metrics at the tile-level may be adopted, such as Peak Signal-to-Noise Ratio (PSNR) or Quality of Experience (QoE) models [16].

overall distortion in weighted-MSE[5] [29], where the tile weights depend on: (i) viewing probability (p_ϕ) and area scaling factor (a_ϕ) across all tiles and (ii) the probability of clients in different classes watching different videos ($f_{\phi,c}$). Eq. (3b) makes sure that the total streamed bitrate to each client class c does not exceed the available bandwidth b_c. Eq. (3c) constrains the consumed storage space within the storage limit S on the server. Eq. (3d) indicates that clients only select the tiles offered by the server. Besides, each client only selects one representation for each tile as enforced by Eq. (3e).

IV. Problem Decomposition

The optimal laddering problem in Eqs. (3a)–(3g) is fairly complicated because of the complex interplay between the bandwidth and storage constraints. More specifically, the best solution that satisfies all the bandwidth constraints may exceed the storage limit, while restricting the storage space for each class causes a huge number of permutations on storage space assignments across videos. Hence, we opt for the *divide-and-conquer* approach, as illustrated in Fig. 5. In particular, we decompose the optimal laddering problem into the following two subproblems.

- *Per-class optimization* problem optimizes the per-class solution under the bandwidth constraint of each class. It takes the video models and viewing probability of video v and bandwidth constraint of class c as the inputs. It then determines the best QP values for tiled-segment (t, n). The output for class c of video v contains the boolean values $\{x^*_{v,t,n,c,q}(= x^*_{\phi,c,q})|\forall t, n, q\}$, which are collectively denoted as $\mathbf{X}^*_{\mathbf{v},\mathbf{c}}$.
- *Global optimization* problem combines and adjusts all per-class solutions into a global solution under the storage limit. It takes the video models, viewing probability, client distribution, and the storage limit of *all* videos as the inputs. It then adjusts $\mathbf{X}^*_{\mathbf{v},\mathbf{c}}$ of each class c to fit all tiled-segments into the storage limit. When the storage limit is loose, all $\mathbf{X}^*_{v,c}$ solutions from the per-class optimization problems may be directly accepted. The output of the global optimization problem contains the revised $\mathbf{X}^*_{\mathbf{v},\mathbf{c}}$ for all v and c, which is collectively written as $\mathbf{X}^*$. Moreover, the output also specifies the optimal QP values for stored tiled-segment (t, n) of video v as $\{y^*_{v,t,n,q}(= y^*_{\phi,q})|\forall v, t, n, q\}$, which is collectively written as $\mathbf{Y}^*$. $\mathbf{Y}^*$ is essentially the *optimal encoding ladder*. It is not hard to see that $\mathbf{Y}^*$ is a function of $\mathbf{X}^*$; that is, $y^*_{v,t,n,q} = 1$ if and only if $\sum_{c=1}^{C} x^*_{v,t,n,c,q} \geq 1, \forall v, t, n, q$.

We solve the per-class optimization problem for each class with a bandwidth constraint in Sec. V. We solve the global optimization problem with the storage limit in Sec. VI.

V. Per-Class Optimization

We first give the formulation, which is followed by two algorithms.

[5]The overall distortion in our formulation is a video-level quality metric, which is essentially a weighted sum of all the tile quality.

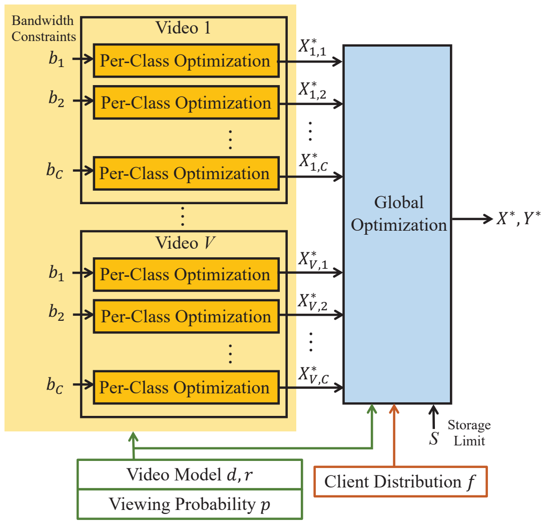


Fig. 5: The overview of our divide-and-conquer approach.

A. Per-Class Formulation

Let $P(v, c)$ be the subproblem of bandwidth class c watching video v, where the storage limit is ignored. That is, this subproblem only considers the constraints in Eqs. (3a)–(3g) that are related to $x_{v,t,n,c,q}$. We formally formulate the problem as:

$$P(v,c): \min \sum_{t=1}^{T}\sum_{n=1}^{N} p_{v,t,n}a_n \sum_{q=1}^{Q} d_{v,t,n}(q)x_{v,t,n,c,q} \tag{4a}$$

$$st: \sum_{n=1}^{N}\sum_{q=1}^{Q} r_{v,t,n}(q)x_{v,t,n,c,q} \leq b_c \qquad t \in [1,T]; \tag{4b}$$

$$\sum_{q=1}^{Q} x_{v,t,n,c,q} = 1 \quad t \in [1,T], n \in [1,N]; \tag{4c}$$

$$x_{v,t,n,c,q} = \{0,1\} \quad t \in [1,T], n \in [1,N], q \in [1,Q]. \tag{4d}$$

Eq. (4a) minimizes the expected distortion for the clients in bandwidth class c who watch video v. Eq. (4b) constrains the consumed bitrate within the available bandwidth b_c. Eq. (4c) ensures that only one representation is selected.

We next propose two algorithms to solve the formulation in Eqs. (4a)–(4d): (i) Per-Class Lagrangian-Based Algorithm (PC-LBA), which leverages the convexity of video models to get the solution, and (ii) Per-Class Greedy-Based Algorithm (PC-GBA), which runs more efficiently. Their performance will be compared in Sec. VII.

B. Lagrangian-Based Algorithm: PC-LBA

PC-LBA consists of two steps: (i) a QP optimizer, which employs Lagrangian multiplier [30] to find the optimal QPs as real numbers, and (ii) optimal rounding algorithm, which solves an ILP formulation to optimally round the QPs to integers supported by the encoder.

QP optimizer. To adopt the Lagrangian approach, we *transform* the discrete decision variables $x_{v,t,n,c,q}$ into continuous decision variables $\kappa_{v,t,n,c}$. We let $\kappa_{v,t,n,c}$ represent the QP value of tiled-segment (t, n) of video v streamed to class c clients. $\kappa_{v,t,n,c}$ is a real number in the range of $[\kappa_{min}, \kappa_{max}]$, where κ_{min} and κ_{max} are the QP bounds from the video encoder. With $\kappa_{v,t,n,c}$, the transformed formulation has fewer decision variables and gets rid of Eq (4c). Moreover, we observe that the decisions on different segments are independent. Hence, we write each $P(v,c)$ into a series of transformed $P'(v,t,c)$ for all $t \in [1,T]$ as:

$$P'(v,t,c) = \min \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n \tag{5a}$$

$$st: \sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) \leq b_c; \tag{5b}$$

$$\kappa_{v,t,n,c} \in [\kappa_{min}, \kappa_{max}]. \tag{5c}$$

In this formulation, Eqs. (5a) and (5b) account for the expected distortion and consumed bitrate for clients in class c watching segment t of video v. The following two lemmas show that the transformed formulation in Eqs. (5a)–(5c) can be solved efficiently. The proofs are given in Appendix B to maintain the flow of the article.

Lemma 2: When the power function in Eq. (1) is adopted as the distortion model, the objective function in Eq. (5a) is convex.

Lemma 3: When the exponential function in Eq. (2) is adopted as the bitrate model, the constraint in Eq. (5b) is convex.

We write $\{\kappa_{v,t,1,c}, \kappa_{v,t,2,c}, \cdots, \kappa_{v,t,N,c}\}$ as $\mathbf{K_{v,t,c}}$ in the following for the sake of presentation. Combining Lemmas 2 and 3, we know that our optimization problem is a convex programming problem, which can be solved using Lagrangian multiplier as follows. We first introduce a Lagrangian multiplier $\mu \in \mathbb{R}^+$, and rewrite our (constrained) convex programming problem into an unconstrained optimization problem:

$$\begin{aligned} \min\ L(\mathbf{K_{v,t,c}}, \mu) = &\sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n \\ &+ \mu\Big(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) - b_c\Big) \end{aligned} \tag{6}$$

Consider Eq. (6) as the primal Lagrangian problem, the Lagrangian dual function g minimizes the Lagrangian value over $\mathbf{K_{v,t,c}}$:

$$\begin{aligned} g(\mu) = \inf_{\mathbf{K_{v,t,c}}} (\mathbf{K_{v,t,c}}, \mu) = &\inf_{\mathbf{K_{v,t,c}}} \Big(\sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,t,n,c}) p_{v,t,n} a_n \\ &+ \mu\Big(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c}) - b_c\Big)\Big). \end{aligned} \tag{7}$$

Lemma 4: The Lagrangian dual function (Eq. (7)) constitutes a lower bound for the objective value of any feasible solution to the Lagrangian primal problem (Eq. (6)). In fact, because the strong duality holds here, the optimal solution of the Lagrangian dual problem is also the optimal solution of the original (primal) problem.

To solve the Lagrangian dual problem, we first calculate the partial derivative w.r.t. each $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$:

$$\begin{aligned} \frac{\partial L}{\partial \kappa_{v,t,n,c}} = &(\alpha^d_{v,t,n} \beta^d_{v,t,n} \kappa_{v,t,n,c}^{\beta^d_{v,t,n}-1}) p_{v,t,n} a_n \\ &+ \mu \alpha^r_{v,t,n} \beta^r_{v,t,n} \mathrm{e}^{\beta^r_{v,t,n} \kappa_{v,t,n,c}} = 0. \end{aligned} \tag{8}$$

Then, we utilize Lambert W function [31] to represent each $\kappa_{v,t,n,c}$ using μ:

$$\kappa_{v,t,n,c} = \frac{1-\beta^d_{v,t,n}}{\beta^r_{v,t,n}} W\Big(\frac{\beta^r_{v,t,n}}{1-\beta^d_{v,t,n}} \mathrm{e}^{\frac{-\ln \frac{\mu \alpha^r_{v,t,n} \beta^r_{v,t,n}}{-\alpha^d_{v,t,n} \beta^d_{v,t,n} p_{v,t,n} a_n}}{1-\beta^d_{v,t,n}}}\Big). \tag{9}$$

Last, we substitute $\kappa_{v,t,n,c}$ into Eq. (7) to derive the optimal μ and the corresponding $\mathbf{K_{v,t,c}}$. Notice that, if some optimal solution $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$ falls outside of $[\kappa_{\text{min}}, \kappa_{\text{max}}]$, the QP optimizer caps $\kappa_{v,t,n,c}$ at κ_{min} or κ_{max}. It then recalculates $\mathbf{K_{v,t,c}}$ until all QP values fall in the practical range of $[\kappa_{\text{min}}, \kappa_{\text{max}}]$.

```
1:  $\mathbf{X^*_{v,c}} \leftarrow \emptyset$
2:  for $t \leftarrow 1$ to $T$ do
3:      // QP optimizer
4:      $\mathbf{K_{v,t,c}} \leftarrow$ Solved with Eqs. (7) and (9)
5:      while $\exists \kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$, where $\kappa_{v,t,n,c} \notin [\kappa_{\text{min}}, \kappa_{\text{max}}]$ do
6:          Set the out-of-range $\kappa_{v,t,n,c}$ to the closest border
7:          $\mathbf{K_{v,t,c}} \leftarrow$ Solved with Eqs. (7) and (9)
8:      // Optimal rounding algorithm
9:      Solve Eqs. (10a)–(10d) for $\mathbf{K^*_{v,t,c}}$
10:     Transform $\mathbf{K^*_{v,t,c}}$ to $\mathbf{X^*_{v,t,c}}$
11:     $\mathbf{X^*_{v,c}} \leftarrow \mathbf{X^*_{v,c}} \cup \mathbf{X^*_{v,t,c}}$
12: return $\mathbf{X^*_{v,c}}$
```

Fig. 6: The pseudocode of the PC-LBA algorithm.

Optimal rounding algorithm. Next, we round the real number QPs in $\mathbf{K}_{v,t,c}$ into integer QPs for the video encoder. We let $\mathbf{K'_{v,t,c}}$ be a subset of $\mathbf{K}_{v,t,c}$ containing tiles in $\mathbf{K_{v,t,c}}$ with non-integer optimal QP values. Our problem is to determine whether to round each $\kappa_{v,t,n,c}$ in $\mathbf{K'_{v,t,c}}$ up or down to minimize the resulting distortion without consuming excessive bandwidth. For $n \in [1, |\mathbf{K'_{v,t,c}}|]$, we define the decision variable $z_{v,t,n,c,0} = 1$ if $\kappa_{v,t,n,c}$ is rounded down, and $z_{v,t,n,c,0} = 0$ otherwise. Similarly, we define $z_{v,t,n,c,1} = 1$ if $\kappa_{v,t,n,c}$ is rounded up and $z_{v,t,n,c,1} = 0$ otherwise. The optimal rounding problem can then be written as:

$$\min \sum_{n=1}^{|\mathbf{K'_{v,t,c}}|} z_{v,t,n,c,0} d_i(\lfloor \kappa'_{v,t,n,c} \rfloor) + z_{v,t,n,c,1} d_{v,t,n}(\lceil \kappa'_{v,t,n,c} \rceil) \tag{10a}$$

$$\begin{aligned} st: &\sum_{n=1}^{|\mathbf{K'_{v,t,c}}|} z_{v,t,n,c,0} r_i(\lfloor \kappa'_{v,t,n,c} \rfloor) + z_{v,t,n,c,1} r_i(\lceil \kappa'_{v,t,n,c} \rceil) \\ &\leq \sum_{n=1}^{|\mathbf{K'_{v,t,c}}|} r_{v,t,n}(\kappa'_{v,t,n,c}); \end{aligned} \tag{10b}$$

$$z_{v,t,n,c,0} + z_{v,t,n,c,1} = 1 \qquad n \in [1, |\mathbf{K'_{v,t,c}}|]; \tag{10c}$$

$$z_{v,t,n,c,0}, z_{v,t,n,c,1} \in \{0,1\} \qquad n \in [1, |\mathbf{K'_{v,t,c}}|]; \tag{10d}$$

In this formulation, Eq. (10a) minimizes the additional distortion due to rounding, while Eq. (10b) makes sure that the bitrate does not exceed the bandwidth constraint. Eq. (3g) ensures that each QP value is either rounded down or up, but not both. The formulation can be optimally solved for $\mathbf{K^*_{v,t,c}}$ using existing solvers, like CPLEX [32] and GLPK [33]. Last, we transform the optimal $\mathbf{K^*_{v,t,c}}$ back to $\mathbf{X^*_{v,t,c}} = \{x^*_{v,t,n,c,q} | q \in [1,Q], n \in [1,N]\}$ of the original problem.

Pseudocode. Fig. 6 presents the pseudocode of our PC-LBA. Lines 4–7 repeatedly solve Eqs. (7) and (9) until there is no out-of-range $\kappa_{v,t,n,c} \in \mathbf{K_{v,t,c}}$. Line 9 rounds the real QP values into integers. Line 10 transforms the QP set $\mathbf{K_{v,t,c}}$ to the binary set $\mathbf{X_{v,t,c}}$. Line 11 collects the solution for each segment. Line 12 returns the optimal solution $x^*_{v,t,n,c,q} \in \mathbf{X^*_{v,c}}$ for clients in bandwidth class c watching video v. The following lemma analyzes the complexity of PC-LBA.

Lemma 5: The PC-LBA algorithm runs in time $O(T2^N)$ with space complexity of $O(N)$.

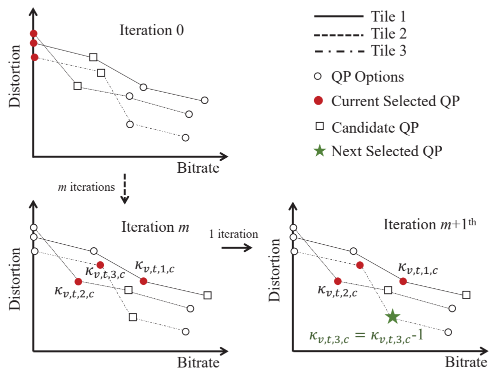


Fig. 7: An illustrative example of PC-GBA with several iterations.

```
1: $\mathbf{X}^*_{\mathbf{v},\mathbf{c}} \leftarrow \emptyset$
2: for $t \leftarrow 1$ to $T$ do
3:     Set $\kappa_{v,t,n,c} \in \mathbf{K}_{\mathbf{v},\mathbf{t},\mathbf{c}}$ as $\kappa_{\max}$, $\forall i = 1, 2, \ldots, N$
4:     $B' = b_c - \sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,t,n,c})$
5:     while $B' > 0$ do
6:         //Tile Selector
7:         $\Theta \leftarrow \{\theta_{v,t,n,c} | n = 1, 2, \ldots, N\}$
8:         $n^* \leftarrow \arg\max \Theta$
9:         //Status Tracker
10:        $B' = B' - (r_{v,t,n^*}(\kappa_{v,t,n^*,c} - 1) - r_{v,t,n^*}(\kappa_{v,t,n^*,c}))$
11:        $\kappa_{v,t,n^*,c} = \kappa_{v,t,n^*,c} - 1$
12:    Transform $\mathbf{K}^*_{\mathbf{v},\mathbf{t},\mathbf{c}}$ to $\mathbf{X}^*_{\mathbf{v},\mathbf{t},\mathbf{c}}$
13:    $\mathbf{X}^*_{\mathbf{v},\mathbf{c}} \leftarrow \mathbf{X}^*_{\mathbf{v},\mathbf{c}} \cup \mathbf{X}^*_{\mathbf{v},\mathbf{t},\mathbf{c}}$
14: return $\mathbf{X}^*_{\mathbf{v},\mathbf{c}}$
```

Fig. 8: The pseudocode of the PC-GBA algorithm.

C. Greedy-based Algorithm: PC-GBA

PC-LBA may suffer from higher computational complexity due to the ILP formulation of optimal rounding[6]. Therefore, we also propose a more efficient greedy algorithm to solve the problem with discrete QPs. Our greedy algorithm contains two components: (i) status tracker and (ii) tile selector. The status tracker keeps track of the current QP selected for each tile, their accumulated bitrate, and the remaining bandwidth. The tile selector selects the tile with the highest *coding efficiency* to allocate more bitrate by reducing its QP. The status tracker then updates the accumulated bitrate and the remaining bandwidth. The above steps *iterate* until there is no remaining bandwidth or all tiles are coded at the smallest QP values.

The crux of the greedy algorithm is the definition of the coding efficiency $\theta_{v,t,n,c}$ when allocating the additional bitrate to tiled-segment (t, n) of video v. We let $\kappa_{v,t,n,c}$ be the current selected QP for tiled-segment (t, n) of v streamed to class c clients. We then define $\theta_{v,t,n,c}$ as:

$$\frac{[d_{v,t,n}(\kappa_{v,t,n,c} - 1) - d_{v,t,n}(\kappa_{v,t,n,c})] p_{v,t,n} a_n}{r_{v,t,n}(\kappa_{v,t,n,c} - 1) - r_{v,t,n}(\kappa_{v,t,n,c})}, \tag{11}$$

where $p_{v,t,n}$ is the viewing probability and a_n is the area scaling factor. $\theta_{v,t,n,c}$ is essentially the slope of the rate-distortion curves at $\kappa_{v,t,n,c}$. Fig. 7 shows an illustrative example of the proposed PC-GBA streaming a video with three tiles. The current QPs are marked with solid dots, which are selected from all QP options marked with circles. Besides, the next candidate QPs of each tile are represented by squares. In this figure, after iteration m, the tile selector chooses tile three to allocate more bandwidth since it has the steepest slope. Note that if all tiles have the same coding efficiency, we use the weights $p_{v,t,n} a_n$ and then the QPs to break the ties. After all the QP values are determined, we transform $\kappa^*_{v,t,n,c} \in \mathbf{K}^*_{\mathbf{v},\mathbf{t},\mathbf{c}}$ to binary indicators $x^*_{v,t,n,c,q} \in \mathbf{X}^*_{\mathbf{v},\mathbf{t},\mathbf{c}}$.

[6]Notice that when optimality is not a major concern, much simpler rounding algorithms, such as rounding down, can be adopted for lower computation complexity.

Pseudocode. Fig. 8 presents the pseudocode of our PC-GBA algorithm that determines the QPs for segments. Lines 3–4 initialize the QP of each tile at the maximum QP and the remaining bandwidth. The while loop between lines 5–11 iteratively allocates more bitrate to the tile with the highest coding efficiency selected in line 8. Lines 10–11 update the status through the status tracker, including the QP value of the selected tile and the remaining bandwidth. Line 12 transforms the determined QP set $\mathbf{K}_{\mathbf{v},\mathbf{t},\mathbf{c}}$ to binary set $\mathbf{X}_{\mathbf{v},\mathbf{t},\mathbf{c}}$. Line 13 collects the solution for each segment. Line 14 returns the optimal solution $x^*_{v,t,n,c,q} \in \mathbf{X}^*_{\mathbf{v},\mathbf{c}}$ for clients in bandwidth class c of video v.

Lemma 6: The PC-GBA algorithm runs in time $O(TN(\log N)Q)$ with space complexity of $O(N)$.

VI. Global Optimization for the Optimal Ladders

```
1: Initialize $\mathbf{Y}^* = \{y_{v,t,n,q} = 0\}$
2: // Per-class optimization
3: $X^* = \{\mathbf{X}^*_{\mathbf{v},\mathbf{c}} | \forall v, c\} \leftarrow$ PC-LBA or PC-GBA
4: for $v \leftarrow 1$ to $V$, $t \leftarrow 1$ to $T$, $n \leftarrow 1$ to $N$, $q \leftarrow 1$ to $Q$ do
5:     if $\sum_{c=1}^{C} x_{v,t,n,c,q} \geq 1$ then
6:         $y_{v,t,n,q} \leftarrow 1$
7: $S' \leftarrow \sum_{v=1}^{V} \sum_{t=1}^{T} \sum_{n=1}^{T} \sum_{q=1}^{Q} r_{v,t,n}(q) y_{v,t,n,q}$
8: // Global optimization
9: while $S' > S$ do
10:    $\mathbf{E} \leftarrow \{\epsilon_{v,t,n,c,q} | v \in [1,V], t \in [1,T], n \in [1,N], c \in [1,C], q \in [1,Q]\}$ using Eq. (12)
11:    $(v^*, t^*, n^*, c^*, q^*) \leftarrow \arg\min E$
12:    $x_{v^*,t^*,n^*,c^*,q^*} \leftarrow 0$
13:    $x_{v^*,t^*,n^*,c^*,q^*+\delta} \leftarrow 1$
14:    Update $y_{v^*,t^*,n^*,q^*}$, $y_{v^*,t^*,n^*,q^*+\delta}$, and $S'$
15: return $\mathbf{X}^*, \mathbf{Y}^*$
```

Fig. 9: The pseudocode of the proposed GL-ITRA algorithm.

```
7: $S_v \leftarrow \frac{S}{V}, \forall v = 1, 2, \cdots, V$
8: for $v \leftarrow 1$ to $V$ do
9:     $S' \leftarrow \sum_{t=1}^{T} \sum_{n=1}^{T} \sum_{q=1}^{Q} r_{v,t,n}(q) y_{v,t,n,q}$
10:    while $S' > S_v$ do
11:        $\mathbf{E} \leftarrow \{\epsilon_{v,t,n,c,q} | t \in [1,T], n \in [1,N], c \in [1,C], q \in [1,Q]\}$ using Eq. (12)
12:        $(t^*, n^*, c^*, q^*) \leftarrow \arg\min E$
13:        $x_{v,t^*,n^*,c^*,q^*} \leftarrow 0$
14:        $x_{v,t^*,n^*,c^*,q^*+\delta} \leftarrow 1$
15:        Update $y_{v,t^*,n^*,q^*}$, $y_{v,t^*,n^*,q^*+\delta}$, and $S'$
16: return $\mathbf{X}^*, \mathbf{Y}^*$
```

Fig. 10: The pseudocode of the proposed GL-ITAA algorithm. Note that lines 1–6 are identical to those in Fig. 9 and thus are omitted.

We propose two greedy algorithms to solve the global optimization problem. The first algorithm directly solves the problem and considers all possible storage space allocation

TABLE III: The Comparisons among Ours and Relevant Algorithms

Method	**Phase**	**Problem**	**Objective**	**Constrains**	**Inputs**			**Tile Qualities per Segment**	**Solution Approach**
					Video Model	Viewing Prob.	Client Dist.		
Our Algorithms	Preparation	Optimal laddering	Viewed distortion minimization	Bandwidth, storage space	✓	✓	✓	Multiple	Lagrangian, ILP, and Greedy
Ozcinar et al. [17] (ISM)	Preparation	Optimal laddering	Overall distortion & cost minimization	Bandwidth, storage space, min. bitrate gap	✓		✓	Single	ILP
Chakareski et al. [28] (ICC)	Streaming	Per-class	Viewed distortion minimization	Bandwidth	✓	✓		Multiple	Convex Optimization
Corbillon et al. [34] (MM)	Streaming	Per-class	Viewed quality maximization	Bandwidth, max. bitrate gap		✓		Multiple	Heuristic

across multiple videos, which we refer to as the Global InTeR-video Algorithm (GL-ITRA). The second algorithm simplifies the problem by: (i) equally dividing the storage space among all videos and (ii) assuming the video popularity is uniformly distributed across all videos. We refer to this algorithm as the GLobal InTrA-video Algorithm (GL-ITAA). We present these two algorithms below and compare their performance in Sec. VII.

GL-ITRA Algorithm. We propose a greedy algorithm to adjust the per-class solutions $\mathbf{X}^*_{\mathbf{v,c}}$ for minimizing the expected distortion while meeting both the client bandwidth constraints and overall server storage limit. First, $y_{v,t,n,q}$ is initialized as 1 if and only if $\sum_{c=1}^{C} x_{v,t,n,c,q} \geq 1$, where $v \in [1,V], t \in [1,T], n \in [1,N]$,and $q \in [1,Q]$; $y_{v,t,n,q}$ is set to be 0, otherwise. We introduce a constant system parameter δ to denote the step size of QP adjustments. We then compute the weight of each tile considering the client distribution as $\epsilon_{v,t,n,c,q} =$

$$\frac{\sum_{v=1}^{V}\sum_{c=1}^{C} f_{v,c} \cdot [d_{v,t,n}(q+\delta) - d_{v,t,n}(q)] p_{v,t,n} a_n x_{v,t,n,c,q}}{[r_{v,t,n}(q) - r_{v,t,n}(q+\delta)(1-y_{v,t,n,q+\delta})] y_{v,t,n,q}}. \quad (12)$$

$\epsilon_{v,t,n,c,q}$ is the ratio between the weighted distortion gain and the storage reduction if the QP value of tiled-segment (t,n) of v increases[7]. In particular, $d_{v,t,n}(q+\delta) - d_{v,t,n}(q)$ is the distortion gain, while $\sum_{v=1}^{V}\sum_{c=1}^{C} f_{v,c}$ and $p_{v,t,n} a_{v,t,n}$ are the weights. $r_{v,t,n}(q) - r_{v,t,n}(q+\delta)(1-y_{v,t,n,q+\delta})$ denotes the reduced storage space, while $(1-y_{v,t,n,q+\delta})$ indicates whether tiled-segment (t,n) of v has already been chosen to streamed. That is, if $y_{v,t,n,q+\delta} = 1$, then the reduced storage space is $r_{v,t,n}(q)$, otherwise the reduced storage space is $r_{v,t,n}(q) - r_{v,t,n}(q+\delta)$. The algorithm iteratively increases the QP of the tile having the lowest $\epsilon_{v,t,n,c,q}$ by step size δ until the storage limit S is not exceeded.

Pseudocode. Fig. 9 shows the pseudocode of the proposed GL-ITRA algorithm. Lines 1–3 initialize $\mathbf{Y}^*$ and compile $\mathbf{X}^*$ using the PC-LBA or PC-GBA algorithms. Lines 4–6 set $y_{v,t,n,q}$ according to $x_{v,t,n,c,q}$. Line 7 initializes the current storage size S'. Lines 9–14 greedily select the tile to adjust its QP value according to Eq. (12) iteratively until the required storage space S' fits the storage limit S. Lines 12–14 update $\mathbf{X}^*$ and $\mathbf{Y}^*$, and the current required storage space. Line 15 returns the decisions $\mathbf{X}^*$ and $\mathbf{Y}^*$.

Lemma 7: The GL-ITRA algorithm runs in time $O(VTNC(\log VTNC)\frac{Q}{\delta})$ with space complexity of $O(VTNC)$.

[7] $\epsilon_{v,t,n,c,q}$ is undefined when $y_{v,t,n,q} = 0$, which is not an issue because the term is never considered in the algorithm.

GL-ITAA Algorithm. We propose a simplified greedy algorithm for lower time and space complexity. In particular, we assume the storage space is uniformly assigned to all videos and the probabilities of videos (w_v^v) are equal. The global optimization problems of individual videos are then decoupled and can be independently solved.

Pseudocode. The pseudocode of GL-ITAA differs from GL-ITRA (in Fig. 9) only from line 7, which is presented in Fig. 10. Line 7 initializes the storage limit S_v for each video v. Lines 8–15 greedily adjust the QP values of the selected tiles for each video v. Line 16 returns the $\mathbf{X}^*$ and $\mathbf{Y}^*$ aggregated from the decision of each video. Because of the optimization problems of individual videos can be independently solved, its time and space complexities are $O(VTNC(\log TNC)\frac{Q}{\delta})$ and $O(TNC)$.

VII. Evaluations

We compare our proposed algorithms against the state-of-the-art ones through extensive experiments in this section.

A. Implementations

Table III summarizes our algorithms and the state-of-the-art algorithms. In this table, Ozcinar et al. [17] and our algorithms solve the optimal laddering problem in the preparation phase, while other algorithms only solve the per-class optimization problem in the streaming phase. Furthermore, the per-class optimization algorithms (Corbillon et al. [34] and Chakareski et al. [28]) do not consider the client distribution. In addition, Corbillon et al. [34] assume all tiles have the same complexity levels and do not take video models into account. Besides, the other optimal laddering algorithm (Ozcinar et al. [17]) does not consider the viewing probability of each tile. *Compared to the abovementioned algorithms [17], [34], [28], our algorithms are the only ones that consider all three features: video models, viewing probability, and client distribution.*

We have implemented our proposed algorithms and three state-of-the-art algorithms [28], [34], [17] for evaluations and comparisons. Note that we use Python for implementations as much as we can. Certain optimization problems, however, can be efficiently solved with specialized solvers that are not written in Python. For example, cvx [35] and CPLEX [32] are tailored for solving convex optimization and ILP problems, respectively. Similarly, MATLAB [36] supports optimization problems with symbolic variables. We opt to call these specialized solvers from Python rather than reimplementing them, in order to understand the performance achieved by real-life implementations. In particular, we use MATLAB [36] and CPLEX [32] to implement PC-LBA's QP optimizer and optimal rounding algorithm, respectively. We present the detailed implementations of the state-of-the-art algorithms in the following.

- **Per-class optimization algorithms:**
  - **Chakareski et al. [28] (ICC)** formulate the tile quality selection problem into an ILP problem and propose to solve it using convex optimization. We use cvx [35] to implement this algorithm.
  - **Corbillon et al. [34] (MM)** propose a heuristic algorithm for bitrate allocation within a 360° video [37]. Their algorithm can be extended for tiled videos by classifying tiles into foreground and background tiles. Their classification [37] is based on the ground truth of user viewport, which is impractical because of the network latency. Therefore, we use 25 percentile of viewing probability as the threshold for the classification[8]. A 3.5 ratio of maximum bitrate gap between the maximum surface bitrate and minimum surface bitrate is set following the recommendation in their paper. Each tile is allocated with the bitrate proportional to the area scaling factor a_n. We use Python to implement this algorithm.
- **Optimal laddering algorithms:**
  - **Ozcinar et al. [17] (ISM)** formulate the optimal laddering problem into an ILP problem without proposing any efficient algorithms. In addition to the storage limit, they also consider different resolutions and introduce a 1.2 ratio of minimum bitrate gap between any two adjacent representations. We use CPLEX to implement this algorithm.

We adopt the same video models in Sec. III-B for all algorithms for fair comparisons.

B. Setup

Parameters, traces, and videos. Several system parameters are varied in our experiments. We fit the bandwidth CDF curve following Cisco's forecast on 2019 fixed broadband bandwidth in North America [38]. If not otherwise specified, we select the bandwidth classes in {3.12, 4.68, 7.02, 10.52, 15.78, 23.67, 35.51, 53.28, 79.91, 119.87} Mbps, which is geometric progression with 1.5 times that covers the bandwidth range of the bandwidth CDF curve. Besides, we adopt the smallest step size of 1 for optimal overall distortion. Such fine-grained step size results in slightly longer running time, which however is insignificant (up to 7% in our experiments) compared to the video encoding time. We randomly select 10 users from the 50 users in a public dataset [26] to evaluate the performance of all algorithms. The remaining 40 users are used to derive the viewing probability for fair comparisons. Each selected user watches six 1-min videos. The videos are classified into three categories:(i) Computer-Generated, Fast-Paced (CG-FP), (ii) Natural Image, FP (NI-FP), and (iii) NI, Slow-Paced (NI-SP). Two videos are selected from each video category at a resolution of 3840×1920 with 6×4 tiles, 6×4 tiles have been shown to achieve the best tradeoff between viewport flexibility and bitrate overhead [39], while other numbers of tiles can also be used with our proposed solutions. The considered encoding QPs are in [1,51] if not otherwise specified. Besides, we take the number of views of the considered videos on YouTube as the video popularity in the evaluation. For example, the most popular Hog Rider and Mega Coaster account for the two highest ratio, which are about 38.69% and 32.36%, respectively.

For conservative comparisons, we let ISM take additional resolutions in {2560×1280, 1920×960} into considerations. Because the production server has limited memory, the ISM algorithm can only consider a QP step of 5 in [1,51] without exceeding a memory consumption of 12 GB. Besides, the ISM algorithm tends to terminate after many days, and we set a practical time limit of 2 hours for each video, which is more than 3 times of the running time of our algorithms.

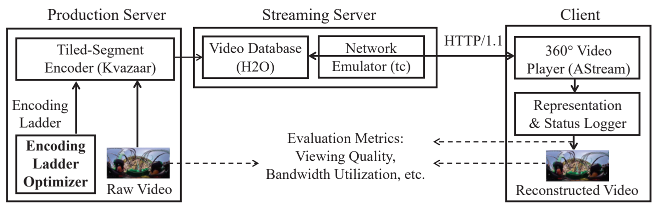

Production Server
Streaming Server
Client
Tiled-Segment Encoder (Kvazaar)
Video Database (H2O)
Network Emulator (tc)
HTTP/1.1
360° Video Player (AStream)
Representation & Status Logger
Encoding Ladder
Encoding Ladder Optimizer
Raw Video
Evaluation Metrics: Viewing Quality, Bandwidth Utilization, etc.
Reconstructed Video

Fig. 11: The overview of our evaluation testbed.

Testbed. To faithful the experiments, we build a real streaming testbed following the one implemented in Yen et al. [16]. The testbed is illustrated in Fig. 11. We setup two Intel i7 workstations with 16 GB RAM running Linux. One of them contains both the production and streaming servers and the other one runs the client. The two workstations are directly connected to each other with a GigE network cable. We employ `tc` [40] in Linux to throttle the network bandwidth. On the server, we adopt *Kvazaar* [27] as the tiled segment encoder to encode the videos. We use *H2O* [41] as the HTTP server to store the representations following the decisions from the encoding ladder optimizer. On the client, we use a Python-based DASH client, *AStream* [42] as the 360° video player. Besides, the status, such as throughput and stalls, and the received representations are logged for further analysis.

We have implemented an ABR algorithm [13] designed for 360° video streaming in the player[9]. Our implemented ABR algorithm performs viewport prediction based on user's previous orientations. The scene is split into three parts: (i) *viewport*, which is the predicted user's viewport, (ii) *extended area*, which is 30° outside the predicted viewport, and (iii) *background*, which is the remaining tiles of the scene. The ABR algorithm first allocates the lowest representation to each part, then allocates the highest affordable representation to the viewport. The residual bandwidth is allocated to the extended

[8]We also tried 50 and 75 percentile and observed similar results, which are left out for clarity.

[9]Other ABR algorithms in the literature can be adopted in our work as well.

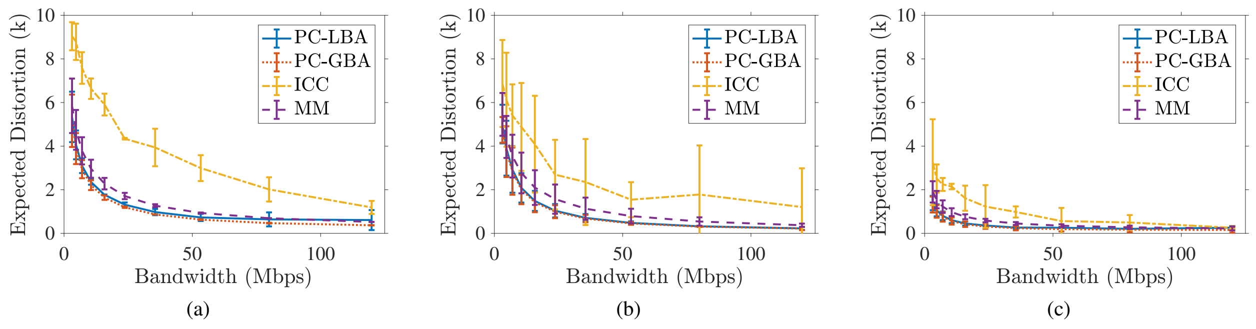


Fig. 12: Expected distortion under different bandwidth: (a) CG-FP, (b) NI-FP, and (c) NI-SP.

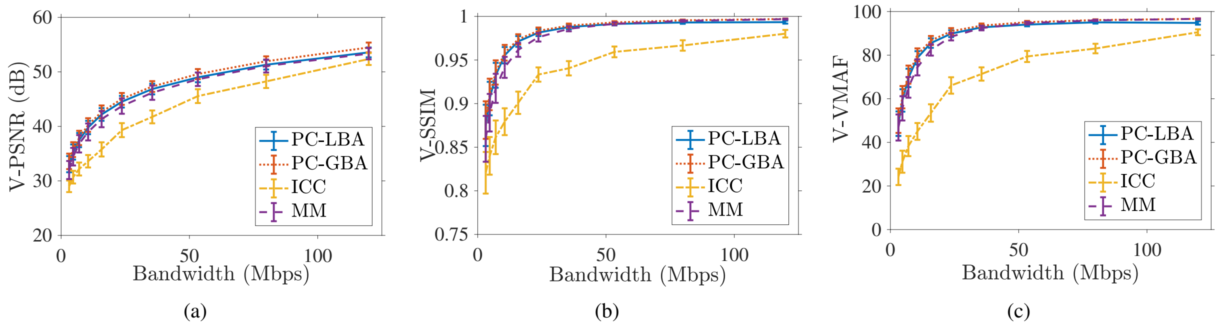


Fig. 13: The viewing quality under different bandwidth: (a) V-PSNR, (b) V-SSIM, and (c) V-VMAF.

area, followed by the background. To accommodate some background traffic, we instruct the ABR algorithm to set the available bandwidth at 70% of the measured throughput.

We evaluate the results using the following metrics:

- **Viewing quality.** We consider V-PSNR [43], V-SSIM, and V-VMAF, which essentially are the PSNR [44], Structural Similarity Index (SSIM) [45], and VMAF [22] in the HMD viewports. The computation of V-VMAF is similar to that in Ozcinar et al. [46]. Because V-SSIM and V-VMAF are not pixel-wise metrics, we cut each circle viewport[10] into its inscribed square when computing them. We believe the negative impacts of removing small areas close to the viewport borders are insignificant as they are far away from the viewport center.
- **Bandwidth utilization.** The ratio between the streamed bitrate to the total bandwidth.
- **Number of stalls.** The total number of stalls of the algorithms throughout each 1-min playout.
- **Running time.** The consumed time of the algorithms for determining the encoding ladder.
- **MPD overhead.** The ratio between the size of the MPD file (meta-data) and the total streamed data in each streaming session.

In the following sections, we first evaluate the per-class optimization algorithms in terms of distortion and viewing quality. After that, we conduct extensive experiments on the testbed to evaluate the performance of optimal laddering algorithms, which solve the global optimization problem. This is followed by a summary of our key findings. The results are reported with 95% confidence interval plotted as error bar.

[10]Our HMD (Oculus DK2) is measured to have a circle viewport [18].

C. Per-Class Optimization Results

Our proposed PC-LBA and PC-GBA algorithms effectively offer lower expected distortion. Fig. 12 plots the expected distortion under different bandwidth levels for different video categories, which is computed with Eq. (4a). This figure shows that our proposed algorithms effectively reduce the expected distortion compared to other state-of-the-art algorithms. ICC sometimes (around $\frac{1}{3}$ of the time) fails to find the solutions using cvx, and thus results in the highest expected distortion across all videos. In contrast, our PC-LBA algorithm is a customized algorithm solving a convex programming problem for low distortion. The greedy PC-GBA algorithm also achieves comparable distortion than the PC-LBA algorithm. MM is a heuristic algorithm that results in slightly higher distortion compared to our algorithms. Across different video categories, videos in NI-SP result in lower expected distortion in general. This is because NI-SP videos contain simpler scenes and slower movements compared to other videos, which lead to higher coding efficiency.

Our proposed algorithms outperform others more in viewing quality at lower bandwidth. We plot the viewing quality of all videos in user's viewport in Fig. 13. All three quality metrics demonstrate the same trend: our proposed algorithms deliver higher viewing quality than others in most bandwidth classes, especially at lower bandwidth. We emphasize that the performance of low bandwidth classes is more crucial, as clients in these classes are more vulnerable to inferior viewing experience. Because three viewing quality metrics show similar trends, we only report the quality in V-VMAF in the rest of the article.

We next plot the overall viewing quality of different video

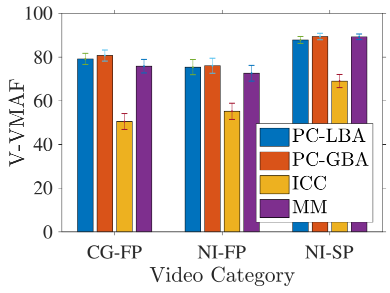


Fig. 14: The viewing quality of different video categories.

TABLE IV: Maximum and Average of Quality Improvement in V-VMAF Compared to State-of-the-Art Algorithms

PC-LBA		
State-of-the-Art	**Avg.**	**Max.**
ICC	22.57	50.19
MM	1.55	24.82

PC-GBA		
State-of-the-Art	**Avg.**	**Max.**
ICC	23.85	52.17
MM	2.83	26.35

TABLE V: The Average Per-Segment Running Time of PC-LBA and PC-GBA

Algorithm	**PC-LBA**			**PC-GBA**
Component	**QP Optimizer**	**Optimal Rounding Algorithm**	**Total**	**Total**
Time (s)	73.8315	44.6170	118.4485	0.1074

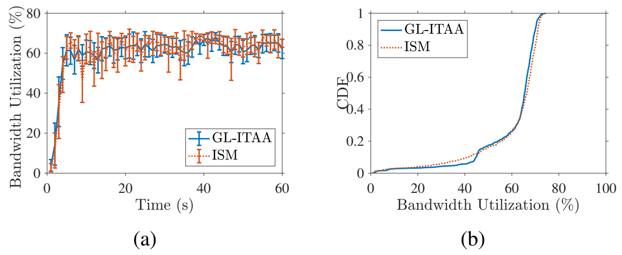


Fig. 15: The bandwidth utilization across 10 users watching a sample video at storage limit S=1200 MB: (a) over time and (b) CDF.

categories in Fig. 14. This figure shows the merits of our algorithms, over ICC and MM algorithms. It also shows that the videos from NI-SP have higher viewing quality for all algorithms, which is consistent with Fig. 12. We report the average and maximum improvements of our algorithms over the state-of-the-art algorithms in Table IV. This table illustrates that our PC-LBA averagely outperforms ICC and MM by 22.57 and 1.55 in V-VMAF. Moreover, PC-GBA outperforms ICC and MM by 23.85 and 2.83. In fact, the improvements of PC-GBA are as high as 52.17 and 26.35. From the figures, we observe that MM achieves reasonably good viewing quality. However, it only supports a single video *and* a single bandwidth class. In contrast, we also study the optimal laddering problem that takes client distribution and storage limit into considerations, which will be evaluated in Sec. VII-D.

PC-GBA achieves even better performance than PC-LBA. Our evaluation results show that PC-GBA offers better viewing quality than PC-LBA (Figs. 12–14). That is, simultaneously limiting multiple out-of-range QPs into the practical range seriously degrades the viewing quality. In contrast, PC-GBA iteratively and gradually adjusts the QP values in discrete manner, which finds better discrete solutions. We report their average computing time for each segment in Table V. This table shows that PC-GBA runs faster, which is due to the much higher complexity of the Lagrangian multiplier approach in PC-LBA. Because of the better performance of PC-GBA compared to PC-LBA in terms of both viewing quality and computing time, *we adopt PC-GBA as the per-class optimization algorithm* in the rest of this article.

D. Optimal Laddering Results

We use our real testbed to evaluate our optimal laddering algorithms compared to the ISM algorithm. We throttle the network bandwidth of users following the distribution in Cisco's report [38]. Note that the ISM algorithm does not take video popularity into considerations, and thus we only compared it against GL-ITAA, where the storage space limit is evenly divided among all videos.

Our testbed effectively performs adaptive streaming over the network. Fig. 15(a) plots the bandwidth utilization across 10 users watching a sample video over time. This figure reveals that our testbed effectively runs ABR algorithm at the clients, which achieves around 60% of bandwidth utilization after 4 seconds. Besides, Fig. 15(b) plots the CDF of the bandwidth utilization indicating that most of the bandwidth utilization is about 65%, which is quite close to the target 70% of the available bandwidth. Throughout the experiments, we observe no stall events.

Our proposed GL-ITAA algorithm outperforms the state-of-the-art algorithm. We then conduct experiments to evaluate the performance under different storage limits. Fig. 16 plots the CDF of the bandwidth utilization achieved by GL-ITAA across 10 users watching 6 videos under different storage limits. This figure shows that the smaller storage limits result in lower bitrates of some representations, which in turn lead to lower bandwidth utilization. This demonstrates the effectiveness of our GL-ITAA algorithm. We next compare the overall viewing quality with the ISM algorithm. Fig. 17 plots the viewing quality under different storage limits. This figure shows that our proposed GL-ITAA algorithm outperforms the ISM algorithm under all considered storage limits. Moreover, our algorithm outperforms ISM algorithm by larger margins as the storage limit decreases. In particular, our GL-ITAA algorithm averagely outperforms the state-of-the-art ISM algorithm by 43.14 in V-VMAF when the storage limit is as low as 400 MB per video. This indicates that our GL-ITAA algorithm scales well under different storage limits. This can be attributed to the fact that our GL-ITAA algorithm effectively reduces the required storage by cutting the bitrate allocated to the less important tiles that are rarely viewed.

To confirm that our proposed algorithm outperforms ISM in terms of Quality of Experience (QoE), we conduct a user study to quantify the real user experience. We randomly select a user trace watching 6 videos and generate the viewport videos using the trace. We then recruit 12 subjects to watch these viewport

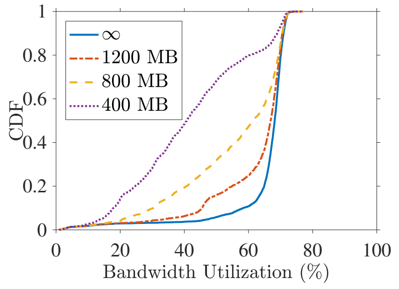


Fig. 16: CDF of viewing quality across different viewers and videos.

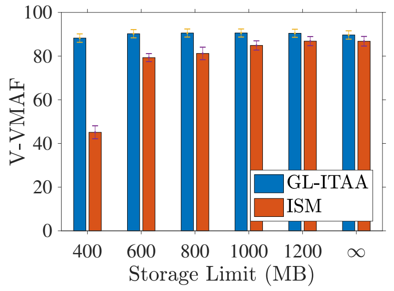


Fig. 17: Viewing quality under different storage limits.

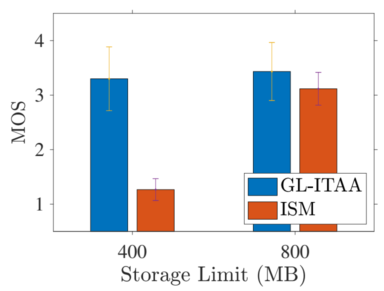


Fig. 18: The MOS scores under different storage limits.

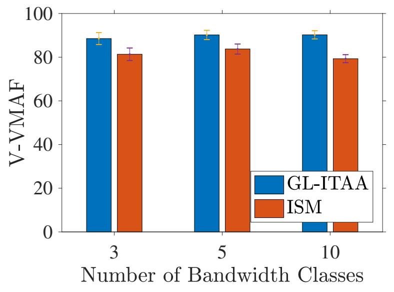


Fig. 19: Viewing quality under different number of representations at storage limit 600 MB.

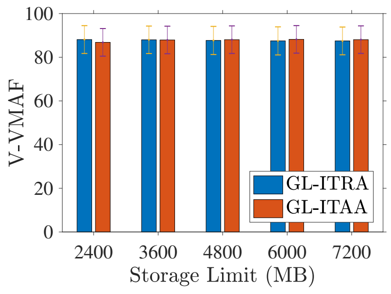


Fig. 20: The viewing quality of GL-ITRA and GL-ITAA.

TABLE VI: The Ratio of the Meta-Data in DASH for Our Proposed Algorithm

Storage Limit	Meta-Data	Video Data
Unlimited	0.112%	99.888%
1200 MB	0.115%	99.885%
1000 MB	0.118%	99.882%
800 MB	0.123%	99.877%
600 MB	0.133%	99.867%
400 MB	0.152%	99.848%

videos in random order and give overall quality scores in [1,5] scale. Fig. 18 plots the Mean Opinion Score (MOS) with 95% confidence intervals from different algorithms under 400 and 800 MB storage limits. In this figure, GL-ITAA outperforms ISM across all storage limits. Besides, the p-value from the ANOVA test is 0.0029 (< 0.05), which shows the statistical significance. As an example, GL-ITAA outperforms ISM by more than 2 points (out of a range of 4 points) in MOS at 400 MB storage limit. The above observations are consistent with our earlier observations on Fig. 17, which are made in V-VMAF. That is, we find that V-VMAF closely follows the user experience derived from time-consuming and expensive user studies. Hence, we use V-VMAF as the quality metric in the rest of this article.

We then consider $C \in \{3, 5, 10\}$, where the considered bandwidth are $\{3.12,\ 10.52,\ 35.52\}$ Mbps and $\{3.12,\ 7.02,\ 15.78,\ 35.51,\ 79.91\}$ Mbps for $C = 3$ and 5, respectively. We report the results across all videos and 10 users with storage limit $S = 600$ MB per video in Fig. 19. This figure shows the good scalability for our algorithm on different number of bandwidth classes. In particular, our GL-ITAA algorithm averagely outperforms the ISM algorithm by 8.2 in V-VMAF under different number of bandwidth classes. In summary, our GL-ITAA algorithm delivers higher viewing quality under various conditions. Next, we conduct experiments to see whether GL-ITRA can further improve the viewing quality.

GL-ITAA runs more efficiently while offering similar viewing quality compared to GL-ITRA. Last, we introduce diverse video popularity and compare the performance of GL-ITAA and GL-ITRA. In this experiment, we let each user watch 1 of the 6 considered videos following the video popularity. Fig. 20 plots the viewing quality of both the GL-ITRA and GL-ITAA under different storage limits. This figure shows that the GL-ITRA and GL-ITAA achieve almost the same viewing quality regardless of the storage limits. This is because both algorithms effectively reduce the resource allocated to the tiles with lower viewing probability. However, the running time of the GL-ITRA is at least an order of magnitude longer than the GL-ITAA, while GL-ITAA averagely spends 39 minutes across different storage limits for each video. Moreover, it is suitable in practical usage scenarios, where new videos are gradually added to the streaming servers without: (i) *recomputing* the optimal encoding ladders and (ii) *re-encoding* the new representations of the existing videos. Last, GL-ITAA is easier to be parallelized and thus is more scalable. Hence, we recommend GL-ITAA for solving the optimal laddering problem.

Our proposed algorithms incur small meta-data overhead. Last, we measure the overhead of our proposed algorithms. In particular, our adopted per-tile-per-segment video models occupy averagely 53.33 KB storage space per 1-min long video. This is equivalent to about 0.013% overhead at 400 MB storage limit. We give the average ratio between the size of the meta-data and the total steamed data under different storage limits in Table VI. This table shows that the meta-data overhead increases as the storage limit decreases, but it never exceeds 0.2% of the total data size.

E. Summary of the Key Findings

The following summarizes findings on the evaluation results.

- **Per-class optimization.** Our PC-LBA and PC-GBA algorithms optimize the viewing quality of the clients in the same bandwidth class. Our evaluation results show that PC-LBA and PC-GBA outperform ICC and MM by up to 52.17 and 26.35 in V-VMAF, respectively. We recommend PC-GBA for per-class optimization for its higher viewing quality and shorter running time.

- **Global optimization under assumptions.** Our GL-ITAA algorithm solves a simplified global optimization problem for optimal laddering. The goal is to optimize the overall viewing quality of the clients, where each video has a pre-determined storage limit. Our evaluation results show that GL-ITAA outperforms ISM by up to 43.14 in V-VMAF when the storage limit is 400 MB per video. Moreover, our GL-ITAA delivers better viewing quality and runs faster than ISM. Our GL-ITAA scales well in terms of both storage limits and number of bandwidth classes.
- **Global optimization.** Our GL-ITRA algorithm solves the most general optimal laddering problem, which jointly optimizes the overall viewing quality of the clients across multiple videos. To the best of our knowledge, none of the existing work in the literature addresses the same problem. While our results show that both GL-ITRA and GL-ITAA achieve high viewing quality, GL-ITAA runs much faster than GL-ITRA.

In summary, we recommend GL-ITAA and PC-GBA for solving the optimal laddering problem, which has not been rigorously solved in the literature.

VIII. Conclusion

We study the optimal laddering problem for tiled 360° video streaming to HMD viewers. We consider video models, viewing probability, and client distribution to maximize the client viewing quality. We formulate the problem into ILP and take a divide-and-conquer approach to solve the problem. In particular, we decompose it into: (i) per-class optimization for each bandwidth class and (ii) global optimization for the overall client viewing quality optimization. We have proposed two algorithms for each of the per-class optimization and global optimization problems. We have performed both analytical analysis and conducted experiments on a real testbed to quantify the performance of our algorithms compared to three state-of-the-art algorithms. We then recommend a combination of the proposed algorithms to solve the optimal laddering problem, which are the PC-GBA and GL-ITAA algorithms. The results show that our recommended algorithms outperform state-of-the-art algorithms by up to 52.17 and 26.35 in V-VMAF in per-class optimization problem and by up to 43.14 in global optimization problem. Moreover, our recommended algorithms scale well under different storage limits and run efficiently.

We plan to extend our work in multiple directions. First, the selection of some parameters should be systematically done. For example, we plan to develop an algorithm to adaptively determine the considered bandwidth classes so as to further maximize the client viewing quality. Second, we plan to develop new optimal laddering algorithms that are more tightly integrated with the ABR algorithms. For example, different objectives can be applied on our optimal laddering problem according to the employed ABR algorithms to further maximize the user experience in different usage scenarios. Last, a content-aware adaptive tiling scheme can be developed and integrated in our proposed optimal laddering algorithms. For example, videos with static scenes or scattered viewing probability may be encoded with fewer tiles to get better coding efficiency. The interplay among the number of tiles, video characteristics, and storage limits can be studied to further maximize the overall viewing quality. Our work can also be adopted by several future applications, such as 6-Degree-of-Freedom (6DoF) streaming [47], which supports the movements of HMD viewers along x-, y-, and z-axes by leveraging view synthesis algorithms [48], [49] or multi-view video coding tools [50], [51], [52]. Another future application is Extending Reality (XR) gaming [53], which may be optimized using in-game context [54].

References

[1] Cisco Systems, Inc, "2020 global networking trends report," 2020, https://www.cisco.com/c/dam/m/en_us/solutions/enterprise-networks/networking-report/files/GLBL-ENG_NB-06_0_NA_RPT_PDF_MOFU-no-NetworkingTrendsReport-NB_rpten018612_5.pdf?ccid=cc001244&oid=rpten018612.

[2] *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*, International Organization for Standardization Standard, March 2012.

[3] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, vol. 18, no. 4, 2011.

[4] D. I. Forum, "Guidelines for implementation: Dash-if interoperability points," *DASH Industry Forum*, November 2018.

[5] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over http," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, 2018.

[6] Netflix Technology Blog, "Per-title encode optimization," 2015, https://medium.com/netflix-techblog/per-title-encode-optimization-7e99442b62a2.

[7] Y. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, "Optimal design of encoding profiles for ABR streaming." in *Proc. of ACM Workshop on Packet Video (PV'18)*, Amsterdam, The Netherlands, June 2018.

[8] K. Misra, A. Segall, M. Horowitz, S. Xu, A. Fuldseth, and M. Zhou, "An overview of tiles in HEVC," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, 2013.

[9] C. Fan, W. Lo, Y. Pai, and C. Hsu, "A survey on 360° video streaming: Acquisition, transmission, and display," *ACM Computing Surveys*, vol. 52, no. 4, 2019.

[10] C. Ozcinar, A. De Abreu, and A. Smolic, "Viewport-aware adaptive 360 video streaming using tiles for virtual reality," in *Proc. of IEEE International Conference on Image Processing (ICIP'17)*, Beijing, China, September 2017.

[11] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming," in *Proc. of ACM International Conference on Multimedia (MM'17)*, Mountain View, CA, October 2017.

[12] P. Alface, J. Macq, and N. Verzijp, "Interactive omnidirectional video delivery: A bandwidth-effective approach," *Bell Labs Technical Journal*, vol. 16, no. 4, 2012.

[13] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck, "An HTTP/2-based adaptive streaming framework for 360 virtual reality videos," in *Proc. of ACM International Conference on Multimedia (MM'17)*, Mountain View, CA, October 2017.

[14] D. Nguyen, H. Tran, A. Pham, and T. Thang, "A new adaptation approach for viewport-adaptive 360-degree video streaming," in *Proc. of IEEE International Symposium on Multimedia (ISM'17)*, Taichung, Taiwan, December 2017.

[15] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. of International Conference on Mobile Computing and Networking (MobiCom'18)*, New Delhi, India, October 2018.

[16] S. Yen, C. Fan, and C. Hsu, "Streaming 360° videos to head-mounted virtual reality using DASH over QUIC transport protocol," in *Proc. of ACM Workshop on Packet Video (PV'19)*, Amherst, MA, June 2019.

[17] C. Ozcinar, A. De Abreu, S. Knorr, and A. Smolic, “Estimation of optimal encoding ladders for tiled 360 VR video in adaptive streaming systems,” in *IEEE International Symposium on Multimedia (ISM’17)*, Taichiung, Taiwan, December 2017.

[18] C. Fan, J. Lee, W. Lo, C. Huang, K. Chen, and C. Hsu, “Fixation prediction for 360° video streaming in head-mounted virtual reality,” in *Proc. of ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’17)*, Taipei, Taiwan, June 2017.

[19] C. Fan, S. Yen, C. Huang, and C. Hsu, “Optimizing fixation prediction using recurrent neural networks for 360° video streaming in head-mounted virtual reality,” *IEEE Transactions on Multimedia*, vol. 22, no. 3, 2019.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[21] G. Sullivan and T. Wiegand, “Rate-distortion optimization for video compression,” *IEEE Signal Processing Magazine*, vol. 15, no. 6, 1998.

[22] Netflix Inc., “VMAF - video multi-method assessment fusion,” 2019, https://github.com/Netflix/vmaf.

[23] B. Li, D. Zhang, H. Li, and J. Xu, “QP determination by lambda value,” in *9th Meeting of the JCT-VC, no. JCTVC-I0426*, May 2012.

[24] B. Li, H. Li, L. Li, and J. Zhang, “λ domain rate control algorithm for high efficiency video coding.” *IEEE Transactions on Image Processing*, vol. 23, no. 9, 2014.

[25] M. Wang, K. Ngan, and H. Li, “An efficient frame-content based intra frame rate control for high efficiency video coding,” *IEEE Signal Processing Letters*, vol. 22, no. 7, 2015.

[26] W. Lo, C. Fan, J. Lee, C. Huang, K. Chen, and C. Hsu, “360° video viewing dataset in head-mounted virtual reality,” in *Proc. of ACM International Conference on Multimedia Systems (MMSys’17)*, Taipei, Taiwan, June 2017.

[27] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. Hämäläinen, “Kvazaar: Open-source HEVC/H.265 encoder,” in *Proc. of ACM International Conference on Multimedia (MM’16)*, Amsterdam, The Netherlands, October 2016.

[28] J. Chakareski, R. Aksu, X. Corbillon, G. Simon, and V. Swaminathan, “Viewport-driven rate-distortion optimized 360° video streaming,” in *Proc. of IEEE International Conference on Communications (ICC’18)*, Kansas, MO, May 2018.

[29] K.-S. Lu, A. Ortega, D. Mukherjee, and Y. Chen, “Perceptually inspired weighted mse optimization using irregularity-aware graph fourier transform,” *arXiv preprint arXiv:2002.08558*, 2020.

[30] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 2014.

[31] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, “On the LambertW function,” *Advances in Computational mathematics*, vol. 5, no. 1, 1996.

[32] IBM Corp., “IBM ILOG CPLEX optimizer,” 2018, http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[33] A. Makhorin, “GLPK (GNU linear programming kit),” 2019, https://www.gnu.org/software/glpk/.

[34] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, “Optimal set of 360-degree videos for viewport-adaptive streaming,” in *Proc. of ACM International Conference on Multimedia (MM’17)*, Mountain View, CA, October 2017.

[35] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.1,” http://cvxr.com/cvx, March 2019.

[36] The MathWorks, Inc., “MATLAB - MathWorks,” 2019, https://www.mathworks.com/products/matlab.html.

[37] Xavier Corbillon, “Optimal set of 360-degree videos for viewport-adaptive streaming,” 2019, https://github.com/xmar/optimal-set-representation-viewport-adaptive-streaming.

[38] Cisco Inc., “Cisco visual networking index: Forecast and trends, 2017-2022 white paper,” 2017, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[39] M. Graf, C. Timmerer, and C. Mueller, “Towards bandwidth efficient adaptive streaming of omnidirectional video over HTTP,” in *Proc. of ACM International Conference on Multimedia Systems (MMSys’17)*, Taipei, Taiwan, June 2017.

[40] Bert Hubert, “tc,” 2019, https://linux.die.net/man/8/tc.

[41] DeNA Co., Ltd. et al., “H2O the optimized HTTP/1.x, HTTP/2 server,” 2019, https://h2o.examp1e.net/.

[42] Parikshit Juluri, “Astream,” 2019, https://github.com/pari685/AStream.

[43] M. Yu, H. Lakshman, and B. Girod, “A framework to evaluate omnidirectional video coding schemes,” in *Proc. of IEEE International Symposium on Mixed and Augmented Reality (ISMAR’15)*, Fukuoka, Japan, September 2015.

[44] Z. Li, M. Drew, and J. Liu, *Lossy Compression Algorithms*. Springer, 2004.

[45] S. Channappayya, A. Bovik, C. Caramanis, and R. Heath, “SSIM-optimal linear image restoration,” in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP’08)*, Las Vegas, NV, March 2008.

[46] C. Ozcinar, J. Cabrera, and A. Smolic, “Visual attention-aware omnidirectional video streaming using optimal tiles for virtual reality,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, 2019.

[47] 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. of IEEE International Conference on Computer Communications (INFOCOM’19)*, Paris, France, April 2019.

[48] D. Tian, P. Lai, P. Lopez, and C. Gomila, “View synthesis techniques for 3D video,” in *Applications of Digital Image Processing XXXII*, vol. 7443. International Society for Optics and Photonics, 2009.

[49] K. Mueller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, “View synthesis for advanced 3D video systems,” *EURASIP Journal on Image and Video Processing*, vol. 2008, 2009.

[50] S. Valizadeh, P. Nasiopoulos, and R. Ward, “Perceptual distortion measurement in the coding unit mode selection for 3D-HEVC,” in *Proc. of IEEE International Conference on Consumer Electronics (ICCE’16)*, Las Vegas, NV, January 2016.

[51] ——, “Perceptual rate distortion optimization of 3D-HEVC using PSNR-HVS,” *Multimedia Tools and Applications*, vol. 77, no. 17, 2018.

[52] S. Schwarz and M. Hannuksela, “Perceptual quality assessment of HEVC main profile depth map compression for six degrees of freedom virtual reality video,” in *Proc. of IEEE International Conference on Image Processing (ICIP’17)*, Beijing, China, September 2017.

[53] B. Thomas, “A survey of visual, mixed, and augmented reality gaming,” *ACM Computers in Entertainment*, vol. 10, no. 1, 2012.

[54] M. Hegazy, K. Diab, M. Saeedi, B. Ivanovic, I. Amer, Y. Liu, G. Sines, and M. Hefeeda, “Content-aware video encoding for cloud gaming,” in *Proc. of ACM International Conference on Multimedia Systems (MMSys’19)*, Amherst, MA, June 2019.

[55] C. Concolato, J. Feuvre, F. Denoual, E. Nassor, N. Ouedraogo, and J. Taquet, “Adaptive streaming of HEVC tiled videos using MPEG-DASH,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 8, 2017.

[56] R. Skupin, Y. Sanchez, Y. Wang, M. Hannuksela, J. Boyce, and M. Wien, “Standardization status of 360 degree video coding and delivery,” in *Proc. of IEEE International Conference on Visual Communications and Image Processing (VCIP’17)*, Taichung, Taiwan, December 2017.

[57] L. D’Acunto, J. Berg, E. Thomas, and O. Niamut, “Using MPEG DASH SRD for zoomable and navigable video,” in *Proc. of ACM International Conference on Multimedia Systems (MMSys’16)*, Klagenfurt, Austria, May 2016.

[58] J. Feuvre and C. Concolato, “Tiled-based adaptive streaming using MPEG-DASH,” in *Proc. of ACM International Conference on Multimedia Systems (MMSys’16)*, Klagenfurt, Austria, May 2016.

[59] T. ParisTech, “MP4Box,” October 2019, https://gpac.wp.imt.fr/mp4box/.

[60] ——, “MP4Client,” October 2019, https://gpac.wp.imt.fr/player/.

[61] A. Zare, A. Aminlou, M. Hannuksela, and M. Gabbouj, “HEVC-compliant tile-based streaming of panoramic video for virtual reality applications,” in *Proc. of ACM International Conference on Multimedia (MM’16)*, Amsterdam, The Netherlands, October 2016.

[62] R. Ju, J. He, F. Sun, J. Li, F. Li, J. Zhu, and L. Han, “Ultra wide view based panoramic VR streaming,” in *Proc. of ACM Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network’17)*, Los Angeles, CA, August 2017.

[63] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, “Viewport-adaptive navigable 360-degree video delivery,” in *Proc. of IEEE International Conference on Communications (ICC’17)*, Paris, France, May 2017.

[64] F. Duanmu, E. Kurdoglu, S. Hosseini, Y. Liu, and Y. Wang, “Prioritized buffer control in two-tier 360 video streaming,” in *Proc. of ACM Workshop on Virtual Reality and Augmented Reality Network (VR/AR Network’17)*, Los Angeles, CA, August 2017.

[65] J. Kua, G. Armitage, and P. Branch, “A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, 2017.

[66] D. Nguyen, T. Huyen, and T. Thang, “An evaluation of tile selection methods for viewport adaptive streaming of 360-degree video,” *ACM Transactions on Multimedia Computing Communications and Applications*, vol. 16, no. 1, 2020.

Appendix A
Related Work

We survey the literature in this section.

A. Tiled Streaming Standards

The MPEG DASH standard includes an amendment on Spatial Representation Description (SRD) for clients to request SRD extends MPD and provides x-axis, y-axis, width, and height as attributes to DASH clients. Concolato et al. [55] discuss the latest High-Efficiency Video Codec (HEVC) and ISO Base Media File Format (ISOBMFF) standards, which used for encoding and encapsulating tiled videos. They demonstrate that a client may merge several tiles into a video stream, and decode it with a single decoder by combining SRD, HEV, and ISOBMF. Recently, MPEG group develops Omnidirectional MediA Format (OMAF) standard for the delivery and storage of 360° videos. Skupin et al. [56] present the application requirements, projection formats, video/audio codec, and DASH integration of the OMAF standard.

Several papers [57], [58], [39] share their experience of developing standard-based tiled 360° video streaming systems. In particular, D'Acunto et al. [57] use SRD to realize navigable video streaming. They summarize the design choices of the SRD-enabled DASH player, such as: (i) definitions of representations, and (ii) seamless switches among representations. Feuvre and Concolato [58] realizing tiled-based adaptive streaming using several open-source projects, including Kvazaar [27], MP4Box [59], and MP4Client [60]. They demonstrate how interactive navigation and bitrate adaptation can be achieved using DASH and SRD. Furthermore, they present their implementation supporting diverse adaptation policies based on the open-source MP4Client. Graf et al. [39] implement various evaluation tools to quantify the pros and cons of different encoding and streaming strategies on tiled-based 360° video streaming systems. They also discuss various options to enable the bandwidth-efficient adaptive streaming of 360° videos. In their evaluation, 6×4 tiles provide the best tradeoff between tiling overhead and bandwidth consumption, which confirm a bitrate saving of 40% compared to the baseline solutions. The aforementioned studies do not consider the diverse viewing probabilities of individual tiles.

B. Viewport-Adaptive Tiled Streaming

Several studies propose to stream the tiles in the viewports at a higher quality, and other tiles at a lower quality, in order to reduce the bandwidth consumption. For example, Zare et al. [61] adopt HEVC tiled-streaming and propose three heuristic schemes for 360° video streaming to HMDs. Their experiment results confirm that their solution utilizes common patterns of head movements to achieve better coding efficiency. Ju et al. [62] stream low-resolution tiles for the whole 360° videos along with high-resolution tiles for the viewports. They also propose to consider the heatmap of viewers' attentions, and stream tiles with higher viewing probability using broadcast. Corbillon et al. [63] take diverse projection models into considerations, and vary both bitrates and viewports when encoding 360° videos into multiple representations. HMD viewers then request for the proper presentations via DASH. Duanmu et al. [64] encode each 360° video into a base and multiple enhancement representations. They adopt separate buffers for different representations, and give the highest priority to the base representation for smooth playback. The residual bandwidth is then used to stream enhancement representations. The aforementioned studies only differentiate the quality of tiles in an ad-hoc way without intelligently allocating resources among tiles.

C. Adaptive BitRate Algorithms

ABR algorithms have been studied for conventional videos [65]. Recently, several studies [10], [11], [12], [13], [14], [15], [16] design ABR algorithms for tiled 360° videos. Ozcinar et al. [10] consider unequal-size tiles. In their algorithm, higher quality level is selected for the tiles in the viewport. The quality levels of the remaining tiles are gradually reduced as the distance between them and the viewport increases. Xie et al. [11] formulate the ABR for 360° videos into an ILP problem considering the viewing probability of each tile. Their objective is to minimize the expected video distortion in terms of MSE and spatial quality variance. Besides, a buffer-based rate control mechanism is proposed for smooth playback. Alface et al. [12] propose a greedy algorithm to select the quality levels according to the ratio between their considered utility function and the tile size, where the utility function is the estimated quality times the viewing probability. Their considered viewing probability is predicted from the previous viewport and filtered by a Gaussian filter, where the standard deviation σ is proportional to the delay.

A few studies [13], [14], [15] group tiles into a number of classes and assign the same quality level to the tiles in the same class. Petrangeli et al. [13] group the tiles into three classes: (i) viewport, (ii) extended area, and (iii) background. Their algorithm estimates the available bandwidth and select the highest affordable representations for viewport, extended area, and background tiles in that order. Similarly, Nguyen et al. [14] also group the tiles into three classes. The size of their extended area is proportional to the estimated viewport prediction error. Their algorithm then searches all possible quality levels for the maximal estimated viewing quality under the constraint of the available bandwidth. Qian et al. [15] group the tiles into four classes. They search for all the possible quality levels for each class and select the one with the highest considered utility, which aims for higher viewing quality and fewer stalls. A measurement study quantitatively compares the above ABR algorithms under various conditions [66].

D. Bitrate Allocation and Optimal Laddering Algorithms

In addition to the ABR algorithms that select the representations stored on the streaming server, some studies [34], [28] propose algorithms for bitrate allocation that encode and stream tiles at different bitrates to the clients under the constraint of a given bandwidth. Corbillon et al. [34] propose an algorithm to allocate bitrates within a 360° video [37]. Their proposed algorithm can be extended for tiled 360°

videos. In particular, they classify the tiles into two categories based on the viewport and assign even bitrate to the tiles in each category. A bitrate gap is introduced to restrict the quality differences between the two categories to avoid sudden quality changes. Their work only adopts two quality levels without considering tile characteristics. Chakareski et al. [28] study an RDO problem for streaming tiled 360° videos. They take viewing probability into account and employ convex optimization to solve the bitrate allocation problem. They do not present the detail of convex optimization and practical concerns, such as discrete and limited QP values. In contrast, we give detailed proofs and propose a rounding algorithm. These studies [34], [28] focus on live bitrate allocation that only consider the constraint of a target bandwidth. Ozcinar et al. [17] share a similar goal as ours. That is, they study the optimal laddering problem for tiled 360° videos. They formulate the problem into ILP and decide the number of representations for each bandwidth class according to its fraction of clients. However, they do not develop efficient algorithms to solve the problem, which incurs extremely long running time. Moreover, they ignore the characteristics of each video tile, such as the complexity level and viewing probability, and do not vary the quality across tiles. In contrast, we propose efficient algorithms and take the diverse characteristics of tiles into considerations. These studies [34], [28], [17] are considered as the state-of-the-art algorithms in our evaluations.

Appendix B
Proofs of Lemmas

Lemma 1: The optimal laddering problem is NP-hard.

The optimal laddering problem can be reduced from the NP-hard Multiple Knapsack Problem (MKP). The MKP problem puts as many objects as possible into multiple knapsacks with various capacities to maximize the total value. If we let $V = 1$, $T = 1$, and $S = \infty$, we can map knapsacks to each class's available bandwidth and objects to tiled-segments without the storage limit. The value and the weight of each tiled-segment are the reciprocal of expected distortion $(\frac{1}{d_{v,t,n}p_{v,t,n}a_n})$ and bitrate $(r_{v,t,n})$, respectively. In this way, we reduce the MKP problem to our optimal laddering problem in polynomial time.

Lemma 2: When the power function in Eq. (1) is adopted as the distortion model, the objective function in Eq. (5a) is convex.

Proof. Note that a multivariate function is convex if it is twice differentiable and its Hessian matrix is positive semidefinite. We observe that $\alpha^d_{v,t,n} \geq 0$, $\beta^d_{v,t,n} \geq 1$, and $\gamma^d_{v,t,n} \geq 0$. We verify the second derivative of the objective function (Eq. (5a)) $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}}$ as:

$$\alpha^d_{v,t,n}\beta^d_{v,t,n}(\beta^d_{v,t,n}-1)\kappa_{v,t,n,c}^{\beta^d_{v,t,n}-2}p_{v,t,n}a_n,\ \forall n \in [1,N] \tag{13}$$

The sphere area a_n is positive constant and viewing probability $p_{v,t,n}$ are non-negative constant. This shows that the objective function is second differentiable and $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} \geq 0$ according to the range of $\alpha^d_{v,t,n}$ and $\beta^d_{v,t,n}$. We then verify the Hessian matrix of the expected distortion:

$$H^D = \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,N,c}} \\ \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,N,c}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^D_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & 0 & \cdots & 0 \\ 0 & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \end{bmatrix} \tag{14}$$

We know that if the eigenvalues of a Hermitian matrix are non-negative, then the Hessian matrix is positive semidefinite. Let λ_D be the eigenvalue of H^D. Let y be a non-zero vector. According to the property of eigenvalue:

$$H^D y - \lambda_D y = 0; \tag{15a}$$

$$(H^D - \lambda_D I)(y) = 0. \tag{15b}$$

Note that y is a non-zero vector, which indicates that $H^D - \lambda_D I = 0$:

$$H^D - \lambda_D I = \begin{bmatrix} \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} - \lambda_D & 0 & \cdots & 0 \\ 0 & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} - \lambda_D & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} - \lambda_D \end{bmatrix} \tag{16a}$$

$$\left(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} - \lambda_D\right)\left(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} - \lambda_D\right)\cdots\left(\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} - \lambda_D\right) = 0 \tag{16b}$$

$$\lambda_D = \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}},\ \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}},\ \cdots,\ \frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \tag{16c}$$

Since $\frac{\partial E^D_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} = \frac{\partial d_{v,t,n}}{\partial^2 \kappa_{v,t,n,c}} p_{v,t,n}a_n \geq 0$, the eigenvalue of H^D are non-negative values and H^D is then proved to be positive semidefinite. This shows that our objective function (Eq. (5a)) is a convex function.

Lemma 3: When the exponential function in Eq. (2) is adopted as the bitrate model, the constraint in Eq. (5b) is convex.

Proof. Similar to Lemma 2, we first verify the second derivative for each κ_n in Eq. (5b):

$$\frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}} = \alpha^r_{v,t,n}(\beta^r_{v,t,n})^2 e^{\beta^r_{v,t,n}\kappa_{v,t,n,c}},\ \forall n \in [1,N]. \tag{17}$$

Eq. (17) shows that the constraint function is second differentiable and $\frac{\partial E^R_{v,t,c}}{\partial \kappa^2_{v,t,n,c}} \geq 0$ according to the range of $\alpha^r_{v,t,n}$ and $\beta^r_{v,t,n}$, where $\alpha^r_{v,t,n} \geq 0$ and $\beta^r_{v,t,n} \leq 0$. We then verify the Hessian matrix of the constraint:

$$H^R = \begin{bmatrix} \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,1,c}\partial \kappa_{v,t,N,c}} \\ \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,2,c}\partial \kappa_{v,t,N,c}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,1,c}} & \frac{\partial E^R_{v,t,c}}{\partial \kappa_{v,t,N,c}\partial \kappa_{v,t,2,c}} & \cdots & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}} & 0 & \cdots & 0 \\ 0 & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \end{bmatrix} \tag{18}$$

Let λ_R be the eigenvalue of H^R. We can then derive λ_R as:

$$\lambda_R = \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,1,c}}, \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,2,c}}, \cdots, \frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,N,c}}. \tag{19}$$

Thus, we found that the eigenvalue of H^R is non-negative since $\frac{\partial E^R_{v,t,c}}{\partial^2 \kappa_{v,t,n,c}}$ are non-negative values. This shows that the constraint function is convex function as well.

Lemma 4: The Lagrangian dual function (Eq. (7)) constitutes a lower bound for the objective value of any feasible solution to the Lagrangian primal problem (Eq. (6)). In fact, because the strong duality holds here, the optimal solution of the Lagrangian dual problem is also the optimal solution of the original problem.

Proof. Let $\mathbf{K}^*_\mathbf{p}$ be the optimal solution set for the primal problem. For any $\mu \geq 0$:

$$g(\mu) \geq \mathbf{K}^*_\mathbf{p} \tag{20}$$

Suppose $\tilde{\mathbf{K}_\mathbf{p}} = \{\kappa_{v,\tilde{t},1,c}, \kappa_{v,\tilde{t},2,c}, \cdots, \kappa_{v,\tilde{t},N,c}\}$ is a feasible solution for Eq. (6)). Then, we have

$$\mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c})) \leq 0. \tag{21}$$

Eq. (21) shows the introduction of non-positive constraint. Therefore,

$$\begin{aligned} L(\tilde{\mathbf{K}}, \mu) &= \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} s_x + \mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c}) \\ &\leq \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} a_n. \end{aligned} \tag{22}$$

Then, we can have $g(\mu) =$

$$\inf_{\mathbf{K}} L_c(\mathbf{K}, \mu) \leq L_c(\tilde{\mathbf{K}}, \mu) \leq \sum_{n=1}^{N} d_{v,t,n}(\kappa_{v,\tilde{t},n,c}) p_{v,t,n} a_n \tag{23}$$

Finding the best lower bound leads to the following optimization problem:

$$\max g(\mu) \tag{24a}$$

$$st: \mu \geq 0. \tag{24b}$$

We denote the optimal solution set of Lagrangian dual problem as $\mathbf{K}^*_\mathbf{d}$. Then we hold the following inequality:

$$\mathbf{K}^*_\mathbf{d} \leq \mathbf{K}^*_\mathbf{p}. \tag{25}$$

To hold the equality of Eq. (25), which indicates the optimal solution set of the dual problem is also the optimal solution set of the primal problem, we verify the strong duality. Since the primal problem is convex problem, the equality condition is hold if it satisfy Slater's condition: there exists and feasible $\tilde{\mathbf{K}} = \{\kappa_{v,\tilde{t},1,c}, \kappa_{v,\tilde{t},2,c}, \cdots, \kappa_{v,\tilde{t},N,c}\}$ such that:

$$\mu(\sum_{n=1}^{N} r_{v,t,n}(\kappa_{v,\tilde{t},n,c}) - b_{v,c}) \leq 0. \tag{26}$$

holds. Let $r^{-1}_{v,t,n}$ be the inverse function of $r_{v,t,n}$ function, which takes bitrate as input and outputs the corresponding QP. Let $r'_{v,t,n} = \frac{b_{v,c}}{N+\alpha}$, $\forall i$, where $\alpha \geq 0$. Then, the set $\mathbf{K} = \{r^{-1}_{v,t,1}(r'_{v,t,n}), r^{-1}_{v,t,2}(r'_{v,t,n}), \cdots, r^{-1}_{v,t,N}(r'_{v,t,n})\}$ is feasible solution that holds the inequality. Therefore, we can solve the original distortion minimization problem by solving its Lagrangian dual problem (Eq. (7)).

Lemma 5: The PC-LBA algorithm runs in time $O(T2^N)$ with space complexity of $O(N)$.

Proof. The dominating time complexity occurs in lines 4 and 9: (i) line 4 solves Lagrangian equations using Newton's Method with $O(IN^3)$, where I is the iteration times in Newton's Method, and (ii) line 9 solves the ILP in $O(2^N)$. With T segments, the time complexity of the PC-LBA algorithm is $T \times O(IN^3 + 2^N) = O(T2^N)$. In addition, the space complexity is $O(N)$, as each of the N tiles records the selected QP value.

Lemma 6: The PC-GBA runs in time $O(TN(\log N)Q)$ with space complexity of $O(N)$.

Proof. The dominating time complexity occurs in lines 5–11: (i) the while-loop starts from line 5 iterates NQ times in the worst case and (ii) lines 7–8 update $\theta_{v,t,n,c}$ values and find out the maximum from them, which can be managed by a max heap with $O(\log N)$ time complexity. Accumulated with T segments, the time complexity of PC-GBA is $T \times O(N(\log N)Q) = O(TN(\log N)Q)$. Besides, the space complexity is N tiles recording the selected QP resulting in $O(N)$ for each segment.

Lemma 7: The GL-ITRA runs in time $O(VTNC(\log VTNC)\frac{Q}{\delta})$ with space complexity of $O(VTNC)$.

Proof. The dominating time complexity occurs in lines 9–14: (i) the while-loop starts from line 9 iterates $VTNC\frac{Q}{\delta}$ times in the worst case and (ii) lines 10–11 update $\epsilon_{v,t,n,c,q}$ values and find out the minimum from them, which can be managed by a min heap with $O(\log(VTNC))$ complexity. Collectively, the time complexity of the GL-ITRA algorithm is $O(VTNC(\log VTNC)\frac{Q}{\delta})$. Besides, the dominating space complexity consists of $VTNC$ QP values, which leads to $O(VTNC)$.