ALPHAS: Adaptive Bitrate Ladder Optimization for Multi-Live Video Streaming

Farzad Tashtarian[1]*, Mahdi Dolati[1]†, Daniele Lorenzi*, Mojtaba Mozhganfar‡, Sergey Gorinsky§, Ahmad Khonsari‡¶, Christian Timmerer*, and Hermann Hellwagner*

*Christian Doppler Laboratory ATHENA, Alpen-Adria-Universität Klagenfurt, Austria
†Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
‡School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
¶School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
§IMDEA Networks Institute, Spain

***Abstract*—Live streaming routinely relies on the Hypertext Transfer Protocol (HTTP) and content delivery networks (CDNs) to scalably disseminate videos to diverse clients. A bitrate ladder refers to a list of bitrate-resolution pairs, or representations, used for encoding a video. A promising trend in HTTP-based video streaming is to adapt not only the client's representation choice but also the bitrate ladder during the streaming session. This paper examines the problem of multi-live streaming, where an encoding service coordinates CDN-assisted bitrate ladder adaptation for multiple live streams delivered to heterogeneous clients in different zones via CDN edge servers. We design ALPHAS, a practical and scalable system for multi-live streaming that accounts for CDNs' bandwidth constraints and encoders' computational capabilities and also supports stream prioritization. ALPHAS, aware of both video content and streaming context, seamlessly integrates with the end-to-end streaming pipeline and operates in real time transparently to clients and encoding algorithms. We develop a cloud-based ALPHAS implementation and evaluate it through extensive real-world and trace-driven experiments against four prominent baseline approaches that encode each stream independently. The evaluation shows that ALPHAS outperforms the baselines, improving quality of experience, end-to-end latency, and per-stream processing by up to 23%, 21%, and 49%, respectively.**

I. Introduction

Video applications account for 38% of downstream traffic in fixed and mobile networks [1]. By the end of 2023, video traffic constitutes 73% of all mobile data traffic [2]. Videos also strongly influence user behavior. For instance, 96% of consumers find product videos helpful in purchasing decisions [3]. Compared to video-on-demand (VoD) content, live video streams gain in popularity and drive greater user engagement.

The *Hypertext Transfer Protocol (HTTP)* and *HTTP Adaptive Streaming (HAS)* are common foundations for both VoD and live delivery over the Internet [4]. In a HAS session, the server divides a video into segments and encodes each segment into multiple representations according to a *bitrate ladder*, where the segment's bitrate and resolution characterize each representation. The server advertises available representations by sending a *manifest file*, and the client employs an *adaptive bitrate (ABR) algorithm* to select from the manifest file a representation for the next requested segment. Existing ABR algorithms seek improvement in multiple metrics, including *quality of experience (QoE)* which captures the end user's overall subjective satisfaction with the streaming session [5]. With each client selecting the representations independently and dynamically, HAS scalably deals with the heterogeneity and variability in network connectivity between servers and clients [4].

The bitrate ladder, which determines the representations available to clients [6]–[10], plays a key role in HAS effectiveness. Per-title encoding [11] and other content-aware methods [12]–[16] rely on intensive offline computations to construct a static ladder tailored to each video. Context-aware techniques adapt the ladder based on session-specific factors, such as the end users' device types and network conditions [17]–[19]. Dynamic bitrate ladders respond to changes in both context (*e.g.*, network bandwidth) and content (*e.g.*, video complexity) during streaming [20,21].

Content delivery networks (CDNs) are indispensable for HAS in practice and increasingly integrate with streaming services. Traditional CDNs cache video content scalably and transparently, delivering it from globally distributed edge servers to end users with low latency [22]. A prominent industry trend in enhancing CDN support for HAS is the use of client-reported network and playback conditions. The *Common Media Client Data (CMCD)* protocol [23] emerges as a standard for this purpose, with major CDNs like Akamai, Cloudflare, and Fastly adopting it. ARTEMIS [24] utilizes CMCD to construct dynamic bitrate ladders effectively.

Compared to VoD, live streaming presents new challenges and opportunities for bitrate ladder adaptation. The need for low end-to-end latency limits the computation that a live encoder can perform. Unlike a VoD encoder, which creates and stores all representations for future playback, the live encoder does not have this requirement. For example, ARTEMIS

[1]The first two authors contributed equally to this work. The research was supported in part by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology, and Development, and the Christian Doppler Research Association through the Christian Doppler Laboratory ATHENA (https://athena.itec.aau.at/). Additionally, it was supported by project PID2022-140560OB-I00 (DRONAC), funded by MICIU/AEI/10.13039/501100011033, ERDF, and EU.

supports live streaming by advertising many representations through a *mega-manifest file* and dynamically composing the bitrate ladder from a small subset of these representations.

In this paper, we introduce and study the problem of *multi-live streaming*, where an encoding service coordinates CDN-assisted adaptation of bitrate ladders for concurrent live streams from multiple streamers to clients in various zones, as shown in Figure 1. Each streamer sends its content to the encoder in a single representation. The live encoder generates lower representations for the current video segment according to the optimized bitrate ladder and passes the segment in multiple representations to the CDNs supporting the streams. Clients assigned to a particular CDN edge server form a zone. They are heterogeneous and can request different representations. The CDN deploying the edge server for a client zone allocates a certain amount of bandwidth for content delivery to this zone [25].

Multi-live streaming entails unique complexities compared to independently encoding each live stream. As more streams share a bottleneck network link, clients' independent bandwidth estimates become less reliable, potentially causing the aggregate bitrate of requested representations to exceed the bandwidth allocated for the zone by the CDN. CDN-assisted construction of dynamic bitrate ladders for multiple streams enhances clients' QoE by adhering to bandwidth constraints set by the CDN. Additionally, multi-live streaming allows the encoding service to prioritize streams, such as based on service agreements with streamers. However, coordinating the encoding of multiple concurrent streams increases computational demands and makes it challenging to meet latency targets.

To address the challenges of multi-live streaming, we propose ALPHAS (Adaptive bitrate Ladder oPtimization for multi-live HAS). ALPHAS builds on an optimization problem that considers CDN bandwidth constraints and stream priorities. Since solving the problem optimally is computationally impractical for large-scale real-time use, we exploit its submodular structure to design a heuristic that quickly computes dynamic bitrate ladders for multiple live streams. ALPHAS advances the state of the art by combining the mega-manifest technique [24] and accurate real-time prediction [26] of *Video Multimethod Assessment Fusion (VMAF)* [27] to dynamically create a content-aware bitrate ladder for each live stream using a small subset of mega-manifest representations. The CDN edge server utilizes the generated representations to provide each client in its zone with the highest available representation that does not exceed the client's request.

We implement ALPHAS on the Amazon Elastic Compute Cloud (EC2) and evaluate it in real-world scenarios against existing solutions for bitrate ladder construction. The results show that ALPHAS outperforms these solutions in QoE, end-to-end latency, and computation time.

This paper makes the following four main contributions:

1) We formulate the problem of CDN-assisted bitrate ladder adaptation for multi-live video streaming. Our formulation considers CDN bandwidth constraints, computation available for encoding, and stream prioritization.

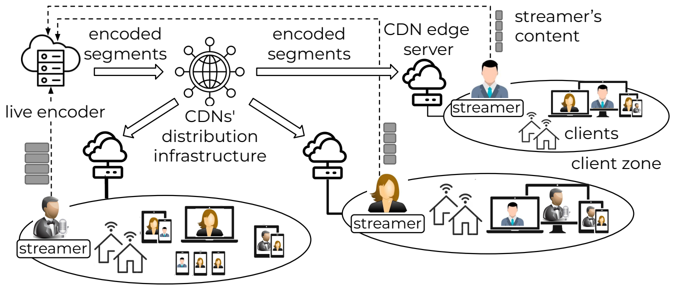

Fig. 1. Multi-live video streaming.

2) We design ALPHAS, an end-to-end system that adjusts the bitrate ladder dynamically during each streaming session. ALPHAS accounts for video content, streaming context, and scalable client feedback via CDNs.
3) We introduce a practical approximation algorithm for ALPHAS that leverages the submodular structure of the optimization problem, ensuring bounded theoretical performance.
4) We evaluate ALPHAS through a cloud-based implementation, comparing it to four prominent baseline solutions that encode streams independently. ALPHAS outperforms these baselines by improving QoE, end-to-end latency, and per-stream processing efficiency by up to 23%, 21%, and 49%, respectively.

II. Motivation

Major streaming platforms provide encoding services for a large number of simultaneous live streams, which vary widely in duration and audience size. As part of this study, we collect real data on the stream duration and average number of concurrent viewers for over 12,000 live streams on YouTube during May and June 2024. Figures 2a and 2b illustrate the distributions of these two metrics. The lower quartile (Q1) and third quartile (Q3) for stream duration span a long range from 326 to 1,104 minutes (5.4 to 18.4 hours). Similarly, the Q1 and Q3 values for average concurrent viewers are 159 and 1,970, respectively. This data confirms that *many live streams are long and exhibit significant variation in duration and popularity.*

Encoding live streams requires significant computational resources. To illustrate this, we employ *Big Buck Bunny* [28] as the source video, the 30 representations from Table 1 of [24], an EC2 `t2.2xlarge` instance, and FFmpeg [29] with the ultra-fast preset to concurrently encode 3–6 of these representations, with each encoded at least 14 times. We measure the relative central processing unit (CPU) utilization and develop a linear regression model, with an R^2 score of 0.95, to estimate the CPU load per representation. Figure 2c shows that higher representations incur greater CPU overhead while providing better video quality. VMAF improves significantly for lower representations but less noticeably for higher ones. Therefore, *composing the bitrate ladder from a small number of representations is computationally efficient and benefits from accounting for their VMAF values.*

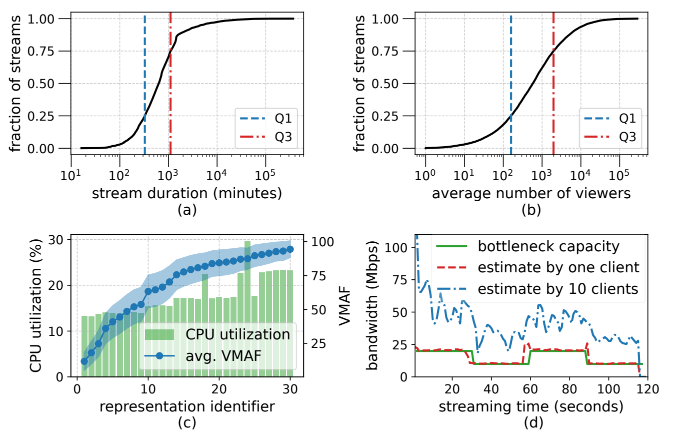

Fig. 2. Motivation for multi-live streaming: (a) long diverse stream duration, (b) diverse stream popularity, (c) CPU overhead and VMAF values of representations, and (d) estimation accuracy of the bottleneck link capacity.

An alternative to multi-live streaming is to independently encode each stream by running separate instances of a bitrate ladder adaptation system. However, this approach lacks stream prioritization and coordinated control over shared computational resources, increasing the risk of resource exhaustion as the number of streams grows. Additionally, uncoordinated adaptation can result in clients within the same zone requesting representations with an aggregate bitrate that exceeds the CDN edge server's bandwidth allocation, thereby degrading QoE. This issue becomes more severe when multiple clients share a bottleneck link and estimate available bandwidth less accurately [30].

To illustrate this, we use ARTEMIS, a state-of-the-art design for single-stream bitrate ladder adaptation, in two EC2 experiments where the bottleneck link alternates between 10 and 20 Mbps. In the first experiment, a single client closely tracks the bottleneck capacity, while in the second, 10 clients independently estimate the available bandwidth at the shared bottleneck link. Figure 2d shows that the aggregate estimates from the 10 clients vary widely, exceeding the bottleneck capacity by a factor of 2 or more. Hence, *multi-live streaming requires a solution to the challenges of independent encoding by coordinating bandwidth usage, managing encoder resources, and prioritizing streams.*

III. Modeling and Problem Formulation

A. System Model

Figure 3 illustrates the conceptual architecture of the proposed ALPHAS system, with Table I summarizing our mathematical notation. ALPHAS employs a live encoder that concurrently handles a set of video streams. The clients of the streams form zones, with a particular CDN edge server being responsible for each zone. V and Z denote the sets of the video streams and client zones, respectively. $C_{z,v}$ captures the number of the clients of stream v in zone z. D_z refers to the bandwidth allocated by the CDN for video delivery to zone z from the respective CDN edge server. The streamer of stream v supplies the content to the live encoder in a single representation that has bitrate $\widehat{q}_v$. Optionally, the live encoder

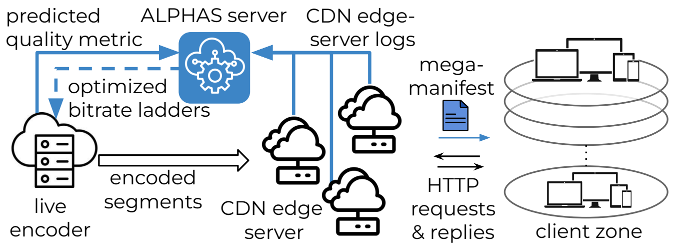

Fig. 3. ALPHAS conceptual architecture.

TABLE I
Mathematical notation.

Symbol	*Description*
V	Set of concurrent video streams
Z	Set of client zones
$C_{z,v}$	Number of clients of stream v in zone z
D_z	CDN bandwidth allocated for video delivery to zone z
$\widehat{q}_v$	Bitrate of the streamer-supplied representation of stream v
Ω	Set of mega-manifest representations
$r_q^{z,v}$	Number of requests for representation q of stream v in zone z
Ψ	Total computational capacity of the live encoder
ψ_q	Computation required to generate representation q
$\nu_{v,p}$	VMAF estimate for representation p of stream v
$p_{z,v}$	Priority of stream v in zone z
ℓ	Maximum number of representations in a bitrate ladder
$x_{v,q}$	Presence of representation q in the bitrate ladder of stream v
$y_{p,q}^v$	Use of representation p to serve the clients of stream v that request representation q
ρ_p	Bitrate of representation p
$\overline{y}_{z,v}$	Average video quality for the clients of stream v in zone z
$\mathcal{I}$	Set of items, each of which is a stream-representation pair
$\mathcal{I}_\tau$	Set of selected items after iteration τ
$\overline{\mathcal{I}}_\tau$	Set of discarded items after iteration τ
λ	Computational budget
w_i^d	Weight of item i for dimension d
ϕ_τ^j	Cost-to-benefit ratio for item j at iteration τ
$z_{d,\tau}$	Cost for dimension d after iteration τ

upscales the received representation to increase $\widehat{q}_v$. ALPHAS sends the clients a mega-manifest file that describes Ω, a large set of candidate representations for encoding of video segments. Each CDN edge server collects and communicates to the ALPHAS server the value of $r_q^{z,v}$, the number of requests for representation q of stream v in zone z.

The ALPHAS server relies on time slots in its operation. During each slot, it gathers the $r_q^{z,v}$ and D_z data from the CDN edge servers. The ALPHAS server also collects information from the live encoder. This information includes Ψ and ψ_q, which refer to the live encoder's total computational capacity and the amount of computation required to generate mega-manifest representation q, respectively. Additionally, the live encoder supplies its real-time VMAF estimate $\nu_{v,p}$ for the video quality of representation p of stream v.

B. Problem Formulation

Based on the inputs from the CDNs and live encoder, the ALPHAS server computes optimized bitrate ladders for all streams. The computation of the bitrate ladders supports stream prioritization, with $p_{z,v}$ denoting the priority of stream v in zone z. To keep the computation and communication overhead low, ALPHAS limits the maximum number of representations in a bitrate ladder to ℓ, which is much smaller than the number of the mega-manifest representations. The live encoder receives from the ALPHAS server, at the end of every time slot, the optimized bitrate ladder for each stream and, during the next time slot, generates segment representations for the stream according to this updated bitrate ladder. When the bitrate ladder does not contain a representation requested by a client, the CDN edge server delivers the segment in the highest lower representation present in the bitrate ladder.

To describe how ALPHAS constructs bitrate ladders and serves client requests, we use vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of binary decision variables $x_{v,q}$ and $y^v_{p,q}$. If $x_{v,q}$ is set to 1, the bitrate ladder of stream v contains representation q; otherwise, $x_{v,q}$ equals 0. $y^v_{p,q} = 1$ means that a client of stream v requests a segment in representation q, and the CDN edge server delivers the segment in representation p. To ensure that ALPHAS serves each client request in the requested or lower representation, the problem formulation incorporates:

$$\sum\nolimits_{p\in\Omega, p\leqslant q} y^v_{p,q} = 1 \quad \forall v \in V, q \in \Omega. \tag{1}$$

The next constraint guarantees that if ALPHAS decides to deliver segments to clients of stream v in representation p, the encoder generates this representation for the stream:

$$\sum\nolimits_{q\in\Omega, q\geqslant p} y^v_{p,q} \leqslant x_{v,p} \times |\Omega| \quad \forall v \in V, p \in \Omega. \tag{2}$$

For each stream v, the encoder does not generate representations with bitrates exceeding the bitrate of the streamer-supplied representation (or its upscaled version):

$$\sum\nolimits_{q>\hat{q}_v} x_{v,q} = 0 \quad \forall v \in V. \tag{3}$$

When the bitrate ladder misses a requested representation, the following constraint assures the use of the highest lower representation present in the bitrate ladder:

$$\sum\nolimits_{p<j\leqslant q} x_{v,j} \leqslant (1 - y^v_{p,q}) \times |\Omega| \;\; \forall v \in V, p, q \in \Omega, p < q. \tag{4}$$

The subsequent constraint ensures that the computation required to generate the representations for all streams in accordance with their bitrate ladders is within the live encoder's total computational capacity:

$$\sum\nolimits_{v\in V} \sum\nolimits_{q\in\Omega} \psi_q \cdot x_{v,q} \leqslant \Psi. \tag{5}$$

We impose the upper limit on the number of representations in a bitrate ladder as follows:

$$\sum\nolimits_{q\in\Omega} x_{v,q} \leqslant \ell \quad \forall v \in V. \tag{6}$$

With ρ_p referring to the bitrate of representation p, the following constraint guarantees that the aggregate bitrate of the representations requested by the clients of all streams in each zone does not exceed the CDN edge server's bandwidth allocated for video delivery to this zone:

$$\sum\nolimits_{v\in V} \sum\nolimits_{q\in\Omega} r^{z,v}_q \Big(\sum\nolimits_{p\in\Omega, p\leqslant q} \rho_p \cdot y^v_{p,q}\Big) \leqslant D_z \quad \forall z \in Z. \tag{7}$$

To construct content-aware bitrate ladders for the streams, ALPHAS combines the VMAF estimates for all mega-manifest representations, the number of requests for each representation, and the number of clients in every zone to express average video quality $\overline{y}_{z,v}$ for the clients of stream v in zone z as follows:

$$\overline{y}_{z,v} \leqslant \frac{1}{C_{z,v}} \sum\nolimits_{q\in\Omega} r^{z,v}_q \Big(\sum\nolimits_{p\in\Omega, p\leqslant q} \nu_{v,p} \cdot y^v_{p,q}\Big). \tag{8}$$

We define the objective function for ALPHAS as the following expression that considers both average video quality and priority of all streams v in all zones z:

$$f(\boldsymbol{y}) = \sum\nolimits_{z\in Z} \sum\nolimits_{v\in V} p_{z,v} \cdot \overline{y}_{z,v}. \tag{9}$$

We note here that Constraint 8 is an intermediate step that computes the auxiliary variables $\overline{y}_{z,v}$ required in $f(\boldsymbol{y})$. Therefore, the following integer linear program (ILP) produces bitrate ladders for all streams in every time slot:

$$\max_{\boldsymbol{x},\boldsymbol{y}}.\; f(\boldsymbol{y}) \quad \text{s.t. Constraints 1 through 7 hold.} \tag{10}$$

IV. ALPHAS Design and Analysis

We refer to Problem 10 in Section III-B as Optimal ALPHAS (OPT-ALPHAS). Since solving it is computationally prohibitive for real-time operation, we address a variation of this problem and propose Submodular ALPHAS (SM-ALPHAS), an algorithm that exploits the submodular structure of the objective function.

A. Submodularity Characterization

To enhance clarity and simplify notation, we reformulate OPT-ALPHAS as an optimization problem over a set and characterize its core component with a submodular function. Specifically, we treat each pair of stream v and representation q as an item and define set $\mathcal{I}$ of items:

$$\mathcal{I} = \{(v, q) \mid v \in V, q \in \Omega\}. \tag{11}$$

We also define set function $\tilde{f}$ that maps $\mathcal{I}$ to $\mathbb{R}$ as follows:

$$\tilde{f}(\mathcal{J}) = \max_{\boldsymbol{y}}.\; f(\boldsymbol{y}) \quad \text{s.t. Constraints 1 through 4 hold} \tag{12}$$

$$x_{v,q} = 1 \leftrightarrow (v, q) \in \mathcal{J} \subseteq \mathcal{I} \tag{13}$$

This reformulation omits Constraints 5 through 7. Inclusion of item (v, q) in set $\mathcal{J}$ is equivalent to the availability of representation q for stream v in OPT-ALPHAS.

Definition 1 (Submodular set function). *Let $\Delta(i \mid \mathcal{J}) = \tilde{f}(\mathcal{J} \cup \{i\}) - \tilde{f}(\mathcal{J})$ denote the increase in set function $\tilde{f}$ after adding item i to set $\mathcal{J}$. Function $\tilde{f}$ is submodular if, for every pair of sets $\mathcal{J}_1$ and $\mathcal{J}_2$ such that $\mathcal{J}_1 \subseteq \mathcal{J}_2 \subseteq \mathcal{I}$, and for any item $i \in \mathcal{I} \setminus \mathcal{J}_2$, increase $\Delta(i \mid \mathcal{J}_1)$ is at least $\Delta(i \mid \mathcal{J}_2)$.*

Theorem 1. *$\tilde{f}(\mathcal{J})$ is a submodular set function.*

Proof. Suppose $i \in \mathcal{I} \setminus \mathcal{J}_2$ and $\mathcal{J}_1 \subseteq \mathcal{J}_2 \subseteq \mathcal{I}$. Since $\mathcal{J}_1 \subseteq \mathcal{J}_2$ and $\tilde{f}(\mathcal{J})$ omits Constraints 5-7, the VMAF increase for a client when adding item i to $\mathcal{J}_1$ is at least as large as when adding it to $\mathcal{J}_2$. Furthermore, because $\mathcal{J}_2$ contains at least as many representations as $\mathcal{J}_1$, at least as many clients benefit from adding i to $\mathcal{J}_1$ as for $\mathcal{J}_2$. Therefore, the marginal gain in $\widetilde{f}$ from adding i to $\mathcal{J}_1$ is at least as large as the gain from adding it to $\mathcal{J}_2$, implying that $\Delta(i \mid \mathcal{J}_1)$ is at least $\Delta(i \mid \mathcal{J}_2)$. By Definition 1, $\widetilde{f}(\mathcal{J})$ is submodular. $\square$

The submodularity of $\widetilde{f}(\mathcal{J})$ is useful because it ensures a diminishing returns property, simplifying our analysis of the objective function and enabling us to develop an efficient approximation algorithm with provable performance bounds. While prior work effectively leverages submodularity to address cache placement [31] and other problems, we adapt the technique for the new problem of multi-live streaming.

To account for Constraints 5 and 6, we associate each item i with $1 + |V|$ weights w_i^d, where dimension $d \in \mathcal{D} = \{0, \ldots, |V|\}$ corresponds to Constraint 5 for $d = 0$ and to one of Constraints 6 for $d \in \mathcal{D} = \{1, \ldots, |V|\}$:

$$w_i^d = \begin{cases} \psi_{q_i}/\Psi, & d = 0, \\ 1/\ell, & d = v_i, \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in \mathcal{I}, d \in \mathcal{D}. \tag{14}$$

Here, v_i and q_i denote the stream and representation that constitute item i. This leads to the following problem formulation:

$$\max_{\mathcal{J} \subseteq \mathcal{I}}. \ \widetilde{f}(\mathcal{J}) \quad \text{s.t.} \ \textstyle\sum_{i \in \mathcal{J}} w_i^d \leqslant 1 \qquad \forall d \in \mathcal{D}. \tag{15}$$

The weight definition above ensures that any feasible solution to Problem 15 satisfies Constraints 5 and 6 of OPT-ALPHAS.

B. Algorithm Outline

This section describes how SM-ALPHAS solves Problem 15 by iteratively selecting items based on their weights and contributions to $\widetilde{f}(\mathcal{J})$ while satisfying Constraint 7. The algorithm builds bitrate ladders for streams in set V by distributing, at each iteration τ, an item from set $\mathcal{I}$ into sets $\mathcal{I}_\tau$ (selected items) or $\overline{\mathcal{I}}_\tau$(discarded items), both initialized as empty.

To account for Constraints 5 and 6, SM-ALPHAS draws inspiration from [32] and, for each dimension d, evaluates cost $z_{d,\tau-1}$ of generating a representation for the corresponding stream. The algorithm keeps the total cost across all $|\mathcal{D}|$ dimensions within budget λ, which depends on the number of dimensions, encoders' computational capacities, computational requirements of all representations, and the maximum number of representations in a bitrate ladder.

For each item j not yet placed in either set, SM-ALPHAS calculates cost-to-benefit ratio ϕ_τ^j where the benefit refers to the increase in the objective function from adding the item to $\mathcal{I}_{\tau-1}$. Iteration τ then selects the item with the lowest cost-to-benefit ratio, adding it to $\mathcal{I}_\tau$ if this action does not violate Constraint 7. The algorithm updates the cost for the corresponding dimension by scaling it exponentially to reflect reduced resource availability, with scaling governed by budget λ and the item's weights. If adding the item violates Constraint 7, iteration τ places it in $\overline{\mathcal{I}}_\tau$. The algorithm terminates when it exhausts budget λ or processes all items in set $\mathcal{I}$.

Algorithm 1: SM-ALPHAS, a submodular approximation algorithm for OPT-ALPHAS.

```
    Input: $\mathcal{I}$, $\mathcal{D}$, $V$, $\Omega$, $\Psi$, $\{\psi_q\}$, $\ell$, $\mathcal{Z}$, $\{D_z\}$, $\{\rho_p\}$, $\{r_q^{z,v}\}$
    Output: $\{x_{v,q}\}$
 1: $W \leftarrow \min\{\ell, \min_{q \in \Omega}\{\frac{\Psi}{\psi_q}\}\}$; $\lambda \leftarrow |D| \cdot \exp(W)$
 2: $\{x_{v,q}\} \leftarrow \mathbf{0}$; $\{y_{p,q}^v\} \leftarrow \mathbf{0}$; $\tau \leftarrow 0$; $\mathcal{I}_\tau \leftarrow \varnothing$; $\overline{\mathcal{I}}_\tau \leftarrow \varnothing$
 3: for $d \in \mathcal{D}$ do
 4:     $z_{d,\tau} \leftarrow 1$
 5: $Z_\tau \leftarrow \sum_{d \in \mathcal{D}} z_{d,\tau}$
 6: while $Z_\tau \leqslant \lambda$ and $\mathcal{I}_\tau \cup \overline{\mathcal{I}}_\tau \neq \mathcal{I}$ do
 7:     $\tau \leftarrow \tau + 1$
 8:     for $j \in \mathcal{I} \setminus (\mathcal{I}_{\tau-1} \cup \overline{\mathcal{I}}_{\tau-1})$ do
 9:         $\phi_\tau^j \leftarrow \frac{\sum_{d \in \mathcal{D}} w_j^d \cdot z_{d,\tau-1}}{\Delta(j | \mathcal{I}_{\tau-1})}$
10:     $j_\tau \leftarrow \arg\min_{j \in \mathcal{I} \setminus \mathcal{I}_{\tau-1} \cup \overline{\mathcal{I}}_{\tau-1}} \phi_\tau^j$
11:     $v_\tau \leftarrow v_{j_\tau}$; $q_\tau \leftarrow q_{j_\tau}$; $\{\widehat{y}_{p,q}^v\} \leftarrow \{y_{p,q}^v\}$; $\tilde{q} \leftarrow q_\tau$
12:     while $x_{v_\tau,\tilde{q}} = 0$ do
13:         for $p \in \Omega$ do
14:             $\widehat{y}_{p,\tilde{q}}^v \leftarrow 0$
15:         $\widehat{y}_{q_\tau,\tilde{q}}^v \leftarrow 1$; $\tilde{q} \leftarrow$ next-largest($\tilde{q}$, $\Omega$)
16:     infeasible $\leftarrow$ FALSE
17:     for $z \in Z$ do
18:         if $\sum_{v \in V} \sum_{q \in \Omega} r_q^{z,v} \left(\sum_{p \in \Omega, p \leqslant q} \rho_p \cdot \widehat{y}_{p,q}^v\right) > D_z$ then
19:             infeasible $\leftarrow$ TRUE
20:             break
21:     if infeasible == FALSE then
22:         $\{y_{p,q}^v\} \leftarrow \{\widehat{y}_{p,q}^v\}$
23:         $x_{v_\tau,q_\tau} \leftarrow 1$; $\overline{\mathcal{I}}_\tau \leftarrow \overline{\mathcal{I}}_{\tau-1}$; $\mathcal{I}_\tau \leftarrow \mathcal{I}_{\tau-1} \cup \{j_\tau\}$
24:         for $d \in \mathcal{D}$ do
25:             $z_{d,\tau} \leftarrow z_{d,\tau-1} \cdot \lambda^{w_{j_\tau}^d}$
26:     else
27:         $\overline{\mathcal{I}}_\tau \leftarrow \overline{\mathcal{I}}_{\tau-1} \cup \{j_\tau\}$; $\mathcal{I}_\tau \leftarrow \mathcal{I}_{\tau-1}$
28:         for $d \in \mathcal{D}$ do
29:             $z_{d,\tau} \leftarrow z_{d,\tau-1}$
30:     $Z_\tau \leftarrow \sum_{d \in \mathcal{D}} z_{d,\tau}$
31: for $d \in \mathcal{D}$ do
32:     if $\sum_{i \in \mathcal{I}_\tau} w_i^d > 1$ then
33:         $x_{v_\tau,q_\tau} \leftarrow 0$
34:         break
35: return $\{x_{v,q}\}$
```

C. Algorithm Details

Algorithm 1 presents the pseudocode for SM-ALPHAS. Lines 1–5 handle initialization, including of cost $z_{d,0}$ for each dimension d to 1. Parameter W, which affects budget λ, reflects the computational bottleneck as the minimum capacity-to-representation weight ratio across all representations.

The main loop in Lines 6–30 executes iteration τ of the algorithm and runs until either total cost Z_τ exceeds computational budget λ or the algorithm processes all items in set $\mathcal{I}$. Lines 8–9 compute cost-to-benefit ratio ϕ_τ^j for each item j that previous iterations have not examined. The cost incorporates dimension-specific weights w_j^d and sums weighted costs $z_{d,\tau-1}$ across all $|\mathcal{D}|$ dimensions. Line 10 detects item j_τ with the lowest cost-to-benefit ratio among such items, with v_τ and q_τ denoting its associated stream and representation, respectively.

Lines 11–20 check whether adding representation q_τ to the bitrate ladder for stream v_τ satisfies Constraint 7. To this end, Lines 11–15 construct potential service arrangement $\widehat{y}^v_{q_\tau,\tilde{q}} = 1$, where representation q_τ serves all representations $\tilde{q}$ that are at least as high as q_τ and currently absent from the bitrate ladder. Then, Lines 16–20 examine whether the CDN bandwidth allocation provides sufficient capacity for this service arrangement in each zone z.

If the check succeeds, Lines 21–25 accept the service arrangement by setting $\{y^v_{p,q}\}$ to $\{\widehat{y}^v_{p,q}\}$, adding representation q_τ to the bitrate ladder for stream v_τ through placement of item j_τ into set $\mathcal{I}_\tau$ of selected items, and scaling costs $z_{d,\tau}$ by $\lambda^{w^d_{j_\tau}}$ for each dimension d. If the check fails, Lines 26–29 place item j_τ into set $\overline{\mathcal{I}}_\tau$ of discarded items without modifying costs $z_{d,\tau}$.

Lines 31–34 check whether the most recently selected item violates the constraint in Problem 15 and remove it from the respective bitrate ladder if a violation occurs. Line 35 concludes Algorithm 1 by outputting the bitrate ladders for all streams.

D. Algorithm Analysis

We begin our analysis by proving that the bitrate ladders constructed by Algorithm 1 meet the constraints on encoding and CDN bandwidth.

Theorem 2. *SM-ALPHAS computes a feasible solution to the multi-live streaming problem.*

Proof. Lines 16–20 and 26–29 of SM-ALPHAS ensure that the algorithm avoids selecting a representation for a bitrate ladder if this selection would violate the constraint in Problem 7. When selecting an item to set $\mathcal{I}_\tau$ violates the constraint in Problem 15 (i.e., the sum of weights w^d_i exceeds 1), then total cost Z_τ, determined by $\lambda^{\sum_{i\in\mathcal{I}_\tau} w^d_i}$, surpasses budget λ. This condition causes the algorithm to exit the main loop, which runs from Line 6 to Line 30. Lines 31–34 then remove the violating item from the set of selected items and restore compliance with the constraint in Problem 15. Therefore, SM-ALPHAS constructs bitrate ladders that satisfy both Constraints 7 and the constraint in Problem 15. □

The following theorem characterizes the computational complexity of the algorithm:

Theorem 3. *SM-ALPHAS executes in polynomial time.*

Proof. The loop in Lines 6-30 terminates after $O(|V| \cdot |\Omega|)$ iterations. In each iteration, the computation of cost-to-benefit ratios takes $O(|V|)$ time, and checking Constraint 7 requires $O(|Z| \cdot |V| \cdot |\Omega|)$ time. Thus, the algorithm runs in polynomial time $O(|V|^2 \cdot |\Omega|^2 \cdot |Z|)$. □

We now derive an upper bound on the marginal value of each item remaining unprocessed in set $\mathcal{I}$ after iteration τ of Algorithm 1, with respect to the items already selected into set $\mathcal{I}_\tau$. First, we consider the case where set $\overline{\mathcal{I}}_\tau$ is empty. Let j_τ denote the item selected into $\mathcal{I}_\tau$ during iteration τ. By construction, item j_τ has the lowest cost-to-benefit ratio among all unprocessed items in set $\mathcal{I}$. Furthermore, its selection respects the CDN bandwidth constraints. Consequently, the following inequality holds for any such item j:

$$\Delta(j \mid \mathcal{I}_\tau) \leqslant \sum\nolimits_{d\in\mathcal{D}} \frac{w^d_j \cdot z_{d,\tau}}{\phi^{j_{\tau+1}}_{\tau+1}}. \tag{16}$$

Suppose that $\mathcal{J}^\star$ represents an optimal solution to the problem, and let $\mathcal{K}_\tau$ denote the set of items selected by this optimal solution but not by SM-ALPHAS after iteration τ. Then, the following inequality holds:

$$\widetilde{f}(\mathcal{J}^\star) \leqslant \widetilde{f}(\mathcal{I}_\tau) + \sum\nolimits_{j\in\mathcal{K}_\tau} \Delta(j \mid \mathcal{I}_\tau). \tag{17}$$

Applying Inequality 16 to Inequality 17 yields:

$$\widetilde{f}(\mathcal{J}^\star) \leqslant \widetilde{f}(\mathcal{I}_\tau) + \sum\nolimits_{j\in\mathcal{K}_\tau} \sum\nolimits_{d\in\mathcal{D}} \frac{w^d_j \cdot z_{d,\tau}}{\phi^{j_{\tau+1}}_{\tau+1}} \tag{18}$$

$$= \widetilde{f}(\mathcal{I}_\tau) + \sum\nolimits_{d\in\mathcal{D}} \frac{z_{d,\tau}}{\phi^{j_{\tau+1}}_{\tau+1}} \sum\nolimits_{j\in\mathcal{K}_\tau} w^d_j. \tag{19}$$

Since $\mathcal{K}_\tau$ is feasible, meaning that $\sum_{j\in\mathcal{K}_\tau} w^d_j \leqslant 1$, we have:

$$\widetilde{f}(\mathcal{J}^\star) \leqslant \widetilde{f}(\mathcal{I}_\tau) + \frac{Z_\tau}{\phi^{j_{\tau+1}}_{\tau+1}}. \tag{20}$$

We express total cost Z_τ as:

$$Z_\tau = \sum\nolimits_{d\in\mathcal{D}} z_{d,\tau-1} \cdot \sqrt[W]{\lambda}^{W\cdot w^d_{j_\tau}} \tag{21}$$

and bound it from above by noting that $W \cdot w^d_{j_\tau}$ is at most 1, since W is, by definition, at most the smallest of $1/w^d_i$:

$$Z_\tau \leqslant \sum\nolimits_{d\in\mathcal{D}} z_{d,\tau-1} \cdot (1 + \sqrt[W]{\lambda} \cdot W \cdot w^d_{j_\tau}). \tag{22}$$

We apply the definitions of Z_τ and ϕ^j_τ to obtain:

$$Z_\tau \leqslant Z_{\tau-1} + \sqrt[W]{\lambda} \cdot W \cdot \Delta(j_\tau \mid \mathcal{I}_{\tau-1}) \cdot \phi^{j_\tau}_\tau. \tag{23}$$

Using Inequality 20, we substitute $\phi^{j_\tau}_\tau$ with an upper bound expressed in terms of the optimal value and $Z_{\tau-1}$:

$$Z_\tau \leqslant Z_{\tau-1} \cdot \left(1 + \frac{\sqrt[W]{\lambda} \cdot W \cdot \Delta(j_\tau \mid \mathcal{I}_{\tau-1})}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_{\tau-1})}\right). \tag{24}$$

Then, we apply $e^x \geqslant (1 + x)$, a well-known inequality, to derive:

$$Z_\tau \leqslant Z_{\tau-1} \cdot \exp\left\{\frac{\sqrt[W]{\lambda} \cdot W \cdot \Delta(j_\tau \mid \mathcal{I}_{\tau-1})}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_{\tau-1})}\right\}. \tag{25}$$

Because Inequality 25 establishes a recursive relation for the upper bound on Z_τ, and Z_0 equals $|D|$, we expand this relation to derive the following upper bound:

$$Z_\tau \leqslant |D| \exp\left\{\sum_{t=1}^{\tau} \frac{\sqrt[W]{\lambda} \cdot W \cdot \Delta(j_\tau \mid \mathcal{I}_{\tau-1})}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_{\tau-1})}\right\}. \tag{26}$$

Next, we determine a lower bound for Z_τ after the loop in Lines 6–30 terminates. When CDN bandwidth is abundant, set $\overline{\mathcal{I}}_\tau$ remains empty, and λ serves as the lower bound. However, when CDN bandwidth becomes a bottleneck, SM-ALPHAS adds items to set $\overline{\mathcal{I}}_\tau$, and the loop terminates after processing all items in set $\mathcal{I}$. Let $\widehat{\nu}$ and $\widehat{r}$ denote, respectively, the maximum bandwidth consumption of any bitrate and the maximum number of clients requesting the same representation of the same stream. Then, the bandwidth consumption resulting from the selection of each representation in each iteration of the loop is at most $\widehat{\nu} \cdot \widehat{r}$. With $\breve{D}$ representing the smallest available CDN bandwidth across all zones, the main loop executes at least $|D| \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor$ iterations, implying:

$$Z_\tau \geqslant |D| \cdot \lambda^{\breve{w} \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor} \geqslant |D| \cdot e^{\breve{w} \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor} \tag{27}$$

where $\breve{w}$ refers to the smallest weight of any item in set $\mathcal{I}$.

By applying Inequalities 26 and 27, we establish the relationship between the optimal solution and the result produced by SM-ALPHAS:

$$\breve{w} \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor \leqslant \sum_{t=1}^{\tau} \frac{\sqrt[W]{\lambda} \cdot W \cdot \Delta(j_\tau \mid \mathcal{I}_{\tau-1})}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_{\tau-1})}. \tag{28}$$

Next, we consider the following theorem, which holds for any submodular function:

Theorem 4 ([32])**.** *Let $\mathcal{J}_0 \subseteq \cdots \subseteq \mathcal{J}_t \subseteq I$ and $\mathcal{J}^\star \subseteq \mathcal{I}$ where $\widetilde{f}(\mathcal{J}^\star) > \widetilde{f}(\mathcal{J}_t)$. Then, the following inequality holds:*

$$\sum_{\tau=1}^{t} \frac{\widetilde{f}(\mathcal{J}_\tau) - \widetilde{f}(\mathcal{J}_{\tau-1})}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{J}_{\tau-1})} \leqslant \ln \left(\frac{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{J}_0)}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{J}_t)} \right). \tag{29}$$

We apply Theorem 4 to simplify Inequality 28 further:

$$\breve{w} \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor \leqslant \sqrt[W]{\lambda} \cdot W \cdot \ln \left(\frac{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_0)}{\widetilde{f}(\mathcal{J}^\star) - \widetilde{f}(\mathcal{I}_t)} \right). \tag{30}$$

Finally, noting that $\widetilde{f}(\mathcal{I}_0) = 0$, we characterize an upper bound on $\frac{\widetilde{f}(\mathcal{I}_t)}{\widetilde{f}(\mathcal{J}^\star)}$ as follows:

$$\frac{\widetilde{f}(\mathcal{I}_t)}{\widetilde{f}(\mathcal{J}^\star)} \leqslant 1 - 1/\exp\left\{ \frac{\breve{w}}{\sqrt[W]{\lambda} \cdot W} \cdot \lfloor \frac{\breve{D}}{\widehat{\nu} \cdot \widehat{r}} \rfloor \right\}. \tag{31}$$

V. Evaluation

A. *Experimental Setup*

Testbed. We implement ALPHAS in Python and deploy it on EC2 as depicted in Figure 4a. Our cloud-based implementation involves eight EC2 *c5d.9xlarge* instances to run the Bitmovin live encoder, ALPHAS server, three emulated CDN edge servers E1, E2, and E3, and three corresponding machines that emulate zones Z1, Z2, and Z3 containing multiple dash.js clients [33]. In the default configuration, the ALPHAS server executes the SM-ALPHAS algorithm. We also experiment in settings where the ALPHAS server relies on Gurobi [34] to find the optimal OPT-ALPHAS solution. The ALPHAS server communicates with the live encoder and CDN edge servers via Transmission Control Protocol (TCP) sockets. The content of streams is the 240-second *Big Buck Bunny* [28] source video encoded live into 1-second segments with FFmpeg [29]. To reproduce live encoding settings, we utilize the Advanced Video Coding (AVC) compression from the libx264 library with ultra-fast preset.

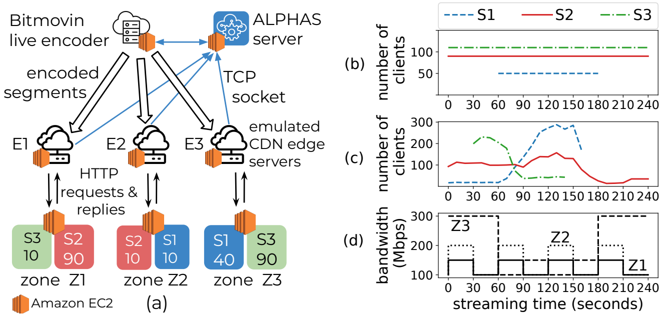

Fig. 4. Experimental setup: (a) cloud-based implementation of ALPHAS, (b) static subscription, (c) dynamic subscription, and (d) bandwidth capacities.

Scenarios. We evaluate ALPHAS against four baselines that encode each stream independently. The baselines include three state-of-the-art static bitrate ladders developed by Twitch [6], YouTube [7], and MUX [9]. The fourth baseline is ARTEMIS [24], which adapts the bitrate ladder during the stream. Each experiment involves three streams S1, S2, and S3 that start at different times and have different duration. We examine two experimental scenarios. In Scenario I, the number of clients of each stream does not change, as shown in Figure 4b. Scenario II leverages our YouTube dataset to vary the number of clients for each stream as depicted in Figure 4c. The bandwidth allocated for video delivery to each of zones Z1, Z2, and Z3 from the respective CDN edge server changes in accordance with the patterns plotted in Figure 4d.

Metrics. We assess computational effectiveness in terms of CPU utilization per stream and execution time. Our evaluation of QoE leverages Comyco's VMAF-based QoE model [35]. Additionally, we examine end-to-end latency (the time between stream capture and playback), stall duration (the total time the playback stalls), VMAF, and VMAF instability as four QoE influence factors.

B. *Experimental Results*

Figure 5 presents the improvement achieved by OPT-ALPHAS over the ARTEMIS, Twitch, YouTube, and MUX baselines in Scenario I with respect to CPU utilization, QoE, stall duration, end-to-end latency, VMAF instability, and average VMAF when all streams in all zones have the same priority. Figure 5a shows that OPT-ALPHAS consistently increases average QoE and decreases CPU utilization per stream. For example, the CPU consumption by stream S1 drops by nearly 60% compared to the Twitch baseline while QoE for this stream improves by 50% vs. the MUX baseline. In comparison to ARTEMIS, OPT-ALPHAS reduces the CPU utilization per stream by around 3% while increasing QoE by up to 10%. Figure 5b shows consistent improvement in end-to-end latency and stall duration as well, with reductions of 54% and 80% vs. MUX in average latency and stall duration, respectively. Figure 5c reveals that OPT-ALPHAS significantly outperforms Twitch and MUX on VMAF instability, with average improvement of 150% vs. MUX. Figure 5c also shows that OPT-ALPHAS might either increase or decrease VMAF compared to the baselines. However, all such increases

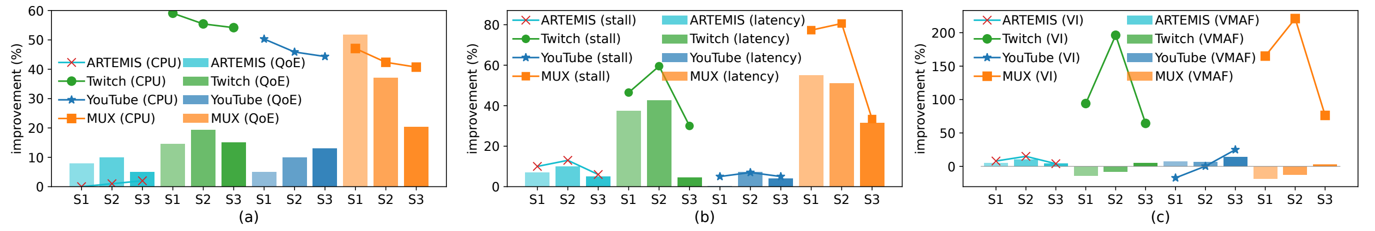
Fig. 5. Improvement by OPT-ALPHAS over the ARTEMIS, Twitch, YouTube, and MUX baselines in Scenario I with respect to: (a) CPU utilization and QoE, (b) stall duration and end-to-end latency, and (c) VMAF instability, denoted as VI, and average VMAF.

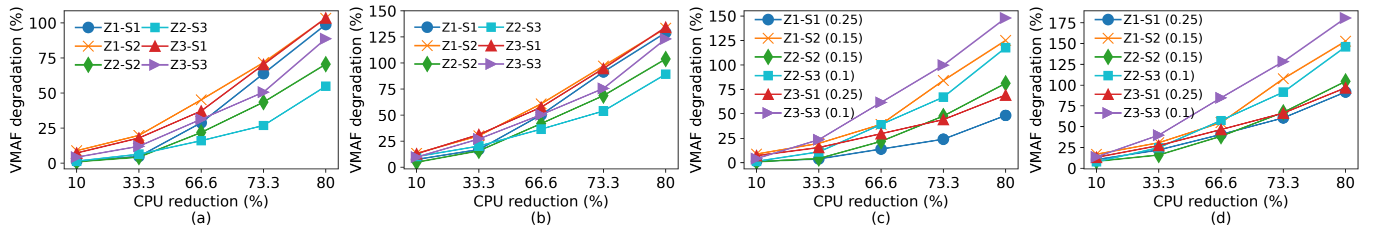
Fig. 6. Impact by CPU reductions on average VMAF degradation in Scenario I for: (a) OPT-ALPHAS with identical 0.17 priorities, (b) SM-ALPHAS with identical 0.17 priorities, (c) OPT-ALPHAS with different 0.1, 0.15, and 0.25 priorities, and (d) SM-ALPHAS with different 0.1, 0.15, and 0.25 priorities.

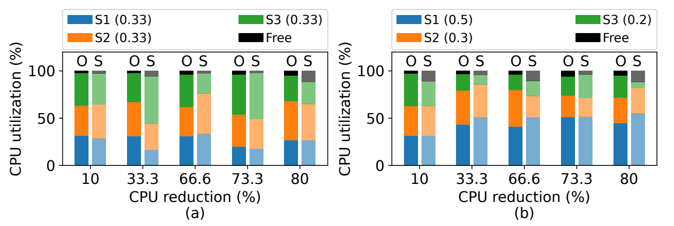
Fig. 7. Impact by CPU reductions on the CPU utilization in Scenario I for OPT-ALPHAS and SM-ALPHAS, denoted as O and S respectively, with: (a) identical 0.33 priorities and (b) different 0.2, 0.3, and 0.5 priorities.

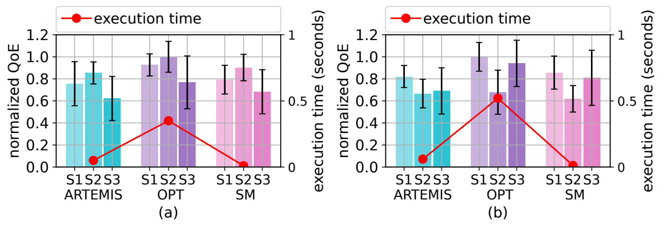
Fig. 8. Average normalized QoE and execution time in Scenario II for ARTEMIS, OPT-ALPHAS, and SM-ALPHAS with: (a) identical priorities and (b) different priorities.

and decreases are fairly small, close to one just-noticeable difference (JND) [36].

Also in Scenario I, Figure 6 examines how CPU reduction, *i.e.*, reduction in the CPU share given to ALPHAS, affects average VMAF degradation expressed as $\bar{y}_{z,v}$ in Constraint 8. Figures 6a and 6b report the effect for OPT-ALPHAS and SM-ALPHAS, respectively, when all streams in all zones have the same priority. The two ALPHAS versions exhibit similar profiles of VMAF degradation across all zones and streams, with the degradation being up to 30% smaller with OPT-ALPHAS. We also explore the impact for OPT-ALPHAS and SM-ALPHAS, respectively, when streams have different priorities. Figures 6c and 6d confirm that stream priorities enable ALPHAS to utilize the computational resources more flexibly. For example, when OPT-ALPHAS assigns priorities 0.25, 0.15, and 0.1 to streams S1, S2, and S3, respectively, the highest priority allows stream S1 to significantly mitigate VMAF degradation, from 130% to 59% across all zones for CPU reduction of 80%. Similarly, stream priorities empower SM-ALPHAS to alleviate the impact of CPU reduction on VMAF degradation for high-priority streams.

Figure 7 shows the effect of stream priorities under CPU reduction on CPU allocation patterns for OPT-ALPHAS and SM-ALPHAS in Scenario I. The optimal OPT-ALPHAS solution and SM-ALPHAS partition the CPU between the streams similarly across all examined levels of CPU reduction. In Figure 7a where all streams have the same priority, streams S1, S2, and S3 receive substantially different CPU shares, *e.g.*, about 17%, 33%, and 45%, respectively, for CPU reduction of 73.3%. Hence, CPU partitioning by ALPHAS significantly depends on not only stream priorities but also other factors, such as the numbers of representation requests. Figure 7b, where the streams have different priorities, confirm that stream priorities do affect CPU allocation. For example, high-priority stream S1 obtains a larger CPU share compared to low-priority stream S3 as CPU reduction becomes more severe.

In Scenario II, Figure 8 reports average normalized QoE for OPT-ALPHAS, SM-ALPHAS, and ARTEMIS. The coordinated encoding of multiple streams enables the ALPHAS variants to outperform ARTEMIS on QoE because, as we discuss in Section II, ARTEMIS is unaware of the CDN edge servers' bandwidth allocations for video delivery to respective client zones, and the aggregate bitrate of the representations requested from a zone might greatly exceed the bottleneck capacity. In contrast, ALPHAS explicitly accounts for the CDNs' bandwidth constraints to avoid the overload. Hence, even the heuristic SM-ALPHAS variant provides higher QoE than the optimal single-live ARTEMIS solution. Compared to Figure 8a where all streams have the same priority, Figure 8b shows that the use of different priorities tangibly affects QoE: high-priority streams S1 and S3 attain substantially higher QoE than low-priority stream S2. With either identical or different priorities, both OPT-ALPHAS and SM-ALPHAS variants consistently deliver better QoE than ARTEMIS.

Figure 8 also plots execution time for OPT-ALPHAS, SM-ALPHAS, and ARTEMIS in Scenario II. With its reliance on Gurobi, OPT-ALPHAS has average execution time of approximately 0.5 seconds, which is acceptably low for the evaluated settings. However, our additional (omitted due to space constraints) experiments reveal that OPT-ALPHAS does not scale up well with the numbers of streams, clients, and zones. In those experiments, execution time of OPT-ALPHAS reaches several minutes, making this ALPHAS variant unsuitable for real-time operation. In contrast, SM-ALPHAS significantly outperforms not only OPT-ALPHAS but also ARTEMIS with respect to execution time in Scenario II (and Scenario I) and exhibits excellent scalability in agreement with Theorem 3.

VI. Related Work

The main focus of this paper is the design and adaptation of bitrate ladders, which are critical for the effectiveness of HAS [37]. A variety of approaches address this challenge.

Static bitrate ladders. HAS typically operates with a static bitrate ladder that remains fixed throughout the streaming session. HTTP Live Streaming (HLS) [38] supersedes one of the earliest solutions introduced by Apple for configuring bitrate ladders and now rivals Dynamic Adaptive Streaming over HTTP (DASH) [39] as a standard for HAS designs. Additionally, video streaming providers, including Twitch [6], YouTube [7], THEO [8], MUX [9], and Bitmovin [10], develop custom settings, guidelines, and recommendations for video encoding. However, a one-size-fits-all bitrate ladder fails to ensure high QoE across diverse video content [11,40]. Inadequate bitrate configurations can degrade video quality for some content and waste bandwidth for others by disregarding content-specific characteristics.

Content-aware bitrate ladders. Per-title encoding, pioneered by Netflix [11], evaluates bitrate-resolution pairs via VMAF to select optimal representations based on rate-distortion curves. Per-chunk bitrate variation adjusts the bitrate according to segment complexity [13]. Similarly, perceptual per-shot bitrate ladders [12] split videos into shots using scene detection, with [41] exploring the benefits of per-shot encoding alongside users' bandwidth and viewport size distributions. Approaches such as [14]–[16] and [42]–[44] propose JND-aware and other content-aware enhancements for video representations. However, these methods do not consider the streaming context.

Context-aware bitrate ladders. [17,19,45,46] design bitrate ladders by accounting for both content complexity and network conditions. Specifically, [19] considers the user's quitting ratio and defines coding conditions to reduce the likelihood of user abandonment. Despite these advancements, such frameworks are still unsuitable for live streaming as they do not adapt to the dynamic nature of the streaming context.

Dynamic bitrate ladders. While LALISA [20] dynamically adjusts the bitrate ladder for a live stream to improve QoE for the viewer, its approach faces scalability challenges, which ARTEMIS overcomes by incorporating the mega-manifest technique. ALPHAS further improves upon ARTEMIS by employing VMAF instead of the peak signal-to-noise ratio (PSNR) for video quality assessment, aligning more closely with human perception. Continuous Bit Rate Slide (COBIRAS) [21] combines the capability of requesting a segment at an arbitrary bitrate and just-in-time encoding of segments. However, [20,21,24] focus on single-stream scenarios and overlook encoder resource constraints, which are considerations addressed in our research.

CDN-assisted bitrate ladder adaptation. Streaming platforms routinely rely on CDNs for scalable caching and increasingly collaborate with CDN providers on broader challenges, in contrast to their historical reluctance to cooperate with network operators [47]. In particular, CMCD [23] emerges as a standard for transmitting client information to CDNs. In [48], a CDN uses a Markov model to predict the next requested bitrate and performs just-in-time transcoding, enabling immediate delivery of video chunks at the needed bitrate. Light-weight Transcoding at the Edge (LwTE) [49] enables edge servers to optimize video storage and delivery by partially transcoding popular chunks and dynamically generating lower bitrates for unpopular chunks using metadata-accelerated transcoding. [41] formulates bitrate-ladder design as an optimization problem with constraints on the CDN capacity, cost, and bandwidth, modeling client bandwidth and viewport sizes as stationary random processes. Our CDN-assisted design targets a different problem of multi-live streaming.

VII. Conclusion

This paper introduces and formulates the problem of multi-live streaming, where an encoding service coordinates CDN-assisted adaptation of bitrate ladders for multiple live streams. In addition to accommodating the CDNs' bandwidth constraints and encoder's computational capabilities, the formulation supports stream prioritization. To solve the problem, we design the ALPHAS system that accounts for video content and streaming context via real-time VMAF estimates and scalable CDN-assisted feedback from clients, respectively. Leveraging the submodular structure of the formulated problem, we equip the system with the practical SM-ALPHAS approximation algorithm that attains a bounded theoretical performance.

We implement and deploy ALPHAS in the EC2 cloud. Our real-world experiments evaluate ALPHAS against the prominent Twitch, YouTube, MUX, and ARTEMIS baselines that encode each of the multiple streams independently. Our evaluation indicates that ALPHAS outperforms the three state-of-the-art static bitrate ladders with multi-objective improvements on QoE, end-to-end latency, stall duration, VMAF instability, CPU utilization per stream, and average VMAF degradation upon CPU reduction by up to 23%, 21%, 37%, 91%, 49%, and 10%, respectively. ALPHAS also delivers consistent multi-objective performance improvements over ARTEMIS, with QoE increases of up to 10% and 20% in the static and dynamic subscription scenarios, respectively.

References

[1] Sandvine, “The Global Internet Phenomena Report,” 2024. [Online]. Available: https://bit.ly/sandvine_report

[2] Ericsson, “Ericsson Mobility Report,” 2024. [Online]. Available: https://bit.ly/ericsson_mobility_report

[3] Z. Zheng, Y. Ma, Y. Liu, F. Yang, Z. Li, Y. Zhang, J. Zhang, W. Shi, W. Chen, D. Li, Q. An, H. Hong, H. H. Liu, and M. Zhang, “XLINK: QoE-Driven Multi-Path QUIC Transport in Large-Scale Video Services,” in *SIGCOMM*, 2021, p. 418–432.

[4] 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, pp. 562–585, 2019.

[5] X. Zuo, J. Yang, M. Wang, and Y. Cui, “Adaptive Bitrate with User-Level QoE Preference for Video Streaming,” in *INFOCOM*, 2022, pp. 1279–1288.

[6] Twitch, “Broadcasting Guidelines.” [Online]. Available: https://help.twitch.tv/s/article/broadcasting-guidelines

[7] YouTube, “Choose Live Encoder Settings, Bitrates, and Resolutions.” [Online]. Available: https://support.google.com/youtube/answer/2853702

[8] THEOlive, “Stream Configuration.” [Online]. Available: https://developers.theo.live/docs/stream-configuration

[9] MUX, “Configure Broadcast Software.” [Online]. Available: https://docs.mux.com/guides/video/configure-broadcast-software

[10] Bitmovin, “Dashboard, Live Encoder.” [Online]. Available: https://bitmovin.com/dashboard/live

[11] A. Aaron, Z. Li, M. Manohara, J. De Cock, and D. Ronca, “Per-Title Encode Optimization,” *Netflix Technology Blog*, 2015. [Online]. Available: https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2

[12] I. Katsavounidis, “Dynamic Optimizer — A Perceptual Video Encoding Optimization Framework,” *Netflix Technology Blog*, 2018. [Online]. Available: https://bit.ly/dynamic_optimizer

[13] J. De Cock, Z. Li, M. Manohara, and A. Aaron, “Complexity-Based Consistent-Quality Encoding in the Cloud,” in *ICIP*, 2016, pp. 1484–1488.

[14] V. V. Menon, H. Amirpour, M. Ghanbari, and C. Timmerer, “Perceptually-Aware Per-Title Encoding for Adaptive Video Streaming,” in *ICME*, 2022, pp. 1–6.

[15] V. V. Menon, P. T. Rajendran, C. Feldmann, K. Schoeffmann, M. Ghanbari, and C. Timmerer, “JND-Aware Two-Pass Per-Title Encoding Scheme for Adaptive Live Streaming,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 34, no. 2, pp. 1281–1294, 2024.

[16] H. Amirpour, R. Schatz, and C. Timmerer, “Between Two and Six? Towards Correct Estimation of JND Step Sizes for VMAF-Based Bitrate Laddering,” in *QoMEX*, 2022, pp. 1–4.

[17] T. Huang, R.-X. Zhang, and L. Sun, “Deep Reinforced Bitrate Ladders for Adaptive Video Streaming,” in *NOSSDAV*, 2021, p. 66–73.

[18] P. Lebreton and K. Yamagishi, “Network and Content-Dependent Bitrate Ladder Estimation for Adaptive Bitrate Video Streaming,” in *ICASSP*, 2021, pp. 4205–4209.

[19] ——, “Quitting Ratio-Based Bitrate Ladder Selection Mechanism for Adaptive Bitrate Video Streaming,” *IEEE Transactions on Multimedia*, vol. 25, pp. 8418–8431, 2023.

[20] F. Tashtarian, A. Bentaleb, H. Amirpour, B. Taraghi, C. Timmerer, H. Hellwagner, and R. Zimmermann, “LALISA: Adaptive Bitrate Ladder Optimization in HTTP-Based Adaptive Live Streaming,” in *NOMS*, 2023, pp. 1–9.

[21] M. Seufert, M. Spangenberger, F. Poignée, F. Wamser, W. Robitza, C. Timmerer, and T. Hossfeld, “COBIRAS: Offering A Continuous Bit Rate Slide to Maximize DASH Streaming Bandwidth Utilization,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2024.

[22] J. Dilley, B. M. Maggs, J. Parikh, H. Prokop, R. K. Sitaraman, and B. Weihl, “Globally Distributed Content Delivery,” *IEEE Internet Computing*, vol. 6, no. 5, pp. 50–58, 2002.

[23] A. Bentaleb, M. Lim, M. N. Akcay, A. C. Begen, and R. Zimmermann, “Common Media Client Data (CMCD): Initial Findings,” in *NOSSDAV*, 2021, p. 25–33.

[24] F. Tashtarian, A. Bentaleb, H. Amirpour, S. Gorinsky, J. Jiang, H. Hellwagner, and C. Timmerer, “ARTEMIS: Adaptive Bitrate Ladder Optimization for Live Video Streaming,” in *NSDI*, 2024, pp. 591–611.

[25] S. Hasan, S. Gorinsky, C. Dovrolis, and R. K. Sitaraman, “Trade-Offs in Optimizing the Cache Deployments of CDNs,” in *INFOCOM*, 2014, pp. 460–468.

[26] H. Amirpour, J. Zhu, P. Le Callet, and C. Timmerer, “A Real-Time Video Quality Metric for HTTP Adaptive Streaming,” in *ICASSP*, 2024, pp. 3810–3814.

[27] Z. Li, C. Bampis, J. Novak, A. Aaron, K. Swanson, A. Moorthy, and J. De Cock, “VMAF: The Journey Continues,” *Netflix Technology Blog*, 2018. [Online]. Available: https://bit.ly/4d6SrRE

[28] Blender, “Big Buck Bunny.” [Online]. Available: https://bit.ly/3YqoaZu

[29] FFmpeg, “FFmpeg.” [Online]. Available: https://www.ffmpeg.org

[30] J. Jiang, V. Sekar, and H. Zhang, “Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE,” in *CoNEXT*, 2012, p. 97–108.

[31] T. X. Tran and D. Pompili, “Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks,” *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, 2019.

[32] Y. Azar and I. Gamzu, “Efficient Submodular Function Maximization under Linear Packing Constraints,” in *ICALP*, 2012, pp. 38–50.

[33] DASH Industry Forum, “Client Implementation for the Playback of MPEG-DASH via Javascript.” [Online]. Available: https://github.com/Dash-Industry-Forum/dash.js

[34] Gurobi Optimization, “Gurobi Optimizer.” [Online]. Available: https://www.gurobi.com/solutions/gurobi-optimizer

[35] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, “Comyco: Quality-Aware Adaptive Video Streaming via Imitation Learning,” in *MM*, 2019, p. 429–437.

[36] D. Yuan, T. Zhao, Y. Xu, H. Xue, and L. Lin, “Visual JND: A Perceptual Measurement in Video Coding,” *IEEE Access*, vol. 7, pp. 29 014–29 022, 2019.

[37] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, “A Survey on Quality of Experience of HTTP Adaptive Streaming,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[38] Apple, “HTTP Live Streaming (HLS) Authoring Specification for Apple Devices,” 2015. [Online]. Available: https://bit.ly/apple_hls

[39] ISO/IEC. 2022, “Information Technology — Dynamic Adaptive Streaming over HTTP (DASH) — Part 1: Media Presentation Description and Segment Formats,” International Organization for Standardization, International Standard 23009-1:2022, 2022.

[40] A. V. Katsenou, F. Zhang, K. Swanson, M. Afonso, J. Sole, and D. R. Bull, “VMAF-Based Bitrate Ladder Estimation for Adaptive Streaming,” in *PCS*, 2021, pp. 1–5.

[41] C. Chen, Y.-C. Lin, S. Benting, and A. Kokaram, “Optimized Transcoding for Large Scale Adaptive Streaming Using Playback Statistics,” in *ICIP*, 2018, pp. 3269–3273.

[42] A. Premkumar, P. T. Rajendran, V. V. Menon, A. Wieckowski, B. Bross, and D. Marpe, “Quality-Aware Dynamic Resolution Adaptation Framework for Adaptive Video Streaming,” in *MMSys*, 2024, p. 292–298.

[43] H. Amirpour, C. Timmerer, and M. Ghanbari, “PSTR: Per-Title Encoding Using Spatio-Temporal Resolutions,” in *ICME*, 2021, pp. 1–6.

[44] M. Bhat, J.-M. Thiesse, and P. L. Callet, “Combining Video Quality Metrics to Select Perceptually Accurate Resolution in A Wide Quality Range: A Case Study,” in *ICIP*, 2021, pp. 2164–2168.

[45] Y. A. Reznik, K. O. Lillevold, A. Jagannath, J. Greer, and J. Corley, “Optimal Design of Encoding Profiles for ABR Streaming,” in *PV*, 2018, p. 43–47.

[46] L. Toni, R. Aparicio-Pardo, K. Pires, G. Simon, A. Blanc, and P. Frossard, “Optimal Selection of Adaptive Streaming Representations,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 11, no. 2s, 2015.

[47] L. Peroni, S. Gorinsky, and F. Tashtarian, “In-Band Quality Notification from Users to ISPs,” in *CloudNet*, 2024, pp. 332–338.

[48] D. K. Krishnappa, M. Zink, and R. K. Sitaraman, “Optimizing the Video Transcoding Workflow in Content Delivery Networks,” in *MMSys*, 2015, pp. 37–48.

[49] A. Erfanian, H. Amirpour, F. Tashtarian, C. Timmerer, and H. Hellwagner, “LwTE: Light-Weight Transcoding at the Edge,” *IEEE Access*, vol. 9, pp. 112 276–112 289, 2021.