Received July 7, 2021, accepted July 26, 2021, date of publication August 5, 2021, date of current version August 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3102633

LwTE: Light-Weight Transcoding at the Edge

ALIREZA ERFANIAN, (Student Member, IEEE), HADI AMIRPOUR, (Student Member, IEEE), FARZAD TASHTARIAN, (Member, IEEE), CHRISTIAN TIMMERER, (Senior Member, IEEE), AND HERMANN HELLWAGNER, (Senior Member, IEEE)

Christian Doppler Laboratory ATHENA, Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt, 9020 Klagenfurt, Austria

Corresponding author: Alireza Erfanian (alireza.erfanian@aau.at)

This work was supported in part by the Austrian Federal Ministry for Digital and Economic Affairs, in part by the National Foundation for Research, Technology and Development, and in part by the Christian Doppler Research Association.

Alireza Erfanian and Hadi Amirpour contributed equally to this work.

ABSTRACT Due to the growing demand for video streaming services, providers have to deal with increasing resource requirements for increasingly heterogeneous environments. To mitigate this problem, many works have been proposed which aim to (*i*) improve cloud/edge caching efficiency, (*ii*) use computation power available in the cloud/edge for on-the-fly transcoding, and (*iii*) optimize the trade-off among various cost parameters, *e.g.*, storage, computation, and bandwidth. In this paper, we propose *LwTE*, a novel *L*ight-*w*eight *T*ranscoding approach at the *E*dge, in the context of HTTP Adaptive Streaming (HAS). During the encoding process of a video segment at the origin side, computationally intense search processes are going on. The main idea of *LwTE* is to store the optimal results of these search processes as metadata for each video bitrate and reuse them at the edge servers to reduce the required time and computational resources for on-the-fly transcoding. *LwTE* enables us to store only the highest bitrate plus corresponding metadata (of very small size) for unpopular video segments/bitrates. In this way, in addition to the significant reduction in bandwidth and storage consumption, the required time for on-the-fly transcoding of a requested segment is remarkably decreased by utilizing its corresponding metadata; unnecessary search processes are avoided. Popular video segments/bitrates are being stored. We investigate our approach for Video-on-Demand (VoD) streaming services by optimizing storage and computation (transcoding) costs at the edge servers and then compare it to conventional methods (store all bitrates, partial transcoding). The results indicate that our approach reduces the transcoding time by at least 80% and decreases the aforementioned costs by 12% to 70% compared to the state-of-the-art approaches.

INDEX TERMS Video streaming, transcoding, video on demand, edge computing.

I. INTRODUCTION

In recent years, video streaming has developed very quickly and, according to the Cisco Visual Networking Index [1], will gain up to 88% of the total Internet traffic by 2022. On the client side, in particular in mobile networks, a wide variety of devices and applications have emerged; that leads to an increasingly heterogeneous environment. To cover the demands of such heterogeneous environments and mitigate network bandwidth fluctuations, it is essential to provide streaming services with different quality levels. *HTTP Adaptive Streaming* (HAS), including *Dynamic Adaptive Streaming over HTTP* (DASH) [2] and *HTTP Live Streaming* (HLS) [3], is the mainstream video delivery technique in the industry today. In a HAS system, a high-quality video source is divided into short intervals known as video segments. By leveraging the transcoding technique, each segment is provided at various bitrates resulting in a *set of representations* (bitrates), *i.e.*, the highest quality representation is converted to other bitrates by decoding and re-encoding processes.

By offering different bitrates of each segment, clients can choose the most appropriate bitrate according to the available network bandwidth. Many strategies have investigated the cost-effective delivery of video, such as on-the-fly transcoding in the network [4], [5], video caching [6], and hybrid approaches [7]–[9].

In the on-the-fly transcoding approaches, the highest bitrate is stored, and the remaining bitrates are transcoded online. The cloud paradigm with virtually unlimited resources enables many cloud service providers like

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

Amazon Web Services or Google Cloud Platform to provide cost-effective transcoding services. However, transcoding tasks are computationally intensive and time-consuming, which impose significant operational costs on service providers. Therefore, on-the-fly transcoding approaches are not commonly used in the industry, especially for large-scale deployments. On the other hand, pre-transcoding methods commonly used in the industry store all bitrates to meet all users' requests. Although storage is becoming cheaper, this approach is not cost-efficient. It incurs high overhead in storage to store all bitrates, especially for video segments/bitrates that are rarely requested.

Some studies try to minimize video streaming costs by combining on-the-fly transcoding and pre-transcoding approaches [7], [10], [11]. In fact, they try to minimize costs by trading off storage costs and computation costs, considering various constraints. They utilize the fact that only a small fraction of videos are popular and account for almost 80% of total views, while other videos receive few requests [12]. To reduce the costs, they store multiple bitrates of popular videos/segments and transcode unpopular videos/segments on-the-fly, using a probability model of video viewing. In other words, when a client requests a popular segment/bitrate, it will be served immediately from the storage, which imposes *storage cost*. In contrast, a request for an unpopular segment/bitrate will be served by on-the-fly transcoding from the highest bitrate to the desired one, which results in *computation cost*. However, since transcoding is inherently a computationally-intensive and time-consuming process, this can impose notable cost and delay.

This paper proposes a novel technique for transcoding called **L**ight-**w**eight **T**ranscoding at the **E**dge (*LwTE*), motivated by the aforementioned issues. Conventional video coding standards like *High Efficiency Video Coding* (HEVC) [13] divide each video frame into blocks of a predefined size and then subdivide them into smaller sub-blocks. The optimal partitioning structure is determined by calculating rate-distortion costs for all sub-blocks [13]. Searching is usually done in a brute-force manner, taking up most of the encoding time as well as of the transcoding time. Our idea is to store the optimal search results/decisions of the encoding process as a metadata file and reuse it in on-the-fly transcoding processes to reduce the transcoding times by avoiding the brute-force search processes. In fact, by leveraging the metadata extracted in the origin server, *LwTE* significantly reduces the transcoding time and computation costs. Edge computing aims at bringing cloud storage/computation resources and services closer to the user. Based on [14], [15], edge computing paradigms can be achieved by employing fog computing [16], cloudlet [17], and mobile edge computing (MEC) [18]. MEC provides resource capabilities by using virtualization techniques in telecommunication networks. Thus, due to the importance of scalability and reliability of the provided service, we leverage the MEC paradigm as edge computing in this paper.

To the best of our knowledge, we are the first to introduce a method to extract metadata during the encoding process and employ them to reduce transcoding efforts at the edge servers. It is worth mentioning that the metadata is extracted during the encoding process in the origin server, which is part of the multi-bitrate video preparation. Thus, *LwTE* does not incur any extra computation cost to extract the metadata. The generated metadata's size is very small compared to the corresponding encoded video segment, as the results indicate. Although the *LwTE* approach is applicable for cloud platforms, we focus on the edge servers in the sense of, *e.g.*, multi-access edge computing in 5G networks, to reveal its potential and capabilities.

The main contributions of this paper are as follows:

- We propose *LwTE* as a novel method that extracts metadata during the encoding process and employs it during the transcoding process at the edge to reduce transcoding time and cost.
- We formulate the problem of minimizing the total cost, including storage and computation (transcoding) costs, as a Mixed-Integer Linear Programming (MILP) model and prove its NP-completeness.
- To mitigate the time complexity of the proposed MILP model, we introduce a polynomial heuristic algorithm to determine a near-optimal solution.
- We compare *LwTE* with state-of-the-art approaches. The implementation results show that *LwTE* achieves at least 80% reduction in transcoding time. Moreover, *LwTE* can decrease the total cost by up to 70% compared with the state-of-the-art approaches.

The remainder of the paper is organized as follows. Section II highlights related work. *LwTE* is described in Section III and evaluated in Section IV. Section V concludes the paper and outlines future work.

II. RELATED WORK

Most of the related works have used a hybrid solution to minimize various cost functions, *e.g.*, for storage, computation, bandwidth, and energy consumption [7], [19]–[21]. Zhao *et al.* [7] formulated a model intending to minimize storage and computation costs for VoD by considering segment popularity and a weighted transcoding graph, which captures each bitrate's computation cost in the representation set to lower bitrates. However, they did not consider the video quality drop caused by using lower bitrate segments for transcoding instead of the highest bitrate segment. A trade-off between the bandwidth cost and the user experience for a cloud-assisted video distribution platform has been studied in [20]. To minimize the backhaul network cost, Tran *et al.* [19] formulated the collaborative joint caching and transcoding problem as an Integer Linear Program (ILP). To mitigate the time complexity and impractical overheads of the proposed model, they presented an online algorithm to determine cache placement and video scheduling decisions. They extended their work such as to minimize the expected video retrieval delay by determining the video request scheduling.

Li *et al.* [22] compared the impact of transcoding and bitrate-aware caching on consumers' *Quality of Experience* (QoE). Jin *et al.* [21] proposed a partial transcoding approach at the edge servers. They store the full representation set for a few popular videos, only keep the highest bitrate for the rest, and prepare the other bitrates on demand by utilizing on-the-fly transcoding. They take into account storage, transcoding, and bandwidth costs and formulate a model to minimize the total cost. Gao *et al.* [11] employed a partial transcoding method based on user viewing patterns in the cloud. Gao *et al.* [23] also addressed resource provisioning issues for transcoding in cloud platforms to maximize financial profit. Wang *et al.* [24] proposed an algorithm to determine edge servers for transcoding operations to improve perceived quality by clients. A distributed platform at the edge named Federated-Fog Delivery Network (F-FDN) has been proposed in [25]. F-FDN stores only one bitrate of non-popular videos and leverages the edge server computing capability to provide requested bitrates at the edge to minimize video streaming latency. Jia *et al.* [26] formulated joint optimization for caching, transcoding, and bandwidth in 5G mobile networks with mobile edge computing.

In [4] and [5], we proposed OSCAR as a framework for real-time streaming. In OSCAR, we served clients' requests by minimizing bandwidth usage and transcoding costs. After aggregating requests at the edge servers, OSCAR transfers the highest requested bitrate from the origin server to the optimal set of Point of Presence (PoP) nodes. After that, virtual transcoders hosted at PoP nodes transcode the highest requested bitrate to those requested by clients. Finally, these bitrates are transferred to the appropriate edge servers and corresponding clients, respectively. However, in OSCAR, we leveraged a conventional transcoding method and executed them in the networks, including the edge. Moreover, OSCAR only targeted a live streaming scenario.

In contrast to other works, *LwTE* employs metadata acquired in the encoding process to reduce the transcoding computation time and cost, which can be applied to the above-mentioned methods to improve their performance.

III. LIGHT-WEIGHT TRANSCODING AT THE EDGE

A. MOTIVATING EXAMPLE

Before explaining an example motivating *LwTE*, let us briefly compare *LwTE* with two basic strategies: *(i)* store all bitrates (*Store-All*) and *(ii)* conventional partial-transcoding (*PT*) [21]. In case of the *Store-All* method, we should only consider the cost of storing all segments in a full representation set for all videos. The *PT* approach stores popular segments/bitrates and only the highest bitrate for unpopular ones. *LwTE* tries to achieve cost reduction as compared to *PT* by employing the metadata to accelerate transcoding operations.

Let us assume 100 segments of a video sequence encoded in a representation set with six bitrates using HEVC HM-16.20 [13] with four-second segment length as shown in Table 1. We are going to measure the storage and computation costs of serving the sequence for one month from an edge server with three request arrival rates: 100, 1000, and 5000 requests per month. As depicted in Fig. 1, the *Store-All* approach stores all segments of the representation set; thus, it introduces just a fixed storage cost regardless of arriving requests (for details of calculating total cost, the reader is referred to Section IV). In case of having request arrival rate 100, *PT* stores around 21% of segments/bitrates as the popular set and serves the others by transcoding, resulting in saving costs by around 55% compared with *Store-All*. However, thanks to the extracted metadata, *LwTE* reduces the total cost by 60% through storing 2% of segments/bitrates as the popular set and performing light-weight transcoding for the other ones. For increasing request arrival rates, the *PT* and *LwTE* approaches store more segments/bitrates in the popular set, which leads to an increase in the storage and computation costs. For instance, in case of request arrival rate 5000, *PT* and *LwTE* store around 57% and 49% of segments/bitrates, respectively.

TABLE 1. Resolutions and bitrates of the encoded video.

QualityId	Resolution	Bitrate (kbps)	Metadata bitrate (kbps)
QId-0	3840x2160	16800	0
QId-1	3840x2160	11600	209
QId-2	2560x1440	8100	86.5
QId-3	1920x1080	5800	66.3
QId-4	1920x1080	4500	47.5
QId-5	1280x720	3400	26

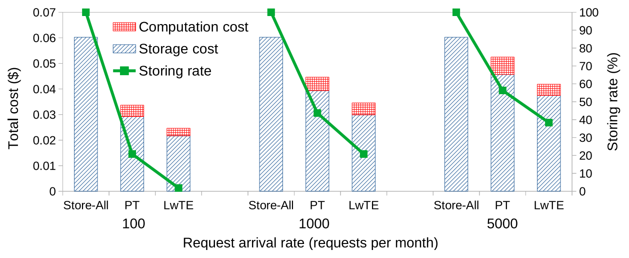

FIGURE 1. Motivating example: Comparison of *LwTE* with two state-of-the-art approaches.

In this paper, by having the metadata for all video segments/bitrates, we address the following problem: For the given set of video segments/bitrates with a wide range of popularity rates and request arrival rates, which segments/bitrates must be stored and which ones should be transcoded in such a way as to minimize the total cost, including storage and computation costs?

B. GENERAL ARCHITECTURE

In HAS, clients adapt to the desired bitrate (resolution/quality level) based on, *e.g.*, network conditions and client characteristics. Consequently, a Content Delivery Network (CDN), *i.e.*, a geographically distributed group of servers providing storage capability for the fast delivery of Internet content, is responsible for distributing all segments (or subsets thereof) within the network towards the clients. The CDN

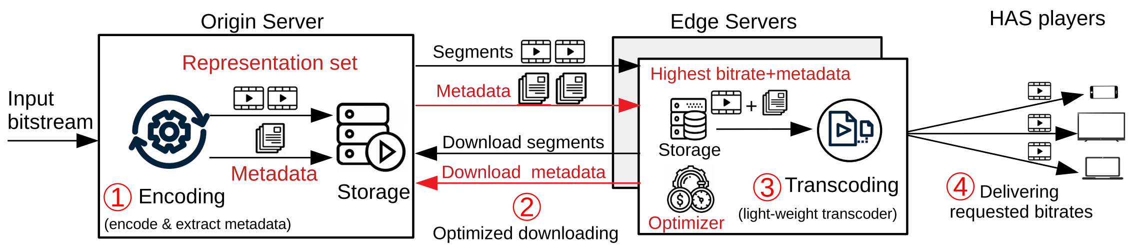

FIGURE 2. ***LwTE*** **architecture.**

paradigm has been generalized and extended by edge computing [27]. On the other hand, edge computing provides storage capability closer to clients, leading to lower latency than CDNs. Moreover, providing computation capabilities within the edge enables us to run compute-intensive services like transcoding, although edge nodes may have limited capacity and serve small, frequently changing client populations [28]. Thus, we use edge computing capabilities and propose a light-weight transcoding approach (*LwTE*). Aiming to reduce the total cost of VoD applications, including storage and computation costs and accelerating the transcoding processes, *LwTE* serves the clients through the following steps (see Fig. 2):

1) **Encoding and extracting metadata**: During the encoding process, selected features are extracted and stored as metadata for all bitrates (except for the highest bitrate) at no additional costs. The metadata is used to reduce the computation time of the transcoding process at the edge. The technical aspects of the metadata generation are discussed in Section III-C.
2) **Optimized downloading**: At this stage, the edge servers employ an optimization model to determine popular and unpopular sets of video segments/bitrates. For popular sets, all video segments/bitrates are downloaded from the origin server. In unpopular sets, only the highest bitrate plus corresponding metadata generated during the encoding process are made available at the edge.
3) **Light-weight transcoding**: This stage will be applied to those segments/bitrates available in the unpopular set. Thanks to the available metadata, the edge server can promptly produce desired bitrates by means of light-weight transcoding.
4) **Delivering requested bitrates**: Finally, requested bitrates are delivered from the edge server to the clients.

C. EXTRACTING METADATA

Video coding is becoming more efficient thanks to sophisticated tools that come with higher demands in terms of computing power and time complexity. In conventional block-based video compression, each video frame is divided into small-sized blocks, and each block is predicted using previously encoded spatial/temporal neighboring blocks.

In HEVC, frames are divided into 64 × 64 pixel blocks, called Coding Tree Units (CTUs) [29], [30]. To encode CTUs, each of them is partitioned into equally sized square blocks

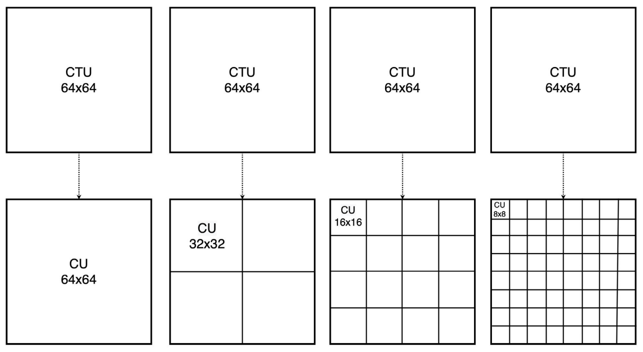

FIGURE 3. **Each CTU is split into CUs with different sizes.**

known as Coding Units (CUs). For a 64 × 64 pixels CTU, four different CU size partitionings including one 64 × 64 pixels CU, four 32 × 32 pixels CUs, sixteen 16 × 16 pixels CUs, and sixty four 8 × 8 pixels CUs is considered. These partitionings for one CTU are shown in Fig. 3. The rate-distortion cost [13] is calculated for all of these CUs to find the optimal CTU partitioning structure with the minimum cost.

The search to find the optimal CTU partitioning into CUs using a brute-force approach takes the largest amount of time in the encoding process. To avoid a brute-force search process at the edge, we extract the optimal partitioning structure for CTUs during encoding in the origin server and store this as metadata for each segment bitrate except the highest bitrate. To save the partitioning structure optimally, a *quad-tree* structure is used for partitioning a CTU recursively into CUs starting from 64 × 64 pixels CU going toward 8 × 8 pixels CUs. When the optimal CTU partitioning is found, '1' is allocated to the partitioned CUs, and '0' is allocated to the non-partitioned CUs. An example of an optimal CTU partitioning structure is shown in Fig. 4. To save this optimal CTU partitioning, thirteen bits are required.

In HEVC, each CU is further split into Prediction Units (PUs) [29]. Eight different PU partitioning modes of each CU in HEVC inter-coding are illustrated in Fig. 5. For each CU, the PU structure with the minimum cost is selected as the optimal PU partitioning mode. In addition to the CTU partitioning structure, the optimal PU partitioning mode for each CU is extracted and added to the metadata. To further reduce the size of the metadata (or bitstream), we use the Huffman algorithm to encode the metadata losslessly.

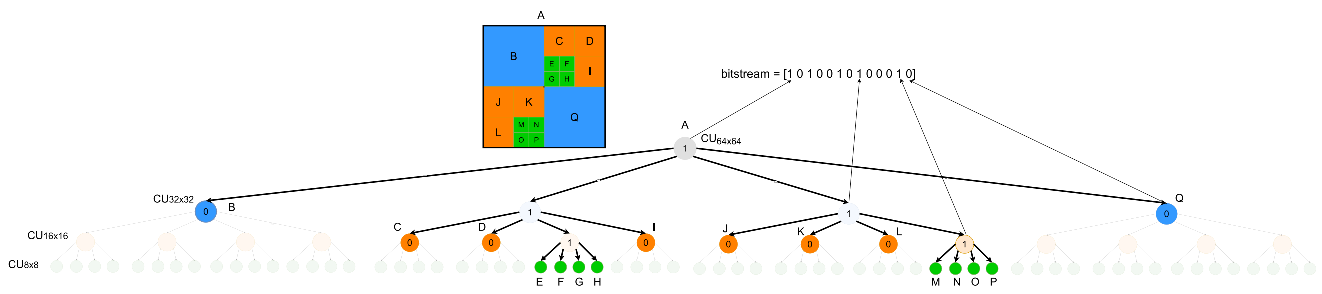

FIGURE 4. Example of an optimal CTU partitioning into CUs and its corresponding quad-tree structure. The optimal decisions in this example are colored. If a CU has been divided to smaller CUs, '1' is added to the bitstream for that CU, otherwise, '0' is added to the bitstream.

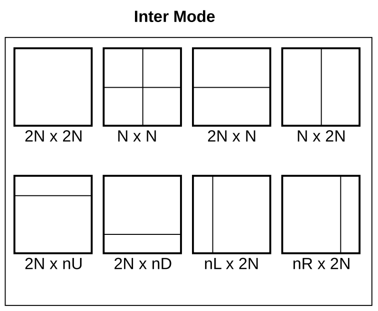

FIGURE 5. Eight different CU partitionings into PUs in inter-coding.

In this paper, *LwTE-M1* (*mode1*) is used if the metadata contains only the optimal CU partitioning structure, and *LwTE-M2* (*mode2*) is used if the metadata contains both the optimal CU partitioning structure and the optimal PU partitioning mode [29].

D. LwTE PROBLEM FORMULATION

Recent studies have shown that the access pattern to video streams follows a long-tail distribution [31]. It means only a small percentage of videos are requested frequently, and the majority of videos are rarely accessed. For instance, in the case of YouTube, it has been shown that only 5% of the videos are popular [32]. Even within a popular video, some segments are accessed more often than others. For example, the beginning portion of a video or a popular highlight part in a video is typically streamed more often than the rest of the video [31]. This paper adopts the long-tail access pattern and the partial transcoding approach.

The main idea of the partial transcoding approach is to store popular video segments/bitrates and keep only the highest bitrate for rarely accessed video segments to be transcoded upon receiving clients' requests. However, the main issues of partial transcoding are (*i*) determining the optimal set of video segments/bitrates which should be stored and (*ii*) the high computation time of the transcoding process that imposes noticeable computation cost and delay.

To cope with the high time complexity of transcoding, *LwTE* leverages the metadata produced during the encoding process. Although, compared to the conventional partial transcoding approaches, *LwTE* transmits corresponding metadata for segments/bitrates that should be transcoded, this transmission cost is worth having since *(i)* it significantly decreases transcoding time at the edge and *(ii)* it introduces only little storage overhead (additional cost) due to its very small size (see Section IV).

We now describe the proposed mixed-integer linear programming (MILP) model that enables *LwTE* to determine the optimal sets of popular and unpopular video segments/bitrates by considering computational resource limitation at the edge (*i.e.*, CPU). Let us define $\mathcal{V}$ as the set of m distinct videos in the edge server and $\mathcal{S} = \{S_i | i = 1, \dots, m\}$ as the set of all segments, where each segment set S_i includes all segments of video i and $s_{i,j} \in S_i$ denotes the jth segment of video i. For the sake of simplicity, let us assume all videos are encoded into a predetermined set $\mathcal{K}$ of representations (bitrates) including k bitrates (see Table 2 for notations). Let $x^r_{i,j}$ be a binary variable that determines if segment $s_{i,j} \in S_i$ in bitrate r must be stored ($x^r_{i,j} = 1$) or served by transcoding ($x^r_{i,j} = 0$). In *LwTE*, we should fetch the highest bitrate to serve requests for popular and unpopular (for transcoding) segments; thus, the first constraint can be stated as follows:

$$x^r_{i,j} = 1, \quad \forall i = \{1, \dots, m\},\ j = \{1, \dots, |S_i|\},\ r = 1 \tag{1}$$

where $r = 1$ shows the highest bitrate of each segment. The storage cost is a function of the storage duration and volume. The storage capacity is consumed by storing the video segments/bitrates for the popular set and the highest bitrate plus the corresponding metadata for the unpopular set. Thus, the storage cost $\mathbf{C}_{str}$ can be formulated as follows:

$$\Delta_s \times \sum_{i=1}^{m} \sum_{j=1}^{|S_i|} \sum_{r=1}^{|\mathcal{K}|} (x^r_{i,j} \times \omega^r_{i,j}) + (1 - x^r_{i,j}) \times \bar{\omega}^r_{i,j} \leq \mathbf{C}_{str}, \tag{2}$$

where $\omega^r_{i,j}$ and $\bar{\omega}^r_{i,j}$ denote the size of bitrate r and the corresponding metadata of segment $s_{i,j}$, respectively. For those segments that must be stored ($x^r_{i,j} = 1$), $x^r_{i,j} \times \omega^r_{i,j}$ represents the size of the bitrate that needs to be stored. In contrast $(1 - x^r_{i,j}) \times \bar{\omega}^r_{i,j}$ shows the required storage for metadata. Moreover, Δ_s refers to the storage cost per byte per θ seconds. The next constraint specifies the required computation cost $\mathbf{C}_{cmp}$ for transcoding processes:

$$\rho \times m \times \Delta_c \times \sum_{i=1}^{m} \sum_{j=1}^{|S_i|} \sum_{r=1}^{|\mathcal{K}|} (1 - x^r_{i,j}) \times F(i,j,r) \times R^r_{i,j} \leq \mathbf{C}_{cmp}, \tag{3}$$

TABLE 2. Notations.

Notation	Description
	Input parameters (MILP model)
$\mathcal{V}$,m	Set of m distinct videos in the edge server
$\mathcal{S}, S_i, s_{i,j}$	Set of all segments of $\mathcal{V}$, where each segment set S_i includes all segments of video i and $s_{i,j} \in S_i$ indicates the jth segment of video i
$\mathcal{K}, k$	Set of representations including k bitrates
Δ_s, Δ_c	Storage cost per byte per θ seconds and resource computation cost per *CPU* per *second*, respectively
$\omega^r_{i,j}$,$\bar{\omega}^r_{i,j}$	Size of segment $s_{i,j}$ in bitrate r and size of corresponding metadata, respectively
Φ	Total available computational resource per second
ρ	Average request arrivals per video at the server during θ seconds
$R^r_{i,j}$	Required resources (*i.e.*, CPU time in seconds) for transcoding segment $s_{i,j}$ into requested bitrate r from its highest bitrate
$F(i,j,r)$	Probability function indicating the request probability of segment $s_{i,j}$ in bitrate r
	Input parameters (heuristic algorithm)
$\mathcal{X}$	Array of all segments of m videos in k bitrates
$P(x)$	Probability function indicating the request probability of $x \in \mathcal{X}$
$\omega_x, \bar{\omega_x}$	Cumulative size of the video segments/bitrates that should be stored up to point x and of the $\lvert\mathcal{X}\rvert - x$ segments/bitrates that should be stored at the highest bitrate plus corresponding metadata, respectively
$\bar{P}_x$	Cumulative required resources (*i.e.*, CPU time in seconds) for transcoding up to point $x \in \mathcal{X}$
R_x	Required resources (*i.e.*, CPU time in seconds) for transcoding segment/bitrate x from its highest bitrate
	Variables (MILP model)
$x^r_{i,j}$	Binary variable that determines if segment $s_{i,j} \in S_i$ in bitrate r must be stored ($x^r_{i,j} = 1$) or served by transcoding ($x^r_{i,j} = 0$)
$\mathbf{C}_{str}$	Storage cost to store the video segments/bitrates in the popular set and the highest bitrate plus the corresponding metadata for the unpopular set during θ seconds
$\mathbf{C}_{cmp}$	computation cost introduced by serving the arrived requests during θ seconds by transcoding
	Variables (heuristic algorithm)
$x^\star$	Integer variable that divides $\mathcal{X}$ into two sub-sets of those video segments/bitrates that should be stored and those that need to be transcoded

where ρ and Δ_c show the average request arrivals per video in θ seconds and resource computation cost per *CPU* per *second*, respectively. $R^r_{i,j}$ defines the required resources (*i.e.*, CPU time in seconds) for transcoding the segment $s_{i,j}$ into requested bitrate r from its highest bitrate. Moreover, $F(i,j,r)$ determines the request probability of segment $s_{i,j}$ in bitrate r based on the given popularity pattern. Considering the computation resource limitation at the edge server (*i.e.*, CPU), we should limit the transcoding to the available computation resource at the edge server through the following constraint:

$$\rho \times m \times \sum_{i=1}^{m} \sum_{j=1}^{|S_i|} \sum_{r=1}^{|\mathcal{K}|} (1 - x^r_{i,j}) \times F(i,j,r) \times R^r_{i,j} \leq \Phi \times \theta, \tag{4}$$

where Φ and θ are the total available computation resource per second and simulation duration in seconds, respectively. Thus, the MILP model can be represented as follows:

$$\begin{aligned} &\textit{Minimize } \mathbf{C}_{str} + \mathbf{C}_{cmp} \\ &\text{subject to: constraints Eq. 1 – Eq. 4,} \\ &\text{variables: } x^r_{i,j} \in \{0,1\}, \quad \mathbf{C}_{str},\ \mathbf{C}_{cmp} \geq 0 \end{aligned} \tag{5}$$

By minimizing the total cost function, we can determine which video segments/bitrates must be stored and which ones should be transcoded according to the given probability function F and the average request arrivals per video (ρ).

Theorem 1: The proposed MILP formulation (Eq. 5) is an NP-complete problem.

Proof: To show that a problem is NP-complete, we need to consider the following two main steps: *(i)* indicating that the problem is in NP, and *(ii)* reducing a well-known NP-complete problem to the addressed problem in polynomial time. To prove that this problem is an NP problem, we should verify whether the proposed solution is a reliable solution in polynomial time or not. By considering the proposed constraints Eq. 1 – Eq. 4, we only need to check whether the solution satisfies these constraints. This examination can be performed in polynomial time for the addressed problem. For the second step, we need to serve all clients' requests in the edge server by either delivering stored segments/bitrates or by transcoding. Let us consider v_i as the obtained value by serving request i that can be proportional to $\frac{1}{c_i}$, where c_i is the cost of serving the request i, and also define a weight parameter w_i as the required computation resource for serving i. Now, by assuming M as the maximum available computational resource at the edge, the addressed problem can be reduced to the NP-complete 0-1 knapsack problem in a polynomial time.

We note here that the current version of the MILP model (Eq. 5) aims to reduce the storage and computation costs concerning the available computational resources. However, we plan to enhance the *LwTE* approach in our future work by considering various metrics in the objective function (*e.g.*, fetch delay, transcoding delay, and bandwidth cost) and adding more realistic constraints (*e.g.*, storage and bandwidth limitations at the edge).

E. HEURISTIC ALGORITHM

Due to the high time complexity of the proposed MILP model (Eq. 5), we introduce an efficient heuristic algorithm to determine a near-optimal solution. Let $\mathcal{X}$ be an array of all segments of m videos in k bitrates. After applying the popularity distribution in [21] over $\mathcal{X}$ and sorting them in descending order, the proposed heuristic algorithm tries to select an optimal integer boundary point, denoted by $x^\star$ where $1 \leq x^\star \leq |\mathcal{X}|$, to divide $\mathcal{X}$ into two sub-sets: *(i)* those video segments/bitrates that should be stored and *(ii)* those that need to be transcoded. We note here that $x^\star$ should be selected regarding the average request arrivals per video (ρ) in such a way that the available computational resource is not violated and the total cost of storage and transcoding is minimized. For a given ρ value, *LwTE* stores $x^\star$ video segments/bitrates to serve requests up to point $x^\star$. The remaining $|\mathcal{X}| - x^\star$ video segments/bitrates are stored in the highest bitrate plus corresponding metadata to serve requests by transcoding.

Let us formulate the storage and computation cost functions for a given integer point x, $1 \leq x \leq |\mathcal{X}|$. The storage capacity is consumed by storing x video segments/bitrates and $|\mathcal{X}| - x$ segments at the highest bitrate plus the corresponding metadata. Thus, the storage cost can be formulated as follows:

$$\mathbf{C}_{str}(x) = (\omega_x + \bar{\omega}_x) \times \Delta_s, \tag{6}$$

where ω_x and $\bar{\omega}_x$ are the cumulative size of the video segments/bitrates that are stored up to point x and $|\mathcal{X}| - x$ segments/bitrates that are stored at the highest bitrate plus corresponding metadata, respectively. Moreover, we can formulate the computation cost in a similar way:

$$\mathbf{C}_{cmp}(x) = (\bar{P}_{|\mathcal{X}|} - \bar{P}_x) \times \rho \times m \times \Delta_c, \tag{7}$$

where $\bar{P}_x$ specifies the cumulative required resources (*i.e.*, CPU time in seconds) for transcoding up to point x; thus, by setting $\bar{P}_0 = 0$, we have:

$$\bar{P}_x = \bar{P}_{x-1} + (P(x) \times R_x), \tag{8}$$

where $P(x)$ and R_x are the request probability function x and the required resources (*i.e.*, CPU time in seconds) for transcoding segment/bitrate x from its highest bitrate, respectively. To consider the computational resource limitation at the edge, we should limit the x in the following equation. Thus, the boundary point $x^\star$ for the given ρ can be obtained as follows:

$$x^\star = \underset{1 \leq x \leq |\mathcal{X}|}{\text{argmin}} \{\mathbf{C}_{str}(x) + \mathbf{C}_{cmp}(x)\} \quad s.t.\ Eq.\ 4 \tag{9}$$

The value of $x^\star$ can be achieved by differentiating the total cost function $(\mathbf{C}_{str}(x) + \mathbf{C}_{cmp}(x))$ with respect to x; however, we first need to estimate the probability function P. Actually, by employing an appropriate method like [33], we can estimate the probability function P. However, to avoid the time complexity of the non-linear function estimation process and its potential error, here we propose a simple heuristic approach based on the binary search algorithm to find $x^\star$ in a limited number of iterations (see Algorithm 1).

First, we define the three input sets as follows: *(i) cuTrans*: the cumulative set of required resources (*i.e.*, CPU time in seconds) for transcoding x video segments/bitrates where $1 \leq x \leq |\mathcal{X}|$, *(ii) cuStorage*1: the cumulative set of required resources for storing x video segments/bitrates where $1 \leq x \leq |\mathcal{X}|$, and *(iii) cuStorage*2: the cumulative set of required resources for storing $|\mathcal{X}| - x$ video segments in their highest bitrates plus their metadata where $1 \leq x \leq |\mathcal{X}|$.

It worth mentioning that in case of need to store the video segments/bitrates, the optimal solution is to store the popular ones. Thus, to satisfy Eq. 4, we should determine the

Algorithm 1 Finding an Optimal Boundary Point $x^\star$

```
Input: cuTrans, cuStorage1, cuStorage2, ρ, m, Φ, k
Output: x⋆
1  x ← len(cuTrans)
2  d ← cuTrans[x] - Φ
3  if d > 0 then
4      lastVisited ← FindElement(d)
5  else
6      lastVisited ← 1
7  end
8  bestCost ← ∞
9  do
10     step ← ⌊ math.abs(x − lastVisited)/2 ⌋
11     cost[x] ← CostFunc(x, cuTrans, cuStorage1,
         cuStorage2, ρ, m)
12     next ← AvgCostFunc(x, x + k, cuTrans,
         cuStorage1, cuStorage2, ρ, m)
13     prev ← AvgCostFunc(x, x − k, cuTrans,
         cuStorage1, cuStorage2, ρ, m)
14     lastVisited ← x
15     if next ≤ prev then
16         x ← x + step
17     else
18         x ← x − step
19     end
20     if cost[x] ≤ bestCost then
21         bestCost ← cost[x]
22         x⋆ ← x
23     end
24 while step > 0;
25 return x⋆
```

amount of violated resources d, in the worst case (when all requests are served by transcoding) based on the cumulative computation set *culTran* (line 2). If the required computational resource is more than the available computational resource (Φ) at the edge server, we set the lower boundary of $x^\star$ to the first element that its value in *culTran* is greater than d (lines 3-4). For example, let assume *culTran* $= [1, 2, 4, 5, 7, 9]$ (the most popular is the first element), $\Phi = 6$, and $d = 9 - 6 = 3$. In this case, the 'FindElement' function returns the third element with the value of 4 as the lower boundary of $x^\star$. In case of available sufficient computational resource, the lower boundary of $x^\star$ is set to 1 (line 6). It should be noted that the upper boundary of $x^\star$ is set to the last element of $\mathcal{X}$ (line 1).

In the *do-while* loop, we try to find $x^\star$ by dividing the search space using variable *step* (line 10). Note that here we prevent unsolicited iterations by restricting the number of examined points to $\lfloor math.abs(x - lastVisited)/2 \rfloor$. In line 11, we call the 'CostFunc' to calculate $\mathbf{C}_{str} + \mathbf{C}_{cmp}$ for x. It is possible to have local minima in the summation of the computation and storage costs. Therefore, to avoid local minima trap and select an accurate search direction in each iteration of the heuristic algorithm, we consider the average $\mathbf{C}_{str} + \mathbf{C}_{cmp}$ of k neighbors before and after $x^\star$ (lines 12-13). Now we update the *lastVisited* and then jump to the new x point according to the value of the *step* variable and determined direction (lines 14-19). In lines 20-23, $x^\star$ and *bestCost* are updated. The search process continues until $step = 0$.

The time complexity of providing the input lists is $|\mathcal{X}|log(|\mathcal{X}|)$ owing to the sort algorithm employed for sorting $\mathcal{X}$. Furthermore, the time complexity of the *while* loop is equal to $log(|\mathcal{X}|)$. Thus the total time complexity of the proposed algorithm is $(|\mathcal{X}| + 1)log(|\mathcal{X}|)$.

IV. PERFORMANCE EVALUATION

A. EVALUATION SETUP AND OVERVIEW

In this paper, aiming to introduce the idea of *LwTE* and to show its feasibility and cost-efficiency compared with conventional and state-of-the-art methods in a simple way, we proposed a mathematical model and heuristic approach to serve client requests by minimizing computation and storage costs. Thus, based on the paper's goals, in this section, we evaluate the feasibility and efficiency of *LwTE* in comprehensive scenarios by varying the considered parameters through implementation/measurement and analytical modeling approaches [34]. To this end, we made the following assumptions: (*i*) there are sufficient computational resources at the edge to serve all incoming requests by transcoding, (*ii*) the performance is evaluated for a fixed time interval, (*iii*) resource costs (*i.e.*, storage and computation) remain fixed during the considered time interval, and (*iv*) arriving requests are distributed uniformly in the time interval.

For the actual implementation, we transcode the full set of representations (*i.e.*, we adopt the bitrate configuration of HEVC/H.265 30 fps from [35]) of the *BasketballDrive* [13] sequence using HEVC HM-16.20 [13] with four-second segment length. It is worth mentioning that transcoding time depends on the video content complexity [36]. However, as the encoding process at the edge (as a part of a transcoding operation) is limited to the optimal decisions stored as metadata, there was not a noticeable difference between "easy to encode" and "hard to encode" videos for both metadata size and transcoding time in our experiments. Thus, for the sake of simplicity, we use one sequence in our evaluations. We generate videos with various lengths, ranging from 2 to 120 minutes, by repeating the *BasketballDrive* sequence segments. We also assume that the number of source videos and video popularity will remain unchanged, and clients can request any bitrate of a segment from the full representation set.

Some studies have shown that the access pattern to video streams follows a long-tail distribution which results in high popularity of only a small set of videos [31], [32]. Requests are generated independently and follow a Poisson process. For the access probability, we use a Zipf-like distribution [31] and for simplicity, we assume that the popularity of each video is known in advance. Considering the set $\mathcal{V}$ including m videos sorted in descending order by videos' popularity, the access probability of video i, denoted by $p_{video}(i)$, is obtained as follows:

$$p_{video}(i) = \frac{l}{i^\alpha}, \quad \text{where } l = \frac{1}{\sum_{i=1}^{m}(1/i^\alpha)}, \tag{10}$$

and $\alpha > 0$ is the level of skewness in the popularity profile. According to [7], [11], [23], we set $\alpha = 0.75$ and use the following cumulative distribution function $q(i, j)$ to determine the access probability of video i up to its jth segment:

$$q(i,j) = \frac{1}{G_i}(1 - \exp^{\gamma j}), \quad \text{where } \gamma = \frac{\sigma}{|S_i|} \; \forall j \in S_i \tag{11}$$

where G_i and S_i are a constant coefficient value and the segment set of video i, respectively. Like [7], we set $G_i = 0.98$ and $\sigma = 4.6$. Therefore, the access probability for each segment $s_{i,j} \in S_i$ is determined as:

$$p_{segment}(i,j) = q(i,j) - q(i,j-1), \tag{12}$$

where $q(i, 0) = 0, \forall i \in \mathcal{V}$. Assuming that the bitrate popularities for various videos are almost identical, the access probability for each bitrate r in representation set $\mathcal{K}$ can be formulated as follows:

$$p_{bitrate}(r) = \frac{e^{-(r-z)^2/2\Gamma^2}}{\sqrt{2\pi}\,\Gamma}, \tag{13}$$

where z and Γ are the mean and variance of the bitrate distribution function, respectively. In general, the middle bitrates in the representation set have a higher demand; thus, we set $z = 6$ and $\Gamma = 0.6$ in our experiments [7]. However, we investigate the performance of *LwTE* regarding different values of α, σ, and Γ in p_{video}, $p_{segment}$, and $p_{bitrate}$, respectively (see Section IV-F). Now, the following probability function F determines the access probability for bitrate r of segment $s_{i,j}$ in video i:

$$F(i,j,r) = p_{video}(i) \times p_{segment}(i,j) \times p_{bitrate}(r) \tag{14}$$

The storage and computation costs are set to 0.024\$ per GB per month and 0.029\$ per CPU per hour, respectively.[1] We calculate the cost values (*i.e.*, storage and computation costs) for a period of one month. Transcoding is performed on Docker containers with one 3.4 GHz CPU and 2 GB memory. We run the evaluation with three and five CPU per second as the total available computational resource (Φ). Since the measured results were very close to each other and follow an identical trend, for ease of explanation, we show the results of $\Phi = 5$ CPU per second in the paper. We also set the $k = 50$ for running the proposed heuristic algorithm.

We evaluate the performance of *LwTE* in six scenarios. In scenario I, we measure the *LwTE* transcoding times and bitrates in two different modes (see Sect. III-C) and compare the results with conventional transcoding. In scenario II, we compare the proposed MILP model and the heuristic algorithm's performance in terms of transcoding rate and

[1]https://calculator.aws/, last access: April 25, 2021.

total cost (*i.e.*, storage and computation costs). In scenario III, the proposed heuristic algorithm's performance utilizing metadata *mode1* is evaluated in terms of storage cost, computation cost, and transcoding rate. In scenario IV, we compare *LwTE* in different modes to state-of-the-art approaches. In scenario V, the impact of various probability distributions on the performance of partial transcoding approaches, including *LwTE* is investigated. In scenario VI, we measure the performance of *LwTE* in different bitrate popularity modes.

B. SCENARIO I

As mentioned earlier, for *LwTE-M1* (*mode1*), we employ the optimal CU partitioning of all CTUs. In addition to optimal CU partitioning, *LwTE-M2* (*mode2*) utilizes the optimal PU mode decisions. Thus, in the first scenario, we measure the *LwTE* performance in terms of transcoding time and the size of generated metadata for the entire representation set for both *LwTE* modes, and compare the results with the conventional transcoding method without the use of metadata.

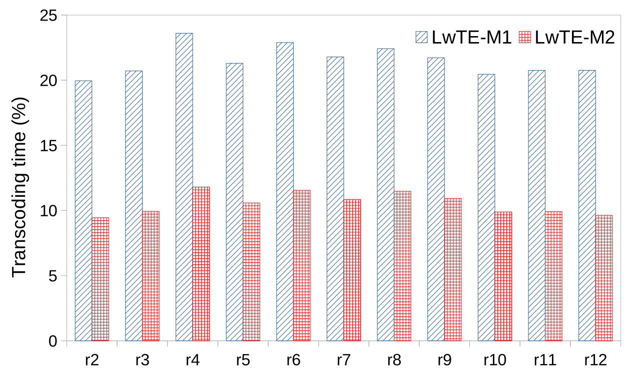

FIGURE 6. Transcoding times of *LwTE-M1* and *LwTE-M2* relative to *conventional mode*'s transcoding times for each bitrate in the representation set.

Fig. 6 shows the transcoding times of *LwTE-M1* and *LwTE-M2* relative to the transcoding times of the *conventional mode* for each bitrate. The transcoding times for *LwTE-M1* and *LwTE-M2* are about 20% and 10% of the *conventional method*, respectively. Since we store the highest bitrate $r1$ to transcode the other bitrates from it, it is omitted from Fig. 6 and the embedded plot in Fig. 7.

Fig. 7 shows the bitrates for all three approaches. The embedded figure shows the bitrates of the metadata relative to the corresponding representations. It is seen that the size of the metadata compared to the corresponding content bitrates is remarkably small. As an example, we take a look at the second bitrate ($r2$). While the bitrate for this representation is 12425 kbps, the metadata's bitrate is 209 kbps for *LwTE-M1* and 318 kbps for *LwTE-M2*, which are significantly smaller than the video bitrate itself (1.7% and 2.5%, respectively). It is clearly seen that by adding more information to the metadata in *LwTE-M2* as compared to *LwTE-M1*, the size of the metadata increases slightly. However, this results in a considerable reduction in transcoding time compared to *LwTE-M1*.

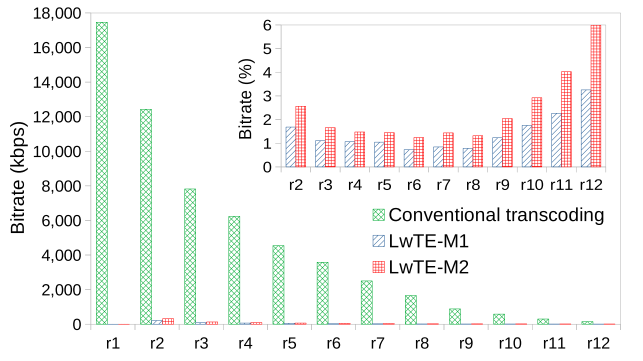

FIGURE 7. Bitrates of all representations and their corresponding metadata. The embedded plot shows the bitrates of the metadata relative to the corresponding representations.

C. SCENARIO II

In this scenario, we investigate the performance of the proposed MILP model and the heuristic algorithm using *LwTE-M1* in terms of the transcoding rate and total cost (*i.e.*, storage and computation costs). Due to the high time complexity of the proposed MILP model (Eq. 5), we conduct the experiments in this scenario with two video sets, including 100 and 1000 distinct videos, and various average request arrivals per video (ρ) for one month. In this paper, we measure the transcoding rate as the percentage of video segments/bitrates in the unpopular set that should be served by transcoding, relative to the total number of video segments/bitrates that is equal to $\frac{|\mathcal{X}|-x^{\star}}{|\mathcal{X}|}\%$.

As depicted in Fig. 8, both the proposed MILP model and the heuristic algorithm show almost identical results in measured normalized total cost and transcoding rate. It means that the proposed heuristic algorithm results in a near-optimal solution. Although the heuristic algorithm produces smaller transcoding rates in some points, *e.g.*, $\rho = 5000$ for the smaller video set ($m = 100$), the measured total costs for both approaches are almost identical. The average measured execution times for the proposed MILP model for video sets with 100 and 1000 videos are 1.2 seconds and 19.2 seconds, respectively. In contrast, the proposed heuristic algorithm determines the solution for both video sets in less than one millisecond.

D. SCENARIO III

In this scenario, we investigate the proposed heuristic algorithm's performance employing *LwTE-M1* for eight video sets and various average request arrivals per video (ρ) for one month in various aspects. As we can see in Fig. 9, the storage and computation costs follow an identical behavior for all video sets (with different numbers of videos). The upper subfigure in Fig. 9 shows a closer look at the storage and computation costs with absolute cost values for 500 videos at various ρ values. The transcoding rate also shows similar behavior for all video sets (see Fig. 10). Since the ρ values are fixed for all video sets, we can conclude that determining

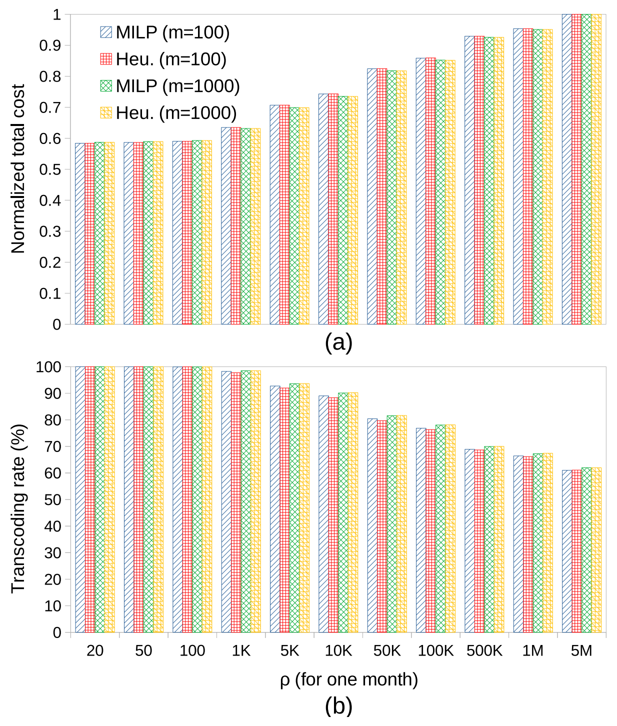

FIGURE 8. Comparison of the proposed MILP model and the heuristic algorithm in terms of (a) normalized total cost and (b) transcoding rate for two video sets, including 100 and 1000 videos, and various average request arrivals per video (ρ) for one month.

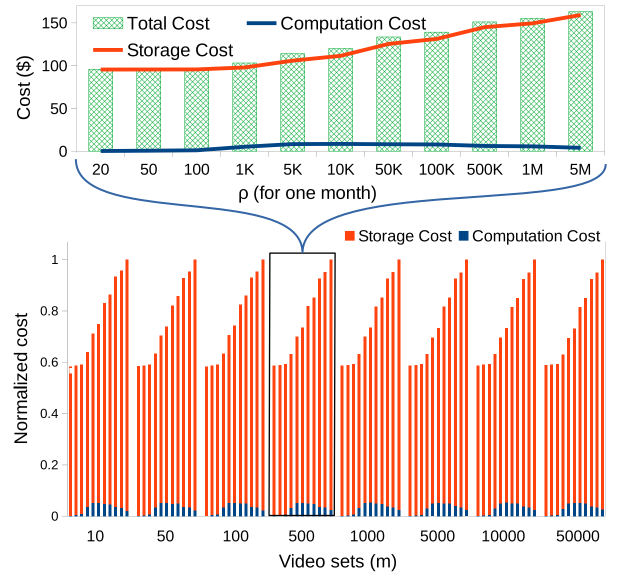

FIGURE 9. Normalized values for storage and computation costs of the proposed heuristic algorithm employing *LwTE-M1* for eight video sets and various average request arrivals per video (ρ) for one month.

the transcoding rate mainly depends on ρ, and the number of video segments/bitrates has a negligible impact on it.

For $\rho \leq 100$, the number of arrived requests is very low, thus *LwTE* prefers to serve almost all requests by transcoding; consequently, there is no change in the storage cost. However, there is a slight increase in the computation cost (see Fig. 9 and Fig. 10). For $100 < \rho \leq 5000$, the transcoding

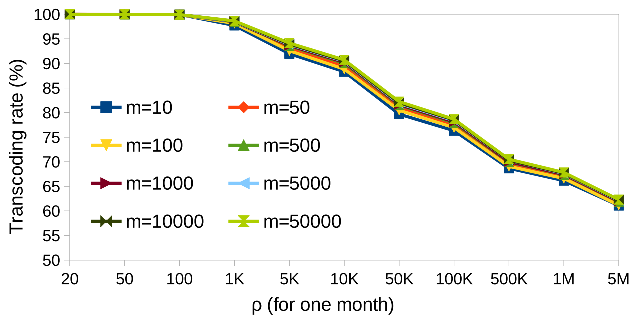

FIGURE 10. Transcoding rate of the proposed heuristic algorithm using *LwTE-M1* for eight video sets and various average request arrivals per video (ρ) for one month.

rate decreases, therefore, more segments/bitrates should be stored, and the storage cost increases moderately. On the other hand, the computation cost increases due to the increase of ρ. For $5000 < \rho \leq 100000$, the computation cost remains almost constant due to the decrease in the transcoding rate, although the ρ value increases. However, the storage cost increases since *LwTE* stores more segments/bitrates. For $\rho > 100000$, the computation cost starts to decrease slightly due to transcoding rate reduction. On the other hand, the storage cost increases in this interval.

E. SCENARIO IV

In this scenario, we compare the *LwTE* approach in different modes with some state-of-the-art and industrial approaches. For this purpose, we select the following methods for comparison in our experiments.

(i) *Store-All*: It stores all segments in a full representation set for all videos.

(ii) *Store-Highest*: Only the highest bitrate is stored for all segments; other bitrates need to be created through transcoding.

(iii) Conventional partial-transcoding (*PT*) [21]: This method stores popular segments/bitrates, while it stores the highest bitrate for unpopular segments. The remaining bitrates in the unpopular set are provided by on-the-fly transcoding in *conventional mode*. Proposed heuristic algorithm (Alg. 1) is used to determine the boundary point between the popular and unpopular segments/bitrates.

(iv) *LwTE-M1* and *LwTE-M2*: The proposed *LwTE* approach in two modes (*i.e.*, *mode1* and *mode2*) employs the proposed heuristic algorithm (Alg. 1) to optimally determine the sets of popular and unpopular video segments/bitrates.

As shown in the previous scenario, the normalized values of the storage and computation costs for various video sets are almost identical. Thus, for ease of explanation, we investigate the performance of the aforementioned approaches for the fixed number of 50000 videos.

As depicted in Fig. 11, the *Store-All* total cost remains fixed regardless of the ρ values due to storing all segments

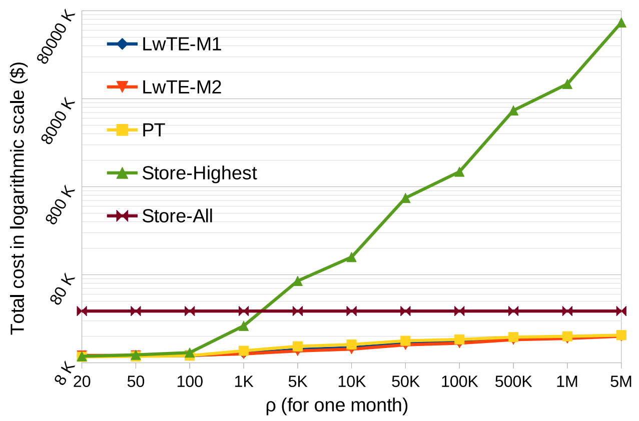

FIGURE 11. Comparison of the proposed ***LwTE*** approach's performance in different modes with some state-of-the-art methods in terms of the total cost.

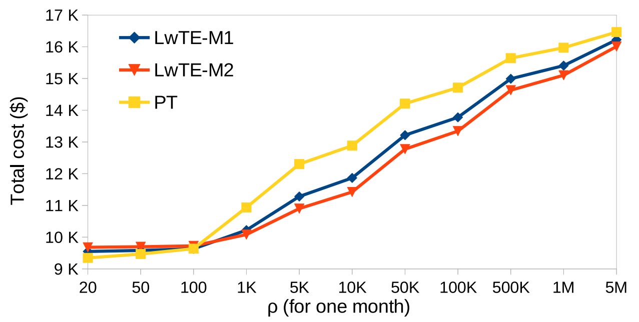

FIGURE 12. Comparison of the proposed ***LwTE*** approach in different modes with ***PT*** in terms of the total cost.

in the full representation set. It also results in the highest cost when $\rho \leq 1000$. In contrast, *Store-Highest* shows better performance for lower ρ values; however, its total cost increases sharply and generates the highest cost when $5000 \leq \rho$. The partial transcoding methods, including *PT*, *LwTE-M1*, and *LwTE-M2* store the popular video segments/bitrates; thus, they result in a better performance than the other approaches (see Fig. 11). Fig. 12 illustrates the obtained total cost for the partial transcoding approaches in a closer inspection. For a low number of arriving requests, $\rho \leq 100$, the computation cost is negligible, so all the partial transcoding methods serve the requests by transcoding (see Fig. 13). However, *PT* results in a lower total cost than *LwTE* approaches, since it does not store the metadata. After this point, the *LwTE* methods have the best performance (see Fig. 11 and Fig. 12). For $\rho > 100$, in the partial transcoding approaches, the transcoding rate starts to reduce. This phenomenon happens in order for *PT*, *LwTE-M1*, and *LwTE-M2* due to the corresponding computation cost (see Figs. 12 and 13). *LwTE-M2* shows the best performance in terms of total cost among all studied approaches due to reduced storage and computation cost. The *LwTE-M1* method has the second-best results.

F. SCENARIO V

In this scenario, we are going to investigate the *LwTE-M1* and *PT* performance with various request

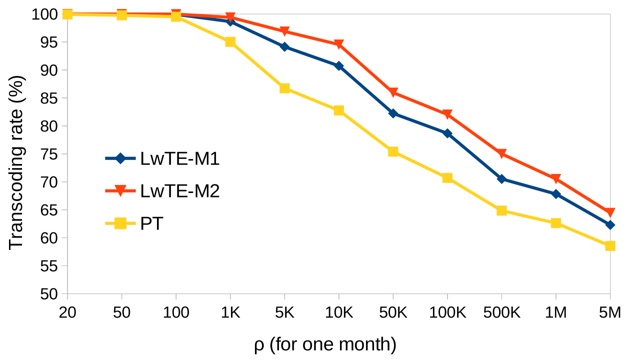

FIGURE 13. Comparison of the proposed ***LwTE*** approach in different modes with ***PT*** in terms of the transcoding rate.

probability distributions for videos, segments, and bitrates. Fig. 14 (a) shows the request probability distribution for 1000 videos achieved from Eq. 10 by setting $\alpha = \{0.75, 1.25, 2\}$. To have various probability distributions for segments within a video and bitrates in the given representation set, we set $\sigma = \{1.2, 4.6, 6.6\}$ and $\Gamma = \{0.6, 1.2, 1.9\}$ in Eq. 11 and Eq. 13, respectively. For example, Fig. 14 (b) and (c) show the probability distributions for 100 segments in a video and 12 bitrates in the representation set. In this scenario, we conduct the experiments for a video set, including 1000 distinct videos, and various average request arrivals per video (ρ) for one month.

Fig. 15(a)–(c) depict the measured total cost and Fig. 15(d)–(f) illustrate the transcoding rates for various probability distributions for videos, segments, and bitrates, respectively. What stands out in Fig. 15 is that, while both approaches illustrate almost identical trends, *LwTE-M1* surpasses the *PT* approach when the ρ is more than 100. Moreover, Fig. 15 also indicates that distributions with a larger standard deviation (*i.e.*, greater α and σ values and smaller Γ values) result in a better performance in terms of total cost and transcoding rate in all cases. In fact, in the case of distributions with a larger standard deviation (*e.g.*, $\alpha = 2$ in Fig. 14(a)), fewer video segments/bitrates are requested with a given probability; thus, the studied approaches need to store fewer video segments/bitrates to serve the arriving requests.

The obtained results for different probability distributions are very close for low ρ values, *i.e.*, less than 100; however, they show a greater difference for higher arrival rates. The bitrate probability distribution results in the broadest differences in terms of total cost and transcoding rate (see $\Gamma = 1.9$ and $\Gamma = 0.6$ in Fig. 15(c) and Fig. 15(f)). On the other hand, the various segment probability distributions result in the smallest differences (see Fig. 15(b) and Fig. 15(e)).

G. SCENARIO VI

In this study, different popularities for each bitrate in the representation set are considered. In this scenario, we are going to investigate the performance of *LwTE* in the following cases: *(i)* considering different popularities for each bitrate

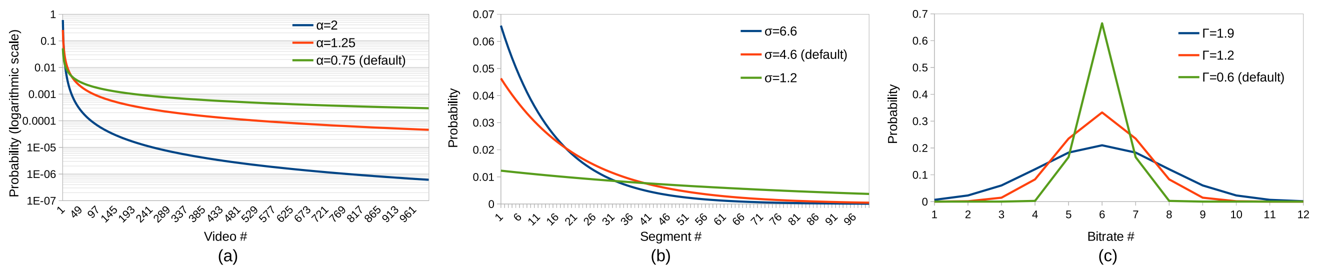

FIGURE 14. Probability distributions for (a) 1000 videos, (b) 100 segments within a video, and (c) bitrates in a representation set.

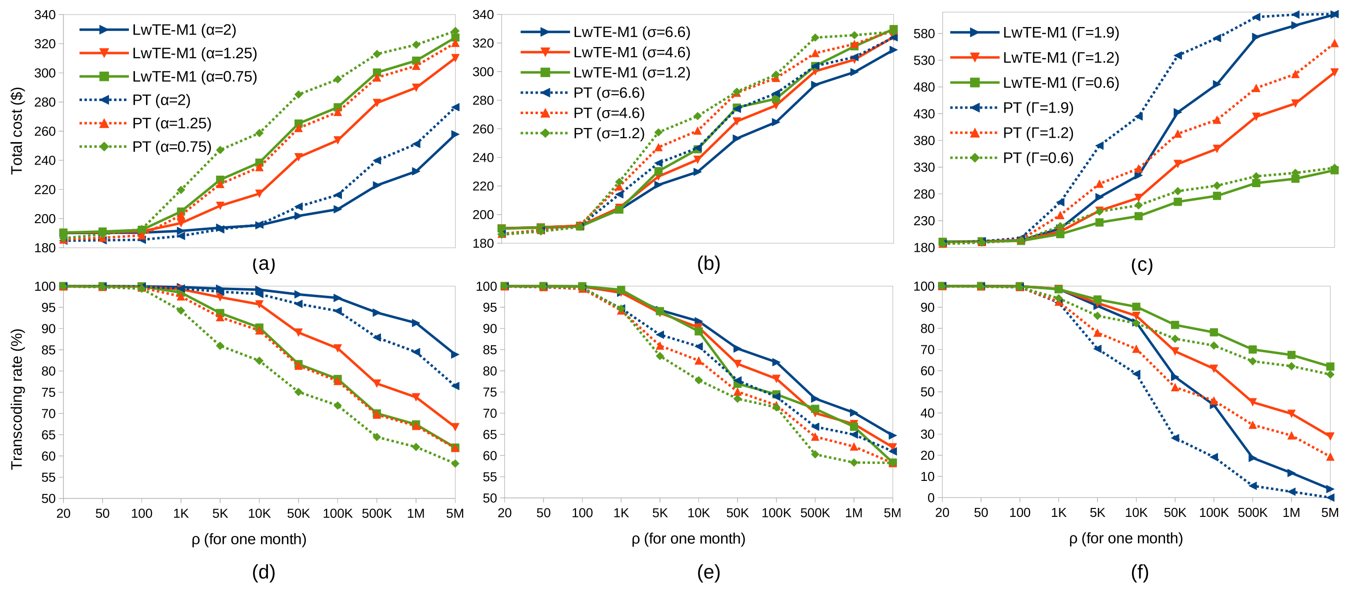

FIGURE 15. Impact of Zipf distribution parameters on total cost with (a) various α values, (b) various σ values, and (c) various Γ values, and on transcoding rate with (d) various α values, (e) various σ values, and (f) various Γ values.

in the representation set (default approach); *(ii)* considering equal popularity for each bitrate in the representation set; and *(iii)* ignoring the bitrate popularity. In fact, in case *(ii)*, each segment's bitrate has equal probability to be selected for the popular set; in the other hand, in case *(iii)*, a segment should be stored with the entire representation set if it is selected for the popular set; for those in the unpopular set, we need to store the highest bitrate plus corresponding metadata for the rest of bitrates.

Both the proposed model (Eq. 5) and the heuristic algorithm support the first and second cases by considering bitrate popularity as an input parameter. However, ignoring the bitrate popularity means identifying popular and unpopular sets according to the video segments' popularity. To support the third approach by the MILP model and the heuristic algorithm, we only need to assume $\mathcal{X}$ as a set of all segments of m videos without considering bitrates.

Here, we investigate the performance of the *LwTE-M1* and *PT* methods for the three cases *(i)*-*(iii)* for a video set with 1000 videos and various ρ values for one month. Fig. 16 and Fig. 17 illustrate the measured total cost and transcoding rate, respectively, for *LwTE-M1* and *PT* using the notations: *(i)* bitrate-based (BB) by considering different popularities for each bitrate in the representation set, *(ii)* equal-popularity (EP) by taking into account equal popularity for each bitrate in the representation set, and *(iii)* segment-based (SB) which does not take into account the bitrate popularity, *i.e.*, only considers video segments' popularity.

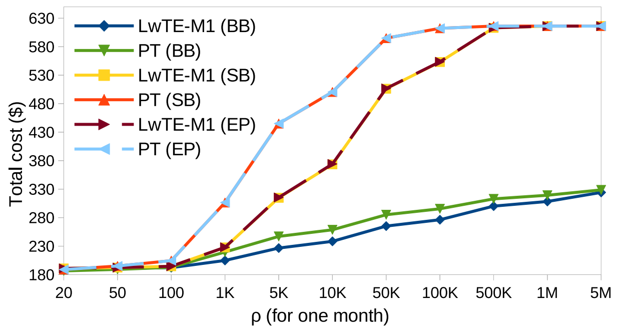

FIGURE 16. Performance comparison of the proposed *LwTE-M1* and *PT* approaches in terms of total cost, considering bitrate-based popularity (BB), equal-bitrate popularity (EP), and segment-based popularity (SB).

From Fig. 16 it can be seen that *LwTE-M1* and *PT* methods for the *BB* case, which is employed as a default approach in this paper, achieve the best performance in terms of the

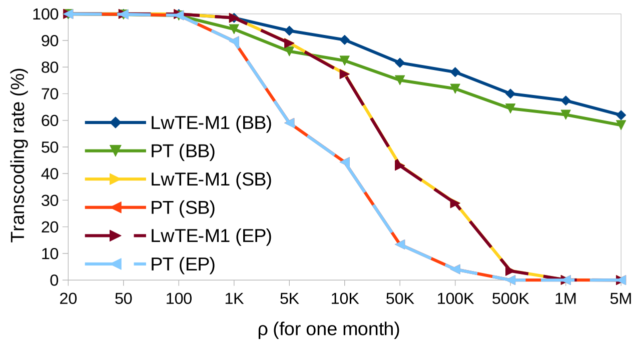

FIGURE 17. Performance comparison of the proposed *LwTE-M1* and *PT* approaches in terms of transcoding rate, considering bitrate-based popularity (BB), equal-bitrate popularity (EP), and segment-based popularity (SB).

total cost with a considerable difference compared to other cases. For the *EP* case, each bitrate of a given segment has an equal probability of selection for the popular set that leads to select all bitrates of the segment. That is, the *LwTE-M1* and *PT* methods for the *EP* and *SB* cases yield almost identical results.

V. CONCLUSION AND FUTURE WORK

In this paper, a novel cost-effective video transcoding approach called *LwTE* was proposed. The main idea of *LwTE* is to store the optimal search results/decisions in the encoding process as metadata for each video bitrate (except the highest) and employ them at the edge server to reduce the transcoding time. For unpopular segments/bitrates, *LwTE* stores only the highest bitrate plus corresponding metadata and serves the desired bitrates by on-the-fly transcoding by employing the metadata. The popular segments/bitrates are stored at the edge server in the usual manner. By utilizing the metadata, unnecessary search processes can be avoided during transcoding, which results in considerable reductions in transcoding time and computation costs. Besides, the storage cost can be reduced as the metadata has a much smaller size than its corresponding segment data. The evaluation results show that *LwTE* does the transcoding processes at least 80% faster than the conventional transcoding method. Moreover, the experimental results indicate up to 70% and 12% cost saving compared to the conventional *Store-All* and *PT* approaches, respectively.

The main goal of this paper was to introduce the idea of *LwTE* and show its feasibility and cost-efficiency compared with conventional and state-of-the-art methods based on simple scenarios. Future work will include more realistic assumptions and constraints, *e.g.*, by considering resource limitations at the edge (*i.e.*, storage, bandwidth, and computation) and multiple time-slots with variable duration into both optimization models and heuristic algorithms.

ACKNOWLEDGMENT

Christian Doppler Laboratory ATHENA: https://athena.itec.aau.at/.

REFERENCES

[1] "Cisco visual networking index: Forecast and trends, 2017–2022," White Paper c11-741490-00. Accessed: Jun. 20, 2021. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/ white-paper-c11-741490.pdf
[2] A. Bentaleb, B. Taani, A. C. Begen, C. Timmerer, and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 562–585, 1st Quart., 2019.
[3] R. Pantos and W. May, Eds., *HTTP Live Streaming*, document RFC 8216, Aug. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8216
[4] A. Erfanian, F. Tashtarian, R. Farahani, C. Timmerer, and H. Hellwagner, "On optimizing resource utilization in AVC-based real-time video streaming," in *Proc. 6th IEEE Conf. Netw. Softwarization (NetSoft)*, Ghent, Belgium, Jun. 2020, pp. 301–309.
[5] A. Erfanian, F. Tashtarian, A. Zabrovskiy, C. Timmerer, and H. Hellwagner, "OSCAR: On optimizing resource utilization in live video streaming," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 1, pp. 552–569, Mar. 2021.
[6] A. O. Al-Abbasi, V. Aggarwal, and M.-R. Ra, "Multi-tier caching analysis in CDN-based over-the-top video streaming systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 835–847, Apr. 2019.
[7] H. Zhao, Q. Zheng, W. Zhang, B. Du, and H. Li, "A segment-based storage and transcoding trade-off strategy for multi-version VoD systems in the cloud," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 149–159, Jan. 2016.
[8] M. Darwich, E. Beyazit, M. A. Salehi, and M. Bayoumi, "Cost efficient repository management for cloud-based on-demand video streaming," in *Proc. 5th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Apr. 2017, pp. 39–44.
[9] F. Tashtarian, A. Erfanian, and A. Varasteh, "S2VC: An SDN-based framework for maximizing QoE in SVC-based HTTP adaptive streaming," *Comput. Netw.*, vol. 146, pp. 33–46, Dec. 2018.
[10] F. Jokhio, A. Ashraf, S. Lafond, and J. Lilius, "A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud," in *Proc. 39th Euromicro Conf. Softw. Eng. Adv. Appl.*, Sep. 2013, pp. 365–372.
[11] G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu, "Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1286–1296, Aug. 2015.
[12] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1357–1370, Oct. 2009.
[13] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[14] E. Ahmed, A. Ahmed, I. Yaqoob, J. Shuja, A. Gani, M. Imran, and M. Shoaib, "Bringing computation closer toward the user network: Is edge computing the solution?" *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 138–144, Nov. 2017.
[15] S. K. U. Zaman, A. I. Jehangiri, T. Maqsood, Z. Ahmad, A. I. Umar, J. Shuja, E. Alanazi, and W. Alasmary, "Mobility-aware computational offloading in mobile edge networks: A survey," *Cluster Comput.*, vol. 24, pp. 1–22, Apr. 2021.
[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. Workshop Mobile Cloud Comput. (MCC)*, 2012, pp. 13–16.
[17] U. Shaukat, E. Ahmed, Z. Anwar, and F. Xia, "Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges," *J. Netw. Comput. Appl.*, vol. 62, pp. 18–40, Feb. 2016.
[18] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
[19] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proc. 13th Annu. Conf. Wireless Demand Netw. Syst. Services (WONS)*, Feb. 2017, pp. 165–172.
[20] J. He, D. Wu, Y. Zeng, X. Hei, and Y. Wen, "Toward optimal deployment of cloud-assisted video distribution services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1717–1728, Oct. 2013.
[21] Y. Jin, Y. Wen, and C. Westphal, "Optimal transcoding and caching for adaptive streaming in media cloud: An analytical approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1914–1925, Dec. 2015.
[22] W. Li, S. M. A. Oteafy, and H. S. Hassanein, "Performance comparison of transcoding and bitrate-aware caching in adaptive video streaming," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[23] G. Gao, H. Hu, Y. Wen, and C. Westphal, ''Resource provisioning and profit maximization for transcoding in clouds: A two-timescale approach,'' *IEEE Trans. Multimedia*, vol. 19, no. 4, pp. 836–848, Apr. 2016.

[24] Z. Wang, L. Sun, C. Wu, W. Zhu, Q. Zhuang, and S. Yang, ''A joint online transcoding and delivery approach for dynamic adaptive streaming,'' *IEEE Trans. Multimedia*, vol. 17, no. 6, pp. 867–879, Jun. 2015.

[25] V. Veillon, C. Denninnart, and M. A. Salehi, ''F-FDN: Federation of fog computing systems for low latency video streaming,'' in *Proc. IEEE 3rd Int. Conf. Fog Edge Comput. (ICFEC)*, May 2019, pp. 1–9.

[26] Q. Jia, R. Xie, H. Lu, W. Zheng, and H. Luo, ''Joint optimization scheme for caching, transcoding and bandwidth in 5G networks with mobile edge computing,'' in *Proc. IEEE 5th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2019, pp. 999–1004.

[27] M. Satyanarayanan, ''The emergence of edge computing,'' *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[28] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, ''Placing dynamic content in caches with small population,'' in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, Apr. 2016, pp. 1–9.

[29] I.-K. Kim, J. Min, T. Lee, W.-J. Han, and J. Park, ''Block partitioning structure in the HEVC standard,'' *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1697–1706, Dec. 2012.

[30] E. Çetinkaya, H. Amirpour, M. Ghanbari, and C. Timmerer, ''CTU depth decision algorithms for HEVC: A survey,'' 2021, *arXiv:2104.08328*. [Online]. Available: http://arxiv.org/abs/2104.08328

[31] L. Cherkasova and M. Gupta, ''Analysis of enterprise media server workloads: Access patterns, locality, content evolution, and rates of change,'' *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, pp. 781–794, Oct. 2004.

[32] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, ''YouTube traffic characterization: A view from the edge,'' in *Proc. 7th ACM SIGCOMM Conf. Internet Meas. (IMC)*, 2007, pp. 15–28.

[33] W. Chen, Y. Tian, and M. Xie, ''Maximum likelihood estimator of the parameter for a continuous one-parameter exponential family under the optimal ranked set sampling,'' *J. Syst. Sci. Complex.*, vol. 30, no. 6, pp. 1350–1363, Dec. 2017.

[34] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. Cambridge, U.K.: Cambridge Univ. Press, 2015.

[35] *HLS Authoring Specification for Apple Devices*. Accessed: Jun. 20, 2021. [Online]. Available: https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices

[36] A. Zabrovskiy, P. Agrawal, R. Mathá, C. Timmerer, and R. Prodan, ''ComplexCTTP: Complexity class based transcoding time prediction for video sequences using artificial neural network,'' in *Proc. IEEE 6th Int. Conf. Multimedia Big Data (BigMM)*, Sep. 2020, pp. 316–325.

ALIREZA ERFANIAN (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer engineering, in 2009 and 2017, respectively. He is currently pursuing the Ph.D. degree in the 5G playground and ATHENA projects with the Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt (AAU). He has been working in the computer networks field for over 12 years. His research interests include multimedia communication and adaptation, software-defined networking, network function virtualization, 5G networks, and optimization. Further information at https://sites.google.com/view/alireza-erfanian/.

HADI AMIRPOUR (Student Member, IEEE) received the dual B.Sc. degree in electrical and biomedical engineering and the M.Sc. degree in electrical engineering. He is currently pursuing the Ph.D. degree with the Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt (AAU). He was involved in the project EmergIMG, a Portuguese consortium on emerging imaging technologies, funded by the Portuguese funding agency and H2020. He is currently working with ATHENA. His research interests include video streaming, image and video compression, quality of experience, emerging 3D imaging technology and medical image analysis. Further information at https://hadiamirpour.github.io/.

FARZAD TASHTARIAN (Member, IEEE) received the Ph.D. degree in computer engineering from the Ferdowsi University of Mashhad. He is currently a Postdoctoral Researcher in the ATHENA project with the Institute of Information Technology (ITEC), Alpen-Adria-Universität Klagenfurt (AAU). Before joining the team, he was an assistant Professor with the Azad University of Mashhad, Iran. His current research interests include end-to-end latency and QoE in video streaming, video networking, software-defined networking, network function virtualization, mathematical modeling, and distributed optimization. He is a member of the technical program committee of several international conferences. Further information at https://tashtarian.net/.

CHRISTIAN TIMMERER (Senior Member, IEEE) is currently an Associate Professor with the Institute of Information Technology (ITEC) and the Director of the Christian Doppler (CD) Laboratory, ATHENA (https://athena.itec.aau.at). His research interests include immersive multimedia communication, streaming, adaptation, and quality of experience, where he coauthored seven patents and more than 200 articles. He was the General Chair of WIAMIS 2008, QoMEX 2013, MMSys 2016, and PV 2018 and has participated in several EC-funded projects, notably DANAE, ENTHRONE, P2P-Next, ALICANTE, SocialSensor, COST IC1003 QUALINET, and ICoSOLE. He also participated in ISO/MPEG work for several years, notably in the area of MPEG-21, MPEG-M, MPEG-V, and MPEG-DASH, where he also served as a Standard Editor. In 2013, he co-founded Bitmovin (http://www.bitmovin.com/) to provide professional services around MPEG-DASH, where he holds the position of the Chief Innovation Officer (CIO)—Head of research and standardization. Further information at http://timmerer.com/.

HERMANN HELLWAGNER (Senior Member, IEEE) is currently a Full Professor of computer science with Alpen-Adria-Universität Klagenfurt (AAU), where he leads the research group Multimedia Communication (MMC), Institute of Information Technology (ITEC). Earlier, he was an Associate Professor with the University of Technology in Munich (TUM) and a Senior Researcher with Siemens Corporate Research, Munich. His current research interests include distributed multimedia systems, multimedia communication and adaptation, QoS/QoE, information-centric networking, and communication in multi-UAV networks. He has published widely on parallel computer architecture, parallel programming, and multimedia communication and adaptation. He is a member of the ACM. He was a member of the Scientific Board of the Austrian Science Fund (FWF) (2005–2016) and an FWF Vice President (2013–2016). He is currently a member of the CD Senate of Christian Doppler Forschungsgesellschaft (CDG). Further information at https://www.itec.aau.at/.