Cost-Effective, Quality-Oriented Transcoding of Live-Streamed Video on Edge-Servers

Dayoung Lee, *Student Member, IEEE,* Younghyun Kim, *Member, IEEE,* and Minseok Song, *Member, IEEE*

Abstract—Live-streaming video requires a lot of CPU-intensive transcoding so that viewers can receive video at bitrates appropriate to their devices and network conditions, which is necessary for a good quality of experience (QoE). We allocate transcoding tasks to edge-servers in a multiple-access edge-computing (MEC) architecture, taking into account server capacity, wireless network coverage, and the cost budget of broadcasters, as well as QoE. Our algorithm first chooses candidate transcoding tasks by giving higher priority to the tasks that make the most cost-effective contribution to popularity-weighted video quality (PWQ). It assigns these tasks to edge-servers in a greedy manner, taking network coverage and computational load into account. Subsequently, it meets a cost budget by reassigning some tasks and removing other assignments altogether, while trying to minimize the effect of these alterations on total PWQ. Simulation results show that our scheme achieves 0.06% to 94.62% (average 25.3%) more PWQ than alternative schemes under the same cost budget.

Index Terms—Edge computing, Streaming media, Multimedia systems, Transcoding

1 Introduction

LIVE-streaming of real-time content such as video sporting events is gaining popularity. For example, Twitch, which is one of the best game live-streaming platforms accounts for 1.8% of the Internet traffic in the United States [1]. In a live-streaming scenario, the channel, which is the source of the broadcast, continuously captures video streams, encodes, and uploads them to its regional servers, which then deliver them to the requested viewers. These continuous computing and transmission processes require a lot of computational resources.

To allow for changing network conditions, live-streaming services typically use dynamic adaptive streaming over HTTP (DASH), in which each source video stream is transcoded into versions with different bitrates, so that the videos delivered to viewers match their network bandwidth [2], [3]. However, transcoding requires more computation than can be performed by a single regional server, so broadcasters supply only a limited number of versions of a video stream. For example, Twitch.tv only offers transcoding services to premium channels [2], [3]. Thus many viewers receive videos at a bitrate which is far from optimal, reducing their quality-of-experience (QoE).

Cloud servers can be used to perform transcoding [4], [5], but the cloud cannot meet the latency requirement of live-streaming, particularly when cloud servers are a long way from the source of the broadcast [4], [6], [7]. Cloud-based transcoding services are therefore considered only to be suitable for video on demand (VoD), and not for live-streaming [4], [6], [7]. Some way needs to be found to make the computational resources of the cloud available for live-streaming.

Multi-access edge computing (MEC) involves the placement of edge-servers (ESs) at the edge of this network, so that they are closer to the sources of data to be processed [8], [9], [10]. MEC is becoming ubiquitous. For example, Gartner [11] predicts that by 2023 more than 50% of large enterprises will have more than six Internet-of-Things (IoT) applications that require edge computing, and there is a growing number of edge infrastructure providers [12], [13].

Live-streaming service providers can use MEC to distribute transcoding workloads previously directed to a central ingest server by transcoding video streams on edge-servers closer to channels' locations. Because these edge-servers are on the path of the streaming data, there is little increase in latency. However, the computational capacity and network coverage of each edge-server is limited, so effective allocation of transcoding tasks is required [14], [15], which must also seek to minimize the total cost of using the edge infrastructure [12], [16].

A lot of work has been done to address the high computational load placed on transcoding servers by real-time and other streaming services. Some schemes for VoD combine the online transcoding of uncommon bitrates with offline transcoding of bitrates that are more frequently requested [17], [18], [19], [20]; others involve the allocation and scheduling of transcoding tasks to satisfy heterogeneous QoE requirements [21]; and the high power consumption of transcoding servers has also been addressed [22], [23]. But these proposals do not make many contributions to live-streaming systems, which require continuous real-time transcoding.

Several techniques have been proposed in which viewers' devices are used for some transcoding tasks in a live-streaming system [4], [6], [7]. However, the instability of viewers' connections and the variable amount of computation available for transcoding makes it difficult to achieve consistent results [7]. To the best of our knowledge, this is the first attempt to use the computing of edge-servers for transcoding specifically for live-

- *Dayoung Lee and Minseok Song are with the Department of Computer Engineering, Inha University, Incheon, Korea. E-mail: dayoung08@inha.edu and mssong@inha.ac.kr*
- *Younghyun Kim is with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison. E-mail: younghyun.kim@wisc.edu*

Manuscript received Dec 9, 2022; revised 14 Oct 2022; accepted 02 Mar. 2023.
(Corresponding author: Minseok Song.)

streaming systems by taking both QoE and cost into account.

We propose a new live-streaming architecture that allows transcoding on edge-servers within the coverage of channels in an MEC architecture. We first examine the effect of transcoding a number of versions of a live-streamed broadcast on perceived quality, aggregated over all users; that quality is clearly related to the popularity of the various versions. We then formulate an optimization problem to determine the versions to be transcoded and the allocation of transcoding tasks to servers. We address this problem with an algorithm that first determines candidate transcoding tasks and allocates them provisionally to servers within the channels' network coverage. It then redistributes these tasks to limit cost while also aiming to maximize overall popularity-weighted video quality.

Specifically, the main contributions of this work are summarized as follows.

- Modeling: We model a MEC-aware transcoding task allocation framework that takes into account each edge server's network coverage and heterogeneous processing requirements. In this framework, we consider the trade-off between the cost model for edge server usage and the quality of the video being delivered.
- Problem formulation: We formulate a multi-objective optimization problem that considers cost and video quality simultaneously, based on transcoding task execution and actual edge server network coverage model.
- Algorithm design: To effectively deal with multi-constrained problems (cost, processing capacity, and network coverage constraints), we design a two-phase algorithm that progressively satisfies these constraints at low overhead, thereby allowing it to be used effectively in a dynamic live-streaming environment.
- Extensive evaluations: We evaluate and demonstrate the effectiveness of the proposed scheme in terms of video quality while varying the cost budget, the number of channels and edge-servers, the allowable bitrates, algorithm complexity, cost model, and so on. To verify its efficacy in practice, we compare our scheme with various algorithms such as baseline methods, greedy heuristics, meta-heuristics, and reinforcement algorithms.

The rest of this paper is organized as follows. We review related work in Section 2. We describe our system architecture in Section 3 and formulate the optimization problem in Section 4. We propose a transcoding task allocation algorithm in Section 5, assess its effectiveness in Section 6, and draw conclusions in Section 7.

2 Related Work

In VoD systems, some or all of the transcoding can be performed offline (i.e. in advance) [17], [18], [19]. As a result, the techniques for managing the transcoding workload are largely irrelevant to live-streaming, and we do not review them here.

Several authors have looked at resource allocation in live-streaming. Zhang et al. [3] try to maximize QoE within a cost budget by migrating the transcoding workloads of unpopular channels to the cloud. Lee and Song [23] allocate transcoding tasks based on CPU utilization with the aim of minimizing power consumption. Gao et al. [24] introduce a scheme for allocating computing resources and scheduling tasks on cloud-based transcoding servers based on a dual-timescale stochastic optimization. They [21] also use priority-based resource provisioning and scheduling to meet heterogeneous quality-of-service (QoS) requirements while accommodating changing workloads.

Li et al. [25] allocate resources using a tradeoff between transcoding time and computation cost, based on predicted transcoding times for heterogeneous virtual machines (VMs) in the cloud. Mada et al. [26] apply dynamic scaling to virtual machines by different cloud providers to reduce costs. Zheng et al. [27] introduce a game video live-streaming architecture in which decisions relating to transcoding, bitrate adaptation and data center allocation, are made dynamically with the aim of minimizing the service provider's operating costs. Most of these approaches largely rely on the cloud infrastructure with limited computational capacity, which is likely to involve latencies that are too long for live-streaming.

The use of edge-servers makes it easier to cope with high computational loads of transcoding in a timely fashion. Dao et al. [28] use a genetic algorithm to make video caching decisions and to select quality levels to achieve a good tradeoff between cache hit-rate and content quality in edge caching systems for video. Wang et al. [29] group users who require similar bandwidths, and then transcode one version of each group to improve the tradeoff between video quality and network traffic. Gao and Wen [5] combine online transcoding with edge-caching decisions with the aim of maximizing QoE while reducing operating costs. Tran et al. [30] allocate both edge-caching and transcoding based on video popularity, CPU and storage requirements, with the aim of improving video quality. Baccoura et al. [31] introduce a caching algorithm that makes caching and transcoding decisions progressively, so as to reflect changes in video popularity. Guo et al. [32] use a deep reinforcement learning algorithm to schedule transcoding tasks on a single edge-server with limits on network bandwidth even when the network state is unknown, with the aim of maximizing streaming quality. Most of these techniques focus on improving the efficiency of video caching on edge-servers close to viewers, and do not consider the overall allocation of computing resources, which is essential for live-streaming systems.

Several authors have addressed the allocation of transcoding tasks in edge-based live-streaming systems. Zhang et al. [2] propose an actor-critic algorithm that schedules video delivery, makes transcoding decisions and also allocates radio spectrum allocation in heterogeneous networks with edge-servers. Zhu et al. [4] propose a collaborative system which allocates video transcoding across viewers' devices and cloud servers, using an auction scheme, which encourages devices with stable network connections to participate in transcoding. An improved algorithm [6] also considers a cost budget. He et al. [7] offload transcoding assignments to the large numbers of viewers' devices that typically participate in live-streaming by detecting stable devices and scheduling task allocation accordingly. Similarly, Dogga et al. [33] treat viewers' devices as edge-servers, in a peer-to-peer transcoding system in which the resulting versions of a video are shared with nearby devices. These techniques require viewers to allow their devices to participate. Instead of attempting to discover which viewers' devices are stable, which will in any case change over time, our scheme relies on edge-servers with fixed stability.

To allocate transcoding tasks to the edge-servers that are close to a channel in an MEC architecture, their computing resources, network coverage, and cost budget must all be considered. None of the work summarized above addresses these issues. To the best

of our knowledge, this paper is the first attempt to maximize aggregate video quality while limiting cost in a live-streaming system with an MEC architecture and heterogeneous edge-servers.

3 System Model

3.1 Architecture

Fig. 1 shows our live-streaming scenario, in which there are four types of server: ingest, central transcoding, delivery and edge. Each channel is a source of broadcasting, and it continually captures, encodes, and uploads video streams to its ingest server, which also manages a list of channels. The original video streams from each channel can be transcoded by the central transcoding server (CTS), which is operated by the live-streaming provider. We also assume that the upload path from the channel to the ingest server passes through, or close to, several edge-servers, and that these edge-servers are available for transcoding. The ingest server determines which server executes each transcoding task. The delivery server is then able to supply a version of the video at the appropriate bitrate to each viewer's device.

If N^{ES} is the number of edge-servers, then the total number of transcoding servers (TSs) is $N^{\mathrm{ES}}+1$, because where TS 0 is the CTS. Table 1 summarizes important notations used in this paper. Different TSs may have different processing capacity, which is the number of operations that can be executed in a given amount of time. For ease of exposition, we express the processing capacity $U_j^{\max}$, $(j=0,...,N^{\mathrm{ES}})$ of TS j as a fraction of that of the TS with the highest processing capacity.

Each edge-server is assigned to a base station to handle transcoding tasks from channels within their network coverage [15]. Thus, each edge-server has a network coverage constraint based on its distance from the channel [15]. We introduce a binary variable $C_{i,j}^{\mathrm{net}}$, which indicates whether each channel i is within the network coverage of TS j; if channel i is within the network coverage of TS j, then $C_{i,j}^{\mathrm{net}}=1$; otherwise, $C_{i,j}^{\mathrm{net}}=0$. Since all the transcoding tasks can be executed on the CTS (which is TS 0), $\forall i, C_{i,0}^{\mathrm{net}}=1$.

3.2 Video quality model

Let N^{ch} be the total number of channels. We define $V_{i,k}$, $(i=1,...,N^{\mathrm{ch}}$ and $k=1,...,N^{\mathrm{ver}})$ to be the kth lowest bitrate version of channel i, where N^{ver} is the number of different bitrates. Since the original highest bitrate version is always available, the number of versions N^{tran} that may be produced by transcoding is $N^{\mathrm{ver}}-1$. Let $\tau_{i,k}$ be the transcoding task for version $V_{i,k}$, $(i=1,...,N^{\mathrm{ch}}$ and $k=1,...,N^{\mathrm{tran}})$.

Live streaming requires real-time transcoding to keep up with playback rates, which can be realized when transcoding for each segment is continuously completed within the duration of that segment [23], [51]. We express the processing capacity requirement $U_{i,k}$ of each task $\tau_{i,k}$ as a fraction of the processor capacity of TS with the highest processing capacity to support real-time transcoding. To derive $U_{i,k}$, the number of CPU cycles required to transcode version $V_{i,k}$ within the segment playback length is measured using programs such as the Linux perf tool [23], [51]. Then the processing capacity requirement $U_{i,k}$ of task $\tau_{i,k}$ can be calculated by dividing this value by the number of cycles provided by the edge server with the highest processing capacity during the segment length. If $U_{i,k}$ is guaranteed to be assigned to task $\tau_{i,k}$, then real-time transcoding for live streaming can be realized [23], [51].

TABLE 1
Important symbols used in this paper.

Symbol	Meaning
N^{ES}	Number of edge-servers
N^{ver}	Number of different bitrate versions
N^{tran}	Number of versions that can be transcoded
$U_j^{\max}$	Processing capacity of transcoding server j
$C_{i,j}^{\mathrm{net}}$	Binary variable which indicates whether each channel i is within network coverage of transcoding server j
N^{ch}	Number of channels
$V_{i,k}$	kth lowest bitrate version of channel i
$\tau_{i,k}$	Transcoding task for version $V_{i,k}$
$U_{i,k}$	Processing requirement $U_{i,k}$ of task $\tau_{i,k}$
$p_{i,k}$	Access probability of version $V_{i,k}$
$Q_{i,k}$	Video quality index of version $V_{i,k}$
$Q_{i,k}^{\mathrm{pwq}}$	Popularity-weighted video quality for version $V_{i,k}$
$H_{i,k}$	Available version with the highest bitrate lower than requested bitrate if the bitrate requested is k for channel i
Q_i^{ch}	Total PWQ over all transcoded versions for channel i
$C_j^{\mathrm{cost}}(u)$	Normalized cost based on CPU usage u
C_j^{onoff}	Normalized cost for using ES j in on-off cost model
C^{budget}	Cost budget
$X_{i,k}$	Binary variable which indicates whether version $V_{i,k}$ is transcoded or not
$Y_{i,k}$	TS index on which task $\tau_{i,k}$ is executed
A_j	Array of task indices assigned to TS j
$D_{i,k}^{\mathrm{task}}$	Total PWQ when eliminating $\tau_{i,k}$ from the set of tasks to be executed
$R_{i,k}^{\mathrm{TDA}}$	Ratio of $D_{i,k}^{\mathrm{task}}$ to $U_{i,k}$ for $\tau_{i,k}$ in TDA phase
$P_{i,j,k}^{\mathrm{TDA}}$	Parameter used in TDA phase when task $\tau_{i,k}$ is assigned to ES j
$I_{i,k}^{\mathrm{ES}}$	ES index for $\tau_{i,k}$ as a result of TDA phase
$X_{i,k}^{\mathrm{tmp}}$	Temporary variable for $X_{i,k}$
$P_{i,k}^{\mathrm{CR}}$	Cost-effectiveness parameter used in CR phase for $\tau_{i,k}$
P_j^{CR}	Cost-effectiveness parameter used in CR phase for ES j
A^{del}	Set of candidate tasks for deallocation
A^{union}	Union of two sets ($A_0 \bigcup A^{\mathrm{del}}$)

The access probability of a version of a channel is the probability that an average viewer will be watching it. Let $p_{i,k}$ be the access probability of version $V_{i,k}$, $(i=1,...,N^{\mathrm{ch}}$ and $k=1,...,N^{\mathrm{ver}})$, where $\sum_{i=1}^{N^{\mathrm{ch}}}\sum_{k=1}^{N^{\mathrm{ver}}} p_{i,k}=1$. The probability for each channel can be derived from its current view counts. Let $Q_{i,k}$ be an index of the video quality of version $V_{i,k}$. Values of $Q_{i,k}$ can be obtained using video quality measurement tools [34], [35], or they can be predicted [36]. We use the video multi-method assessment fusion (VMAF) index developed by Netflix, which is known [35] to express QoE more accurately than other video quality indices. With the aim of providing higher quality for more popular channels, we introduce popularity-weighted video quality (PWQ) for version $V_{i,k}$, $Q_{i,k}^{\mathrm{pwq}}$ which we define as follows:

$$Q_{i,k}^{\mathrm{pwq}} = Q_{i,k}p_{i,k}. \tag{1}$$

We use the binary variable $X_{i,k}$ to indicate whether version $V_{i,k}$, $(i=1,...,N^{\mathrm{ch}}$ and $k=1,...,N^{\mathrm{tran}})$ of a video stream is transcoded or not. If $\tau_{i,k}$ is executed, then $X_{i,k}=1$; otherwise, $X_{i,k}=0$. In live-streaming, each viewer's device can be expected to request the highest-bitrate version that can be supported by that device and the current network situation. As described in many previous studies of bitrate selection algorithms such as Pensieve and BOLA, choosing a high bitrate drains the playback buffer, causing frequent rebuffering [37], [38]. Rebuffering is also known to be a major factor in reducing QoE. For example, subjective

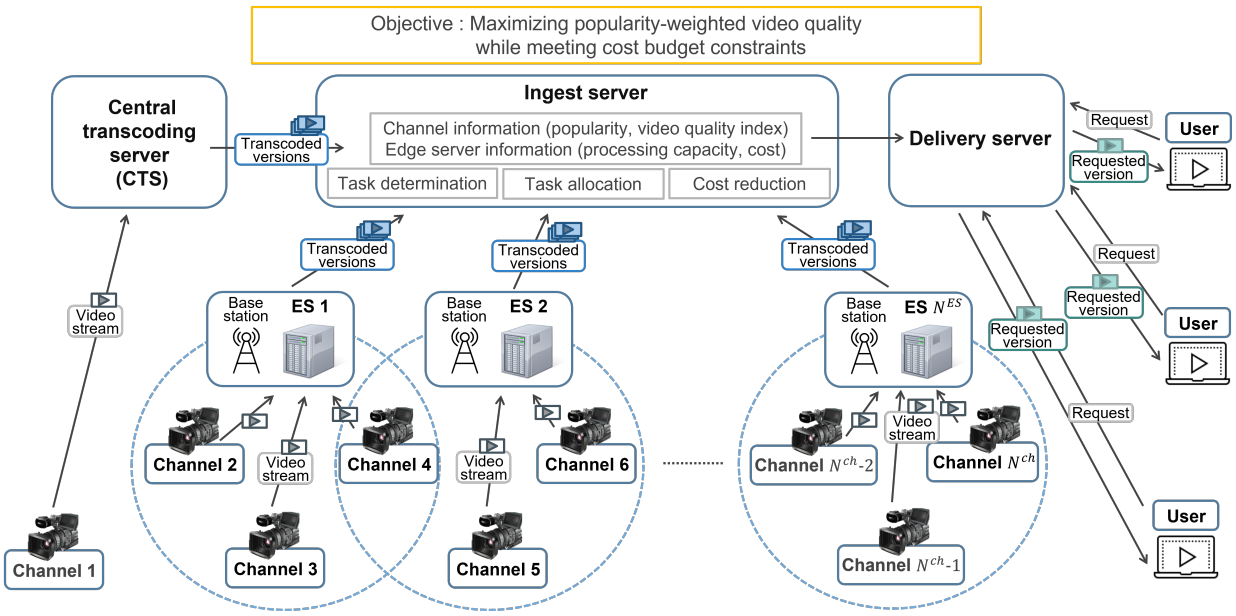


Fig. 1. Live-streaming scenario.

QoE analysis shows that the length and frequency of rebuffering can degrade QoE greatly [39], [40], [41]. To avoid this in any case, every video must be transcoded to the lowest bitrate offered by its channel.

If the version with the kth bitrate is not available because it has not been transcoded, then the available version with the highest bitrate lower than the requested bitrate is transmitted. If the bitrate requested is k, then we call this version of the video stream $H_{i,k}$, and it can be found as follows:

$$H_{i,k} = \begin{cases} \arg\max_{m=1,...,k} X_{i,m} = 1, & \text{if } k = 1, ..., N^{\text{tran}} \\ V_{i,N^{\text{ver}}}, & \text{if } k = N^{\text{ver}}. \end{cases} \tag{2}$$

Since $H_{i,k}$ is determined by the combination of $X_{i,k}$ values, the total PWQ for channel i can be calculated as follows:

$$Q_i^{\text{ch}}(X_{i,1}, ..., X_{i,N^{\text{tran}}}) = \sum_{k=1}^{N^{\text{ver}}} Q_{i,H_{i,k}}^{\text{pwq}}. \tag{3}$$

3.3 Costing the use of edge-servers

The live-streaming provider must pay to use edge-servers. We primarily consider a linear cost model, in which cost is proportional to CPU usage [42], [43], [44]. Let α_j be the normalized cost changing-rate for CPU usage of ES j, calculated as a proportion of that of the most expensive ES. The cost $C_j^{\text{cost}}(u)$ to use ES j is then proportional to the normalized CPU usage u, where $0 \leq u \leq U_j^{\max}$, and can be expressed as follows:

$$C_j^{\text{cost}}(u) = \alpha_j u. \tag{4}$$

We also consider an on-off cost model [45], [46], in which the cost of using an edge-server is determined by whether or not it is used during a given period, regardless of the amount of processing that is performed. If C_j^{onoff} be the normalized cost of using ES j, calculated as a proportion of that of the most expensive ES during some period, then $C_j^{\text{cost}}(u)$ can be determined as follows:

$$C_j^{\text{cost}}(u) = \begin{cases} 0 & u = 0 \\ C_j^{\text{onoff}} & \text{otherwise.} \end{cases} \tag{5}$$

4 Problem Formulation

We now introduce $Y_{i,k}$, which is the index of transcoding server on which each task $\tau_{i,k}$ is executed on TS j. If a TS is not available and $X_{i,k} = 0$, then $Y_{i,k}$ is set to -1. The goal is then to find values of $X_{i,k}$ and $Y_{i,k}$ for each task $\tau_{i,k}$ that maximize the overall PWQ under the following three constraints:

- **C1:** (Processing capacity constraint) The total CPU usage required to run all the tasks on TS j must not exceed its maximum processing capacity $U_j^{\max}$, as follows:

$$\sum_{\forall \tau_{i,k},\ Y_{i,k}=j} U_{i,k} \leq U_j^{\max},\ (j = 0, ..., N^{\text{ES}}). \tag{6}$$

- **C2:** (Network coverage constraint) Each transcoding task must be executed on an edge-server within the network coverage of the channel or the central transcoding server, as follows:

$$\forall \tau_{i,k},\ \text{where } X_{i,k} = 1, C_{i,Y_{i,k}}^{\text{net}} = 1. \tag{7}$$

- **C3:** (Cost budget constraint) The total cost incurred must not exceed the live-streaming service provider's cost budget, C^{budget}, as follows:

$$\sum_{j=1}^{N^{\text{ES}}} C_j^{\text{cost}}\Big(\sum_{\forall \tau_{i,k},\ Y_{i,k}=j} U_{i,k}\Big) \leq C^{\text{budget}}. \tag{8}$$

We can now formulate the transcoding task allocation problem (TTAP) that finds the values of $X_{i,k}$ and $Y_{i,k}$ for each task $\tau_{i,k}$ with the aim of maximizing total PWQ as follows:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{N^{\text{ch}}} Q_i^{\text{ch}}(X_{i,1}, ..., X_{i,N^{\text{tran}}}) \\ \text{Subject to} \quad & \forall \tau_{i,k} \text{ where } X_{i,k} = 1, C_{i,Y_{i,k}}^{\text{net}} = 1 \\ & \textstyle\sum_{\forall \tau_{i,k},\ Y_{i,k}=j} U_{i,k} \leq U_j^{\max},\ (j = 0, ..., N^{\text{ES}}) \\ & \textstyle\sum_{j=1}^{N^{\text{ES}}} C_j^{\text{cost}}\big(\sum_{\forall \tau_{i,k},\ Y_{i,k}=j} U_{i,k}\big) \leq C^{\text{budget}}. \end{aligned}$$

The TTAP can be reduced to the multiple-choice multiple-knapsack problem with assignment restriction (MMPAR), which

is known to be NP-hard [47]. In the MMPAR, there are classes consisting of a finite number of items, each of which requires a capacity and returns a profit. Also, there are multiple knapsacks, each with a capacity constraint, with assignment restrictions between classes and knapsacks. The objective of the MMPAR is to select one item from each class and then to allocate that item to a knapsack with the aim of maximizing total profit while satisfying knapsack capacity limits and assignment restrictions.

In the TTAP, the connection between each edge-server and each channel may be limited due to network coverage (assignment restriction). Each edge-server has processing capacity and cost budget constraints (knapsack capacity constraint). A combination of transcoding tasks (items) must be selected from each channel (class), and each task needs to be allocated to one of edge-servers (knapsack) with the aim of maximizing total PWQ. Therefore, the TTAP problem can be reduced to the MMPAR, which is NP-hard.

5 ALGORITHM

5.1 Algorithm Concept

If the number of channels and edge-servers will be sufficiently large, then it is not feasible to enumerate all combinations of $X_{i,k}$ and $Y_{i,k}$, eliminating those that do not meet conditions (**C1**,**C2** and **C3**), and selecting the remaining combination with highest PWQ. Therefore, to satisfy these constraints progressively with low computational overhead, we introduce a heuristic with two phases: task determination and allocation (TDA) and cost reduction (CR).

The TDA phase first determines a set of candidate tasks to be executed in order to satisfy the constraint **C1**, and then these tasks are assigned to the appropriate ESs to satisfy the constraint **C2** while taking cost into account. The CR phase is further divided into task removal and redistribution steps to satisfy the constraint **C3**. For this purpose, we introduce a cost-effectiveness parameter for each task allocated in the TDA phase, allowing cost-effective tasks to run first. All these steps are executed differently taking into account the characteristics of different cost models. In more detail:

1) The TDA phase chooses the tasks to be run and assigns them to appropriate TSs with adequate processing capacity **C1** and network coverage **C2**, while maximizing $\sum_{i=1}^{N^{\mathrm{ch}}} Q_i^{\mathrm{ch}}(X_{i,1},...,X_{i,N^{\mathrm{tran}}})$. To achieve this, it chooses candidate tasks in a greedy way, giving higher priority to tasks with a higher ratio of PWQ to CPU requirement, and then allocates them to edge-servers, taking processing cost into account.
2) The CR phase progressively removes the task allocations made in the first phase until **C3** is satisfied, while minimizing the concomitant loss in PWQ due to versions not being transcoded. This step returns values of $X_{i,k}$ and $Y_{i,k}$ which satisfy all three constraints **C1**,**C2** and **C3**.

5.2 Task determination and allocation (TDA) phase

The output of the TDA phase is an array A_j, $(j = 0,...,N^{\mathrm{ES}})$ which specifies the task assigned to each TS j. This assignment must satisfy constraint **C1** and **C2**. We introduce a temporary binary variable $X_{i,k}^{\mathrm{tmp}}$ which holds temporary values of $X_{i,k}$ for each task $\tau_{i,k}$ during algorithm execution; if the execution of $\tau_{i,k}$ is scheduled, then $X_{i,k}^{\mathrm{tmp}} = 1$; otherwise, $X_{i,k}^{\mathrm{tmp}} = 0$. A further variable $D_{i,k}^{\mathrm{task}}$ expresses the effect on PWQ of eliminating $\tau_{i,k}$ from the set of tasks to be executed. $D_{i,k}^{\mathrm{task}}$ can be expressed in terms of $X_{i,k}^{\mathrm{tmp}}$ as follows:

$$D_{i,k}^{\mathrm{task}} = Q_i^{\mathrm{ch}}(X_{i,1}^{\mathrm{tmp}},...,X_{i,N^{\mathrm{tran}}}^{\mathrm{tmp}}) - Q_i^{\mathrm{ch}}(X_{i,1}^{\mathrm{tmp}},...,X_{i,k-1}^{\mathrm{tmp}},0,X_{i,k+1}^{\mathrm{tmp}},...X_{i,N^{\mathrm{tran}}}^{\mathrm{tmp}}).$$

All the values of $X_{i,k}^{\mathrm{tmp}}$ are initialized to 1, so $D_{i,k}^{\mathrm{task}}$ initially represents the change in PWQ caused by eliminating $\tau_{i,k}$, alone. We can now determine $R_{i,k}^{\mathrm{TDA}}$, which is the ratio of $D_{i,k}^{\mathrm{task}}$ to $U_{i,k}$ for each task $\tau_{i,k}$, as follows:

$$R_{i,k}^{\mathrm{TDA}} = \frac{D_{i,k}^{\mathrm{task}}}{U_{i,k}}. \tag{9}$$

The higher the corresponding value of $D_{i,k}^{\mathrm{task}}$, the less desirable it is to eliminate $\tau_{i,k}$, because of the large impact on PWQ. Therefore, tasks with higher values of $R_{i,k}^{\mathrm{TDA}}$ are allocated first.

During task allocation, a parameter $P_{i,j,k}^{\mathrm{TDA}}$ is maintained which expresses the feasibility and (if feasible) the desirability of allocating task $\tau_{i,k}$ to each ES j. The value of $P_{i,j,k}^{\mathrm{TDA}}$ is a combined expression of remaining processing capacity of ES j and its cost when assigning task $\tau_{i,k}$ to ES j, and is defined differently depending on the cost model:

- Linear cost model:

$$P_{i,j,k}^{\mathrm{TDA}} = \frac{U_j^{\max} - \sum_{\forall \tau_{m,n} \in A_j \bigcup \tau_{i,k}} U_{m,n}}{C_j^{\mathrm{cost}}(\sum_{\forall \tau_{m,n} \in A_j \bigcup \tau_{i,k}} U_{m,n})}. \tag{10}$$

  Using the linear cost model, the numerator of the $P_{i,j,k}^{\mathrm{TDA}}$ of this expression represents the remaining processing capacity when the task $\tau_{i,k}$ is assigned, while the denominator is the cost that will be incurred by currently allocated tasks. To increase the CPU utilization of edge-servers cost-efficiently, tasks with higher values of $D_{i,k}^{\mathrm{task}}$ should be allocated to the ES with the lowest processing cost. it is thus sufficient to divide the remaining processing capacity by the cost incurred by the task allocation when the $\tau_{i,k}$ task is allocated.
- On-off cost model:

$$P_{i,j,k}^{\mathrm{TDA}} = \frac{\sum_{\forall \tau_{m,n} \in A_j \bigcup \tau_{i,k}} U_{m,n}}{C_j^{\mathrm{cost}}(\sum_{\forall \tau_{m,n} \in A_j \bigcup \tau_{i,k}} U_{m,n})}. \tag{11}$$

  When the on-off cost model is in use, assigning one task to an ES incurs the whole of the fixed cost. Thus it is disadvantageous to spread the execution of tasks across ESs. In Equation (11), the numerator of the expression for $P_{i,j,k}^{\mathrm{TDA}}$ represents the processing capacity already allocated when the task $\tau_{i,k}$ is assigned. Using this model, tasks are preferentially allocated to ESs with higher values of $P_{i,j,k}^{\mathrm{TDA}}$ to make the best use of those low-cost ESs.

Having set up these structures, the TDA phase runs in four steps, as follows:

1) Since the lowest-bitrate versions must always be transcoded, the corresponding tasks $\tau_{i,1}$, $(i = 1,...,N^{\mathrm{ch}})$ are allocated to the ES j first where $C_{i,j}^{\mathrm{net}} = 1$. For this purpose, this phase finds the highest value of $R_{i,1}^{\mathrm{TDA}}$ from S^{R}, which is a set of $R_{i,k}^{\mathrm{TDA}}$ values, and then allocates the corresponding task successively to the ES with the highest value of $P_{i,j,k}^{\mathrm{TDA}}$. However, if the cost budget is very small and exceeds this, the task $\tau_{i,1}$ is assigned to the ES with the next highest value of $P_{i,j,k}^{\mathrm{TDA}}$, but if there is no ES available, the task $\tau_{i,1}$ is assigned to the CTS.

2) The task with the highest value of $R_{i,k}^{\mathrm{TDA}}$, $(k \neq 1)$ is selected from S^{R}. Then the corresponding task $\tau_{i,k}$ is allocated to the ES j with the highest value of $P_{i,j,k}^{\mathrm{TDA}}$ within the network coverage of the channel associated with that task.
3) If there is no ES with adequate processing capacity remaining within that coverage, then this task is assigned to CTS. If there is no remaining processing capacity at the CTS, then this task cannot be allocated to any server.
4) The $R_{i,k}^{\mathrm{TDA}}$ value associated with this task is removed from S^{R} and steps 2 to 3 are repeated until S^{R} is empty.

We can finally build an array of tasks assigned to TS j, A_j as a result of the TDA phase. The results of this algorithm are stored in an array A_j $(j = 0, ..., N^{\mathrm{ES}})$. Each element of A_j contains the set of tasks assigned to the corresponding TS. Algorithm 1 shows the details of the TDA phase.

The TDA phase repeatedly finds the highest value of $R_{i,k}^{\mathrm{TDA}} \in S^{\mathrm{R}}$ (lines 4–9). Since the number of elements in S^{R} is $N^{\mathrm{ch}}N^{\mathrm{tran}}$, sorting this set requires $O(N^{\mathrm{ch}}N^{\mathrm{tran}}log(N^{\mathrm{ch}}N^{\mathrm{tran}}))$ time complexity when merge sorting is used. Next, the selected task is assigned to one of the edge-servers with the highest value of $P_{i,j,k}^{\mathrm{TDA}}$ (lines 10–20). Again, since the number of $P_{i,j,k}^{\mathrm{TDA}}$ is at most N^{ES}, sorting requires $O(N^{\mathrm{ES}}logN^{\mathrm{ES}})$ time complexity when merge sorting is used. Overall, the time complexity of the TDA phase is calculated as: $O(N^{\mathrm{ch}}N^{\mathrm{ES}}N^{\mathrm{tran}}log(N^{\mathrm{ch}}N^{\mathrm{tran}})logN^{\mathrm{ES}})$.

Algorithm 1: Task determination and allocation (TDA) phase.

Input: S^{R}: Set of $R_{i,k}^{\mathrm{TDA}}$ values;
Output: A_j, $(j = 0, ..., N^{\mathrm{ES}})$;
1 Temporary variables: $X_{i,k}^{\mathrm{tmp}}$, $(i = 1, ..., N^{\mathrm{ch}}$ and $k = 1, ..., N^{\mathrm{tran}})$;
2 All the values of $X_{i,k}^{\mathrm{tmp}}$ are initialized to 1 ;
3 **while** $S^{\mathrm{R}} \neq \phi$ **do**
4 **if** $\exists i$, $R_{i,1}^{\mathrm{TDA}} \in S^{\mathrm{R}}$ **then**
5 Find the task $\tau_{i,1}$ with the highest value of $R_{i,1}^{\mathrm{TDA}} \in S^{\mathrm{R}}$ for which $i = W$ and $1 = M$, and remove $R_{W,1}^{\mathrm{TDA}}$ from S^{R};
6 **end**
7 **else**
8 Find the task $\tau_{i,k}$, $(k \neq 1)$ with the highest value of $R_{i,k}^{\mathrm{TDA}} \in S^{\mathrm{R}}$ for which $i = W$ and $k = M$, and remove $R_{W,M}^{\mathrm{TDA}}$ from S^{R};
9 **end**
10 Find the ES index j, $(j = 1, ..., N^{\mathrm{ES}})$ with the highest value of $P_{W,j,M}^{\mathrm{TDA}}$ where $\sum_{\forall \tau_{i,k} \in A_j} U_{i,k} + U_{W,M} \leq U_j^{\max}$ and $C_{W,j}^{\mathrm{net}} = 1$, for which $j = L$;
11 **if** *ES* L *exists* **then**
12 $A_L = A_L \bigcup \{\tau_{W,M}\}$;
13 **end**
14 **else if** $\sum_{\forall \tau_{i,k} \in A_0} U_{i,k} + U_{W,M} \leq U_0^{\max}$ **then**
15 $A_0 = A_0 \bigcup \{\tau_{W,M}\}$;
16 **end**
17 **else**
18 $X_{i,k}^{\mathrm{tmp}} \leftarrow 0$;
19 **end**
20 **end**

5.3 Cost reduction (CR) phase

If the constraint **C3** is not satisfied during the TDA phase, then some of the allocations of tasks to ESs must be removed while attempting to produce the least possible reduction in overall PWQ. This deallocation problem can be reduced to a variant of the multiple knapsack problem in which overall profit is maximized subject to the capacity constraint of the knapsacks [48]. In our context, a knapsack is an ES and profit is PWQ. We use a greedy technique to find a candidate set of tasks that can be removed from a set of tasks, A^{ES} assigned only to ES in the TDA phase, where $A^{\mathrm{ES}} = A_1 \cup ... \cup A_{N^{\mathrm{ES}}}$.

However, the PWQ to CPU utilization ratios of the servers to which these tasks are assigned may be greater than the ratio that applies to the central transcoding server. In this case, these removed tasks can be redistributed to the central transcoding server. Details of task removal and redistribution follow, and are also shown in Algorithm 2.

5.3.1 Task removal step

This step differs with the cost model:

- Linear cost model: The cost of using each ES increases with processor usage. For ease of exposition, we introduce variables $I_{i,k}^{\mathrm{ES}}$ to indicate the ES index for $\tau_{i,k}$, $(\tau_{i,k} \in A^{\mathrm{ES}}, k \neq 1)$ after the TDA phase. We also introduce a cost-effectiveness parameter $P_{i,k}^{\mathrm{CR}}$ for each task, $\tau_{i,k}$ as follows:

$$P_{i,k}^{\mathrm{CR}} = \frac{D_{i,k}^{\mathrm{task}}}{C_{I_{i,k}^{\mathrm{ES}}}^{\mathrm{cost}}(U_{i,k})}, \tag{12}$$

  where the numerator and denominator respectively represent the decrease in PWQ and cost that result from removing task $\tau_{i,k}$ from $A_{I_{i,k}^{\mathrm{ES}}}$. The task with the lowest value of $P_{i,k}^{\mathrm{CR}}$ is added to a set A^{del} which contains candidates for deallocation, and this process is repeated until $\sum_{j=1}^{N^{\mathrm{ES}}} C_j^{\mathrm{cost}}(\sum_{\forall \tau_{i,k} \in A_j} U_{i,k}) \leq C^{\mathrm{budget}}$.
- On-off cost model: Using this model, we need to eliminate the use of one or more ESs altogether to satisfy **C3**, which will result in the deletion of all the tasks that have been allocated to ES j. Then cost-effectiveness parameters of P_j^{CR}, $(j = 1, ..., N^{\mathrm{ES}})$ can be determined for each ES j, $(j = 1, ..., N^{\mathrm{ES}})$ as follows:

$$P_j^{\mathrm{CR}} = \frac{\sum_{\forall \tau_{i,k} \in A_j} D_{i,k}^{\mathrm{task}}}{C_j^{\mathrm{cost}}(\sum_{\forall \tau_{i,k} \in A_j} U_{i,k})}. \tag{13}$$

  Obviously, the ESs with the lowest values of P_j^{CR} are then eliminated one by one until $\sum_{j=1}^{N^{\mathrm{ES}}} C_j^{\mathrm{cost}}(\sum_{\forall \tau_{i,k} \in A_j} U_{i,k}) \leq C^{\mathrm{budget}}$. A set, A^{del} of candidate tasks for deallocation can then be allocated to the deleted ESs.

5.3.2 Task redistribution step

A new set of tasks to run on the CTS is now determined from the union of the set A_0 of tasks found in phase 1, and the set A^{del} of tasks deallocated from ESs. To achieve this, A^{del} is replaced by $A^{\mathrm{del}} \bigcup A_0$, and then A_0 is replaced with the subset of the new A^{del} which consists of tasks that transcode lowest-bitrate versions: $A^{\mathrm{del}} = A^{\mathrm{del}} \bigcup A_0$, and then $A_0 = \{\tau_{i,1} | \forall \tau_{i,1} \in A^{\mathrm{del}}\}$.

The overall PWQ is then increased within the capacity limit of the CTS by repeatedly moving the task with the highest value of

Algorithm 2: Cost reduction (CR) phase.

Input: C^{budget}, A_j $(j = 0, ...N^{\text{ES}})$, $I^{\text{ES}}_{i,k}$, $V_{i,k}$ and $D^{\text{task}}_{i,k}$ $(i = 1, ..., N^{\text{ch}}$ and $k = 1, ..., N^{\text{tran}})$;
Output: $X_{i,k}$ and $Y_{i.k}$, $(i = 1, ..., N^{\text{ch}}$ and $k = 1, ..., N^{\text{tran}})$;
Sets of tasks: A^{del} and A^{ES}, ($A^{\text{del}} \leftarrow \phi$ and $A^{\text{ES}} \leftarrow A_1 \bigcup ... \bigcup A_{N^{\text{ES}}}$) ;
/* *Task removal step* */;
if *Linear cost model is used* **then**
  while $A^{\text{ES}} \neq \phi$ **do**
    Find the lowest value of $P^{\text{CR}}_{i,k}$ for which $i = L$ and $k = M$;
    $A_{I_{L,M}} \leftarrow A_{I_{L,M}} - \{\tau_{L,M}\}$;
    $A^{\text{ES}} \leftarrow A^{\text{ES}} - \{\tau_{L,M}\}$;
    $A^{\text{del}} \leftarrow A^{\text{del}} \bigcup \{\tau_{L,M}\}$;
    if $\sum_{j=1}^{N^{\text{ES}}} C^{\text{cost}}_j (\sum_{\forall \tau_{i,k} \in A_j} U_{i,k}) \leq C^{\text{budget}}$ **then**
      Break the loop;
    end
  end
end
if *On-off cost model is used* **then**
  while $A^{\text{ES}} \neq \phi$ **do**
    Find the ES index j with the lowest value of P^{CR}_j for which $j = L$;
    $A_L = \phi$;
    $A^{\text{ES}} \leftarrow A^{\text{ES}} - A_L$;
    $A^{\text{del}} \leftarrow A^{\text{del}} \bigcup A_L$;
    if $\sum_{j=1}^{N^{\text{ES}}} C^{\text{cost}}_j (\sum_{\forall \tau_{i,k} \in A_j} U_{i,k}) \leq C^{\text{budget}}$ **then**
      Break the loop;
    end
  end
end
/* *Task redistribution step* */;
$A^{\text{del}} \leftarrow A^{\text{del}} \bigcup A_0$;
$A_0 \leftarrow \{\tau_{i,1} | \forall \tau_{i,1} \in A^{\text{del}}\}$;
while $\sum_{\forall \tau_{i,k} \in A_0} U_{i,k} \leq U^{\max}_0$ **do**
  Find a task $\tau_{i,k} \in A^{\text{union}}$ with highest value of $\frac{Q^{\text{pwq}}_{i,k}}{U_{i,k}}$ for which $i = L$ and $k = M$;
  $A_0 \leftarrow A_0 \bigcup \{\tau_{L,M}\}$;
end
/* *Finalization* */;
for $i = 1$ *to* N^{ch} **do**
  for $k = 1$ *to* $N^{\text{ver}} - 1$ **do**
    if $\tau_{i,k} \in A^{\text{ES}} \bigcup A_0$ **then**
      $X_{i,k} \leftarrow 1$;
      if $\tau_{i,k} \in A_j$ **then**
        $Y_{i,k} \leftarrow j$;
      end
    else
      $X_{i,k} \leftarrow 0$;
      $Y_{i,k} \leftarrow -1$;
    end
  end
end

TABLE 2
Resolutions, bitrates and VMAF values used in simulations [23], [51], [52].

Resolution	Bitrate (kbps)	Mean VMAF
400×224	200, 400, 600	40, 60, 72
640×360	1000, 1500	90, 92.5
1280×720	2000	95
1920×1080 (Original)	2750	100

$\frac{Q^{\text{pwq}}_{i,k}}{U_{i,k}}$ from A^{del} to A_0. Only the tasks with the lowest values of $\frac{Q^{\text{pwq}}_{i,k}}{U_{i,k}}$ remain in A^{del} for de-allocation.

When the task redistribution step has finished, the arrays $X_{i,k}$ and $Y_{i,k}$ can be filled in from the sets A^{ES} and A_0, and the array A_j, $(j = 0, ..., N^{\text{ES}})$, as follows:

- If task $\tau_{i,k}$ belongs to $A^{\text{ES}} \bigcup A_0$, then $X_{i,k}$ is set to 1; otherwise, it is set to 0.
- If task $\tau_{i,k}$ belongs to A_j, $(j = 0, ..., N^{\text{ES}})$, then $Y_{i,k}$ is set to j; otherwise, it is set to -1.

The task removal step involves sorting of elements in set A^{ES} in the linear cost model, which requires $O(n(A^{\text{ES}})logn(A^{\text{ES}}))$. However, in the on-off cost model, it repeatedly finds the edge-server index with the lowest value of P^{CR}_j, which incurs $O(N^{\text{ES}} log N^{\text{ES}})$. The redistribution step requires the elements in set A^{union} to be sorted, with a time complexity of $O(n(A^{\text{union}})logn(A^{\text{union}}))$.

6 Experimental Results

6.1 Simulation Setup

We perform simulations to evaluate our scheme in terms of PWQ values, under the cost limit. Next, we describe the simulation environment used, such as live-streaming, edge-servers, and task models.

6.1.1 Live-streaming model

By default, N^{ch} is set to 6,000, which is the median number of channels simultaneously broadcast on Twitch.tv [51]. Table 2 shows the set of bitrate versions with 7 different resolutions that we considered. These bitrates are recommended by Zencoder, a public transcoding cloud provider [23], [51], [52]. The VMAF values are modeled by normal distributions, with the mean values shown in Table 2 [23], [52]. The popularity of each channel is modeled by a gamma distribution with $k = 0.399$ and $\theta = 14260$, values which were observed from Twitch.tv [53].

The relative popularity of each bitrate of streaming video can be modeled as a normal distribution with parameters of mean μ and variance σ [19]. Based on this, we consider four scenarios, with $\sigma = 1$ and values of μ determined as follows [19], [20]: when higher-bitrate versions are popular (HVP), μ is N^{ver}; when medium-bitrate versions are popular (MVP), μ is $\frac{1+N^{\text{ver}}}{2}$; when lower-bitrate versions are popular (LVP), μ is 1; when random versions are popular (RVP), and μ varies between 1 and N^{ver}. We use the MVP scenario by default, because medium-bitrate versions are the most commonly requested in practice [19].

6.1.2 Edge-server and task models

To determine the locations of the edge-servers, we consider real cellular network location data from Melbourne, Australia, which includes the latitudes and longitudes of 816 wireless access points

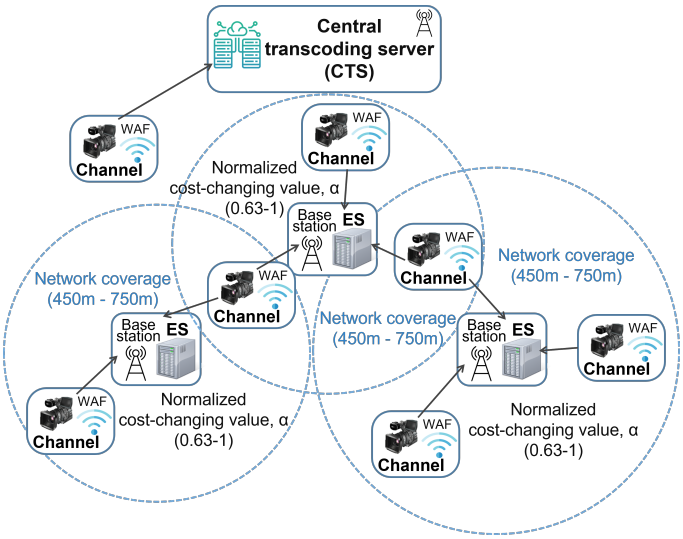


Fig. 2. Network topology used for simulation based on real cellular networks [14], [15].

TABLE 3
Edge-server specification [54].

Server	Processor
Asus RS620SA-E10-RS12	2.25GHz, 384cores
Dell PowerEdge R7525	2.00GHz, 512cores
HPE Apollo XL225n Gen10 Plus	2.45GHz, 128cores
Fujitsu Primergy RX4770 M6	2.9GHz, 112cores
Lenovo Technology ThinkSystem SR665	2.45GHz, 128cores

(WAPs) and 125 base stations [14], [15]. One edge-server is assumed to be connected to each base station with network coverage randomly chosen between 450m and 750m [14], [15]. A channel which connects to a WAP can then use edge-servers within its network coverage. Fig. 2 shows the network topology used for the simulation.

We base the CPU characteristics of edge-servers on those of 5 commercial servers released in 2020 or 2021, shown in Table 3 [54]. A regression model from [51] is built based on the measured data to relate transcoding bitrates to CPU usage [51]. The output of this model [51] is used to derive the values of CPU usage. These values are randomly perturbed by between -5% and +5% to reflect changes in different video types [23].

We obtain the normalized cost-changing values from six cost models (c4, c5n, c5a, c5, c6gn, and c6g) suitable for media transcoding on AWS Wavelength, an edge computing service [42], [49], [50], where the α_j is found to be between 0.63 and 1. The value of α_j is randomly chosen in this range.

We compare our scheme against two task selection methods that prioritize popular channels: in the 'all popular channels' (AP) scheme, the channels are split into two groups, with versions at all bitrates transcoded for those in the popular group, but only the lowest-bitrate version for those in the unpopular group; in the 'highest popularity first' (HPF) scheme, the lowest-bitrate version is transcoded for all channels and then higher-bitrate versions are transcoded, starting with the most popular, until the cost budget is exhausted.

We consider three benchmark methods of allocating edge-server, which attempt to balance computational load and cost: the 'random allocation' (RA) scheme assigns each task to a randomly selected ES; and the 'lowest utilization first' (LUF) scheme assigns each task in turn to the ES with the lowest CPU utilization; the 'lowest cost first' (LCF) scheme tries to save cost by allocating each task to the ES on which the lowest cost has been incurred in a linear cost model, whereas in an on-off cost model, it chooses the cheapest ES and assigns tasks to it until the processing capacity of that ES is used up. Each of these schemes continues allocating tasks until the cost budget is exhausted.

We combine these benchmarks to create six different benchmark combinations of ES allocation and task selection: RA+AP, RA+HPF, LUF+AP, LUF+HPF, LCF+AP and LCF+HPF. We compare these with our two-phase algorithm, which we now refer to as TDA+CR, in terms of cost budget, number of channels and edge-servers, version popularity, and bitrate sets. The parameters used in the simulation are summarized in Table 4, with default values in bold.

TABLE 4
Parameters used in the simulation.

C^{ratio}	N^{ch}	N^{ES}	Popularity Distribution	Bitrate Setting Recommendations
20%	2,000	50	HVP	**Zencoder [23], [51], [52]**
30%	4,000	75	**MVP**	Youtube [55]
40%	**6,000**	**100**	LVP	Netflix [56]
50%	8,000	125	RVP	IBM Watson Media [57]
60%	10,000	150		D. Stohr et al. [58]

6.2 Effect of Cost Budget

We investigate the effect of cost budget on total PWQ when N^{ch} is 6,000 and N^{ES} is 100. For this purpose, we introduce a variable, C^{ratio} which is the ratio of the cost budget to the cost using each ES j as its maximum utilization $U_j^{\max}$ so that $C^{\text{ratio}} = \frac{C^{\text{budget}}}{\sum_{j=1}^{N^{\text{ES}}} C_j^{\text{cost}}(U_j^{\max})}$.

Fig. 3 shows how the total PWQ varies for different values of C^{ratio}. TDA+CR always achieves the best performance, yielding 5.12% to 69.79% (average 29.35%) more PWQ than the other methods. The gap usually tends to increase as C^{ratio} decreases, indicating that our method is effective in coping with a reduced cost budget. Except for ours, the best ES selection method is LCF in all cases, which is expected because it takes cost into account. In terms of task selection, the AP method is better than the HPF method, and the higher the budget, the greater the difference.

6.3 Effect of Number of Channels

Fig. 4 shows the total PWQ when the number of channels ranges between 2,000 and 10,000. Again, TDA+CR performs best, achieving 0.06% to 75.01% (average 24.31%) more PWQ than the other methods. In particular, it produces a PWQ that is 0.06% to 11.63% (average 6.5%) more than the LCF+HPF, the best among the benchmark schemes. As the number of channels N^{ch} increases, fewer high-bitrate versions are transcoded, so the PWQ drops. However, this effect is less with TDA+CR than with the comparative benchmark schemes.

6.4 Effect of Number of Edge-Servers

Fig. 5 shows how the total PWQ varies with N^{ES} when C^{ratio} is 50% and N^{ch} is 6,000. As expected, the total PWQ increases with N^{ES}, as more processing capacity becomes available for transcoding. TDA+CR shows the best performance at all values

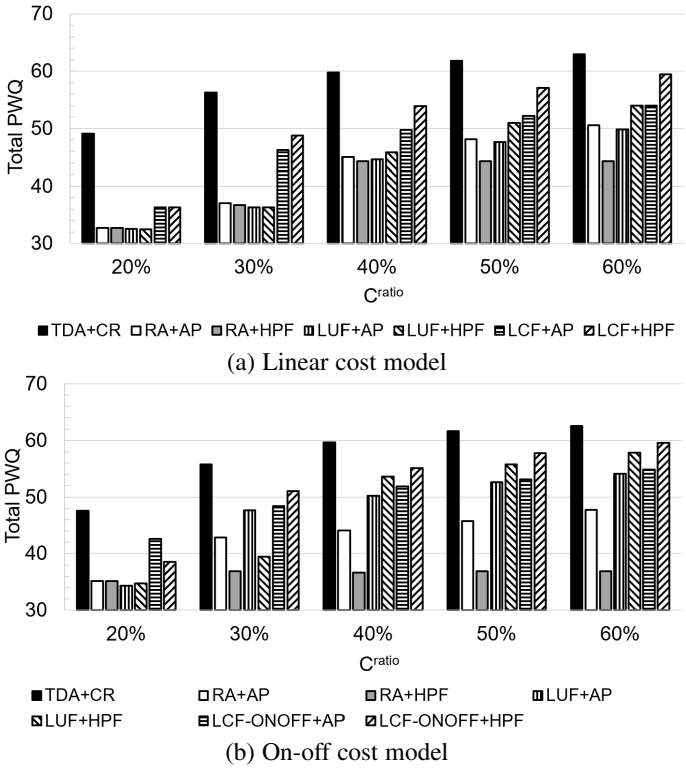


Fig. 3. Total PWQ against C^{ratio}, for our scheme and six benchmarks.

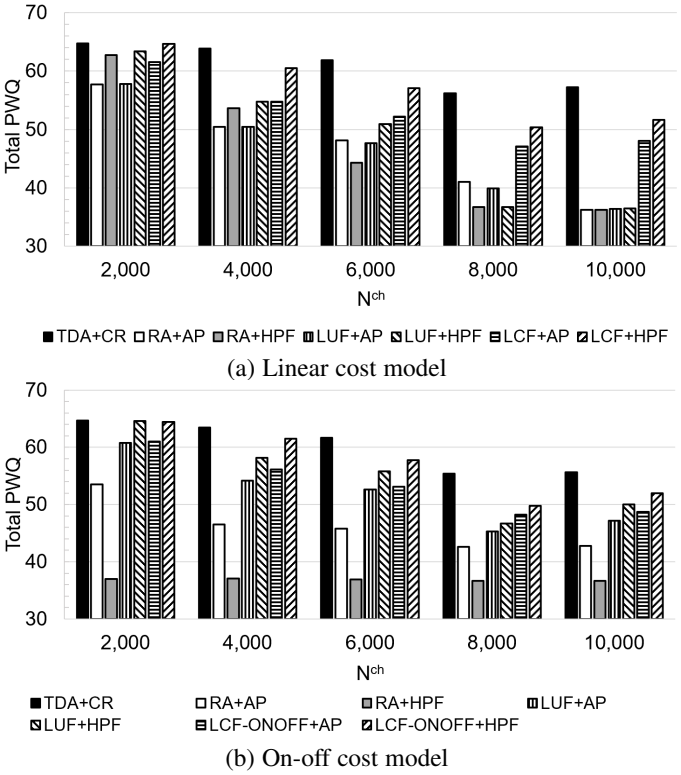


Fig. 4. Total PWQ against number of channels, for our scheme and six benchmarks.

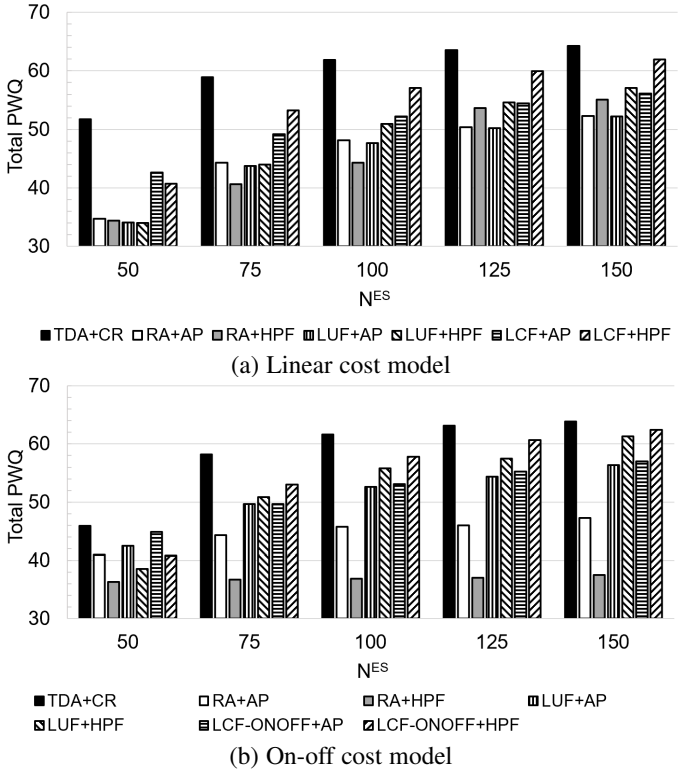


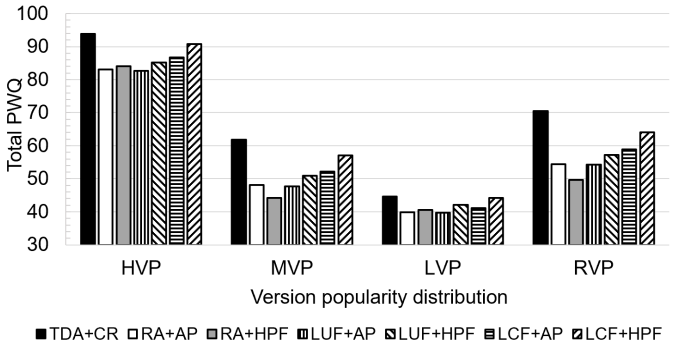
Fig. 5. Total PWQ against number of ESs, for our scheme and six benchmarks.

of N^{ES}, yielding 2.33% to 70.76% (average 24.19%) more PWQ than the other methods. This difference tends to become larger when there are fewer edge-servers, suggesting that our scheme is more effective, when the total processing capacity of the edge-servers is low.

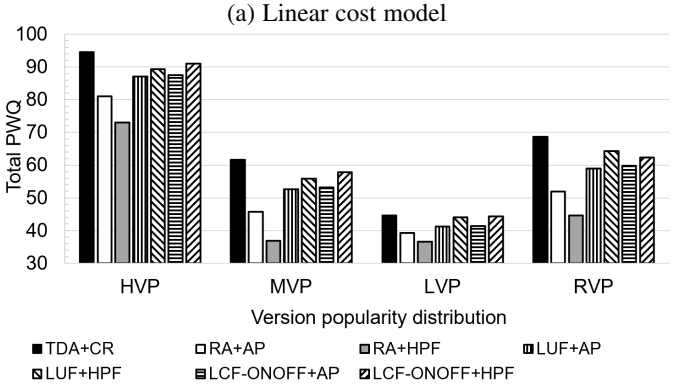
6.5 Effect of Version Popularity

Fig. 6 shows the total PWQ values for the HVP, MVP, LVP, and RVP popularity distributions, when C^{ratio} is 50%, N^{ch} is 6,000 and N^{ES} is 100. Again, TDA+CR outperforms all the other schemes, yielding 0.45% to 67.11% (average 17.09%) more PWQ. The performance difference between TDA+CR and other methods is highest in MVP and RVP cases (both are similar), and lowest in LVP case. The reasons for these results can be explained as follows:

- For LVP distributions where lower bitrate versions are popular, the difference in VMAF with bitrate is not high, so the video quality degradation due to inappropriate bitrate selection is not significant. This creates marginal PWQ differences among schemes.
- Differences in VMAF with bitrate tend to be higher in the medium bitrate versions, allowing more scope for optimization. TDA+CR distributes the transcoded version to each edge-server more efficiently than other methods, making better use of each server's available CPU processing capacity. Therefore, there is a large difference in PWQ compared to other methods in MVP distribution.
- Transcoding the high bitrate version requires higher CPU processing capacity compared to the low bitrate version. For HVP distribution where higher bitrate versions are popular, fewer versions are transcoded compared to MVP cases due

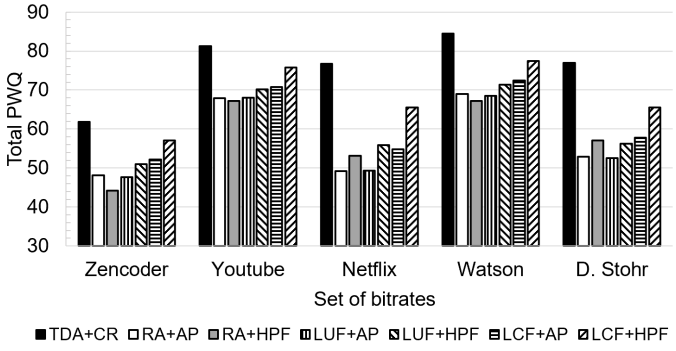


(a) Linear cost model

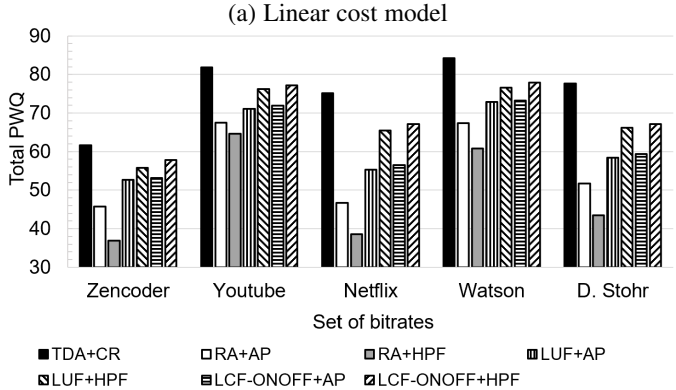


(b) On-off cost model

Fig. 6. Total PWQ against version popularity distribution, for our scheme and six benchmarks.



(a) Linear cost model



(b) On-off cost model

Fig. 7. Total PWQ against bitrate setting recommendations, for our scheme and six benchmarks.

TABLE 5
Recommended bitrates.

Source	N^{ver}	Resolution range	Bitrate range (kbps)
Youtube	11	426×240 ∼ 3840×2160	500 ∼ 35,000
Netflix	10	320×240 ∼ 1920×1080	235 ∼ 5,800
IBM Watson Media	7	480×270 ∼ 3840×2160	400 ∼ 11,000
D. Stohr et al.	9	480×270 ∼ 1920×1080	253 ∼ 4,000

to CPU processing capacity limitations on edge-servers. As a result, compared with MVP, the difference in PWQ between TDA+CR and other schemes is decreased.

6.6 Effect of Bitrate Setting Recommendations

Video streaming service providers recommend different sets of bitrates, and to examine this effect, we consider four sets of bitrate ranges, recommended by Youtube [55], Netflix [56], IBM Watson Media [57] and D. Stohr et al. [58], as listed in Table 5, in addition to the Zencoder bitrates of Table 2. Fig. 7 shows how the set of bitrates affects total PWQ when C^{ratio} is 50%, N^{ch} is 6,000 and N^{ES} is 100. TDA+CR scheme yields 5.95% to 94.62% (average 27.58%) more PWQ than the other schemes.

The performance of TDA+CR is especially good in the bitrate setting recommendations by D. Stohr et al. and Netflix. Their rather low bitrates seem to create a similar scenario to MVP in the previous section, which provides flexibility for optimization; whereas performance on the other sets of recommendations, with generally higher bitrates, is more dependent on the availability of processing power.

6.7 Effect of Cost Model

We apply TDA+CR with the two cost models under the same cost limitation, with the results shown in Fig. 8 when N^{ch} is 6000 and N^{ES} is 100. The linear cost model always results in higher PWQ than the on-off cost model. The difference is between 0.25% and 3.18% (average 1.03%), which increases as C^{ratio} decreases. This is because the on-off cost model forces the use of relatively few ESs, while the linear cost model allows the ES which is most suitable for a particular task to be allocated more frequently.

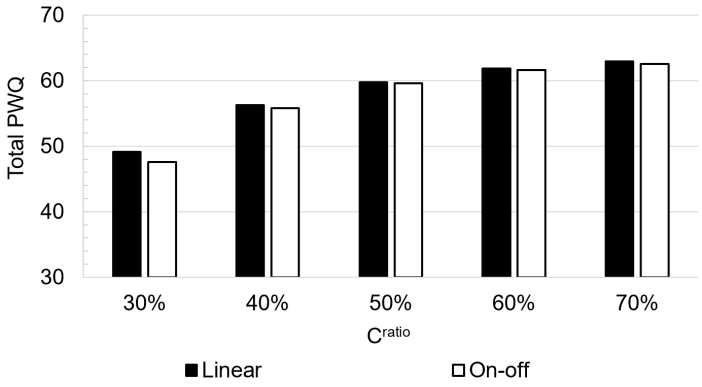


Fig. 8. Total PWQ for the linear and on-off cost models.

6.8 Execution Time Analysis

We measure algorithm execution times, averaged across five runs, on an Intel Xeon W-2235 CPU. Table 6 shows the results for different values of C^{ratio}, N^{ch} and N^{ES}. The execution time ranges between 6.4 ms and 41.4 ms. It is hardly affected by the values of C^{ratio}, but increases with N^{ch} and N^{ES} because the allocation problem addressed by the greedy algorithm is larger.

TABLE 6
Average execution times for our scheme (ms).

Effect of the cost budget					
C^{ratio}	20%	30%	40%	50%	60%
Linear	29.2	28.8	27.2	26.6	25.8
On-off	26.6	25.4	23.4	22.8	22.2
Effect of the number of channels					
N^{ch}	2,000	4,000	6,000	8,000	10,000
Linear	8.2	19	26.6	33.2	41.4
On-off	6.4	15	22.8	30.2	38.2
Effect of the number of edge-servers					
N^{ES}	50	75	100	125	150
Linear	16	21.4	26.6	33.4	38.4
On-off	15.6	18	22.8	29.4	33.4

6.9 Comparison between TDA+CR and Optimal Solution

Since TTAP is NP-hard, it is impossible to derive an optimal solution in a reasonable amount of time. In particular, in the case of a live streaming environment with a large number of channels and edge servers, it is impossible to derive the optimal solution by examining all combinations. In order to verify whether TDA+CR produces results close to optimum, we reduce the workload size (number of channels and edge servers) and compare TDA+CR with the method of examining all combinations. Table. 7 shows the percentage difference between TDA+CR and optimum according to the values of N^{ch} when $N^{\text{ES}} = 10$, which implies that TDA+CR produces a near-optimal solution for small workloads.

TABLE 7
Percentage difference in total PWQ between TDA+CR and optimum against N^{ch}.

N^{ch}	100	200	300	400
Linear	0%	0%	0.01%	0.01%
On-off	0%	0%	0.08%	0.02%

6.10 Comparison in Dynamically Changing Workloads

In live streaming, the number of concurrent channels has been reported to follow a diurnal access pattern [59]. For example, the maximum diurnal variation of the number of channels on Twitch is known to be 0.7 : 1 [59]. Based on the actual number of channels that change every hour on the Twitch system [59], we examine the average PWQ over the course of a day as shown in Fig. 9.

Like static workloads, TDA+CR scheme shows the best performance again, yielding 4.94% to 67.11% (average 26.29%) more PWQ than the other schemes. The execution time of our algorithm is negligible, as tabulated in Table 6, so it can be effectively applied to dynamic workloads regardless of the cost model.

6.11 Comparison with Other Heuristic Algorithms for the Knapsack Problem

Since TTAP is a variant of the knapsack problems, we examine other heuristic algorithms for the knapsack problems for comparison. For example, we compare our scheme with five heuristic algorithms: Martello and Toth's heuristic (MTHM) [48], Dyer and Zemel's heuristic (Dyer-Zemel) [60], simulated annealing (SA) [61], genetic algorithm (GA) [62] and deep Q-learning (DQN) [63].

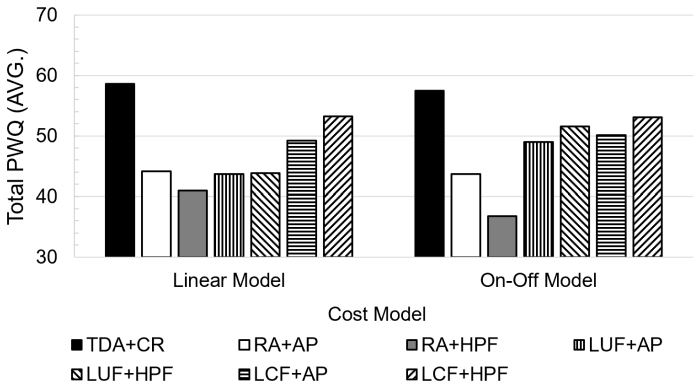


Fig. 9. Average PWQ over the day with dynamically changing workloads, for our scheme and six benchmarks.

MTHM [48] and Dyer-Zemel [60] are heuristic algorithms based on a greedy approach to solving the knapsack problem, both modified to apply to TTAP. For example, the Dyer-Zemel scheme uses a partitioning technique to determine the threshold ratio value of PWQ to CPU usage for each channel and progressively prunes a candidate task set for each channel [60], whereas the MTHM scheme assigns tasks with high PWQs first and allocates them considering the remaining cost budget and CPU capacity of each ES [48].

Both SA and GA are a kind of meta-heuristic, which repeats the process of randomly generating and changing solutions to find the best result [61], [62]. In SA, temperate values are set, starting at 1 and going down to 0.0001. The cooling coefficient is set to 0.000005 and the Boltzmann constant to 1. In GA, the population size and the number of generations are set to 100 and 10000, respectively. In addition, 10% of elite chromosomes are set to be passed to the next generation, and the probability of mutation is set to 1%. For the next generation, a two-point crossover and roulette wheel selection are used. All of these parameters have been empirically derived to give the best results.

DQN is one of the deep reinforcement learning methods. It trains the result of each combination as a neural network through exploration and exploitation without the agent knowing the environment, and finds a solution based on this trained model [63]. The ϵ-greedy method is used, in which ϵ, (the probability of selecting the greedy method) was set to 20%. We use a replay memory that stores 100,000 recent experiences to train the neural network. 512 experiences are randomly selected for every state transition to update the neural network model. The neural network consists of 2 input nodes, 4 hidden nodes, and 1 output node. DQN runs on a server with dual NVIDIA GeForce RTX 2080 Ti GPU cards.

In practice, SA, GA, and DQN are not algorithms with fixed completion times, so their completion times depend on the number of combinations generated. Therefore, the parameters of each algorithm (SA and GA) were set so that the running time was approximately 1 hour. Likewise, the training time for DQN is set to be limited to 1 hour.

Fig. 10 shows the overall PWQ values when N^{ch} is 6000 and N^{ES} is 100. TDA+CR outperforms all the other schemes, by yielding 2.62% to 22.94% (average 10.61%) more PWQ than the other schemes. Except for TDA+CR, Dyer-Zemel and MTHM show the best results, while SA and GA yield the lowest PWQ. DQN may not be suitable for use in a live streaming environment, because solving TTAP with DQN requires training on a new set

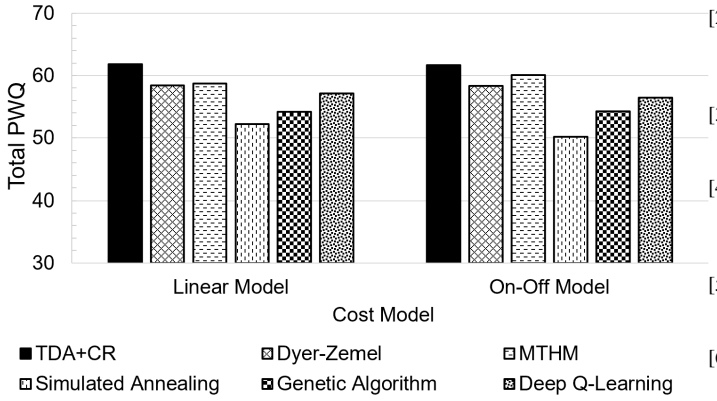


Fig. 10. Overall PWQ comparison between the five heuristics and TDA+CR.

of channels and ES each time the workload changes.

7 Conclusions

Edge-servers can be used to improve QoE in live-streaming systems, but the extra computation that they provide comes at a cost. We proposed a scheme for the allocation of video transcoding tasks to edge-servers close to channels, so as to improve QoE in live-streaming. We formulated this task allocation as an optimization problem which takes account of network coverage, server processing capacities, channel popularity, and the cost budget. We then proposed a two-phase algorithm to solve this problem. In the first phase, candidate transcoding tasks are identified, and then allocated to edge-servers within the channels' network coverage. In the second phase, some of these allocations are changed or canceled, so as to reduce cost within a specified budget, while minimizing the effect of these alterations on overall PWQ. In each step, a greedy optimization operates on parameters which represent a combination of PWQ, processing capacity, and cost.

We conducted simulations to assess our scheme in terms of total PWQ while varying the cost budget, the number of channels and edge-servers, the allowable bitrates, version popularity, and so on. The results showed that our scheme achieves 0.06% to 94.62% (average 25.3%) more PWQ than other benchmark schemes. This difference is more pronounced when the cost budget is tight, when there are many channels, and when there are few edge-servers, suggesting that our scheme is particularly effective for heavy workloads.

Experimental results suggest that our scheme can contribute to the effective cost management of live-streaming systems in an MEC environment. In future work, we plan to explore techniques that address the high power consumption issues of the edge-servers used in live-streaming systems.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT). (NRF2022R1A2C1007237 and NRF2021R1F1A1064053)

References

[1] https://www.businessofapps.com/data/twitch-statistics/

[2] Z. Zhang, R. Wang, F. R. Yu, F. Fu, and Q. Yan. QoS aware transcoding for live streaming in edge-clouds aided hetnets: an enhanced actor-critic approach. *IEEE Transactions on Vehicular Technology*, 68(11):11295–11308, November 2019.

[3] C. Zhang, J. Liu, and H. Wang. Cloud-assisted crowdsourced livecast. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 13(3s):1—22, August 2017.

[4] Y. Zhu, J. Liu, Z. Wang, and C. Zhang. When cloud meets uncertain crowd: an auction approach for crowdsourced livecast transcoding. In *Proceedings of the ACM International Conference on Multimedia*, pages 1–7, October 2017.

[5] G. Gao and Y. Wen. Video transcoding for adaptive bitrate streaming over edge-cloud continuum. *Digital Communications and Networks (Early Access)*, pages 1–7, 2020.

[6] Y. Zhu, Q. He, J. Liu, B. Li, and Y. Hu. When crowd meets big video data: cloud-edge collaborative transcoding for personal livecast. *IEEE Transactions on Network Science and Engineering*, 7(1):42–53, October 2018.

[7] Q. He, C. Zhang, and J. Liu. Crowdtranscoding: Online video transcoding with massive viewers. *IEEE Transactions on Multimedia*, 19(6):1365–1375, June 2017.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T.Skeie. Mobile edge computing: a survey. *IEEE Internet of Things Journal*, 5(1):450–465, February 2018.

[9] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong. Edge computing enabled smart cities: a comprehensive survey. *IEEE Internet of Things Journal (Early access)*, 2020.

[10] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan. Toward computation offloading in edge computing: a survey. *IEEE Access*, 7:131543–131558, August 2019.

[11] https://blog.stratus.com/12-gartner-edge-computing-use-cases-win-edge-computing/

[12] C. Anglano, M. Canonico, and M. Guazzone. Profit-aware resource management for edge computing system. In *Proceedings of the ACM International Workshop on Edge Systems, Analytics and Networking*, pages 25–30, May 2018.

[13] D. Nguyen, L. Le, and V. Bhargava. Price-based resource allocation for edge computing: a market equilibrium approach. *IEEE Transactions on Cloud Computing*, 9(1):302–317, Jan 2021.

[14] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *Proceedings of the IEEE International Conference on Service-Oriented Computing*, pages 230–245, November 2018.

[15] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang. A game-theoretical approach for user allocation in edge computing environment. *IEEE Transactions on Parallel and Distributed Systems*, 31(3):515–529, March 2020.

[16] Y. Hao, L. Hu, Y. Qian, and M. Chen. Profit maximization for video caching and processing in edge cloud. *IEEE Journal on Selected Areas in Communications*, 37(7):1632–1641, July 2019.

[17] D. Krishnappa, M. Zink, and R. Sitaraman. Optimizing the video transcoding workflow in content delivery networks. In *Proceedings of the ACM Multimedia Systems Conference*, pages 37–48, March 2015.

[18] 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 *Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications*, pages 1–4, September 2013.

[19] 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 Transactions on Multimedia*, 19(1):149–159, January 2017.

[20] B. Shen, S. Lee, and S. Basu. Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks. *IEEE Transactions on Multimedia*, 6(2):940–950, 2004.

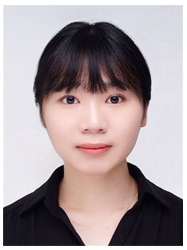
[21] G. Gao, Y. Wen, and C. Westphal. Dynamic priority-based resource provisioning for video transcoding with heterogeneous QoS. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(5):1515–1529, May 2019.

[22] D. Lee, J. Lee, and M. Song. Video quality adaptation for limiting transcoding energy consumption in video servers. *IEEE Access*, 7:126253–126264, December 2019.

[23] D. Lee and M. Song. Quality-aware transcoding task allocation under limited power in live-streaming system. *IEEE Systems Journal (Early Access)*, pages 1–12, 2021.

[24] G. Gao, H. Hu, Y. Wen, and C. Westphal. Resource provisioning and profit maximization for transcoding in clouds: a two-timescale approach. *IEEE Transactions on Multimedia*, 19(4):836–848, April 2017.

[25] X. Li, M. Salehi, Y. Joshi, M. Darwich, B. Landreneau, and M. Bayoumi. Performance analysis and modeling of video transcoding using heterogeneous cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):910–922, September 2018.
[26] B. Mada, M. Bagaa, and T. Taleb. Efficient transcoding and streaming mechanism in multiple cloud domains. In *Proceedings of the IEEE Global Communications Conference*, pages 1–6, December 2017.
[27] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen, and G. Zhang. Online cloud transcoding and distribution for crowdsourced live game video streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(8):1777–1789, August 2017.
[28] N. Dao, D. Ngo ad N. Dinh, T. Phan, N. Vo, S. Cho, and T. Braun. Hit ratio and content quality tradeoff for adaptive bitrate streaming in edge caching systems. *IEEE Systems Journal (Early Access)*, pages 1–4, 2020.
[29] D. Wang, Y. Peng, X. Ma, W. Ding, H. Jiang, F. Chen, and J. Liu. Adaptive wireless video streaming based on edge computing: Opportunities and approaches. *IEEE Transactions on Services Computing*, 12(5):685–697, September 2018.
[30] A. Tran, N. Dao, and S. Cho. Bitrate adaptation for video streaming services in edge caching systems. *IEEE Access*, 8:3135844–135852, July 2020.
[31] E. Baccoura, A. Erbada, K. Bilalb, A. Mohameda, and M. Guizani. PCCP: proactive video chunks caching and processing in edge networks. *Future Generation Computer Systems*, 105:44–60, April 2020.
[32] Y. Guo, F. Yu, J. An, K. Yang, C. Yu, and V. Leung. Adaptive bitrate streaming in wireless networks with transcoding at network edge using deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(4):3879–3892, April 2020.
[33] P. Dogga, S. Chakraborty, Subrata Mitra, and R. Netravali. Edge-based transcoding for adaptive live video streaming. In *Proceedings of the USENIX Workshop on Hot Topics in Edge Computing*, pages 1–7, July 2019.
[34] https://netflixtechblog.com/vmaf-the-journey-continues-44b51ee9ed12/
[35] R. Rassool. VMAF reproducibility: Validating a perceptual practical video quality metric. In *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, pages 1–2, June 2017.
[36] D. Ghadiyaram, J. Pan, and A. Bovik. Learning a continuous-time streaming video QoE model. *IEEE Transactions on Image Processing*, 27(5):2257–2271, January 2018.
[37] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with Pensieve. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 197—210, August 2017.
[38] K. Spiteri, R. Urgaonkar, and R. Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. In *Proceedings of the IEEE International Conference on Computer Communications*, pages 1—9, April 2016.
[39] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing*, 11(1):154–166, September 2016.
[40] H. Nam, K. Kim, and H. Schulzrinne. QoE matters more than QoS: Why people stop watching cat videos. In *Proceedings of the IEEE International Conference on Computer Communications*, pages 1—9, April 2016.
[41] A. Ahmed, Z. Shafiq, H. Bedi, and A. Khakpour. Suffering from buffering? Detecting QoE impairments in live video streams In *Proceedings of the IEEE International Conference on Network Protocols*, pages 1—9, October 2017.
[42] https://aws.amazon.com/ko/wavelength/
[43] https://aws.amazon.com/ko/lambda/edge/
[44] https://cloud.google.com/blog/ko/products/anthos/anthos-for-telecom-puts-google-cloud-partners-apps-at-the-edge
[45] https://azure.microsoft.com/en-us/pricing/details/azure-stack/edge/
[46] https://aws.amazon.com/ko/snowball/pricing/
[47] G. Dahl and N. Foldnes. LP based heuristics for the multiple knapsack problem with assignment restrictions. *Annals of Operations Research*, 146(1):91–104, 2006.
[48] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990.
[49] https://aws.amazon.com/ko/ec2/instance-types/
[50] https://aws.amazon.com/ko/ec2/pricing/on-demand/
[51] R. Aparicio-Pardo, K. Pires, A. Blanc, and G. Simon. Transcoding live adaptive video streams at a massive scale in the cloud. In *Proceedings of the ACM Multimedia Systems Conference*, pages 49–60, March 2015.
[52] Y. Qin, S. Hao, K. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue. Quality-aware stategies for optimizing ABR video streaming QoE and reducing data usage. In *Proceedings of the ACM Multimedia Systems Conference*, pages 189–200, June 2019.
[53] C. Zhang and J. Liu. On crowdsourced interactive live streaming: A twitch.tv-based measurement study. In *Proceedings of the IEEE International Conference on Service-Oriented Computing*, pages 55–60, March 2015.
[54] http://www.spec.org/power_ssj2008/results
[55] https://support.google.com/youtube/answer/2853702
[56] https://netflixtechblog.com/per-title-encode-optimization-7e99442b62a2
[57] https://support.video.ibm.com/hc/en-us/articles/207852117-internet-connection-and-recommended-encoding-settings
[58] D. Stohr, A. Frömmgen, A. Rizk, M. Zink, and R. Steinmetz. Where are the sweet spots?: A systematic approach to reproducible DASH player comparisons. In *Proceedings of the ACM International Conference on Multimedia*, pages 1113–1121, October 2017.
[59] K. Pires, G. Simon. YouTube Live and Twitch: A Tour of User-Generated Live Streaming Systems In *Proceedings of the ACM Multimedia Systems Conference*, pages 225–230, March 2015.
[60] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Berlin, Heidelberg, 2004.
[61] A. Liu, J. Wang, G. Han, S. Wang, and J. Wen. Improved Simulated Annealing Algorithm Solving for 0/1 Knapsack Problem In *Proceedings of International Conference on Intelligent Systems Design and Applications*, pages 1159-1164, October 2006.
[62] S. K. Shil. *An Approach to Solve MMKP: Using Genetic Algorithm: Basic, Solving Procedure and Discussion*. LAP Lambert Academic Publishing, 2010.
[63] Q. Cappart, T. Moisan, L. Rousseau, I. Prémont-Schwarz, and A. Cire. Combining Reinforcement Learning and Constraint Programming for Combinatorial Optimization In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 3677-368, February 2021.



Dayoung Lee received the B.S. and M.S. degrees in computer engineering from Inha University, Korea, in 2016 and 2018, respectively. She is currently pursuing the Ph.D. degree from the Department of Computer Engineering at Inha University. Her current research interests include distributed-file systems and multimedia systems.

Younghyun Kim received the B.S. degree (Hons.) in computer science and engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 2007 and 2013, respectively. From 2013 to 2016, he was a Postdoctoral Research Assistant with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA. From 2009 to 2011, he was a Visiting Scholar with the University of Southern California, Los Angeles, CA, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI, USA. His current research interests include energy-efficient computing and the security and privacy of Internet-of-Things.

Minseok Song (M'07) received the B.S., M.S., and Ph.D. degrees in computer engineering from Seoul National University, Korea, in 1996, 1998, and 2004, respectively. Since September 2005, he has been with the Department of Computer Engineering at Inha University, Korea, where he is currently a professor. His research interests include embedded systems, multimedia and IoT systems.