Online Orchestration of Collaborative Caching for Multi-Bitrate Videos in Edge Computing

Song Yang, *Member, IEEE,* Lei Jiao, *Member, IEEE,* Ramin Yahyapour, and Jiannong Cao, *Fellow, IEEE*

Abstract—In the traditional video streaming service provisioning paradigm, users typically request video contents through nearby Content Delivery Network (CDN) server(s). However, because of the uncertain wide area networks delays, the (remote) users usually suffer from long video streaming delay, which affects the quality of experience. Multi-Access Edge Computing (MEC) offers caching infrastructures in closer proximity to end users than conventional Content Delivery Networks (CDNs). Yet, for video caching, MEC's potential has not been fully unleashed as it overlooks the opportunities of collaborative caching and multi-bitrate video transcoding. In this paper, we model and formulate an Integer Linear Program (ILP) to capture the long-term cost minimization problem for caching videos at MEC, allowing joint exploitation of MEC with CDN and real-time video transcoding to satisfy arbitrary user demands. While this problem is intractable and couples the caching decisions for adjacent time slots, we design a polynomial-time online orchestration framework which firstly relaxes and carefully decomposes the problem into a series of subproblems solvable in each individual time slot and then converts the fractional solutions into integers without violating constraints. We have formally proved a parameterized-constant competitive ratio as the performance guarantee for our approach, and also conducted extensive evaluations to confirm its superior practical performance. Simulation results demonstrate that our proposed algorithm outperforms the state-of-the-art algorithms, with 13.6% improvement on average in terms of total cost.

Index Terms—Online Caching, Multi-Bitrate Video, Multi-Access Edge Computing, Quality of Experience.

1 Introduction

Along with the rapid development of personal mobile devices and proliferation of video content providers (e.g., YouTube, Netflix, etc.), mobile video streaming has become one of the most popular applications on the users' mobile devices. According to Cisco [1], video traffic will account for around 79 percent of the whole mobile data traffic by 2022, up from 59 percent in 2017. The nowadays' video streaming service usually requires low latency and large bandwidth [2] in order to satisfy the Quality of Experience (QoE) of users. Content Distribution Network (CDN) is the current main video service provisioning paradigm [3], in which CDN servers are deployed across geo-distributed backbone Internet nodes. In this sense, users can access the video file from their nearby CDN server. However, this paradigm still suffers from the high streaming delay [4] because of the "long" transmission distance from end users to CDN server deployed on Internet backbone. Moreover, with the increasing amount of video traffic, it will cause the network bottleneck and hence results in high latency, which affects the QoE of users. Consequently, the current video provisioning paradigm cannot keep pace with the ever-increasing video demands with more stringent QoE requirement, which poses big challenges for video service providers.

The concept of Multi-Access Edge Computing (MEC) [5] has been proposed by ETSI. MEC aims at pushing cloud service from network core to network edge, by building small-scale cloud infrastructures (called edge cloud) in close proximity to end users. In this sense, the video file can be cached in edge cloud for nearby end users, so as to shorten the video content transmission delay to guarantee the QoE. Since the edge cloud contains a limited number of servers, it is suggested each individual edge cloud should connect to one another via local area network or wired peer-to-peer links in order to work collaboratively [6] and expose a greater storage and processing capability. Moreover, it is known that a video file with a higher bitrate level has a higher quality, but consumes a larger capacity. Therefore, there exists a tradeoff in caching which bitrate level(s) for a certain video file to satisfy the QoE requirement without violating the capacity of the edge cloud. Nevertheless, it still happens that when all the edge clouds are fully utilized and there are still remaining video requests to be served. In that case, the central CDN server(s) with more sufficient capacity will be in help to store and provision requested video file, but this comes at the expense of higher delay. This deals with how to strategically place requested video files with appropriate bitrate level on edge clouds and CDN server(s) to satisfy the QoE of users. We refer this problem to the video caching problem in edge computing in this paper. Fig. 1 shows a framework about video caching at the network edge.

The existing efforts [7], [8] on solving the video caching problem in edge computing can be categorized into the offline and online scenarios. On the one hand, the offline sce-

- *Song Yang is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. E-mail: S.Yang@bit.edu.cn*
- *Lei Jiao is with the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403, USA. E-mail: jiao@cs.uoregon.edu*
- *Ramin Yahyapour is with Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) and Institute of Computer Science, University of Göttingen, 37077 Göttingen, Germany. E-mail: Ramin.Yahyapour@gwdg.de*
- *Jiannong Cao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: csjcao@comp.polyu.edu.hk.*

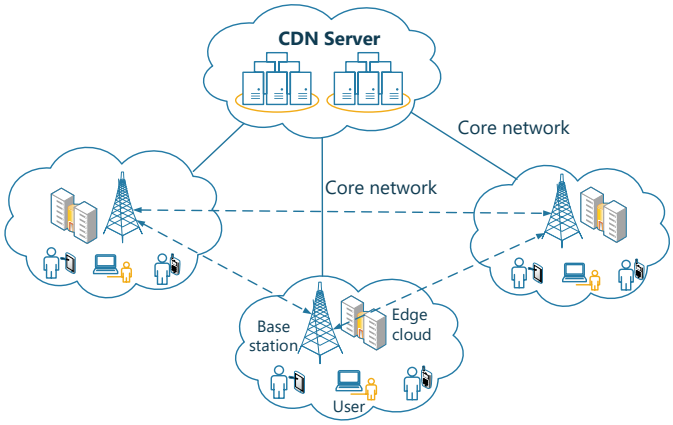


Fig. 1: A framework about video caching at network edge.

nario assumes that the whole traffic requests are known as a prior. However, in practice the traffic requests are usually time-varying and difficult to obtain accurately. The offline or the one-shot solutions therefore cannot solve the dynamic video caching problem appropriately without knowing all the traffic information. On the other hand, the current work on the online video caching problem fails to jointly optimize the QoE factor such as delay for users and OPerational EXpenditures (OPEX)[1] for service providers in a collaborated MEC network as we elaborate above. For instance, some work [10]- [11] for video caching only focus on guaranteeing the QoE for the end users without considering the cost expenditure of service providers. Moreover, the current work [12], [13] usually consider the scenario containing one edge cloud, ignoring the collaboration of multiple edge clouds. In this paper, we tackle the online video caching problem in edge computing, where the network lifetime consists of a set of time slots, and in each time slot the video requests arrive at the network dynamically. We jointly consider three kinds of costs, namely (1) the operational cost incurred by OPEX, (2) the deployment cost due to launching new video file, and (3) video streaming delay cost which consists of video delivering delay cost and transcoding delay cost. The goal of the considered problem in this paper is to minimize the total sum of all these kinds of costs.

In order to solve the considered online video caching problem in edge computing, we first formulate it as an offline Integer Linear Programming (ILP). After that, we leverage the regularization-based technique [14] to divide the relaxed ILP into a series of regularized subproblems. In particular, we identify deployment cost in the objective function as the connection between consecutive time slots and apply regularization technique to eliminate its correlation between consecutive time slots. By doing this, each of the subproblems can be solved efficiently in each time slot using only current information, and therefore the caching decisions in each time slot constitute the final feasible solution to the relaxed offline problem. Subsequently, we design a randomized dependent rounding scheme [15] to round the fractional solutions for the regularized subproblems to integer solution. The randomized dependent rounding scheme continuously selects a pair of fractional variables and rounds them up and down in a compensative manner respectively, such that the sum of them remains largely the same as before without violating the related constraints. The online orchestration video caching framework, consisting of the online regularization-based algorithm and randomized dependent rounding scheme, is able to achieve a provable competitive ratio compared to the offline optimum. In all, our key contributions are as follows:

- We analyze the video caching model in edge computing by considering three kinds of costs and formulate the online video caching problem in edge computing.
- We propose an online orchestration framework by leveraging the regularization technique and randomized dependent rounding scheme to solve the considered problem with provable competitive ratio.
- We validate the performance of the proposed solution with three existing benchmark algorithms via extensive simulations.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces the network model, formally defines the considered problem and presents its formulation. Section 4 presents the online orchestration framework for video caching in edge computing, and Section 5 analyzes and proves its overall competitive ratio. Section 6 provides the performance evaluation results and we conclude in Section 7.

2 Related Work

A survey about video caching, content delivery and video streaming in MEC can be found in [7], [8].

2.1 Offline Video Caching in Edge Computing

Li *et al.* [10] formulate the video caching problem in MEC as a submodular maximization problem to maximize the aggregated utility. They subsequently devise a k-cost benefit greedy algorithm to solve this problem and prove it has an approximation ratio when $k = 2$. Qu *et al.* [16] address the caching problem in MEC to maximize the QoE value over all the users under the condition that the QoE function has general or linear function with received bitrate. They present approximation algorithms by leveraging the idea to solve the multiple-choice knapsack problem to solve this problem in these two cases, respectively. Chen *et al.* [11] formulate the multi-bitrate video caching problem in edge computing as a Stackelberg game in order to jointly maximize the profit of service provider and video providers. A dynamic programming algorithm and a adjustive caching algorithm are proposed in [11] to find the Stackelberg equilibrium. Nevertheless, the above literature only works in the offline scenario, and cannot properly solve the online video caching problem where the future requests are unknown.

Moreover, Wang *et al.* [17] present a deep reinforcement learning approach to solve the offline video caching (placement) problem by jointly minimizing streaming delay, channel switching latency and bitrate mismatch level as well as system costs. Wang *et al.* [18] later develop a multi-agent deep reinforcement learning approach to solve the similar

1. OPEX [9] mainly refers to the cost factors of the network operation such as the maintenance of running services, operating, testing equipment to detect failures, etc.

TABLE 1: Difference between our work and existing work.

Ref	Online	Multi-Edges	Multi-Costs	Transcode
[10]	no	no	yes	no
[16]	no	yes	yes	no
[11]	no	no	yes	yes
[17]	no	no	yes	no
[18]	no	no	yes	no
[19]	yes	no	no	yes
[20]	yes	no	yes	no
[21]	yes	yes	yes	no
[22]	yes	yes	no	no
[12]	yes	no	yes	no
[13]	yes	no	yes	no
[23]	yes	no	yes	no
[24]	yes	no	no	no
[25]	yes	no	yes	no
[26]	yes	no	no	no
Our work	yes	yes	yes	yes

problem. However, as mentioned in [17], the action space in the proposed deep reinforcement learning-based algorithm will increase exponentially with the number of requests, which means that the proposed DRL-based algorithm cannot concurrently return the solution for a number of (online) requests within a reasonable time.

2.2 Online Video Caching and Streaming in Edge Computing

Tran and Pompili [19] decompose the collaborative video caching problem in MEC into the cache placement subproblem and request scheduling subproblem. They respectively devise an Adaptive Bitrate (ABR)-aware proactive cache placement algorithm and an online video request scheduling algorithm to solve these two subproblems. Li *et al.* [20] present an online content placement, node association and power allocation strategy based on Lyapunov optimization and Generalized Benders decomposition. Similarly, Xia *et al.* [21] apply Lyapunov technology to develop an online data caching framework in edge computing. Chiang *et al.* [22] apply Auto Regressive Moving Average (ARMA) model to predict social behaviour of users and the popularity of videos, and then present a collaborative social-aware online video caching framework. Choi *et al.* [27] study the dynamic video delivery problem in wireless caching networks. More specifically, when the mobile users are moving around and the wireless network condition is stochastic between the user and caching node, they investigate how to control and receive different chunks of desired files to guarantee the video quality. Hao *et al.* [12] propose a Multi-Armed Bandit (MAB)-based online algorithm to place video caching to maximize the service provider's profit on edge cloud. Wang *et al.* [13] study the dynamic configuration adaptation and bandwidth allocation problem in edge-based video analytics systems for multiple video streams, where energy consumption of mobile device and service latency are taken into account. They present a Lyapunov-based algorithm to solve this problem. However, only one edge server (cloud) is considered in [12], [13].

Mehrabi *et al.* [23] develop a network-assisted bitrate adaptation heuristic for video streaming in edge computing, which uses a self-tuning of the values of the optimization problem parameters. Mu *et al.* [24] present a Deep Reinforcement Learning (DRL) algorithm by capturing the content features and channel estimation to make playout rate and bitrate decision for video streaming. Li *et al.* [25] study the trusted cooperative computation offloading problem in MEC, and devise a online learning-aided offloading mechanism based on Lyapunov optimization and Online Gradient Descent (OGD) method. Galanopoulos *et al.* [26] develop an online learning-based configuration algorithm for video analytic based on Bayesian GP technique with bandit learning and safe constraint exploration. While [23]-[26] respectively deals with video streaming, computation offloading and video analytics problems in MEC, our work which addresses the online video caching problem is different from them. In all, Table 1 provides a table showing the difference between our work and the existing work.

Nevertheless, none of above work addresses our considered online video caching problem in a collaborated MEC network by considering various kinds of costs and multi-bitrate video transcoding. We comprehensively take into account the operational cost, deployment cost and video streaming delay cost. Since the deployment cost couples two successive time slots, the above mentioned work solving online video caching problem by applying Lyapunov technique and MAB theory cannot solve our problem appropriately. This stimulates us to develop an online algorithm based on regularization technique to decouple the time-consecutive costs together with randomized dependent rounding technique to make the algorithm can execute in polynomial time.

3 Network Model and Problem Formulation

3.1 Video Caching Edge System

We assume there is a set of edge cloud $\mathcal{N}_e$. Co-located with each edge cloud $n \in \mathcal{N}_e$, there is a base station for sending and receiving signals. An edge cloud consisting of a limited number of edge servers processes and caches video files requested by the users. We use $\lambda(n)$ to represent the available capacity of the edge cloud n. $\lambda(n)$ is usually less than the full capacity of n to guarantee that the server can work normally and steadily so that the tasks performed on it will not affect each other. Moreover, there is one CDN server $\mathcal{N}_c$ for simplicity, which is assumed to store as many video files as possible without loss of generality. Different edge clouds as well as the CDN server are inter-connected with each other, and we use $T(u, v)$ to denote the transmitting delay from u to v, where $u, v \in \mathcal{N}_e \cup \mathcal{N}_c : u \neq v$. As a result, the network can be represented by $\mathcal{G}(\mathcal{N}, \mathcal{L})$, where $\mathcal{N} = \mathcal{N}_e \cup \mathcal{N}_c$ denotes the set of N nodes and $\mathcal{L}$ represents the set of L links. In fact, the video file consists of a number of same-sized chunks, and different chunks of a video file can be stored on different edge nodes. In this sense, the video file can be further divided and stored on different nodes. For ease of presentation, we assume that the video file cannot be further divided and stored on different nodes in this paper, similar to [12], [18], [19]. The reason is that, if each request r asks for a video file consisting of m chunks, we can also equivalently change it to the case where m requests demands m chunks (files). We denote F as the set of total $|F|$ different video files, and B as the set of total $|B|$

TABLE 2: Notations.

Notation	Description
$\mathcal{G}(\mathcal{N}, \mathcal{L})$	A network with set of nodes $\mathcal{N}$ and links $\mathcal{L}$.
$\mathcal{N}_e, \mathcal{N}_c$	The set of edge cloud and CDN server.
$\lambda(n)$	Available capacity of edge cloud $n \in \mathcal{N}_e$.
$T(u, v)$	Transmitting delay from u to v.
F, B	The set of total $\|F\|$ video files, the set of total $\|B\|$ bitrate levels.
Δ_n	The transcoding cost per unit.
Φ_n	The per-unit cost for caching a video file.
$C_O(t), C_D(t), C_W(t)$	The total operational cost, deployment cost and video streaming cost at time t.
e_n	The per-unit cost for deploying a file with a certain bitrate level in an edge node n.
$\Gamma_v(f, b, \beta)$	The transcoding delay for f from current cached bitrate level β to the requested bitrate level b.
R_t	The set of requests R_t at time t. For each request $r(u, f, b) \in R_t$, u indicates his/her local edge cloud, and f denotes the requested video file with bitrate level b.
$\mathbf{P}$, $\mathbf{P'}$, $\mathbf{P_t}$, $\widetilde{\mathbf{P}}_\mathbf{t}$, $\mathbf{D}$	Original OVCE problem in Eqs. (5a)-(5e), relaxed OVCE problem of $\mathbf{P}$, one shot OVCE problem of $\mathbf{P'}$ at time t, regularized OVCE problem at time t, Lagrange dual problem of relaxed OVCE problem $\mathbf{P'}$
P, P', P_t, $\widetilde{P}_t$	Objective function of original OVCE problem $\mathbf{P}$, relaxed OVCE problem $\mathbf{P'}$, one shot OVCE problem $\mathbf{P_t}$, and regularized OVCE problem $\widetilde{\mathbf{P}}_\mathbf{t}$
$X_{n,t}^{r,\beta}$	A boolean variable. It is 1 (true) if r is accommodated by the requested cached file with bitrate level β on node n at time t; and 0 (false) otherwise.
$Y_{n,t}^{f,\beta}$	A boolean variable. It is 1 (true) if f with bitrate level β is placed on n at time t, and 0 (false) otherwise.

bitrate levels. We use S_β^f to represent the size of file $f \in F$ with bitrate level $\beta \in B$. Clearly, a video file with higher bitrate level has a larger size than it with lower bitrate level. For ease of reading, Table 2 provides all the used notations and variables in this paper.

Nevertheless, our work can also be extended to the scenario of multiple geo-distributed CDN servers, where the CDN server set $\mathcal{N}_c$ contains more than one geo-distributed CDN servers. In that case, each CDN server node $n \in \mathcal{N}_c$ is connected with the edge node $n \in \mathcal{N}_e$ with a transmitting delay value. Compared to the single CDN server scenario, in the multiple geo-distributed CDN scenario, the network topology $G'(\mathcal{N}, \mathcal{L})$ contains more than one CDN server nodes, where $\mathcal{N} = \mathcal{N}_c \cup \mathcal{N}_e$. However, since the CDN server node is assumed to have infinite storage capability, one CDN server is always enough to host all the video files (at the expense of the higher transmitting delay compared to placing the video file on edge node). In the multiple CDN servers scenario, the algorithm's procedure and performance ratio keep the same with the single CDN server scenario, but only to take $G'(\mathcal{N}, \mathcal{L})$ as the problem input.

The video caching edge system works in a time-slotted fashion over a continuous time span of $\mathcal{T} = \{1, 2, \ldots, T\}$. Each time slot $t \in \mathcal{T}$ indicates a decision interval, and usually set to be longer than video streaming delay in order to avoid frequent update configurations and reduce overheads [13], [28]. We use R_t to represent the set of traffic requests in time slot t. In each time slot t, the user sends his/her request $r(u, f, b) \in R_t$ to his/her local edge cloud u, where f denotes the requested video file with bitrate level b. We define two boolean variables $X_{n,t}^{r,\beta}$ and $Y_{n,t}^{f,\beta}$ for calculating the caching decision. More specifically, $X_{n,t}^{r,\beta}$ is 1 (true) if r is accommodated by the requested cached file with bitrate level β on node n at time t, and 0 (false) otherwise. $Y_{n,t}^{f,\beta}$ is 1 (true) if f with bitrate level β is placed on n at time t, and 0 (false) otherwise.

3.2 Cost Structure

3.2.1 Operational cost

In general, the operational cost consists of (1) transcoding cost and (2) caching cost. For a request $r(u, f, b)$ accessing video file on edge node n, the transcoding cost happens when n only stores f with the bitrate level(s) (say β) higher than b. In that context, the transcoding cost[2] for r can be calculated as $X_{n,t}^{r,\beta} \cdot (S_\beta^f - S_b^f)\Delta_n$, where Δ_n represents the transcoding cost per unit (e.g., Gb). The caching cost indicates the cost for storing file on an edge or CDN server node. Let Φ_n denote the per-unit cost for caching a video file, then the caching cost for file f with bitrate level β at time slot t on node n can be calculated as: $Y_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Phi_n$. As a result, the total operational cost (denoted by $C_O(t)$) for all the requests at time slot t can be formulated as:

$$C_O(t) = \sum_{r \in R_t} \sum_{n \in \mathcal{N}} \sum_{\beta \in B} X_{n,t}^{r,\beta} \cdot (S_\beta^f - S_b^f)\Delta_n + \sum_{f \in F} \sum_{n \in \mathcal{N}} \sum_{\beta \in B} Y_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Phi_n \quad (1)$$

3.2.2 Deployment cost

We use e_n to represent the per-unit cost for deploying a file with a certain bitrate level in an edge node n, while there is no such video file in n in the previous time slot. The deployment cost is mainly due to the effort of transferring/duplicating a video file in/out of the edge cloud, which consumes network bandwidth [30] and video file duplication and transferring time [31]. The deployment cost can also reflect the QoE influence for the users, since transmitting a video file consumes some time and may result in a "delayed" service. As a result, The total deployment cost (denoted by $C_D(t)$) for all the requests at time slot t can therefore be calculated as:

2. In this paper, we assume that the size of a video file is proportional with the bitrate level and adopt the transcoding cost model similar to [29].

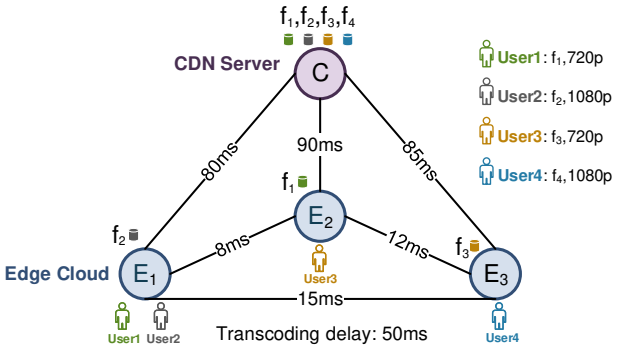


Fig. 2: An example of video streaming delay calculation.

$$C_D(t) = \sum_{f\in F}\sum_{n\in\mathcal{N}}\sum_{\beta\in B} e_n S_\beta^f \cdot (Y_{n,t}^{f,\beta} - Y_{n,t-1}^{f,\beta})^+ \quad (2)$$

where $(Y_{n,t}^{f,\beta} - Y_{n,t-1}^{f,\beta})^+ \stackrel{\text{def}}{=} \max\{0, Y_{n,t}^{f,\beta} - Y_{n,t-1}^{f,\beta}\}$.

3.2.3 Video streaming delay cost

In general, the video streaming delay consists of transmitting delay and transcoding delay. For a user whose local edge cloud is u requires f with bitrate level b, the video streaming delay is represented as:

$$T(u,v) + \Gamma_v(f,b,\beta) \quad (3)$$

where $T(u,v)$ denotes the transmitting delay from u to v where f is cached. Clearly, if f is cached on u (in this case $u=v$), then the transmission delay is 0. $\Gamma_v(f,b,\beta)$ indicates the transcoding delay for f from current cached bitrate level β to the requested bitrate level b. Video transcoding is to encode video content into multiple representations, and we consider that a lower bitrate variant can be obtained from a higher bitrate variant via transcoding [19].

For example in Fig. 2, the video caching edge system contains 3 edge clouds and 1 CDN server, and they are interconnected with each other where the transmission delays are shown on respective links. The transmission delay between edge cloud and CDN server is set to be larger than the transmission delay between edge clouds because of longer physical distance, which reflects the practical scenario. Suppose there are 4 requested video files, and it is assumed that each edge cloud only caches one video file with the bitrate level 1080p because of capacity constraint and CDN server stores all the video files with all the bitrate levels. For simplicity, the transcoding delay from 1080p to 720p for all the files is assumed to be 50 *ms*. Within the coverage of each edge area, the users request video files with different bitrate levels. For example, since user 3 requires f_3 with 720p bitrate level and f_3 is cached in E_3 with 1080p bitrate level, the streaming delay is equal to: 12 (transmission delay) + 50 (transconding delay) = 62 *ms*. Even though f_3 is also cached on the CDN server, since it consumes much longer transmission delay, it is preferable to accommodate the request of user 3 by using the edge cloud. On the contrary, since the requested f_4 is not cached on any edge cloud, user 4 has to fetch the video file from remote CDN server, causing a streaming delay of 85 *ms*. From this example we can see that caching a video file on edge cloud to accommodate a user quest can result in less delay cost compared to provisioning a request by the CDN server.

Consequently, the video streaming delay cost (denoted by $C_W(t)$) for all the requests at time slot t can be calculated as:

$$C_W(t) = \sum_{r\in R_t}\sum_{n\in\mathcal{N}}\sum_{\beta\in B} X_{n,t}^{r,\beta} \cdot (T(n,u) + \Gamma_n(f,b,\beta)) \quad (4)$$

3.3 Problem Definition and Formulation

In this subsection, we formally define the Online orchestration of collaborative Caching for multi-bitrate Videos in Edge computing (OCVE) problem as follows:

Definition 1. *Given is a network $\mathcal{G}(\mathcal{N},\mathcal{L})$ and for each time slot $t\in\mathcal{T}$, there is a set of video requests R_t. For each $r(u,f,b)\in R_t$, the OCVE problem is to place f on $v\in\mathcal{N}$ in order to minimize the total sum of operational cost, deployment cost and delay cost for all the requests over all the time slots.*

Subsequently, we present an exact solution to formulate the OCVE problem.

$$\mathbf{P}: \quad \min \quad \sum_{t\in\mathcal{T}} H_1\cdot C_O(t) + H_2\cdot C_D(t) + H_3\cdot C_W(t) \quad (5a)$$

$$\text{s.t.} \quad \sum_{n\in\mathcal{N}}\sum_{\beta\in B'} X_{n,t}^{r,\beta} \geq 1 \;\; \forall t\in\mathcal{T}, r\in R_t \quad (5b)$$

$$Y_{n,t}^{f,\beta} \geq X_{n,t}^{r,\beta} \quad \forall \beta\in B, n\in\mathcal{N}, f\in F, r\in R_t : r.f = f \quad (5c)$$

$$\sum_{\beta\in B, f\in F} Y_{n,t}^{f,\beta}\cdot S_\beta^f \leq \lambda(n) \;\; \forall t\in\mathcal{T}, n\in\mathcal{N}_e \quad (5d)$$

$$X_{n,t}^{r,\beta}\in\{0,1\},\; Y_{n,t}^{f,\beta}\in\{0,1\} \; \forall t\in\mathcal{T}, r\in R_t, \beta\in B, f\in F, n\in\mathcal{N} \quad (5e)$$

(5a) minimizes the total sum of operational cost, deployment cost and video streaming delay cost for all the requests over all the time slots, where H_1, H_2 and H_3 denotes the weight for each cost, respectively. (5b) ensures that in each time slot, each request should be accommodated by placing the requested video file with a higher (or equal) bitrate level than b of r on one node. Here, B' denotes the bitrate level set that is no lower than b of r. (5c) draws the relation between $X_{n,t}^{r,\beta}$ and $Y_{n,t}^{f,\beta}$. (5d) ensures that the capacity of each edge cloud cannot be exceeded. (5e) ensures that all the decision variables are binary.

The OCVE problem is NP-hard in general. To prove it, for simplicity we assume there is only one time slot and we assume that transmission delay are all 0. We also do not allow the video transcoding and therefore the transcoding delay is 0. After this simplification, only operational cost is considered in the OVCE problem. We further assume that the operational cost coefficient is the same for all the network nodes and each request asks for the unique video file. Consequently, the simplified OVCE problem can be reduced to the NP-hard multiple knapsack problem [32], which is to pack a subset of given items into N knapsacks

to maximize the total profit such that the total weight on each knapsack does not exceed its maximum capacity. Since the simplified OVCE problem is a special case of the original OVCE problem, the OVCE problem is in general NP-hard.

4 An Online Orchestration Framework for Video Caching in Edge Computing

4.1 Challenges and Algorithm Idea

Challenges: There area two major challenges for solving the OVCE problem:

The first challenge is the online uncertainty. The deployment cost couples two successive time slots concerning with non-convex expression $\max\{0, Y_{n,t}^{f,\beta} - Y_{n,t-1}^{f,\beta}\}$, so we first need a way to better formulate it as convex expression without changing the nature of the problem. Moreover, the OVCE problem requires to make online decision only according to the current information, but this decision would influence the decision made in next time slot when considering e.g, switching cost. Therefore, an online algorithm that has guaranteed competitive ratio is desired, in which case the total cost over all time slots of our online decisions made without future knowledge should not exceed a constant times that of the optimal offline optimal algorithm.

The second challenge is the intractability. As we prove that the OVCE problem is NP-hard even for one time slot scenario, it is also intractable to solve in the offline scenario with multiple time slots, not to mention to devise an online algorithm. It is desired to design a polynomial-time algorithm to return an integer solution that incurs a total cost no greater than a constant times of the optimal solution.

Algorithm idea: In this section, we present an Online oRchestration Framework for video Caching (ORFC) in edge computing. In general, ORFC first relaxes the original problem and leverages regularization-based technique to decompose the correlation about the deployment cost across successive time slots. By doing this, the original problem is decomposed into a series of one-time slot fractional optimization problems. After that, ORFC applies the randomized dependent rounding scheme to convert the fractional solutions to obtain an integer solution with bounded integrality gap.

4.2 Online Regularization-based Video Caching Fractional Algorithm

We first transform problem $\mathbf{P}$ of (5a)-(5e) into a linear programming by relaxing $X_{n,t}^{r,\beta}$ and $Y_{n,t}^{f,\beta}$ into fractional numbers.

$$\begin{aligned}\mathbf{P}': \quad \min \quad & H_1 \cdot C_O(t) + H_2 \cdot C_D(t) + H_3 \cdot C_W(t) \\ \text{s.t.} \quad & \text{Constraints } (5b)-(5d) \\ & 0 \le X_{n,t}^{r,\beta} \le 1,\ 0 \le Y_{n,t}^{f,\beta} \le 1 \ \ \forall r, \forall f, \forall \beta, \forall n, \forall t\end{aligned} \tag{6a}$$

The main difficulty in solving problem $\mathbf{P}'$ lies in minimizing the deployment cost in Eq. (2) which couples every two consecutive time slots. Moreover, since Eq. (2) is not convex, it causes problem $\mathbf{P}'$ intractable to solve. To overcome it, we adopt the regularization technique [14], which approximates and substitutes $(Y_{n,t}^{f,\beta} - Y_{n,t-1}^{f,\beta})^+$ in Eq. (2) with convex regularizer relative entropy function as follows:

$$\Delta(Y_{n,t}^{f,\beta} || Y_{n,t-1}^{f,\beta}) = Y_{n,t}^{f,\beta} \ln \frac{Y_{n,t}^{f,\beta}}{Y_{n,t-1}^{f,\beta}} + Y_{n,t-1}^{f,\beta} - Y_{n,t}^{f,\beta} \tag{7}$$

where $Y_{n,t}^{f,\beta} \ln \frac{Y_{n,t}^{f,\beta}}{Y_{n,t-1}^{f,\beta}}$ indicates the relative entropy term and $Y_{n,t-1}^{f,\beta} - Y_{n,t}^{f,\beta}$ is the linear term and implies the movement. Since the relative entropy function is convex and differentiable, it has been widely used to approximate the non-convex term such as L1-distance term (e.g., the deployment cost in this paper). In order to ensure the fraction is still feasible when $Y_{n,t-1}^{f,\beta} = 0$, we add ε on denominator and nominator of the fraction in the relative entropy term. Moreover, to normalize the deployment cost by regularization, we define a factor $\sigma = \ln(1 + \frac{1}{\varepsilon})$ and multiply the improved relative function by $\frac{1}{\sigma}$. As such, the regularized deployment cost at time t (denoted by $C_D^*(t)$) can be formulated [3] as the following convex function:

$$C_D^*(t) = \sum_{f,\beta,n} \frac{e_n}{\sigma} \cdot \left((Y_{n,t}^{f,\beta} + \varepsilon) \ln \frac{Y_{n,t}^{f,\beta} + \varepsilon}{Y_{n,t-1}^{f,\beta} + \varepsilon} - Y_{n,t}^{f,\beta} \right) \tag{8}$$

By substituting $C_D(t)$ with $C_D^*(t)$ in the objective of $\mathbf{P}'$, we obtain the regularized relaxed problem $\widetilde{\mathbf{P}}$. Since $C_D^*(t)$ only requires the current information and previous decision in time $t-1$, we can partition the regularized relaxed problem $\widetilde{\mathbf{P}}$ into a series of one-shot convex optimization problems $\widetilde{\mathbf{P}}_\mathbf{t}$, which can be solved in each individual time slot t based on the decision in time $t-1$ (but does not depend on future traffic information). Consequently, the solutions returned by solving $\widetilde{\mathbf{P}}_\mathbf{t}$ in each time slot constitute a feasible solution to the relaxed problem $\mathbf{P}'$. More specifically, we have:

$$\begin{aligned}\widetilde{\mathbf{P}}_\mathbf{t}: \quad \min \quad & \widetilde{P}_t = H_1 \cdot C_O(t) + H_2 \cdot C_D^*(t) + H_3 \cdot C_W(t) \\ \text{s.t.} \quad & \text{Constraints } (5b)-(5d), (6a), \text{ without } t\end{aligned}$$

Convex optimization problem is a problem where the objective function is either a maximization of a concave function or a minimization of a convex function, and its constraints are all convex. Convex optimization problems can usually be solved quickly and accurately with convex optimization solvers. We notice that problem $\widetilde{\mathbf{P}}_\mathbf{t}$ is a standard convex optimization problem, since its objective is to minimize the a convex function, and its constrains are all convex (linear). The convex optimization problem can usually be solved in polynomial time by some classical methods, like subgradient projection, interior point method, cutting-plane method, etc. We choose to apply interior point method [33] to solve Problem $\widetilde{\mathbf{P}}_\mathbf{t}$, which makes use of self-concordant barrier functions and self-regular barrier functions. In all, Algorithm 1 reflects the whole procedure of our proposed online regularization-based video caching fractional algorithm.

3. We eliminate $Y_{n,t-1}^{f,\beta}$ in $C_D^*(t)$ for brevity since this value is already known and cannot be further optimized at time slot t.

Algorithm 1: Online Regularization-based Video Caching Fractional Algorithm

Input: $\mathcal{G}(\mathcal{N},\mathcal{L}), \boldsymbol{S}, \boldsymbol{\Delta}, \boldsymbol{\Phi}, \boldsymbol{e}, \boldsymbol{T}, \boldsymbol{\Gamma}, \boldsymbol{\lambda}$
Output: $\boldsymbol{X}, \boldsymbol{Y}$
1 **foreach** *time slot* $t \in \mathcal{T}$ **do**
2 Observe values $\boldsymbol{X_{t-1}}$, R_t
3 Use the interior point method to solve Problem $\widetilde{\mathbf{P}}_{\mathbf{t}}$
4 Return solution $\boldsymbol{X_t}$, $\boldsymbol{Y_t}$

4.3 Randomized Dependent Rounding Algorithm

Algorithm 1 only returns the fractional caching solutions, while the caching decision should be boolean in practice. Therefore, we need to round the fractional solutions returned by Algorithm 1 to boolean number (either 0 or 1). Since the constraints are coupled and dependent in Problem $\widetilde{\mathbf{P}}_{\mathbf{t}}$, the traditional randomized rounding, where each variable is rounded up and down independently, cannot guarantee the constraints are always obeyed after being rounded. To this end, we present a randomized dependent rounding algorithm in this subsection to deal with this issue.

The main idea of the proposed randomized dependent rounding algorithm is that it iteratively selects two variables each time. Based on respective calculated probabilities, a variable is rounded up and another variable is rounded down correspondingly. We first round $\widetilde{\boldsymbol{Y}}$ using this routine in order to get boolean decision $\overline{\boldsymbol{Y}}$, and then calculate rounded $\overline{\boldsymbol{X}}$ based on $\overline{\boldsymbol{Y}}$. More specifically in Algorithm 2, taking rounding $\widetilde{\boldsymbol{Y}}$ for example, for each f and β, we maintain a set $\mathcal{I}_t^\beta$ that stores all the nodes with ρ_{nt}^β either fractional or integral, and another set $\mathcal{I}'^\beta_t$ that contains all the nodes whose ρ_{nt}^β value is fractional (Step 5-10). As long as $\mathcal{I}'^\beta_t$ contains more than one element, we repeatedly select two nodes n_1 and n_2 in Step 12, and define two weights Ψ_1 and Ψ_2 for them as in Step 13-14, respectively. After that, the ρ_{nt}^β value for n_1 and n_2 are accordingly updated in Step 15-18. As shown in Step 19-22, if a node ρ_{nt}^β becomes 0 (or 1), then we round down (or round up) $\overline{Y}_{n,t}^{f,\beta}$. If there is only single node in $\mathcal{I}'^\beta_t$, we round $\overline{Y}_{n,t}^{f,\beta}$ up (Step 23-24).

The time complexity of Algorithm 2 can be analyzed as this: Algorithm 2 rounds $\widetilde{\mathbf{Y}}_t$ and $\widetilde{\mathbf{X}}_t$ respectively, so we take analyzing the complexity of rounding $\widetilde{\mathbf{Y}}_t$ for example, and the case of rounding $\widetilde{\mathbf{X}}_t$ follows similarly. When $\kappa = f$ in Step 5, it takes $O(|F|)$ time. Step 6 has a time complexity of $O(|B|)$, and Step 8 consumes $O(N)$ time. There are at most $O(N)$ elements in $\mathcal{I}'^\beta_t$, so Step takes $O(N)$ time. As a result, the time complexity of Algorithm 2 for rounding $\widetilde{\mathbf{Y}}_t$ is $O(|F||B|N)$. Similarly, time complexity of Algorithm 2 for rounding $\widetilde{\mathbf{X}}_t$ is $O(|R_t||B|N)$. In all, the time complexity of Algorithm 2 is $O(|B|N\cdot(|F|+|R_t|))$.

Algorithm 2 holds three properties. (i) either $\rho_{n_1t}^\beta$ or $\rho_{n_2t}^\beta$, or both of them are rounded into integers. (ii) the total weighted sum of two variables stay unchanged after updating either in Step 16 or Step 18. For instance, in Step 16, $(\widetilde{\rho}_{n_1t}^\beta+\Psi_1)\lambda(n_1)+(\widetilde{\rho}_{n_2t}^\beta-\frac{\lambda(n_2)}{\lambda(n_1)}\Psi_1)\lambda(n_2)=\widetilde{\rho}_{n_1t}^\beta\lambda(n_1)+\widetilde{\rho}_{n_2t}^\beta\lambda(n_2)$. (iii) The expectation of the integral solution is equal to the fractional value. For instance, $E(\overline{X}_{n_1,t}^{r,\beta}) = (\widetilde{X}_{n_1,t}^{r,\beta}+\Psi_1)\frac{\Psi_2}{\Psi_1+\Psi_2}+(\widetilde{X}_{n_1,t}^{r,\beta}-\Psi_2)\frac{\Psi_1}{\Psi_1+\Psi_2}=\widetilde{X}_{n_1,t}^{r,\beta}$.

Algorithm 2: Randomized Dependent Rounding Algorithm

Input: $\mathcal{G}(\mathcal{N},\mathcal{L})$, F, B, R_t, $\widetilde{\mathbf{X}}_t$, $\widetilde{\mathbf{Y}}_t$
Output: $\overline{\mathbf{X}}_t$, $\overline{\mathbf{Y}}_t$
1 First, round $\widetilde{\mathbf{Y}}_t$
2 Denote $\overline{Y}_{n,t}^{f,\beta}$ as $\overline{u}_{n,t}^{f,\beta}$, $\widetilde{Y}_{n,t}^{f,\beta}$ as $\hat{u}_{n,t}^{f,\beta}$
3 Then, based on $\overline{\mathbf{Y}}_t$, round $\widetilde{\mathbf{X}}_t$
4 Denote $\overline{X}_{n,t}^{r,\beta}$ as $\overline{u}_{n,t}^{r,\beta}$, $\widetilde{X}_{n,t}^{r,\beta}$ as $\hat{u}_{n,t}^{r,\beta}$
5 **foreach** $\kappa \in F$ *or* R_t ($\kappa = f$ *or* r) **do**
6 **foreach** $\beta \in B$ **do**
7 $\mathcal{I}_t^\beta = \emptyset$, $\mathcal{I}'^\beta_t = \emptyset$
8 **foreach** $n \in \mathcal{N}$ **do**
9 $\rho_{nt}^\beta = \hat{u}_{n,t}^{\kappa,\beta} - \lfloor \hat{u}_{n,t}^{\kappa,\beta} \rfloor$, $\mathcal{I}_t^\beta = \mathcal{I}_t^\beta \cup \{n\}$
10 $\mathcal{I}'^\beta_t = \mathcal{I}'^\beta_t \cup \{n\}$ if $\rho_{nt}^\beta \notin \{0,1\}$
11 **while** $|\mathcal{I}'^\beta_t| \geq 1$ **do**
12 Randomly select n_1 and n_2 from $\mathcal{I}'^\beta_t$
13 $\Psi_1 = \min\{1-\rho_{n_1t}^\beta, \frac{\lambda(n_2)}{\lambda(n_1)}\rho_{n_1t}^\beta\}$
14 $\Psi_2 = \min\{1-\rho_{n_2t}^\beta, \frac{\lambda(n_2)}{\lambda(n_1)}\rho_{n_2t}^\beta\}$
15 With the probability $\frac{\Psi_2}{\Psi_1+\Psi_2}$ set
16 $\rho_{n_1t}^\beta = \rho_{n_1t}^\beta + \Psi_1$, $\rho_{n_2t}^\beta = \rho_{n_2t}^\beta - \frac{\lambda(n_2)}{\lambda(n_1)}\Psi_1$
17 With the probability $\frac{\Psi_1}{\Psi_1+\Psi_2}$ set
18 $\rho_{n_1t}^\beta = \rho_{n_1t}^\beta - \Psi_2$, $\rho_{n_2t}^\beta = \rho_{n_2t}^\beta + \frac{\lambda(n_2)}{\lambda(n_1)}\Psi_2$
19 **if** $\rho_{n_1t}^\beta = 0$ *or* $\rho_{n_1t}^\beta = 1$ **then**
20 $\overline{u}_{n_1,t}^{\kappa,\beta} = \rho_{n_1t}^\beta + \lfloor \hat{u}_{n_1,t}^{\kappa,\beta} \rfloor$, $\mathcal{I}_t^\beta = \mathcal{I}_t^\beta \backslash \{n_1\}$
21 **if** $\rho_{n_2t}^\beta = 0$ *or* $\rho_{n_2t}^\beta = 1$ **then**
22 $\overline{u}_{n_2,t}^{\kappa,\beta} = \rho_{n_2t}^\beta + \lfloor \hat{u}_{n_2,t}^{\kappa,\beta} \rfloor$, $\mathcal{I}_t^\beta = \mathcal{I}_t^\beta \backslash \{n_2\}$
23 **if** $|\mathcal{I}'^\beta_t| = 1$ **then**
24 $\overline{u}_{n,t}^{\kappa,\beta} = \lceil \hat{u}_{n_2,t}^{\kappa,\beta} \rceil$ for the only $n \in \mathcal{I}'^\beta_t$

5 Performance Analysis

In this section, we rigorously analyze the theoretical performance in terms of overall competitive ratio of our proposed approach. The overall competitive ratio represents the worst case of the performance achieved by our approach ORFC compared to the offline optimal solution, i.e., $CR = \max \frac{C_{ORFC}}{C_{opt}}$, where C_{ORFC} and C_{opt} represent the cost returned by ORFC and offline optimum solution, respectively. In general, we prove the overall competitive ratio of ORFC as $CR = CR_1 \cdot CR_2$, where CR_1 represents the competitive ratio of online regularization-based video caching fractional algorithm (Algorithm 1) and CR_2 denotes the integrality gap associated to our randomized dependent rounding algorithm (Algorithm 2). Fig. 3 shows the main idea of the performance analysis of our proposed approach.

Throughout the rest of the paper, we introduce some additional notations: $\mathbf{P}$ is our original problem, $\mathbf{P'}$ is the relaxed problem of $\mathbf{P}$, $\mathbf{P_t}$ is one shot problem at time t of $\mathbf{P'}$, $\widetilde{\mathbf{P}}_{\mathbf{t}}$ is the regularized problem corresponding to $\mathbf{P_t}$. $\mathbf{D}$

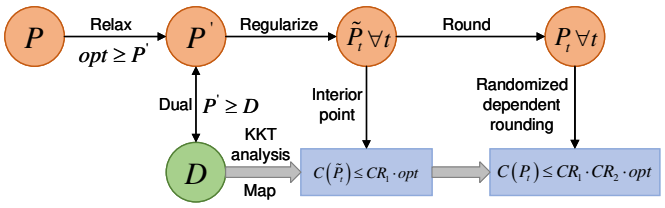


Fig. 3: Main idea of performance analysis of ORFC.

is the Lagrange dual problem of problem $\mathbf{P}'$ and D is the objective function of this dual problem. P, P', P_t, $\widetilde{P}_t$ are the corresponding problem objective functions. $\mathbf{X_t}$ and $\mathbf{Y_t}$ are shorthand for $X_{n,t}^{r,\beta}$ and $Y_{n,t}^{f,\beta}$, $\forall r, f, \beta, n, t$. We also use different diacritical marks with such symbols to represent respective solutions. For instance, $\widetilde{X}_{n,t}^{r,\beta}$ represents the fractional solution returned by regularization-based fractional algorithm, and $\overline{\mathbf{X}}_\mathbf{t}$ denotes the integral solution achieved by the randomized dependent rounding algorithm.

5.1 Competitive Ratio

5.1.1 An Equivalent LP Transformation

Since Problem $\mathbf{P}'$ contains the time-coupling constraint, it is not straightforward to write its dual problem. As such, we introduce an auxiliary variable $Z_{n,t}^{r,\beta}$ and equivalently transform $\mathbf{P}'$ into the following LP formulation:

$$\min \quad H_1 \cdot C_O(t) + H_3 \cdot C_W(t) + H_2 \cdot \sum_f \sum_n e_n S_\beta^f \cdot Z_{n,t}^{f,\beta}$$

$$\text{s.t.} \quad \text{Constraints } (5b) - (5d)$$

$$0 \le X_{n,t}^{r,\beta} \le 1,\ 0 \le Y_{n,t}^{f,\beta} \le 1,\ Z_{n,t}^{f,\beta} \ge 0\ \forall r, f, \beta, n \tag{9a}$$

5.1.2 Deriving the Dual Lagrange problem

We first define the Lagrange dual variables $\alpha_{r,t}$, $\pi_{f,r,\beta,n,t}$, $\theta_{n,t}$, $\omega_{r,\beta,n,t}$, $\mu_{f,\beta,n,t}$, which corresponds to the Constraints (5b)-(5d) and (9a) in above LP formulation. As a result, its dual problem can be written as following:

$$\max \quad D = \sum_{r,t} \alpha_{r,t} - \sum_{n,t} \theta_{n,t}\lambda(n) - \sum_{r,\beta,n,t} \omega_{r,\beta,n,t}$$

$$\text{s.t.} \quad \alpha_{r,t} - \pi_{f,r,\beta,n,t} - \mu_{f,\beta,n,t} + \mu_{f,\beta,n,t-1} - \omega_{r,\beta,n,t} \le \sum_{n,r,\beta,t} H_1(S_\beta^f - S_b^f)\Delta_n + H_3 T(n,u) + H_3\Gamma_n(f,b,\beta) \tag{10a}$$

$$\pi_{f,r,\beta,n,t} - \theta_{n,t}\sum_{\beta,f} S_\beta^f \le H_1 \sum_{f,\beta,n} S_\beta^f \cdot \Phi_n\ \forall f, \beta, r, n, t \tag{10b}$$

$$\mu_{f,\beta,n,t} \le H_2 e_n S_\beta^f\ \ \forall f, \beta, n, t \tag{10c}$$

5.1.3 Characterizing the Regularized Problem

The optimal fractional solution achieved by Algorithm 1 satisfies the Karush-Kuhn-Tucker (KKT) conditions, i.e., the first order necessary conditions for optimal solution. By using the Lagrange dual variables α_r, $\pi_{f,r,\beta,n}$, θ_n, ω_n corresponding to Constraints (5b)-(5d) and (6a) in Problem $\mathbf{P}'$, we have the $\widetilde{\mathbf{P}}_\mathbf{t}$'s KKT conditions as follows:

$$H_1 \sum_{r,\beta}(S_\beta^f - S_b^f)\Delta_n + H_3 \sum_{r,n,\beta} T(n,u) + H_3\Gamma_n(f,b,\beta) + H_2 \sum_{r,n} \frac{e_n S_\beta^f}{\sigma} \ln \frac{Y_{n,t}^{f,\beta} + \varepsilon}{Y_{n,t-1}^{f,\beta} + \varepsilon} - \alpha_r + \pi_{f,r,\beta,n} + \omega_{r,\beta,n} = 0 \tag{11a}$$

$$\sum_{f,\beta,n} S_\beta^f \Phi_n - \pi_{f,r,\beta,n} + \sum_{\beta,f} S_\beta^f \theta_n = 0 \tag{11b}$$

$$\alpha_r\Big(\sum_{n,\beta} X_{n,t}^{r,\beta} - 1\Big) = 0 \tag{11c}$$

$$\pi_{f,r,\beta,n}(Y_{n,t}^{f,\beta} - X_{n,t}^{r,\beta}) = 0 \tag{11d}$$

$$\theta_n\Big(\sum_{\beta,f} Y_{n,t}^{f,\beta} S_\beta^f - \lambda(n)\Big) = 0 \tag{11e}$$

$$\omega_{r,\beta,n}(X_{n,t}^{r,\beta} - 1) = 0 \tag{11f}$$

5.1.4 Constructing the mapping

We now construct a mapping denoted from $\widetilde{\mathbf{P}}$'s primal and dual solution to a feasible solution of the relaxed dual problem at time t as following. It can be verified that the constructed solutions from $\widetilde{\mathbf{P}}$ satisfies constraints (10a)-(10c).

$$\alpha_{r,t} = \alpha_r, \forall r;\ \pi_{f,r,\beta,n,t} = \pi_{f,r,\beta,n},\ \forall f, r, \beta, n;$$
$$\theta_{n,t} = \theta_n, \forall n;\ \omega_{r,\beta,n,t} = \omega_{r,\beta,n},\ \forall r, \beta, n;$$
$$\mu_{f,\beta,n,t} = \frac{e_n S_\beta^f}{\sigma} \ln \frac{1+\varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta} + \varepsilon},\ \forall f, \beta, n.$$

5.1.5 Bounding

We are now ready to bound total cost returned by Algorithm 1. We respectively bound the sum of operational cost and video streaming delay cost as "static cost", and deployment cost as "dynamic cost". Accordingly, we have the two following lemmas.

Lemma 1. *The sum of operational cost and video streaming delay cost achieved by Algorithm 1 is no larger than D.*

Proof. To bound the sum of operational and video streaming delay cost, we have following equations:

$$H_1 \sum_{r,\beta,t} \widetilde{X}_{n,t}^{r,\beta} \cdot (S_\beta^f - S_b^f)\Delta_n + H_1 \sum_{f,\beta,n,t} \widetilde{Y}_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Phi_n + H_3 \sum_{r,n,\beta,t} (T(n,u) + \Gamma_n(f,b,\beta))\widetilde{X}_{n,t}^{r,\beta} \tag{12a}$$

$$= \sum_t (\alpha_r - H_2 \sum_{r,n} \frac{e_n}{\sigma} \ln \frac{\widetilde{Y}_{n,t}^{f,\beta} + \varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta} + \varepsilon} - \pi_{f,r,\beta,n,t} - \omega_{r,\beta,n,t})\widetilde{X}_{n,t}^{r,\beta} + \lambda(n) \cdot \Phi_n \tag{12b}$$

$$\le \sum_t (\alpha_{r,t} - \omega_{r,\beta,n,t} + \lambda(n)\Phi_n) \tag{12c}$$

$$\le \sum_{r,t} \alpha_{r,t} - \sum_{n,t} \theta_{n,t}\lambda(n) - \sum_{r,\beta,n,t} \omega_{r,\beta,n,t} \tag{12d}$$

$$= D \tag{12e}$$

(12b) holds because of (11a) and (11e). (12c) follows from (11c) and $\sum_t \widetilde{Y}_{n,t}^{f,\beta} \ln \frac{\widetilde{Y}_{n,t}^{f,\beta} + \varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta} + \varepsilon} \ge 0$, which will be proved

later. (12d) is due to (11b). Similar to [34], we are ready to prove $\sum_t \widetilde{Y}_{n,t}^{f,\beta} \ln \frac{\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta}+\varepsilon} \geq 0$ as follows:

$$
\begin{aligned}
&\sum_t \widetilde{Y}_{n,t}^{f,\beta} \ln \frac{\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta}+\varepsilon} \\
=&\sum_t (\widetilde{X}_{n,t}^{r,\beta}+\varepsilon) \ln \frac{\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta}+\varepsilon} - \sum_t \varepsilon \ln \frac{\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon}{\widetilde{Y}_{n,t-1}^{f,\beta}+\varepsilon} \\
\overset{\diamond}{\geq}&(\sum_t (\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon)) \ln \frac{\sum_t (\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon)}{\sum_t (\widetilde{X}_{n,t-1}^{r,\beta}+\varepsilon)} + (\widetilde{Y}_{n,0}^{f,\beta}+\varepsilon) \ln \frac{\widetilde{Y}_{n,0}^{f,\beta}+\varepsilon}{\widetilde{Y}_{n,T}^{f,\beta}+\varepsilon} \\
\geq&\sum_t (\widetilde{Y}_{n,t}^{f,\beta}+\varepsilon) - \sum_t (\widetilde{Y}_{n,t-1}^{f,\beta}+\varepsilon) + \widetilde{Y}_{n,0}^{f,\beta} - \widetilde{Y}_{n,T}^{f,\beta} \\
=&0
\end{aligned}
$$

where $\diamond$ comes from the fact that $(\sum_n p_n) \ln \frac{\sum_n p_n}{\sum_n q_n} \leq \sum_n p_n \ln \frac{p_n}{q_n}$ and $p - q \leq p \ln \frac{p}{q}$. □

Lemma 2. *The deployment cost achieved by Algorithm 1 is no larger than* $(1+\varepsilon)\ln(1+\frac{1}{\varepsilon})D$.

Proof. We set that $\eta = (1+\varepsilon)\sigma$ and $\mathcal{N}' \overset{\text{def}}{=} \left\{ n \in \mathcal{N} | \widetilde{X}_{n,t}^{r,\beta} > \widetilde{X}_{n,t-1}^{r,\beta} \right\}$. We now bound the deployment cost as follows:

$$
\begin{aligned}
&H_2 \sum_{f,n,t} e_n S_\beta^f (\widetilde{Y}_{n,t}^{f,\beta} - \widetilde{X}_{n,t-1}^{r,\beta})^+ \\
\leq& H_2 \sum_{t,f} \sum_{n \in \mathcal{N}'} e_n S_\beta^f (\widetilde{Y}_{n,t}^{f,\beta} - \widetilde{Y}_{n,t-1}^{f,\beta}) && \text{(13a)} \\
\leq& H_2 \sum_{t,f} \sum_{n \in \mathcal{N}'} e_n S_\beta^f (Y_{n,t}^{f,\beta}+\varepsilon) \ln \frac{Y_{n,t}^{f,\beta}+\varepsilon}{Y_{n,t-1}^{f,\beta}+\varepsilon} && \text{(13b)} \\
\leq& \eta H_2 \sum_{t,f} \sum_{n \in \mathcal{N}'} \frac{e_n S_\beta^f}{\sigma} \ln \frac{Y_{n,t}^{f,\beta}+\varepsilon}{Y_{n,t-1}^{f,\beta}+\varepsilon} && \text{(13c)} \\
=& \eta \sum_{t,f} \sum_{n \in \mathcal{N}'} (\alpha_{r,t} - \pi_{r,\beta,n,f,t} - \omega_{r,\beta,n,t} - H_1 \sum_{r,\beta} (S_\beta^f - S_b^f)\Delta_n \\
&- H_3 \sum_{r,n,\beta} T(n,u) - H_3 \Gamma_n(f,b,\beta) && \text{(13d)} \\
\leq& \eta \sum_{t,r} \sum_{n \in \mathcal{N}'} (\alpha_{r,t} - \pi_{r,\beta,n,f,t} - \omega_{r,\beta,n,t}) && \text{(13e)} \\
\leq& \eta \left(\sum_{r,t} \alpha_{r,t} - \sum_{n,t} \theta_{n,t} \lambda(n) - \sum_{r,\beta,n,t} \omega_{r,\beta,n,t} \right) && \text{(13f)} \\
=&(1+\varepsilon)\ln(1+\frac{1}{\varepsilon})D && \text{(13g)}
\end{aligned}
$$

(13a) follows from the definition of $\mathcal{N}'$. (13b) is due to the fact that $a - b \leq a \ln \frac{a}{b}$. (13c) follows from the definition of η and σ. (13d) follows from (11a). (13e) follows from $\sum_{r,n,\beta} T(n,u) + \gamma_n(f,b,\beta) \geq 0$ and $\sum_{r,\beta} (S_\beta^f - S_b^f)\Delta_n \geq 0$. (13f) holds because of (11b), (11d)-(11f). More specifically, we have $\pi_{f,r,\beta,n} \geq \sum_{f,\beta,n} S_\beta^f \theta_{n,t}$ in (11b). From (11d)-(11f) we obtain $\lambda(n) = \sum_{f,\beta,n} S_\beta^f$. In all, we arrive at $\pi_{f,r,\beta,n} \geq \lambda(n) \cdot \theta_{n,t}$, and therefore (13f) holds. □

Theorem 1. *Algorithm 1 can achieve an overall cost no larger than CR_1 times of the offline optimum, where $CR_1 = 1 + (1+\varepsilon)\ln(1+\frac{1}{\varepsilon})$.*

Proof. The proof follows from Lemma 1 and 2. □

5.2 Integrality Gap

We analyze the integrality gap of Algorithm 2. We respectively bound the expectation of three kinds of costs considered in $\mathbf{P}'$. To that end, we first upper-bound $\sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} S_\beta^f$. Based on it, we draw connection from all the terms in three kinds of costs and derive bounds.

Lemma 3. *For the random variable $\overline{Y}_{n,t}^{f,\beta}, \forall f, \beta, n, t$ and every possible value it can take, we have: $\sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} S_\beta^f \leq \tau P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t)$, where $\tau = \frac{(1+\frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f})}{\min_n \Phi_n}$.*

Proof.

$$
\begin{aligned}
&\sum_t \sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} S_\beta^f \\
=&\sum_t \sum_f \sum_\beta \sum_{n \in \mathcal{I}_t^\beta \setminus \mathcal{I}'^\beta_t} \overline{Y}_{n,t}^{f,\beta} S_\beta^f + \sum_t \sum_f \sum_\beta \sum_{n \in \mathcal{I}'^\beta_t} S_\beta^f && \text{(14a)} \\
\leq& \sum_t (\sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f + \max_{f,\beta} S_\beta^f) && \text{(14b)} \\
\leq& \sum_t (\sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f + \sum_{r,n,\beta} \widetilde{X}_{n,t}^{r,\beta} \max_{f,\beta} S_\beta^f) && \text{(14c)} \\
\leq& \sum_t (\sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f + \sum_{f,n,\beta} \widetilde{Y}_{n,t}^{f,\beta} \max_{f,\beta} S_\beta^f) && \text{(14d)} \\
\leq& \sum_t (\sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f + \frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f} \sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f) && \text{(14e)} \\
=&(1 + \frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f}) \sum_t \sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f && \text{(14f)} \\
\leq& \frac{(1 + \frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f})}{\min_n \Phi_n} \sum_t \sum_{f,\beta,n} \widetilde{Y}_{n,t}^{f,\beta} S_\beta^f \Phi_n && \text{(14g)} \\
\leq& \frac{(1 + \frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f})}{\min_n \Phi_n} P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t) && \text{(14h)} \\
\leq& \frac{(1 + \frac{\max_{f,\beta} S_\beta^f}{\min_{f,\beta} S_\beta^f})}{\min_n \Phi_n} P(\mathbf{X}_\mathbf{t}, \mathbf{Y}_\mathbf{t}, \forall t) && \text{(14i)}
\end{aligned}
$$

(14a) holds because of the definition of $\mathcal{I}_t^\beta$ and $\mathcal{I}'^\beta_t$ in Algorithm 2. Since there is at most one element in $\mathcal{I}'^\beta_t$ after executing Algorithm 2, we reach (14b). (14c) follows from constraint (5a), and (14d) is by constraint (5b). (14e)-(14g) is due to some simple algebra, and (14h) holds due to the definition of objective function P'. (14i) holds because the optimal solution of the relaxed problem provides a lower bound of optimal solution of the original problem. □

Theorem 2. *Algorithm 2 ensures that $E(P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t)) \leq CR_2 \cdot P'(\overline{\mathbf{X}}_\mathbf{t}, \overline{\mathbf{Y}}_\mathbf{t}, \forall t)$, where $CR_2 = \delta_o + \delta_d + \delta_w$ and*

$$\delta_o = \max_n (\Delta_n + \Phi_n) \cdot \tau$$

$$\delta_d = \max_n e_n \cdot \tau$$

$$\delta_w = \frac{\max_{n,b,\beta} \mathcal{T}_{f,b,\beta}^{n,u}}{\min_{f,\beta} S_\beta^f} \cdot \tau$$

Proof. We respectively bound the operational cost, deployment cost and delay cost. We start with operational cost and prove δ_o.

$$E(\sum_t C_O(t))$$
$$\leq \sum_t (\sum_r \sum_\beta \overline{X}_{n,t}^{r,\beta} \cdot S_\beta^f \cdot \Delta_n + \sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Phi_n) \tag{15a}$$
$$\leq \sum_t (\sum_f \sum_\beta \overline{Y}_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Delta_n + \sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} \cdot S_\beta^f \cdot \Phi_n) \tag{15b}$$
$$\leq \max_n(\Delta_n + \Phi_n) \sum_t \sum_f \sum_\beta \sum_n \overline{Y}_{n,t}^{f,\beta} \cdot S_\beta^f \tag{15c}$$
$$\leq \max_n(\Delta_n + \Phi_n) \cdot \tau \cdot P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t) \tag{15d}$$
$$\leq \max_n(\Delta_n + \Phi_n) \cdot \tau \cdot P(\mathbf{X}_\mathbf{t}, \mathbf{Y}_\mathbf{t}, \forall t) \tag{15e}$$

(15a) is by definition of operational cost in Eq. (1). (15b) follows from Constraint (5b). (15c) is due to simple algebra, and (15d) holds according to Lemma 3. (15d) follows from the fact that the optimal solution of the relaxed problem provides a lower bound of optimal solution of the original problem. We next bound deployment cost and prove δ_d.

$$E(\sum_t C_D(t))$$
$$\leq \sum_t \sum_f \sum_n e_n S_\beta^f \overline{Y}_{n,t}^{f,\beta} \tag{16a}$$
$$\leq \max_n e_n \sum_t \sum_f \sum_n \overline{Y}_{n,t}^{f,\beta} S_\beta^f \tag{16b}$$
$$\leq \max_n e_n \cdot \tau \cdot P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t) \tag{16c}$$
$$\leq \max_n e_n \cdot \tau \cdot P(\mathbf{X}_\mathbf{t}, \mathbf{Y}_\mathbf{t}, \forall t) \tag{16d}$$

(16a) uses the definition of deployment cost in Eq. (2). (16b)-(16c) follows analogously from (15b)-(15d).

Finally, we will bound delay cost and prove δ_w. We denote $\mathcal{T}_{f,b,\beta}^{n,u} = T(n,u) + \Gamma_n(f,b,\beta)$ for simplicity.

$$E(\sum_t C_W(t))$$
$$\leq \max_{n,b,\beta} \mathcal{T}_{f,b,\beta}^{n,u} \cdot \sum_t \sum_{f,\beta,n} \overline{Y}_{n,t}^{f,\beta} \tag{17a}$$
$$\leq \frac{\max_{n,b,\beta} \mathcal{T}_{f,b,\beta}^{n,u}}{\min_{f,\beta} S_\beta^f} \cdot \sum_t \sum_{f,\beta,n} \overline{Y}_{n,t}^{f,\beta} S_\beta^f \tag{17b}$$
$$\leq \frac{\max_{n,b,\beta} \mathcal{T}_{f,b,\beta}^{n,u}}{\min_{f,\beta} S_\beta^f} \cdot \tau \cdot P'(\widetilde{\mathbf{X}}_\mathbf{t}, \widetilde{\mathbf{Y}}_\mathbf{t}, \forall t) \tag{17c}$$
$$\leq \frac{\max_{n,b,\beta} \mathcal{T}_{f,b,\beta}^{n,u}}{\min_{f,\beta} S_\beta^f} \cdot \tau \cdot P(\mathbf{X}_\mathbf{t}, \mathbf{Y}_\mathbf{t}, \forall t) \tag{17d}$$

The proof of (17a)-(17c) follows analogously from (16a)-(16d) and we therefore omit it. □

5.3 The overall competitive ratio

Based on the derived competitive ratio of Algorithm 1 and integrality gap of Algorithm 2, we can now get the overall competitive ratio of ORFC in Theorem 3.

Theorem 3. *The overall competitive ratio of ORFC is $CR = CR_1 \cdot CR_2$, where CR_1 and CR_2 are derived in Theorem 1 and 2, respectively. That is, the overall cost achieved by ORFC is no larger than CR times of the cost returned by the offline optimum solution.*

6 Performance Evaluation

6.1 Evaluation Setup

We first generate one MEC network topology with a total network nodes $N = 8$ which consists of $N_e = 7$ edge nodes and $N_c = 1$ CDN server. The capacity of each edge node is set to 7 *Gb* first and we assume the CDN server has unlimited storage. The delay between each edge node pair is randomly picked in $[0.01, 0.05]$ seconds, and the delay between edge node and CDN server falls in the range of $[0.1, 0.15]$ seconds, considering that the cloud is geographically far away from edge clouds, which is similar to [35].

There are in total $|F| = 12$ video files with $|B| = 5$ bitrate levels, which are 360p, 480p, 720p, 1080p and 1440p. The transcoding delay for a video file with different bitrate levels is randomly chosen in $[0.01, 0.05]$ seconds. The video access pattern is assumed to follow the Zipf distribution according to [36], and each video file f_i ($1 \leq i \leq 12$) with bitrate level j ($1 \leq j \leq 5$) has an accessing probability:

$$p_{ij} = \frac{(i*j)^{-z}}{\sum_{1 \leq a \leq A^{a-z}}}, \forall 1 \leq a \leq A \tag{18}$$

where $A = 12*5 = 60$ is the number of all the distinct video files with different bitrate levels and z is set to 0.8 which denotes the aggregation degree of video requests similar to [16]. Without loss of generality, we regard that the video file consists of a series of same-duration chunks (the duration is set 10 seconds in this case), and each 10-seconds chunk takes up 0.5 *Mb* according to [37]. We first set $S_\beta^f \in [3, 10]$ *Gb*, where $\beta = 1440$p. Afterwards, for a same video file f with bitrate level β' in {360p, 480p, 720p, 1080p}, we set $S_{\beta'}^f = \frac{\beta'}{1440} \cdot S_{1440}^f$. Accordingly, we set $\Delta_n \in [0.001, 0.01]$\$ per GB, $\Phi_n \in [0.01, 0.12]$\$ per GB [38], and $e_n \in [1, 1.5]$\$ per GB. Due to the lack of workload trace of public accessible video requests in MEC, we randomly generate 50 requests among all the edge areas in each time slot, and we generate such 50 requests for 100 time slots (rounds) in total. We let all 3 kinds of costs have weight of 1 in the simulation. We set $\varepsilon = 0.001$ in ORFC, and compare it with the following algorithms[4]:

- Online Randomized Rounding (OnRR): It first executes Algorithm 1 to obtain online fractional solutions, and then applies traditional randomized rounding to convert the fractional solutions to integer solutions. In case a rounded solution for request r violates the edge node's capacity constraint, r will be delivered to the CDN server.

4. The offline optimum solution consumes extreme long running time (more than 1 day) without returning a feasible solution. We therefore do not present the offline optimum solution's results in the current simulation setup. Later, we will accordingly reduce the problem size to obtain the performance of the offline optimum algorithm to provide the ground truth.

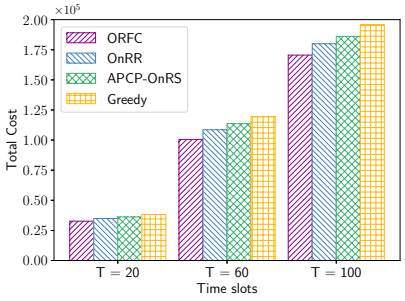


Fig. 4: Overall total cost under different time slots.

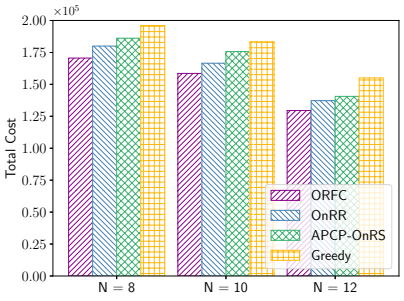


Fig. 5: Overall total cost under different number of edge clouds.

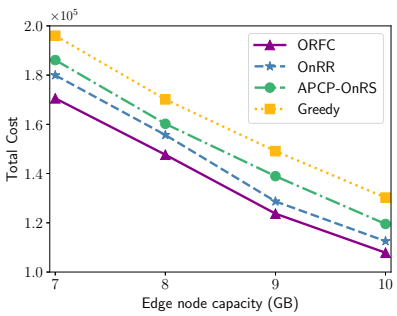


Fig. 6: Overall total cost under different capacities.

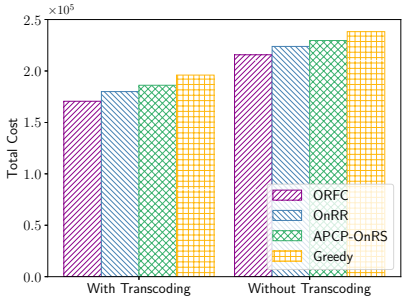


Fig. 7: Overall total cost with and without transcoding.

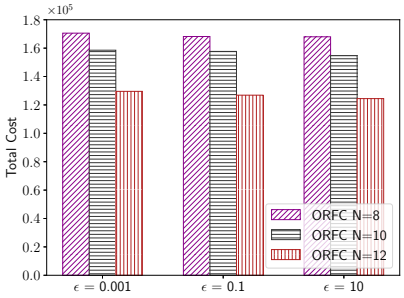


Fig. 8: The impact of ε on ORFC.

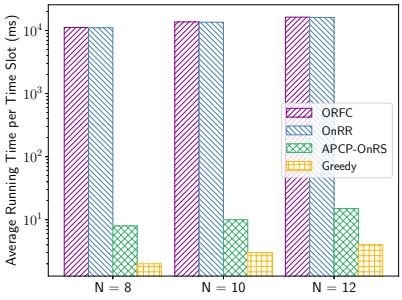


Fig. 9: Running time per time slot.

- APCP-OnRS [19]: This is a two-step algorithm. In each time slot, APCP iteratively selects the most popular video file and places it in the most appropriate edge cloud in order to maximize the total cost. After that, OnRS accommodates each request by using the video file cached in the priority of (1) local edge cloud, (2) neighbor edge cloud, and (3) CDN server.
- Greedy: In each edge area n, it sequentially accommodates each request r by caching its required video file f with requested bitrate b, if such bitrate level video file is not stored on n and the capacity of n is still sufficient. If the capacity of n is full, Greedy will try to use the nearest edge node to accommodate r, otherwise r will be delivered to CDN server.

The simulations are run on a high-performance desktop PC with 8 core 3.40GHz Intel(R) Core(TM) i7-6700 processors and 16 GB memory. We use Matlab CVX [39] to implement ORFC and OnRR, and use C# to implement APCP-OnRS and Greedy.

6.2 Evaluation Results

We first evaluate the overall total cost achieved by all the algorithms with different time slots. Fig. 4 shows that ORFC can always achieve the minimum total cost, followed by OnRR and APCP-OnRS, and Greedy returns the largest total cost. We calculate that ORFC outperforms them in terms of cost savings by 9.5% and 17.5%, respectively. It is worthwhile to mention that with more time slots by setting $T = 200$, we rerun simulations and found that ORFC can outperform them regarding the cost savings by around 12% and 21%. We omit showing this result in the figure for brevity. APCP-OnRS and Greedy behave worse than ORFC and OnRR, since both of them do not essentially capture the "online" nature of the problem, and only "greedily" or "locally" make the caching decisions. On the one hand, the adopted strategies for APCP-OnRS and Greedy cannot guarantee to return (sub)optimal solution in the current (one) time slot, since they are two heuristics and do not provide performance-guaranteed solution. On the other hand, both of them pay attention on caching video files for only current time slot, and they ignore the influence of current caching decision on the future time slots. This leads to the inefficiency of these two heuristics, and makes them to achieve lower total cost. On the contrary, our proposed ORFC (1) applies regularization-based technique to get the online per-time slot fractional solutions with proved competitive ratio, and then (2) leverages randomized dependent rounding technology to return the integer solution with theoretical approximation ratio. By combining these two procedures, ORFC is able to make online decisions with guaranteed performance. This is the reason why ORFC minimum cost consumption and also Fig. 4 verifies its efficiency. Because of applying randomized dependent rounding scheme, ORFC outperforms OnRR, which uses traditional randomized rounding mechanism. It therefore shows the superiority of randomized dependent rounding.

We next show the performance of all the algorithms when we respectively set the number of network nodes N to be 6 (with $N_e = 5$ and $N_c = 1$), 8 (with $N_e = 7$ and $N_c = 1$) and 10 (with $N_e = 9$ and $N_c = 1$), but set the number of requests to be 50 in all this cases (unchanged). We see from Fig. 5 that when the number of network nodes increases,

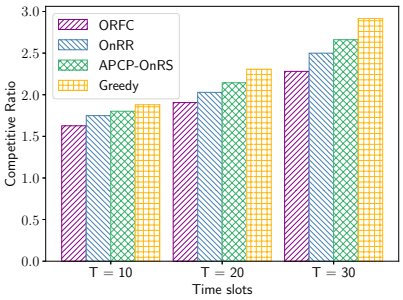


Fig. 10: Competitive ratio.

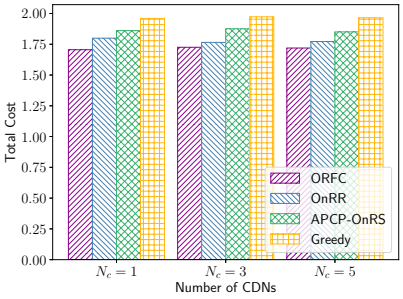


Fig. 11: The total cost for different number of CDNs.

the overall total cost of all the algorithms decreases. The reason is that when the number of network nodes increases, more requested video files can be cached on edge nodes. This saves more delay cost and leads to less total cost for all the algorithms. For the similar reason, when we keep the number of network nodes unchanged ($N = 8$) but vary the capacity of edge nodes, Fig. 6 illustrates that the total cost achieved by all the algorithms decreases when the capacity of edge node increases. Nevertheless, ORFC can always achieve the minimum total cost in these two figures, which also verifies its effectiveness and efficiency.

Fig. 7 shows the overall total cost of all the algorithms with and without transcoding when $N = 8$. With the aid of transcoding, the requests which access the same video file (with different bitrate levels) can be accommodated by using the same video file with a higher bitrate level. This saves more capacity for edge cloud and reduces transmission delay cost of placing the video files on CDN server for the case when transcoding is not allowed. Fig. 8 shows that when we increase ε, the overall total cost of ORFC decreases slightly. Although this corresponds to Lemma 2, we find that the effect of ε on the overall performance is very limited. Finally, Fig. 9 gives the average running time per time slot (in log scale) of all the algorithms with different number of network nodes. It can be seen that ORFC for $N = 12$ takes less than 17 seconds. We believe with more advanced servers, the running time of ORFC can be further reduced.

In the above simulation setup, the offline optimum algorithm cannot return a feasible solution in a reasonable time due to the relative large problem input size. We therefore reduce the problem input size by setting $N = 4$, $|R| = 25$ and $|T| = 30$. By doing this, the offline optimum algorithm can return the feasible solution in order to provide a ground truth. We show in Fig. 10 the competitive ratio of all the compared algorithms. We see from the figure that with the number of time slots increases, the competitive ratio of all the algorithms increases accordingly. Nevertheless, ORFC can always achieve the minimum competitive ratio value (i.e., the most close performance with offline optimum), which confirms its superiority. Moreover, we also increase the number of CDN servers from 1 to 3 and 5, respectively, and draw links between the CDN sever and all the other edge clouds following the similar way to the above simulation setup. We see from Fig 11 that the number of CDN servers have very tiny impact on the performance of all the algorithms. For different number of CND servers, all the algorithms return the similar total cost. This can be explained that all the CDN servers are assumed to have infinite storage capacity, so it makes no difference in placing video files on any specific CDN server.

7 Conclusion

In this paper, we have presented the online orchestration framework for multi-bitrate video caching in Edge Computing. The online orchestration framework consists of two main components, namely, (i) an online regularization-based video caching fractional algorithm that decomposes the original problem into a series of one-shot optimization subproblems, and (ii) a randomized dependent rounding scheme that converts the fractional solutions for the regularized subproblems to integer solution. We have rigorously proved that the online orchestration framework can achieve an upper-bounded competitive ratio compared to the offline optimum. The extensive simulation results confirm the superiority of our proposed online orchestration framework over alternative algorithms. In our future work, we will implement our proposed algorithm in a large-scale real experimental testbed. We are also interested in predicting the traffic information and incorporate the traffic prediction algorithm in the proposed online algorithm.

Acknowledgments

The authors would like to thank the anonymous reviewers and the editor for their valuable suggestions. The work is partially supported by the National Natural Science Foundation of China under Grants No. 62172038 and No. 61802018, by Beijing Institute of Technology Research Fund Program for Young Scholars, by the U.S. National Science Foundation under Grant CNS-2047719, by Hong Kong RGC TRS under Grant T-41-603/20R and by Hong Kong GRC GRF PolyU under Grant 15217919. Song Yang is the corresponding author.

References

[1] "Cisco visual networking index (vni) global and americas/emear mobile data traffic forecast, 20172022," 2019. [Online]. Available: https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/190320-mobility-ckn.pdf

[2] J. Wu, R. Tan, and M. Wang, "Streaming high-definition real-time video to mobile devices with partially reliable transfer," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 458–472, 2018.

[3] E. Ghabashneh and S. Rao, "Exploring the interplay between CDN caching and video streaming performance," in *IEEE INFOCOM*, 2020, pp. 516–525.

[4] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee, "Awstream: Adaptive wide-area streaming analytics," in *ACM SIGCOMM*, 2018, pp. 236–252.

[5] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[6] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *IEEE INFOCOM*, 2020.

[7] B. Jedari, G. Premsankar, G. Illahi, M. Di Francesco, A. Mehrabi, and A. Ylä-Jääski, "Video caching, analytics, and delivery at the wireless edge: A survey and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 1, pp. 431–471, 2020.

[8] X. Jiang, F. R. Yu, T. Song, and V. C. Leung, "A survey on multi-access edge computing applied to video streaming: Some research issues and challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[9] C. M. Machuca, O. Moe, and M. Jäger, "Impact of protection schemes and network component's availability on operational expenditures," *Journal of Optical Networking*, vol. 7, no. 2, pp. 142–150, 2008.

[10] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, 2018.

[11] N. Chen, W. Xing, D. Zhang, M. Guo, and L. Gao, "Multi-bitrate video caching and processing in edge computing: A stackelberg game approach," in *IEEE ICC*, 2020, pp. 1–6.

[12] 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*, vol. 37, no. 7, pp. 1632–1641, 2019.

[13] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao, "Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics," in *IEEE INFOCOM*, 2020, pp. 1–10.

[14] N. Buchbinder, S. Chen, and J. Naor, "Competitive analysis via regularization," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014, pp. 436–444.

[15] R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, "Dependent rounding and its applications to approximation algorithms," *Journal of the ACM*, vol. 53, no. 3, pp. 324–360, 2006.

[16] Z. Qu, B. Ye, B. Tang, S. Guo, S. Lu, and W. Zhuang, "Cooperative caching for multiple bitrate videos in small cell edges," *IEEE Transactions on Mobile Computing*, vol. 19, no. 2, pp. 288–299, 2020.

[17] F. Wang, C. Zhang, F. Wang, J. Liu, Y. Zhu, H. Pang, and L. Sun, "Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized QoE," in *IEEE INFOCOM*, 2019, pp. 910–918.

[18] F. Wang, F. Wang, J. Liu, R. Shea, and L. Sun, "Intelligent video caching at network edge: A multi-agent deep reinforcement learning approach," in *IEEE INFOCOM*, 2020.

[19] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, 2019.

[20] R. Li, L. Wang, Y. Gong, M. Song, M. Pan, and Z. Han, "Dynamic cache placement, node association, and power allocation in fog aided networks," in *IEEE Global Communications Conference*, 2019, pp. 1–6.

[21] X. Xia, F. Chen, Q. He, J. Grundy, M. Abdelrazek, and H. Jin, "Online collaborative data caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 2, pp. 281–294, 2021.

[22] Y. Chiang, C.-H. Hsu, and H.-Y. Wei, "Collaborative social-aware and QoE-driven video caching and adaptation in edge network," *IEEE Transactions on Multimedia*, pp. 1–15, 2021.

[23] A. Mehrabi, M. Siekkinen, and A. Yl-Jski, "Edge computing assisted adaptive mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 787–800, 2019.

[24] P. K. Mu, J. Zheng, T. H. Luan, L. Zhu, M. Dong, and Z. Su, "Amis: Edge computing based adaptive mobile video streaming," in *IEEE INFOCOM*, 2021, pp. 1–10.

[25] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 12, pp. 2833–2849, 2020.

[26] A. Galanopoulos, J. A. Ayala-Romero, D. J. Leith, and G. Iosifidis, "AutoML for video analytics with edge computing," in *IEEE INFOCOM*, 2021, pp. 1–10.

[27] M. Choi, A. No, M. Ji, and J. Kim, "Markov decision policies for dynamic video delivery in wireless caching networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 12, pp. 5705–5718, 2019.

[28] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *ACM SIGCOMM*, 2013, pp. 15–26.

[29] Y. Jin, Y. Wen, and C. Westphal, "Optimal transcoding and caching for adaptive streaming in media cloud: An analytical approach," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 12, pp. 1914–1925, 2015.

[30] Y. Ma, T. Nandagopal, K. P. Puttaswamy, and S. Banerjee, "An ensemble of replication and erasure codes for cloud file systems," in *IEEE INFOCOM*, 2013, pp. 1276–1284.

[31] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[32] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman & Co., 1979.

[33] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[34] W. You, L. Jiao, J. Li, and R. Zhou, "Scheduling DDoS cloud scrubbing in ISP networks via randomized online auctions," in *IEEE INFOCOM*, 2020, pp. 1658–1667.

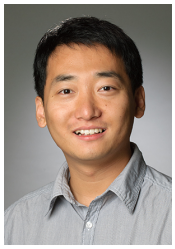
[35] B. P. Rimal, D. P. Van, and M. Maier, "Mobile-edge computing versus centralized cloud computing over a converged fiwi access network," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 498–513, 2017.

[36] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 15–28.

[37] J. Summers, T. Brecht, D. Eager, and B. Wong, "To chunk or not to chunk: Implications for http streaming video server performance," in *Proceedings of the 22nd international workshop on Network and Operating System Support for Digital Audio and Video*, 2012, pp. 15–20.

[38] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? distributed service caching in mobile edge clouds," in *IEEE INFOCOM*, 2020, pp. 1–10.

[39] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.2," http://cvxr.com/cvx, 2020.



Song Yang received the Ph.D. degree from Delft University of Technology, The Netherlands, in 2015. From August 2015 to July 2017, he worked as postdoc researcher for the EU FP7 Marie Curie Actions CleanSky Project in Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG), Göttingen, Germany. He is currently an associate professor at School of Computer Science and Technology in Beijing Institute of Technology, China. His research interests focus on data communication networks, cloud/edge computing and network function virtualization. He is a member of IEEE and ACM.

Lei Jiao received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is currently an assistant professor at the Department of Computer and Information Science, University of Oregon, USA. Previously he worked as a member of technical staff at Alcatel-Lucent/Nokia Bell Labs in Dublin, Ireland and also as a researcher at IBM Research in Beijing, China. He is interested in the mathematics of optimization, control, learning, and mechanism design, applied to computer and telecommunication systems, networks, and services. He publishes papers in journals such as IEEE/ACM Transactions on Networking, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Mobile Computing, and IEEE Journal on Selected Areas in Communications, and in conferences such as INFOCOM, MOBIHOC, ICNP, and ICDCS. He is a recipient of the NSF CAREER Award. He also received the Best Paper Awards of IEEE LANMAN 2013 and IEEE CNS 2019, and the 2016 Alcatel-Lucent Bell Labs UK and Ireland Recognition Award. He served as a guest editor for IEEE JSAC. He was on the program committees of conferences including INFOCOM, MOBIHOC, ICDCS, and IWQoS, and was also the program chair of multiple workshops with INFOCOM and ICDCS.

Ramin Yahyapour is full professor at the Georg-August University of Göttingen. He is also managing director of the GWDG, a joint compute and IT competence center of the university and the Max Planck Society. Dr. Yahyapour holds a doctoral degree in Electrical Engineering and his research interest lies in the area of efficient resource allocation in its application to service-oriented infrastructures, clouds, and data management. He is especially interested in data and computing services for eScience. He gives lectures on parallel processing systems, service computing, distributed systems, cloud computing, and grid technologies. He was and is active in several national and international research projects. Ramin Yahyapour serves regularly as reviewer for funding agencies and consultant for IT organizations. He is organizer and program committee member of conferences and workshops as well as reviewer for journals.

Jiannong Cao received the BSc degree in computer science from Nanjing University, China, in 1982, and the MSc and PhD degrees in computer science from Washington State University, Pullman, Washington, in 1986 and 1990, respectively. He is currently the Otto Poon Charitable Foundation professor in data science and the chair professor of distributed and mobile computing with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He is also the director of the Internet and Mobile Computing Lab, Department and the associate director of University Research Facility at Big Data Analytics. His research interests include parallel and distributed computing, wireless networks and mobile computing, big data and cloud computing, pervasive computing, and fault tolerant computing. He has coauthored five books in Mobile Computing and Wireless Sensor Networks, co-edited nine books, and published more than 600 papers in major international journals, and conference proceedings. He is a distinguished member of ACM and a senior member of China Computer Federation (CCF)