Transcoding-Enabled Cloud-Edge-Terminal Collaborative Video Caching: A Online Learning Approach with Unknown Time-Varying Information

Journal:	*IEEE Internet of Things Journal*
Manuscript ID	IoT-29809-2023
Manuscript Type:	Special issue on Cloud-Edge-Terminal Collaboration Enabled AIoT: Services, Technologies, and Applications
Date Submitted by the Author:	13-May-2023
Complete List of Authors:	Xiao, Han; Beijing University of Posts and Telecommunications, Zhuang, Yirong; China Telecom Corporation Limited Xu , Changqiao wang, wendong; Beijing University of Posts and Telecommunications, Zhang, Hongke; Beijing Jiaotong University Ding, Renjie; Beijing University of Posts and Telecommunications Cao, Tengfei; Qinghai University Zhong, Lujie; Capital Normal University, Information Engineering College Muntean, Gabriel-Miro; Dublin City University, Performance Engineering Laboratory
Keywords:	Video Cache, Online Learning, Cloud-Edge-Terminal, Internet of Thing

Transcoding-Enabled Cloud-Edge-Terminal Collaborative Video Caching: A Online Learning Approach with Unknown Time-Varying Information

Han Xiao, Yirong Zhuang, Changqiao Xu, *Senior Member, IEEE,* Wendong Wang, Honeke Zhang, *Fellow, IEEE,* Renjie Ding, Tengfei Cao, Lujie Zhong, and Gabriel-Miro Muntean *Fellow, IEEE*

***Abstract*—As a key enabling technology in intelligent Internet of Thing(IoT), edge caching provides important support for reducing core network load and improving network service efficiency, especially for high bandwidth demand services represented by multimedia applications. However, external time-varying information is hard to be obtained comprehensively in complicated IoT environment. Meanwhile, there exists the interchangeability between content data (e.g., videos with different bitrates), which is difficult to make caching decisions online in real-time to achieve fast feedback with low latency and avoid useless deployment. To this end, this paper designs a transcoding-enabled online cache scheme for IoT video service with cloud-edge-terminal collaboration. Firstly, we design a variable bitrate video routing strategy to dynamically retrieve content from cloud/edge according to user demands. Furthermore, the video caching problem is considered as an online convex optimization problem to learn utility gradient and determine the optimal caching strategy in real-time without any prior information. On this basis, we extend the problem to elastic networks with dynamic available resources, and prove the sublinear regret and sublinear constraint violation. Finally, we summarized 5 video request datasets, and carried out differentiated multiple verifications based on different request habits and content requirements. Compared with the most advanced algorithms in terms of delay, we evaluated the performance advantages of the proposed scheme.**

***Index Terms*—Video Cache, Online Learning, Cloud-Edge-Terminal, Internet of Thing**

I. Introduction and Motivation

Edge caching can significantly reduce the delivery distance by storing content on edge servers closer to terminals, and satisfy the request quickly. Benefiting from the advantage, caching is especially suitable for future expanded IoT applications with the requirement of high-bandwidth and low latency represented by video media, and has been considered as a promising technology with great potential to improve network transmission efficiency and quality of experience (QoE). Recently, Apple Inc. has begun to push the actual deployment of edge cache (i.e., Apple Edge Cache, AEC [1]), which becomes a commercial endorsement.

As the most popular IoT services, according to Cisco data report [2], nearly four-fifths of mobile data traffic will be generated from multimedia video. Meanwhile, the existing research in [3] shows that, most of video traffic comes from repeated requests for the same video, especially popular content, e.g. video clips, game replay, etc. This provides greater operation space for caching mechanism. However, with the development of media technology and user requirement, video caching strategy has not always presented a universal solution. The facing challenges mainly include the following aspects.

Interchangeability on requested content. The data in IoT has duplication and potential substitution, e.g., videos with various bitrates. Video is coded to multiple versions (e.g., multi-rate video) to adaptive accommodate dynamic networks [4], [5]. Generally, the content with different qualities is cached independently, although a video with any bitrate can be scheduled to meet the requests for the same video with another bitrate. The conflict between adaptive bitrate and cache technology leads to the waste of resources. For this case, how to balance delivery latency and storage resources become an important issue to be considered in IoT multimedia service.

Unknown time-varying information. The video popularity is dynamic. The requests and network conditions is time-varying [12], [13]. The relevant information is difficult to be obtained comprehensively in real-time, while caching is expected to adapt dynamically to the information mentioned above. Cache point has to determine what should be cached and how to be more profitable in the future without any prior information. It is still a pending issue and has become a significant barrier to design caching mechanism.

Limited resources on edge. The cache points at the edge are often resource-poor due to the severe deployment conditions [6], [7]. On one hand, the missed requests still need to be satisfied from cloud, and the existence of cache module makes no sense here. More crucial is that the cache points at edge often carry multiple types of services (e.g., payment, navigation, etc.). The resources allocated to video services may be elastic. How to make efficient caching decisions with limited resources should be discussed carefully.

The above challenges require that the caching policy should

H. Xiao, C. Xu, W. Wang, R. Ding are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China. E-mail: {xiaohan, cqxu, wdwang, 2021140795}@bupt.edu.cn

Y. Zhuang is with the Research Institute of China Telecom, Ave Zhongshan, Guangzhou 510630, China. E-mail: 13316094433@chinatelecom.cn.

H. Zhang is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China. E-mail: hkzhang@bjtu.edu.cn.

T. Cao is with the School of Computer, Qinghai University, Xining, China. E-mail: tfcao@qhu.edu.cn.

L. Zhong is with the Information Engineering College, Capital Normal University, Beijing 100048, China. E-mail: zhonglj@cnu.edu.cn.

G.-M. Muntean is with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin 9, Ireland. E-mail: gabriel.muntean@dcu.ie.

be designed to accommodate complex environments, schedule resources efficiently, and online cache beneficial videos without prior information. This is difficult, as complex environments naturally affect the efficiency of mechanisms, and time-varying requests conflict with timeliness. To this end, in this paper, we expect to be able to leverage computing resources to bridge substitutable videos, and transcode cached video segments to another bitrate when the discrepancy between request and caching occurred [8]–[10]. Meanwhile, based on real-time terminal requests and cloud state, the edge caching benefit curve should be learned online to evaluate the future caching performance in the absence of complete information. Thus, the caching state is adjusted and updated quickly, and the solution is expected to achieve long-term caching optimization through simple and efficient online policy updates. Specifically, the contributions are summarized as follows:

1) We design a transcode-enabled online caching (**ToC**) architecture, embed the computation module into caching mechanism, and model it as an online convex optimization problem. To measure online performance, regret is introduced to judge the gap between online decisions and the offline optimal decision.
2) We propose a flexible routing policy to adjust the request pattern of terminals. By maintaining a dynamic matching pool, routes are skillfully determined to reduce the overhead of transmission and transcoding.
3) We propose an online caching scheme in a stable IoT network (**ToC-S**) to learn utility gradient and update cache status online without any prior information. Through rigorous mathematical proof, sublinear regret is achieved when the step size is set as $\Delta_x/J\sqrt{T}$.
4) We further extend it into an elastic IoT network with time-varying resources. A partial Lagrangian function is introduced to design a dual update policy (**ToC-E**) and search the saddle points. Strict theoretical proof supports the sublinearity of constraint violation, i.e., $\mathcal{O}\left(T^{\frac{3}{4}}\right)$.
5) We summarized 5 various video request datasets, and carried out differentiated multiple verifications based on different request habits and content requirements. The evaluation results show that ToC has superior learning ability and can achieve better performance than other state-of-the-art schemes under unknown or even adversarial IoT video requests.

This paper is organized as follows. Section I provides background information, motivation, and contribution details. Section II introduces the latest work on IoT video caching. Section III gives the system model and optimization goal. Section IV discusses the property of routing and caching problems. Section V and Section VI design ToC-S and ToC-E, respectively. Finally, Section VII presents the evaluation results and Section VIII concludes.

II. Related Work

This section mainly introduces the latest work on caching and consists of the following three main parts.

A. Video-oriented Caching Policy

IoT video service is the most popular application and suitable for caching, which has been favored by researchers. For example, Chiang $et\ al.$ design a two-layer MEC caching architecture in literature [14], utilizing social information to actively cache popular content. Li $et\ al.$ propose a lifecycle-aware video caching strategy in [15]. The content is clustered according to the lifecycle and integrated into a general caching system, which can significantly reduce cache replacement frequency while maintaining high cache hit ratio. In addition, in [16], Ayoub $et\ al.$ propose a video-on-demand (VoD) service framework, including edge caching and routing to reduce the transmission energy consumption by sharing cache content with each other.

It should be noted that the above research does not distinguish video quality. In this case, video caching strategy is close to the traditional content caching mode(e.g., files). To this end, Zhang $et\ al.$ [17] consider the video caching problem with scalable video coding (SVC), and develop an economical caching strategy. It establishes the correlation between cache overhead and service performance by efficiently scheduling the interaction between base layer and enhancement layer. Jedari $et\ al.$ [18] consider the caching problem of SVC video as an auction between network operators and content providers. The proposal solves the social welfare maximization problem through the iterative trading strategy of double auction, so as to improve the delivery efficiency and maintain the economic utility. In addition, Guo $et\ al.$ [19] consider adaptive bitrate video and propose a dynamic caching scheme with multiple time scales. The video bitrate decision and cache decision are executed on a large time scale, while the data transmission is performed on a small time scale. Our previous work in [9] also discussed the collaborative caching problem of emerging immersive media, which requires the implementation of media processing and the acquisition of prior.

B. Online Caching Policy

Online policy means that decision makers should cache without knowing future information (e.g., popularity), which is the objective condition under actual environment. Some research works treat the scene as static, i.e., the popularity does not change over time, the popularity in the future is consistent with the present popularity, which is difficult to be applied in practice [20]. To this end, Mehrizi $et\ al.$ [21] consider the spatiotemporal dependence of content popularity online, and establish a probabilistic dynamic model of popularity prediction. The model parameters are estimated by variational Bayes, which reduces the network service costs notably. On the other hand, Zhang $et\ al.$ [22] consider the vehicle as the cache carrier, and take the mobility of requester and cache vehicle into account. Further, the authors propose an online Lyapunov-based algorithm to optimize energy consumption and improve cache hit rate.

Meanwhile, with the development of artificial intelligence (AI) technology, caching strategies based on AI have also been studied. Wu $et\ al.$ [23] design a multi-agent reinforcement learning-based caching method, utilizing neural network to

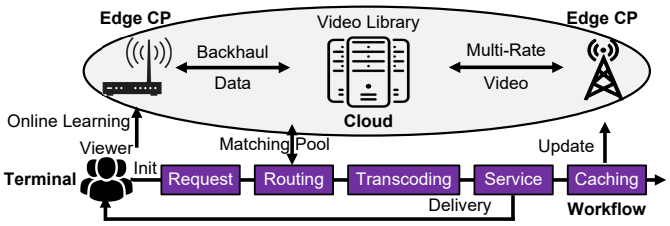

Fig. 1. Scenario and workflow

simulate the real-time cache utility function. The evaluation shows that the service load of small cell with limited storage space is reduced dramatically. Qian *et al*. [24] integrated the recommendation and cache into a framework and expressed it as an average-cost Markov decision problem. Hierarchical reinforcement learning is used to mine the mutual influence relationship between recommendation and cache, so as to maximize the utilization of bandwidth and reduce the total amount of data transmitted. Liu *et al*. [25] propose a privacy-preserving caching policy based on deep deterministic policy gradient algorithm to improve the cache hit rate and meet the privacy protection constraints, which ensures the reliability of cache data. In fact, due to the fixed structure of neural networks, AI-based methods currently face the scalability problem and need a large amount of data to support the training, which brings some troubles to the practical application.

C. Difference with existing works

This paper introduces the online caching problem of substitutable content in dynamic external IoT environment with unknown information, which addresses three issues that have not been properly discussed in the past. First, it embeds a computational transcoding module in caching and bridges video with various bitrates. Second, it can adapt to external information dynamically, including differentiated user needs, time-varying network conditions, and resource status, etc. Third, it can achieve fast, online and low latency feedback in a simple and effective manner, and has theoretical guarantees.

III. Model and Goal

In this section, the involved models and the optimization goal are introduced as follows. The related scenario and workflow is shown in Fig. 1.

Note: This paper adopts lowercase *italic* symbols as scalars. The uppercase symbols express constants. The **bold** symbols are vectors. The calligraphic symbols demonstrate variable space. The norm $|\cdot|$ denotes the cardinality of set or vector. The symbols with the subscript t represent the value at that slot, otherwise it is the general attribute.

A. Network Model

IoT video service is considered as a discrete system with time slot in this paper. It mainly includes the following participants.

Cloud is the content provider, which stores processed video clips. When receiving data requests, it starts the transmission process and delivers the video content to edge close to terminals.

Cache point (CP) is the powerful edge node with storage and computing capacity, which can connect cloud and provide services to terminals. For convenience, CPs are organized as $\mathcal{M} = \{1, ..., m, ..., M\}$. The storage space size and computing power of any node m are denoted as C_m and H_m, respectively.

Terminal corresponds to a video requester, i.e., viewer, which can communicate with CP or cloud to obtain content. The requests are generated according to dynamic interactive behavior. In addition, it deploys an adaptive bitrate (ABR) module [27], [28] to generate the bitrate demands. For clarity, the terminal set is denoted as $\mathcal{N} = \{1, ...n, ..., N\}$.

The network scenario is designed based on the traditional bipartite cache network [29]. Two participants have a many-to-many mapping, i.e. one cache point m serves multiple terminals and one terminal n can access multiple CPs to obtain content. Let $l_{mn} = 1$ be the positive connection relationship between m and n, otherwise $l_{mn} = 0$.

B. Content Model

The video library is denoted as $\mathcal{K} = \{1, ..., k, ..., K\}$, and each content is encoded to multi-rate for accommodating dynamic networks when uploading. The quality set is expressed as $\mathcal{B}_k = \{1, ..., b, ..., B\}$. The video with higher bitrate can provide terminals with higher QoE. The request initiated by the viewer n is expressed as $\{u_k, u_b\}$, where $u_k \in \mathcal{K}$ is the requested content and $u_b \in B_k$ is the required bitrate level. Therefore, the requested content space is denoted as $\mathcal{U} = \{u_k \in \mathcal{K}, u_b \in B_k\}$. The request sequence of the system is expressed as $\{u_t\}_{t=1}^{T}$. The requests from viewers are generally related to the popularity, which is highly dynamic and completely unknown to CPs. Even there exist adversarial requests, which go against the popularity.

C. Cache Model

Multi-rate video is cached through maximum distance separable (MDS) codes [31]. The content is split into several interrelated pieces and organized by random linear combinations. Terminals can decode original video after receiving enough amount of pieces. For clarity, we denote the cache state of any CP m as x_t. Each element $x_m^{k,b} \in [0, 1]$ denotes the amount of random coded pieces, and the feasible region is denoted as $\mathcal{D}$. $s^{k,b}$ is the unit size of video k with bitrate b. The cache space of any CP m can be expressed as $\mathcal{X} = \{x_m^{k,b} | \sum_{k=1}^{K} \sum_{b=1}^{B} x_m^{k,b} s^{k,b} \leq C_m, \forall m, \forall k, \forall b\}$, where C_m is the storage space of CP m. Thus, the cache policy is represented as: $\mu : \{\mathcal{U} \times \mathcal{X}\}_t \rightarrow \boldsymbol{x_{t+1}}$.

It should be noted that when the bitrate level of cached video is higher than the requested quality u_b, CP can achieve fast transcoding at edge by scheduling computing resources and providing services. To measure the computational cost, the required computing resource is denoted as $\zeta||b - u_b||^{\epsilon}$, where $||b - u_b||$ is the difference between the routed version b, ϵ is the computing density and ζ is considered as the scaling factor.

D. Optimization Goal

Considering the transcoding-embedding caching function at edge, we design the service utility from two parts: (1) the traffic offloaded from cloud to edge, (2) the transcoding cost related to terminal requests. The utility is denoted as follows.

$$f_t(x_t) = \sum_{b=1}^{B} l_{m,n} z_{t,m,n}^{k,b} s^{k,b} \left(w_{m,n}^{k,b} - \zeta ||b - u_b||^{\epsilon} \right) \quad (1)$$

where $z_{t,m,n}^{k,b} \in [0, x_{t,m}^{k,b}]$ is the amount data of routing and $l_{m,n}$ is the indicator about the connectivity. $w_{m,n}^{k,b}$ is the delivery utility, which can be measured based on RTT [30].

For an online algorithm, influenced by dynamic, random, and even maliciously adversarial user requests, the incoming requirement is likely to deviate from current cache status, which makes caching module difficult to work. Thus, the *regret* is designed to measure the performance difference between the online decision and optimal configuration in hindsight.

$$x^* = \arg\max_{x} \sum_{t=1}^{T} f_t(x) \quad (2)$$

Thus, the *regret* is expressed as follows.

$$R_T = \mathbb{E}\left[\sum_{t=1}^{T} f_t(x^*) - f_t(x_t) \right] \quad (3)$$

According to the above description, we expect to gain performance advantages during the long-term service process rather than the online caching status being optimal at an exact slot. It requires CP to capture the potential change trend without knowing popularity information.

Algorithm 1 Routing Algorithm

Input: CPs set $\mathcal{M}$. User request $u_n = \{u_k, u_b\}, n \in \mathcal{N}$. The cache status $\boldsymbol{x}$.
Output: Routing result Ψ.
/* ***Initialization:*** */
Routing result $\Psi = \{\emptyset\}$.
for CP $m \in \mathcal{M}$ **do**
    CP m order the cached video with various bitrate.
    Υ add the content $k_m^{b^{(1)}}$ with fewest computational transcoding overhead.
end for
/* ***Routing:*** */
while $\sum_{i=1}^{|\Psi|} \psi_i < 1$ **do**
    Υ select the routing decision $k_m^{b^{(p)}}$ with optimal utility $f_t(x_t)$.
    Υ remove the routed content $k_m^{b^{(p)}}$ and add the content $k_m^{b^{(p')}}, p' > p, x_m^{k,b^{(p')}} > 0$ with least transcoding overhead.
    if Ψ is $\emptyset$ **then**
        Ψ add first element $\psi_1 : \min\{1, x_{m,n}^{k,b^{(p)}} | p = 1\}$
    else
        Ψ add $\psi_2 : z_{m,n}^{k,b^{(p)}} = \min\{x_{m,n}^{k,b^{(p)}}, 1 - \sum_{i=1}^{|\Psi|} \psi_i\}$
    end if
end while

IV. Routing and Caching Problem

The service capability of edge CP ultimately depends on the user request. We consider each unit of high bitrate video can be transcoded to the lower required one in an equal amount through additional computation. Various CP can provide different utilities for terminal n. How to help terminals route from the appropriate node is the first problem we consider.

A. Routing with Various Bitrate

CP and terminal are 2 independent sets during the delivery process. It means that the relationship only exists in the unidirectional association from one set to another, i.e. the topology is bipartite cache graph. For CP, the traffic scale and transcoding overhead is the key. As for terminals, it usually pays attention to the delivery benefit w_{mn}(e.g., transmission rate, extra overhead, etc.).

With a given cache configuration x_t, terminal n determines the priority according to the delivery utility $w_{m,n}, (m \in \mathcal{M}, l_{m,n} = 1)$, while the utility of CP m is reflected by Equ. (1). For multi-rate video, CP can provide flexible services to terminals through transcoding as follows. First, CP $m, m \in \mathcal{M}$ order the priorities of cached video with various bitrates based on transcoding overhead as $\{k_m^{b^{(1)}}, k_m^{b^{(2)}}, ...\}$, where $b^{(1)}$ is the bitrate that demands least overhead to be transcoded to u_b, i.e., $\min_{b=b^{(1)}} ||b - u_b||$. $b^{(2)}$ is the bitrate with the second least transcoding overhead[1] similarly. Meanwhile, the scheme maintains a matching pool[2] $\Upsilon = \{k_m^{b^{(p)}}\}_{m=1}^{M}$ to iteratively determine $z_{m,n}^{k,b^{(p)}}$, where p is the highest priority cached video (no routed), and a routing vector $\Psi = \{\psi_1, ..., \psi_e, ... | \forall \psi_e := z_{m,n}^{k,b} > 0\}$ to keep the route result. The amount of routing data in the first round is denoted as $z_{m,n}^{k,b^{(1)}} = \min\{1, x_{m,n}^{k,b^{(1)}}\}$, the subsequent routing result is $z_{m,n}^{k,b^{(p)}} = \min\{x_{m,n}^{k,b^{(p)}}, 1 - \sum_{i=1}^{|\Psi|} \psi_i\}$. For the sake of notation, let $z_{m,n}^{k} = \sum_{b=1}^{B} z_{m,n}^{k,b}$ be the amount of routing data to CP m. The detailed description is listed as **Algorithm 1**.

Notice that the subscript t is hidden during routing. The algorithm can execute at any time with cache state x_{t-1} (cache is empty at slot 0).

B. Online Caching Optimization

After obtaining the routing result at slot t, CP needs to update the cache status in time according to requests. We consider it as a structured, repeated game. Benefiting from the powerful modeling and online learning ability of online convex optimization (OCO) [32], it provide a feasible idea to establish a framework with flexibility and adaptation.

Definition 1. ***Online Convex Optimization*** *is expected to minimize utility functions on convex feasible set under dynamic input sequence.*

Proposition 1. *Transcoding-enabled Online Caching (ToC) is an OCO problem.*

[1]The transcode function can only be enabled from higher bitrate to lower bitrate. The super-resolution technique requires efficient models and a lot of computational resource, which beyond the scope of this paper.

[2]Matching pool dynamically maintains the optimal routable information for accessible CPs.

First, the feasible region (i.e., $\mathcal{D}$) of ToC is a continuous closed set, which is a convex set obviously. Then, the requests $\{u_t\}_{t=1}^{T}$ under dynamic preferences are dynamic sequences. As for the caching utility function, it is designed based on the typical femtocaching model [29], which is convex. We especially enable the edge transcoding function for multi-rate video services. To analyze the convexity, Equ.(1) with specific m and n ($k = u_k$) can be simplified as follows:

$$\begin{aligned} f(x) &\triangleq \max_{z^b \in [0, x^b]} \sum_{b=1}^{B} z^b h(b) \\ s.t. &\sum_{b=1}^{B} z^b < 1, \\ &0 \le z^b \le x^b \end{aligned} \tag{4}$$

where $h(b) = s^b \cdot (w^b - \zeta ||b - u_b||^{\epsilon})$ is determined by request and matching relation between CP and user.

As shown in (4), the goal is to maximize the utility. We consider to verify the function is concave, so that the convexity of $-f(x)$ can be easily obtained.

Proof. For a concave function $f(x)$, the following inequation should be satisfied for any feasible caching vector x_1,x_2:

$$f(\lambda x_1 + (1-\lambda)x_2) \ge \lambda f(x_1) + (1-\lambda) f(x_2), \forall \lambda \in [0, 1] \tag{5}$$

Meanwhile, consider z_1, z_2 as the routing result of x_1,x_2, respectively, (i.e., $z_i = \sum_{b=1}^{B} z_i^b \le \sum_{b=1}^{B} x_i^b = x_i$), $f(x_i) = \sum_{b=1}^{B} z_i^b h(b)$. Let $x_3 = \lambda x_1 + (1-\lambda)x_2$ and $z_3 = \lambda z_1 + (1-\lambda)z_2$, the following relation can be presented.

$$z_3 = \lambda \sum_{b=1}^{B} z_1^b + (1-\lambda) \sum_{b=1}^{B} z_2^b \le \lambda \sum_{b=1}^{B} x_1^b + (1-\lambda) \sum_{b=1}^{B} x_2^b = x_3 \tag{6}$$

Meanwhile, according to the utility definition in Formula (4),

$$f(x_3) = \max_{z^{b*} \ge 0} \sum_{b=1}^{B} z^{b*} h(b) \ge z_3^b h(b) \tag{7}$$

Thus,

$$\begin{aligned} f(x_3) &\ge z_3^b h(b) \\ &= \lambda \sum_{b=1}^{B} z_1^b h(b) + (1-\lambda) \sum_{b=1}^{B} z_2^b h(b) \\ &= \lambda f(x_1) + (1-\lambda) f(x_2) \end{aligned} \tag{8}$$

The Equ.(5) and proposition are proofed. □

In fact, through the convexity is proofed, its differentiability is difficult to be guaranteed, which can be illustrated by the following simple case. For example, when $\sum_{b=u_b}^{b'} x^b > 1$, for any bitrate $b, u_b \le b < b'$, increasing the amount of cached data can improve utility obviously. For the cached video of higher bitrate $b, b \ge b'$, the existing cached video can already satisfy the request, and the increase of quantity has no effect on utility. It means that there exists a critical point with various gradients. To avoid accidents, we discuss supergradient ∂f instead of gradient ∇f of $f(x)$ below.

Algorithm 2 Transcoding-enabled Online Caching under Stable Network (ToC-S)

Input: User request $u_n = \{u_k, u_b\}$. The cache status x_t at current slot t. The step size η_t.
Output: Caching status x_{t+1} at next slot.
while $t \le T$ **do**
  Select update step η_t.
  Generate routing result $z^{k,b}$ according to Algorithm 1.
  Content index, transcoding, and obtain utility $f(x)$.
  Generate cache candidate decision $\hat{x}_t$ based on supergradient ∂f_t.
  if $\hat{x}_t \in \mathcal{D}$ **then**
    $x_{t+1} = \hat{x}_t$.
  else
    $x_{t+1} = \Pi(\hat{x}_t)$.
  end if
end while

V. Transcoding-enabled Online Caching under Stable IoT Network

Next, we expect to design the OCO-based cache optimization scheme to improve online utility during cache service. In this section, we first discuss the case of a stable IoT network, in which the utility and constraints are known and stable over time. The cache update process can be expressed as follows.

$$x_{t+1} = \Pi(x_t + \eta_t \partial f_t) \tag{9}$$

where $\Pi(\cdot)$ is the projection function to map out-of-scope decisions back into scope and η_t is the step size. The projection here is considered as Euclidean projection:

$$\Pi(\hat{x}_t) = \arg\min_{x_t^*} ||\hat{x}_t - x_t^*|| \tag{10}$$

where $\hat{x}_t = x_t + \eta_t \partial f_t$ is the candidate decision. If $\hat{x}_t \in \mathcal{D}$, the projection step can be skipped. Otherwise, the projection should be operated to find the nearest point x_t^* within domain. The detailed process is shown in **Algorithm 2** (ToC-S).

Actually, the update step and the supergradient of algorithm can directly affect the algorithm performance. Next, we first analyze regret, and determine the step size to minimize it.

A. Regret Performance

The theoretical performance of the algorithm is introduced as follows.

Theorem 1. *The regret of ToC-S is:*

$$R_T^{ToC,S} \le \frac{3\Delta_x J \sqrt{T}}{2} \tag{11}$$

Proof. According to Pythagorean theorem [33], given $\hat{x}$ outside the domain $\mathcal{D}$ and the projection $x = \Pi(\hat{x})$, for $\forall y \in \mathcal{D}$, we have:

$$||y - x|| \le ||y - \hat{x}|| \tag{12}$$

which also is called the non-expansive property of projection. Let $x^* \in \arg\min_{x \in \mathcal{D}} \sum_{t=1}^{T} f_t(x)$. The difference between the

updated caching x_{t+1} and the optimal decision x^* in hindsight can be expressed as follows.

$$\begin{aligned} &||\Pi(\hat{x}) - x^*||^2 \\ &\leq ||x_t + \eta_t \partial f_t - x^*||^2 \\ &= ||x_t - x^*||^2 + 2\eta_t \partial f_t(x_t - x^*) + \eta_t^2 ||\partial f_t||^2 \end{aligned} \tag{13}$$

Rearrange it and obtain,

$$2\partial f_t(x^* - x_t) \leq \frac{||x_t - x^*||^2 - ||\Pi(\hat{x}) - x^*||^2}{\eta_t} + \eta_t ||\partial f_t||^2 \tag{14}$$

Meanwhile, according to the concavity of $f(x)$, the following inequation with supergradient holds.

$$f_t(x_t) - f_t(x^*) \leq \partial f_t(x_t - x^*) \tag{15}$$

Considering the long-term nature of online caching, summing equation (15) along time series and combining equation (14), we have:

$$\begin{aligned} &2\left(\sum_{t=1}^{T}(f_t(x_t) - f_t(x^*))\right) \leq 2\sum_{t=1}^{T}\partial f_t(x_t - x^*) \\ &= \sum_{t=1}^{T}\left(\frac{||x_t - x^*||^2 - ||x_{t+1} - x^*||^2}{\eta_t} + \eta_t ||\partial f_t||^2\right) \\ &= \frac{||x_1 - x^*||^2 - ||x_{T+1} - x^*||^2}{\eta_t} + \sum_{t=1}^{T}\eta_t ||\partial f_t||^2 \end{aligned} \tag{16}$$

It can be observed that the upper bound of regret is related to the change of the cache state, the supergradient, and the update step. For $\forall x \in \mathcal{D}$, the maximum changed scope of cache states is expressed as the diameter Δ_x of the feasible set.

$$\begin{aligned} \Delta_x &= \sum_{k=1}^{K}\sum_{b=1}^{B}\sqrt{||x_1^{k,b}s^{k,b} - x_2^{k,b}s^{k,b}||^2} \\ &\overset{(a)}{=} \sqrt{||C^{\mathbf{1}} - C^{\mathbf{2}}||^2} \overset{(b)}{\leq} \sqrt{2C_m} \end{aligned} \tag{17}$$

where $C^{\mathbf{1}}$ and $C^{\mathbf{2}}$ are different caching configuration for the derivation (a). In the worst case, where the configurations are completely deviated from each other, i.e., a configuration caches something that has been abandoned by another configuration. The difference is up to $2C_m$ for the derivation (b).

As for the supergradient, the calculation of utility at slot t is directly affected by the routing result $z^{k,b}$ at current slot t, given the cache state x_{t-1}. $z^{k,b}$ is constrained by the amount of caching. The lowest bound of ∂f is 0, which occurs in the cases that the increase of caching cannot improve the utility as the simple example mentioned above. The upper bound is determined by delivery utility $w^{k,b}$ and transcoding cost. To be specific, the upper bound occurred with $b^* = \arg\max_b\{w^{k,b} - \zeta||b - u_b||^\epsilon\}$. For the sake of convenience, let $J = \max\{w^{k,b} - \zeta||b - u_b||^\epsilon\} \geq ||\partial f||$.

Combined with Equation (17), Equation (16) will be converted to the following inequation.

$$\sum_{t=1}^{T}(f_t(x_t) - f_t(x^*)) \leq \frac{\Delta_x^2}{2\eta_t} + \sum_{t=1}^{T}\frac{\eta_t J^2}{2} \tag{18}$$

The step size should be carefully chosen to minimize the upper bound of regret, which makes the performance of the algorithm more stable and more efficient. Taking the first-order derivative of the right term of Eq. 18 and the optimal step size is obtained as $\eta^* = \Delta_x / J\sqrt{T}$ until slot T. Since we cannot obtain T in advance, the caching function could stop at any slot and wait to be enabled again. Thus, for each slot t, the optimal step size is $\Delta_x / J\sqrt{t}$. Thus,

$$\begin{aligned} \sum_{t=1}^{T}(f_t(x_t) - f_t(x^*)) &\leq \frac{\Delta_x J\sqrt{T}}{2} + \frac{\Delta_x J}{2\sum_{t=1}^{T}\sqrt{t}} \\ &\overset{(a)}{\leq} \frac{3\Delta_x J\sqrt{T}}{2} \end{aligned} \tag{19}$$

where the derivation (a) is deduced from $1/\sqrt{t}$ series summation. The theorem is proofed. □

As shown in theorem 1, the growth rate of regret is sublinear. As the time T goes to infinity($+\infty$), the regret of the online algorithm approaches zero, i.e. $\lim_{T\to\infty} R_T/T \to 0$. It means the performance of online decision is not weaker than the optimal solution in hindsight during long-term running.

VI. Transcoding-enabled Online Caching with Elastic IoT Network

In fact, in practice, under the influence of network operators, service providers, and terminals, the external conditions of cache modules are likely to be changed. For example, the edge server carries various services (e.g. multimedia, games, shopping, etc.) at the same time. The reserved resources are usually elastic. For multimedia services, we describe this dynamic in terms of cache space and computational resources, and introduce the following constraints:

$$\mathbf{C1:} \sum_{t=1}^{T} g_t^1(x) = \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{b=1}^{B} x^{k,b}s^{k,b} - C_t \leq 0 \tag{20}$$

$$\mathbf{C2:} \sum_{t=1}^{T} g_t^2(x) = \sum_{t=1}^{T} z^{k,b}s^{k,b}||b - u_b^t||^\vartheta - H_t \leq 0 \tag{21}$$

where C_t and H_t are the capacity of storage and computing resource, respectively. Facing burst requests, resource capacity limits can be temporarily breached to meet user needs through the compensation of subsequent feasible decisions. It means that to balance the requirements of each service, the long-term resource requirements should be limited. For the sake of presentation, let the constraint violation of various resources be denoted as $V_T^i = \sum_{t=1}^{T} g_t^i, i \in \{1, 2\}$. The corresponding regret is denoted as follows.

$$R_T^{ToC,E} = \mathbb{E}\left[\sum_{t=1}^{T}(f_t(x_t^*) - f_t(x_t))\,\middle|\, g_t^i(x) \leq 0\right] \tag{22}$$

Algorithm 3 Transcoding-enabled Online Caching under Elastic Network (ToC-E)

Input: User request $u_n = \{u_f, u_b\}$. The cache status x_t at current slot t. The capacity constraint of storage and computing resource, C_t and H_t. The step size η. The control factor δ.
Output: Caching status x_{t+1} at next slot.
while $t \leq T$ **do**
    Select update step η_t.
    Generate routing result $z^{k,b}$ according to Algorithm 1.
    Content index, transcoding, and obtain utility $f(x)$.
    Obtain $\nabla_x \mathcal{L}_t(x_t, \lambda_t^i) = \nabla_x f_t(x_t) + \sum_{i=1}^{2} \lambda_t^i \nabla_x g_t^i(x)$ and $\nabla_{\lambda^i} \mathcal{L}_t(x_t, \lambda_t^i) = \nabla_{\lambda^i} g_t(x_t)$.
    Update cache x_{t+1} and weights λ^i according to Eq. (24) and Eq. (25).
end while

where x_t^* is the optimal dynamic decision, corresponding to the various resources condition.

The necessary and sufficient condition for primal-dual optimality of convex optimization problems is to find Lagrangian saddle points. Inspired by this, we introduce the partial Lagrangian as follows.

$$\mathcal{L}_t(x, \lambda) = f_t(x) + \lambda_t^1 g_t^1(x) + \lambda_t^2 g_t^2(x) - \frac{\delta\eta}{2}||\lambda||^2 \tag{23}$$

where $\lambda^i, i \in \{1, 2\}$ is the Lagrangian multiplier, δ is the control factor. The regularizer $\delta\eta||\lambda^i||^2/2$ is introduced to reduce the influence from the rapid growth of λ^i to $\nabla_x \mathcal{L}_x$, which leads to the fluctuation of the proposal.

The algorithm is expected to ensure the advantages of cache performance while taking constraint violation into account. This requires the algorithm to dynamically control cache status x and multipliers λ_i based on the environment. Different from Eq.(9) and Eq. (10), the update process is listed as follows:

$$\begin{aligned} x_{t+1} &= \Pi(x_t - \eta \nabla_x \mathcal{L}_t(x_t, \lambda_t^i)) \\ &= \Pi\left(x_t - \eta\left(\nabla_x f_t(x_t) + \sum_{i=1}^{2} \lambda_t^i \nabla_x g_t^i(x)\right)\right) \end{aligned} \tag{24}$$

Meanwhile, the dual variable λ_{t+1} is updated as follows.

$$\begin{aligned} \lambda_{t+1}^i &= \Pi_{R_+}(\lambda_t^i + \eta \nabla_{\lambda^i} \mathcal{L}_t(x_t, \lambda_t^i)) \\ &= \Pi_{R_+}\left(\lambda_t^i + \eta \nabla_{\lambda^i} g_t(x_t)\right) \end{aligned} \tag{25}$$

where Π_{R_+} is the projection to ensure the dual variable is feasible. The detailed description is shown in **Algorithm 3**.

At slot t, the algorithm automatically adjusts the weight λ_i according to the constraint violation to control the algorithm performance. Next, the algorithm performance is analyzed.

A. Performance Analyze

In order to verify the regret bound, we first consider the inequality relation of the Lagrangian function:

$$\begin{aligned} &\mathcal{L}_t(x_t, \lambda) - \mathcal{L}_t(x, \lambda_t) \\ &\leq \frac{1}{2\eta}\left(||x - x_t||^2 - ||x - x_{t+1}||^2 + ||\lambda - \lambda_t||^2 - ||\lambda - \lambda_{t+1}||^2\right) \\ &+ \frac{\eta}{2}\left(||\nabla_x \mathcal{L}_t(x_t, \lambda_t)||^2 + ||\nabla_\lambda \mathcal{L}_t(x_t, \lambda_t)||^2\right) \end{aligned} \tag{26}$$

Proof. Considering the convexity of $\mathcal{L}_t(\cdot, \lambda)$ of given dual variable and the concavity of $\mathcal{L}_t(x, \cdot)$ of given primal variable, the following inequation is obtained.

$$\mathcal{L}_t(x, \lambda_t) \geq \mathcal{L}_t(x_t, \lambda_t) + \nabla_x \mathcal{L}_t(x_t, \lambda_t)(x - x_t) \tag{27}$$

$$\mathcal{L}_t(x_t, \lambda) \leq \mathcal{L}_t(x_t, \lambda_t) + \nabla_x \mathcal{L}_t(x_t, \lambda_t)(\lambda - \lambda_t) \tag{28}$$

Combining these two inequation, we have:

$$\begin{aligned} &\mathcal{L}_t(x_t, \lambda) - \mathcal{L}_t(x, \lambda_t) \\ &\leq \nabla_x \mathcal{L}_t(x_t, \lambda_t)(x_t - x) + \nabla_\lambda \mathcal{L}_t(x_t, \lambda_t)(\lambda - \lambda_t) \end{aligned} \tag{29}$$

According to the nonexpansive property of projection mentioned above and Eq. (16), $\nabla_x \mathcal{L}_t(x_t, \lambda_t)(x_t - x)$ in the RHS of Eq. (29) can be deduced.

$$\begin{aligned} ||x - x_{t+1}||^2 \leq ||x - x_t||^2 + \eta^2||\nabla_x \mathcal{L}_t(x_t, \lambda_t)||^2 \\ - 2\eta \nabla_x \mathcal{L}_t(x_t, \lambda_t)(x_t - x) \end{aligned} \tag{30}$$

Similarly, $\nabla_\lambda \mathcal{L}_t(x_t, \lambda_t)(\lambda - \lambda_t)$ can be obtained. Plugging these two item to Eq. (29), Eq. (26) is obtained. □

From a long-term perspective, under the dynamic benchmark x_t^*, the upper bound can be denoted as follows.

$$\begin{aligned} &\sum_{t=1}^{T} \mathcal{L}_t(x_t, \lambda) - \mathcal{L}_t(x_t^*, \lambda_t) \\ &\leq \sum_{t=1}^{T} \frac{1}{2\eta}\left(||x_t^* - x_t||^2 - ||x_t^* - x_{t+1}||^2\right) \\ &+ \sum_{t=1}^{T} \frac{1}{2\eta}\left(||\lambda - \lambda_t||^2 - ||\lambda - \lambda_{t+1}||^2\right) \\ &+ \sum_{t=1}^{T} \frac{\eta}{2}\left(||\nabla_x \mathcal{L}_t(x_t, \lambda_t)||^2 + ||\nabla_\lambda \mathcal{L}_t(x_t, \lambda_t)||^2\right) \end{aligned} \tag{31}$$

Considering in turn, the first term is expressed as follows:

$$\begin{aligned} &\sum_{t=1}^{T}\left(||x_t^* - x_t||^2 - ||x_t^* - x_{t+1}||^2\right) \\ &= ||x_1^* - x_1||^2 - ||x_T^* - x_{T+1}||^2 \\ &+ \sum_{t=2}^{T}\left(||x_t - x_t^*||^2 - ||x_t - x_{t+1}^*||^2\right) \\ &\leq ||x_1^* - x_1||^2 + ||x_T^*||^2 + 2\sum_{t=2}^{T} ||x_t||||x_{t-1}^* - x_t|| \\ &\leq 3 + 2\Delta_x \quad (||x|| \leq 1, ||x^* - x|| \leq \Delta_x) \end{aligned} \tag{32}$$

As for the second term, $\sum_{t=1}^{T}\left(||\lambda-\lambda_t||^2-||\lambda-\lambda_{t+1}||\right)=||\lambda-\lambda_1||^2-||\lambda-\lambda_{T+1}||^2\leq||\lambda||^2$. For the last term, we have:

$$\begin{aligned}||\nabla_x\mathcal{L}_t(x_t,\lambda_t)||^2&=||\nabla f_t(x_t)+\sum_{i=1}^{2}\lambda_t^i\nabla g_t^i(x_t)||^2\\&\overset{(a)}{\leq}3\left(||\nabla f_t(x_t)||^2+\sum_{i=1}^{2}(\lambda_t^i)^2||\nabla g_t^i(x_t)||^2\right)\end{aligned}\tag{33}$$

According to Eq. (20) and Eq. (21), the constraint is a linear function, and the gradient can be obtained as $G_1=s^{k,b}$ and $G_2=s^{k,b}||u-u_b||^{\vartheta}$. Let $G=\max\{G_1,G_2\}$.Meanwhile, the upper bound of constraint violation within a single slot is limited by the maximum resource number of nodes, which we simply denote it as M here. Thus, Eq. (33) can be converted as follows.

$$||\nabla_x\mathcal{L}_t(x_t,\lambda_t)||^2\leq3\left(J^2+||\lambda_t||^2G^2\right)\tag{34}$$

Similarly,

$$\begin{aligned}||\nabla_\lambda\mathcal{L}_t(x_t,\lambda_t)||^2&=||g_t(x_t)-\delta\eta\lambda_t||^2\\&\leq2\left(||g_t(x_t)||^2+\delta^2\eta^2||\lambda_t||^2\right)\\&\leq2\left(M^2+\delta^2\eta^2||\lambda_t||^2\right)\end{aligned}\tag{35}$$

In summary, the Eq. (31) is deduced to the following inequation.

$$\begin{aligned}&\sum_{t=1}^{T}\mathcal{L}_t(x_t,\lambda)-\mathcal{L}_t(x_t^*,\lambda_t)\\&\leq\frac{1}{2\eta}\left(3+2\Delta_x+||\lambda||^2\right)+\frac{T\eta}{2}\left(3J^2+2M^2\right)\\&+\frac{\eta}{2}\left(3G^2+2\delta^2\eta^2\right)\sum_{t=1}^{T}||\lambda_t||^2\end{aligned}\tag{36}$$

Next, to obtain the upper bound of regret, we substitute Eq. (23) into Eq. (36) and obtain:

$$\begin{aligned}&\sum_{t=1}^{T}\left[f_t(x_t)-f_t(x_t^*)\right]\\&+\sum_{i=1}^{2}\sum_{t=1}^{T}\left[\lambda^i g_t^i(x_t)-\lambda_t^i g_t^i(x_t^*)\right]-\frac{\delta\eta T}{2}||\lambda||^2\\&\leq\frac{1}{2\eta}\left(3+2\Delta_x+||\lambda||^2\right)+\frac{T\eta}{2}\left(3J^2+2M^2\right)\\&+\frac{\eta}{2}\left(3G^2+2\delta^2\eta^2-\delta\right)\sum_{t=1}^{T}||\lambda_t||^2\end{aligned}\tag{37}$$

By choosing $\delta=5G^2\geq3G^2+2\delta^2\eta^2$ properly, the last term can be eliminated. Meanwhile, we group the terms with λ together into the LHS of the inequality:

$$\begin{aligned}&\sum_{t=1}^{T}\left[f_t(x_t)-f_t(x_t^*)\right]\\&+\sum_{i=1}^{2}\left[\lambda^i\sum_{t=1}^{T}g_t^i(x_t)-\left(\frac{\delta\eta T}{2}+\frac{1}{2\eta}\right)\left(\lambda^i\right)^2\right]\\&\leq\sum_{t=1}^{T}\lambda_t^i g_t^i(x_t^*)+\frac{1}{2\eta}\left(3+2\Delta_x\right)+\frac{\eta T}{2}\left(3J^2+2M^2\right)\end{aligned}\tag{38}$$

For the cached benchmark decision x^*, the constraint violation should be zero or even negative. Thus, the first term on the RHS with $\lambda\geq0$ can be eliminated. Meanwhile, note that the second term on the LHS is a concave function about lambda. λ can be maximized by the first-order derivative, and it is easy to obtain:

$$\lambda_t^{i*}=\frac{\left[\sum_{t=1}^{T}g_t^i(x_t)\right]^+}{\delta\eta T+\frac{1}{\eta}}\tag{39}$$

And the Eq. (38) can be converted as follows.

$$\begin{aligned}&\sum_{t=1}^{T}\left[f_t(x_t)-f_t(x_t^*)\right]+\sum_{i=1}^{2}\frac{\left(\left[\sum_{t=1}^{T}g_t^i(x_t)\right]^+\right)^2}{2(\delta\eta T+\frac{1}{\eta})}\\&\leq\frac{1}{2\eta}\left(3+2\Delta_x\right)+\frac{\eta T}{2}\left(3J^2+2M^2\right)\end{aligned}\tag{40}$$

Obviously, the second term on the LHS is greater than zero. By properly choosing the step size $\eta=\sqrt{\frac{\Delta_x}{T}}$, the regret can be estimated as follows:

$$R_T^{TOC,E}\leq\frac{3}{2}\sqrt{\frac{T}{\Delta_x}}+\sqrt{T\Delta_x}\left(1+\frac{3J^2}{2}+M^2\right)\tag{41}$$

Then the long-term constraint violation can also be derived from Eq. (40) as follows.

$$\begin{aligned}&\sum_{t=1}^{T}g_t^i(x_t)\leq\left[\sum_{t=1}^{T}g_t^i(x_t)\right]^+\\&\leq\sqrt{2\left(\Omega+\frac{1}{2\eta}(3+2\Delta_x)+\frac{\eta T}{2}(3J^2+2M^2)\right)(\delta\eta T+\frac{1}{\eta})}\\&\leq\sqrt{2\left(\Omega+\frac{3}{2}\sqrt{\frac{T}{\Delta_x}}+\sqrt{\Delta_x T}\left(1+\frac{3}{2}J^2+M^2\right)\right)}\\&\times\sqrt{\delta\sqrt{\Delta_x T}+\sqrt{\frac{T}{\Delta_x}}}\end{aligned}\tag{42}$$

where $\Omega=\sum_{t=1}^{T}\left[f_t(x_t^*)-f_t(x_t)\right]$ is the accumulated difference. In the worst case, the user requests the highest bitrate video with the complete data (i.e., 100% routing). The optimal decision in hindsight should be $x^{K,B}=1$, while the worst decision generated by the proposed policy is zero(i.e. the utility is zero). Let $\xi_t=\max\{f_t(x_t^*)-f_t(x_t)\}=f_t(x_t^*)=s^{K,B}w^{K,B}$.

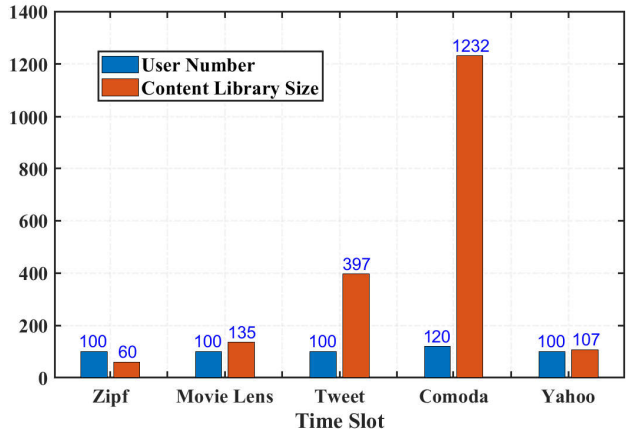

Fig. 2. Open Dataset Property

Thus, we have $\Omega \leq \sum_{t=1}^{T} \xi_t = \xi T$ and the constraint violation $\sum_{t=1}^{T} g_t^i(x_t) \leq \mathcal{O}\left(T^{\frac{3}{4}}\right)$ holds.

Therefore, on the one hand, the performance of the algorithm in elastic networks over time is not worse than that of the dynamic benchmark $\{x_t^*\}$. On the other hand, the TOC-E can maintain sublinear constraint violation, which means that the violation tends to be non-positive over time.

VII. Performance Evaluation

In this section, we evaluate the proposed scheme under different request patterns to complete the multiple verification. It is mainly introduced from four aspects: open datasets, experimental scenario, comparing algorithm and evaluation results.

A. Open Datasets

First, to simulate the actual user request, we introduced the following datasets and processed the data to fit the transcoding-enabled video caching scenario.

- LDOS-CoMoDa[3]. The LDOS-CoMoDa is a context-rich movie recommend dataset. It contains ratings for the movies and some contextual information describing the situation where the movies were consumed. The concerned items are: userID (15 - 200), itemID (1 - 4138), and rating (1-5).
- MovieLens[4]. MovieLens is a real movie rating dataset that consists of tens of millions of ratings and tag applications, which was recently updated in 2018.
- Movie Tweet[5]. Movie Tweet is a dataset consisting of ratings on movies that were contained in well-structured tweets on Twitter.
- Yahoo[6]. It contains movies rating generated on Yahoo Movies up to Nov. 2003. It provides content and ratings information on a 1-5 scale and records the ratings which are collected from 4,000 users on more than 2,000 items.

[3]https://www.lucami.org/en/research/ldos-comoda-dataset.
[4]https://grouplens.org/datasets/movielens/
[5]https://github.com/sidooms/MovieTweetings
[6]https://github.com/sisinflab/LinkedDatasets/tree/master/yahoo_23MB

It should be noted that some datasets are widely used in recommendation systems. For suiting our scenario, we take the rated level (e.g., recommendation degree) as the preference information of users. Meanwhile, we notice that the number of requests for part of videos is very scarce (e.g., 0-2 times). Thus, video without adequate information is screened out in data pre-processing to reduce video library size. In addition, for the consideration of differences, various numbers of users are loaded from datasets as request initiators in the scenario. The specific attributes of open datasets are shown in Fig. 2.

B. Experimental Scenario

Next, we introduce the experimental scenario. We consider the scene as a $500 * 500m^2$ area, which is shown in Fig. 3. The geographical location of users and cache points follows 2-dimensional random Poisson point distribution with the density of 8 and 60, respectively. The communication range of each CP is set as a circular area with a radius of 200 meters. The storage space of CP follows uniform distribution between 1800 and 3200, which can store about 40 videos with high bitrate. The computing resource is a random number between 0.5 and 2.5. The video library size is set as 100 in simulation when the requests follows Zipf distribution, and the video bitrate set includes four levels from high bitrate to low bitrate.

C. Comparing Algorithm

We introduce several comparisons to verify the performance of the proposed scheme.

- LFU(Least Frequently Used). By evicting the least recently used video items, LFU ensures that cached video is likely to be requested again.
- LRU(Least Recently Used). LRU improve the cache hit probability by maintaining video content with high usage frequency.
- FIFO(First In First Out). The video item which is stored first will also be ejected first.
- SB(Static Benchmark). Given the information of request sequence, SB output a optimal caching policy and keep it unchanged to improve the utility.

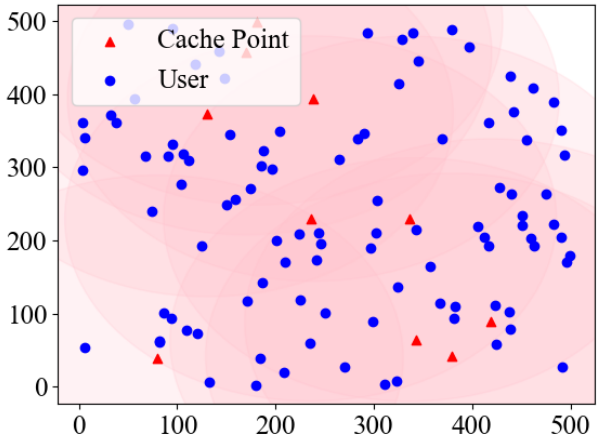

Fig. 3. Experimental Scenario

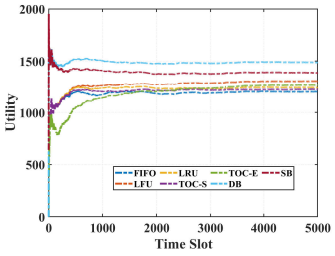

Fig. 4. Service Utility (Zipf)

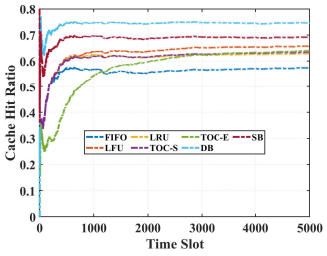

Fig. 5. Cache Hit Ratio (Zipf)

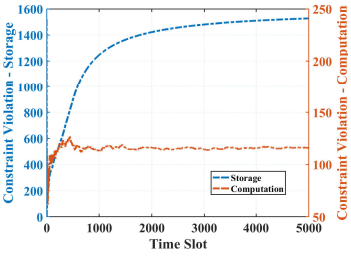

Fig. 6. Constraint Violation(Zipf)

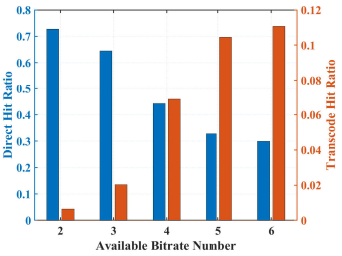

Fig. 7. Direct Hit versus Transcode Hit (Zipf)

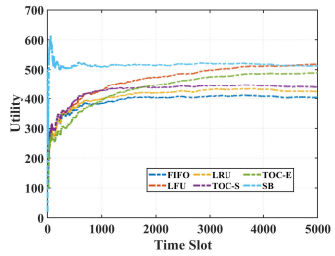

Fig. 8. Service Utility(Yahoo)

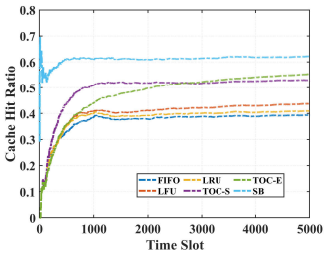

Fig. 9. Cache Hit Ratio(Yahoo)

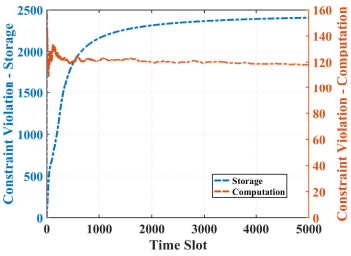

Fig. 10. Constraint Violation (Yahoo)

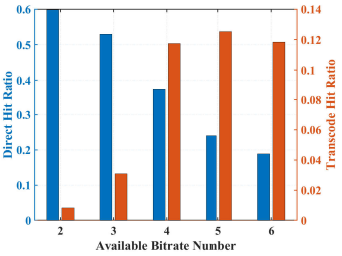

Fig. 11. Direct Hit versus Transcode Hit (Yahoo)

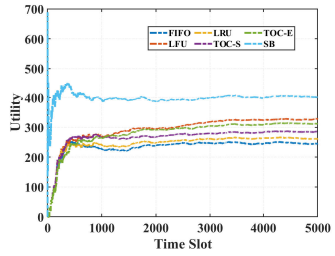

Fig. 12. Service Utility (Movie Lens)

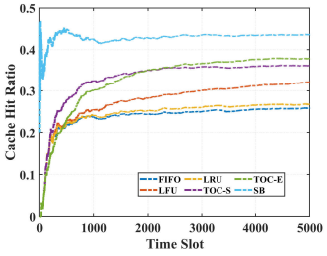

Fig. 13. Cache Hit Ratio(Movie Lens)

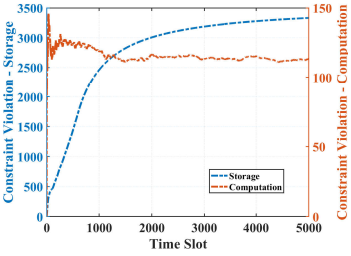

Fig. 14. Constraint Violation (Movie

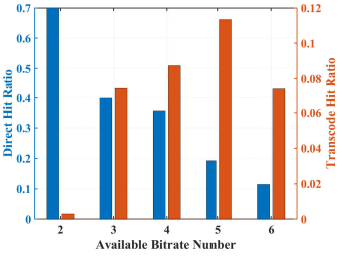

Fig. 15. Direct Hit versus Transcode Hit(Movie Lens)

- DB(Dynamic Benchmark). Given the request sequence, DB generate caching policy based on various requests and resource conditions at a certain moment t.

D. Evaluation Result

In this section, several schemes are evaluated on different request modes generated from various datasets. We discussed the aspects of cache service utility, cache hit rate, constraint violation under elastic network, and the impact of the number of available bitrates on edge hit rate in detail, as follows.

1) Zipf: First, we evaluate the cache performance under the request mode with Zipf distribution by performing caching decisions at edge CDN nodes. The video library size is set to 60. Each video is assigned a request probability based on the subscript. As shown in Fig. 4, without prior knowledge, online learning algorithms explore the potential benefits of cache decisions from complex information, and the service utility is gradually improved with iteration. On the other hand, due to the fixed mode of the request, LFU, LRU show adaptive advantages in terms of regularity, and their performance is better than that of online learning schemes in long-term process, only slightly worse than that of dynamic benchmark algorithms and static benchmark algorithms. Finally, SB and DB achieve greater benefits with the full knowledge of request information.

Correspondingly, the probability of cache hit is shown in Fig. 5. 7 strategies listed can be divided into three categories. FIFO, LFU, and LRU are traditional cache strategies that play well in most of them. TOC-E and TOC-S show their learning attributes, while DB and SB have relatively higher hit ratios.

In addition, we also test the constraint violation under elastic network conditions in Fig. 6. According to the distribution of cache space and computing resources, the violation status varies, but eventually converges to a relatively stable state. This shows that constraint violation does not increase linearly with algorithm running and validates our previous conclusion about constraint violation.

Finally, we evaluate whether video cache hits came from transcoding in Fig. 7. By adjusting the number of bitrates available, the conclusions vary. For the first case, it does not need to schedule computing resources, and can directly index the video content at edge, which is called a direct hit. On the contrary, it is called transcoding hit when computational transcoding is required. For example, there is only one available bitrate, and the video hit must all be direct hits. When there exist 2 available bitrates, more than 70% of the requests are directly hit, while transcoding hits less than 1%. As the number of available bitrates increases, it increases the size of the content library in effect. The overall video hit ratio tends to decrease, which can be observed from the sum of the

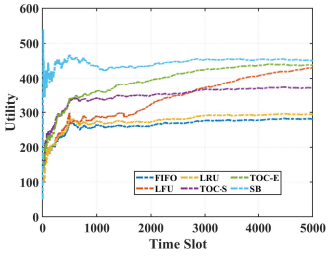

Fig. 16. Service Utility (Tweet)

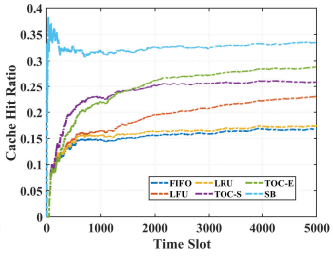

Fig. 17. Cache Hit Ratio (Tweet)

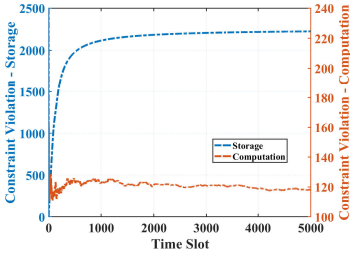

Fig. 18. Constraint Violation (Tweet)

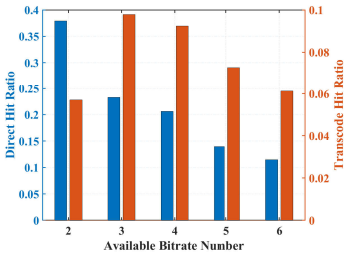

Fig. 19. Direct Hit versus Transcode Hit (Tweet)

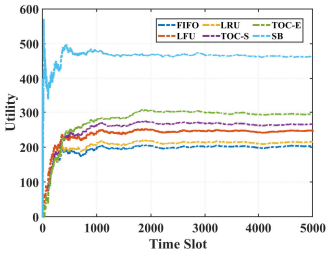

Fig. 20. Service Utility (Comoda)

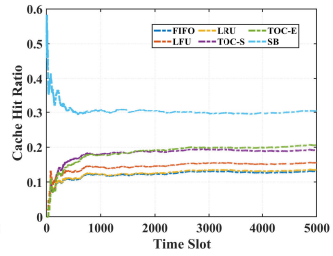

Fig. 21. Cache Hit Ratio (Comoda)

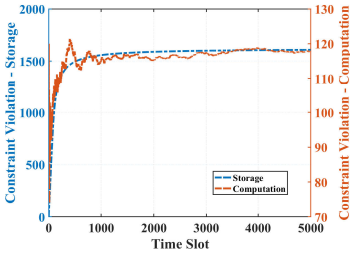

Fig. 22. Constraint Violation (Comoda)

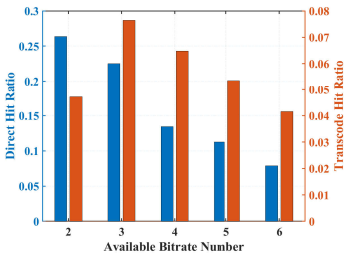

Fig. 23. Direct Hit versus Transcode Hit (Comoda)

subsequent direct hit ratio and transcoding hit ratio. On the other hand, the direct hit rate is significantly reduced, while the transcoding hit rate is further improved, and when the number of additional available bit rates reaches 6, the transcoding hit rate is more than 10%.

2) Movie Yahoo: It should be clear before we discuss the following results, due to the prophetic information in DB, the utility, cache hit rate remain at higher level. To improve the readability, we mainly display the caching service performance of the other 6 strategies. From Fig. 8 to Fig. 11, we notice that TOC-E has strong learning properties, and its utility and hit ratio are often weaker than other superior strategies in the early stage, but with the progress of iteration, the utility gradually catches up with and surpasses the latter. However, LFU shows its performance superiority again. Its utility is close to the static benchmark algorithm. As for constraint violations in Fig. 14, the database (Yahoo) contains more content, resulting in larger storage violations, while the computation is still on the same level as before. With the increase in the number of available bitrates as shown in Fig. 15, the transcode hit rate of the dataset shows a Stationary trend in the later period.

3) Movie Lens: The basic attributes of Movie Lens is similar to both scenarios mentioned before, which we describe below. First, in terms of service utility in Fig. 12, the learning class strategy shows its ability in irregular requests, and its performance is ahead of LRU and FIFO. LFU still holds good performance thanks to the maintenance of usage frequency in space. Through the observation of hit ratio in Fig. 13, it is found that the actual hit ratio of the proposed scheme is higher than that of LFU. This is because the LFU mainly provides services for the request directly, while the learning policy needs to consume extra overhead in the case of more hits. The constraint violation increases further with the number of content libraries, implying that its numerical size may increase with the content library size, all things being equal. Finally, the inflection point of transcoding hit and direct hit appears. When the number of available code rates reach 6, both transcoding hit and direct hit fall.

4) Movie Tweet: The content library size is significantly increased in the tweet dataset, and the effects are demonstrated as follows. The utility and hit rate of each strategy are decreased obviously. With the same storage space and computing resource configuration as the previous experiments, the hit ratio is reduced to about 25%. However, the constraint violation in this experiment shows it has no direct relationship with the content library size, and it can still enter the stationary state after iteration as before. Meanwhile, with the increase in the number of available bitrates, the direct hit and transcoding hit probability are lower. It shows that the content library size and the available bitrate have intrinsic relations to cache hit ratio. We conclude that when the content library size is more than 200, the number of available bit rates is 3, which is the most conducive to the work of transcoding module.

5) Movie Comoda: The number of file libraries is further increased in the comoda. The storage space configuration is the same as that in the previous experiments. In this scenario, the performance of the learning policy is superior. Specifically, in the case of a similar hit ratio, various transcoding hits produce differentiated utility. However, since the number of requests is determined, and the performance is evaluated under the environment with 3 available bitrates, the computational resource violation remains the same. Meanwhile, the trend of transcoding hit and direct hit is consistent as mentioned before, and the overall hit rate is still maintained at 20%.

VIII. Conclusions

To satisfy the requirements of low latency and high QoE for real-time IoT video services, this paper embeds computation into caching mechanisms, establishes bridging connections between substitutable multi-rate video, and proposes a cloud-edge-terminal collaborative online caching framework that enables transcoding in unknown time-varying environments.

By maintaining a dynamic matching pool, content routing is proactively determined to reduce overhead. Furthermore, the cache online update strategy under stable IoT networks is designed to adapt to unknown prior environmental information. It is further extended to time-varying IoT networks, where the resources are limited and the status is difficult to be evaluated. A partial Lagrangian function is introduced to search the saddle points. Strict theoretical proof supports the sublinearity of regret and constraint violation. Finally, we utilized multiple datasets to form a differentiated environment and completed multiple verifications, demonstrating the superiority of the proposed scheme compared to representative schemes.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (NSFC) under grant No. 62225105, 62072047 and by the Postdoctoral Science Foundation of China under grant No. 2022M720518. G.-M. Muntean wishes to acknowledge the Science Foundation Ireland (SFI)'s support via grant nos. 16/SP/3804 (Enable) and 12/RC/2289_P2 (Insight).

REFERENCES

[1] Apple Edge Cache, https://cache.edge.apple/.
[2] Cisco, Cisco Annual Internet Report (2018−2023) White Paper, 2020. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html
[3] G. Li et al., "Understanding User Generated Content Characteristics: A Hot-Event Perspective," 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1-5.
[4] D. Schroeder, A. Ilangovan, M. Reisslein and E. Steinbach, "Efficient Multi-Rate Video Encoding for HEVC-Based Adaptive HTTP Streaming," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, no. 1, pp. 143-157, Jan. 2018.
[5] E. Çetinkaya, H. Amirpour, C. Timmerer and M. Ghanbari, "Fast Multi-Resolution and Multi-Rate Encoding for HTTP Adaptive Streaming Using Machine Learning," in IEEE Open Journal of Signal Processing, vol. 2, pp. 484-495, 2021.
[6] J. L. D. Neto, S. -Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar and S. Secci, "ULOOF: A User Level Online Offloading Framework for Mobile Edge Computing," in IEEE Transactions on Mobile Computing, vol. 17, no. 11, pp. 2660-2674, 1 Nov. 2018, doi: 10.1109/TMC.2018.2815015.
[7] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016.
[8] X. Chen, Changqiao Xu, M. Wang, Z. Wu, L. Zhong and G.-M. Muntean, "A Universal Transcoding and Transmission Method for Livecast with Networked Multi-Agent Reinforcement Learning," Proceeding of the 2021 IEEE Conference on Computer Communications (INFOCOM 2021), 2021.
[9] H. Xiao, C. Xu, Z. Feng, R. Ding, S. Yang, L. Zhong, J. Liang, and G.-M. Muntean, "A Transcoding-enabled 360○ VR Video Caching and Delivery Framework for Edge-enhanced Next-generation Wireless Networks," IEEE Journal on Selected Areas in Communications, vol. 40, no. 5, pp. 1615-1631, May 2022.
[10] J. Yoon and S. Banerjee, "Hardware-Assisted, Low-Cost Video Transcoding Solution in Wireless Networks," in IEEE Transactions on Mobile Computing, vol. 19, no. 3, pp. 581-597, 1 March 2020.
[11] Guanyu Gao, Yonggang Wen, "Video transcoding for adaptive bitrate streaming over edge-cloud continuum", Digital Communications and Networks, Volume 7, Issue 4, 2021, Pages 598-604, ISSN 2352-8648.
[12] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang,"Learning-based content caching and sharing for wireless networks," IEEE Trans. Commun., vol. 65, no. 10, pp. 4309–4324, Oct. 2017.
[13] Z. Zhang and M. Tao, "Deep Learning for Wireless Coded Caching With Unknown and Time-Variant Content Popularity," in IEEE Transactions on Wireless Communications, vol. 20, no. 2, pp. 1152-1163, Feb. 2021.
[14] Y. Chiang, C. -H. Hsu and H. -Y. Wei, "Collaborative Social-Aware and QoE-Driven Video Caching and Adaptation in Edge Network," in IEEE Transactions on Multimedia, vol. 23, pp. 4311-4325, 2021.
[15] T. Li, T. Braud, Y. Li and P. Hui, "Lifecycle-Aware Online Video Caching," in IEEE Transactions on Mobile Computing, vol. 20, no. 8, pp. 2624-2636, 1 Aug. 2021.
[16] O. Ayoub, F. Musumeci, M. Tornatore and A. Pattavina, "Energy-Efficient Video-On-Demand Content Caching and Distribution in Metro Area Networks," in IEEE Transactions on Green Communications and Networking, vol. 3, no. 1, pp. 159-169, March 2019.
[17] X. Zhang, T. Lv, Y. Ren, W. Ni, N. C. Beaulieu and Y. J. Guo, "Economical Caching for Scalable Videos in Cache-Enabled Heterogeneous Networks," in IEEE Journal on Selected Areas in Communications, vol. 37, no. 7, pp. 1608-1621, July 2019.
[18] B.Jedari and M. D. Francesco, "Auction-based Cache Trading for Scalable Videos in Multi-Provider Heterogeneous Networks," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 2019, pp. 1864-1872.
[19] Y. Guo, Q. Yang, F. R. Yu and V. C. M. Leung, "Cache-Enabled Adaptive Video Streaming Over Vehicular Networks: A Dynamic Approach," in IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pp. 5445-5459, June 2018.
[20] N. Golrezaei, A. F. Molisch, A. G. Dimakis and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," in IEEE Communications Magazine, vol. 51, no. 4, pp. 142-149, April 2013.
[21] S. Mehrizi, S. Chatterjee, S. Chatzinotas and B. Ottersten, "Online Spatiotemporal Popularity Learning via Variational Bayes for Cooperative Caching," in IEEE Transactions on Communications, vol. 68, no. 11, pp. 7068-7082, Nov. 2020.
[22] Y. Zhang, C. Li, T. H. Luan, Y. Fu, W. Shi and L. Zhu, "A Mobility-Aware Vehicular Caching Scheme in Content Centric Networks: Model and Optimization," in IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3100-3112, April 2019.
[23] X. Wu, J. Li, M. Xiao, P. C. Ching and H. V. Poor, "Multi-Agent Reinforcement Learning for Cooperative Coded Caching via Homotopy Optimization," in IEEE Transactions on Wireless Communications, vol. 20, no. 8, pp. 5258-5272, Aug. 2021.
[24] Y. Qian, R. Wang, J. Wu, B. Tan and H. Ren, "Reinforcement Learning-Based Optimal Computing and Caching in Mobile Edge Network," in IEEE Journal on Selected Areas in Communications, vol. 38, no. 10, pp. 2343-2355, Oct. 2020.
[25] S. Liu, C. Zheng, Y. Huang and T. Q. S. Quek, "Distributed Reinforcement Learning for Privacy-Preserving Dynamic Edge Caching," in IEEE Journal on Selected Areas in Communications, vol. 40, no. 3, pp. 749-760, March 2022.
[26] Z. Shi, Y. Zhou, D. Wu and C. Wang, "PPVC: Online Learning Toward Optimized Video Content Caching," in IEEE/ACM Transactions on Networking, vol. 30, no. 3, pp. 1029-1044, June 2022.
[27] K. Spiteri, R. Urgaonkar and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," in IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1698-1711, Aug. 2020, doi: 10.1109/TNET.2020.2996964.
[28] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. "From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player." in ACM Trans. Multimedia Comput. Commun. Appl., 2019
[29] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in Proc. IEEE INFOCOM, Mar. 2012, pp. 1107–1115.
[30] Satadal Sengupta, Hyojoon Kim, and Jennifer Rexford, Continuous in-network round-trip time monitoring, In Proceedings of the ACM SIGCOMM 2022 Conference (SIGCOMM '22), New York, NY, USA, 473–485, 2022.
[31] S. Emara et al., "Low-Latency Network-Adaptive Error Control for Interactive Streaming," in IEEE Transactions on Multimedia, vol. 24, pp. 1691-1706, 2022.
[32] E. Hazan, "Introduction to online convex optimization," FNT Optim., 2016.
[33] Jean-Daniel Boissonnat, Mariette Yvinec, Herve Bronniman, "Algorithmic Geometry", Cambridge University Press,2001.