Rosevin: Employing Resource- and Rate-Adaptive Edge Super-Resolution for Video Streaming

Xiaoxi Zhang*, Haoran Xu*, Longhao Zou†, Jingpu Duan§†, Chuan Wu‡, Yali Xue†, Zuozhou Chen†, Xu Chen§*
*School of Computer Science and Engineering, Sun Yat-sen University
†Department of Communications, Peng Cheng Laboratory
‡Department of Computer Science, The University of Hong Kong
Email: *{zhangxx89,chenxu35}@mail.sysu.edu.cn, *xuhr6@mail2.sysu.edu.cn
†{zoulh,duanjp,xueyl,chenzzh}@pcl.ac.cn, ‡cwu@cs.hku.hk

***Abstract*—Today's video streaming service providers have exploited cloud-edge collaborative networks for geo-distributed video delivery. The existing content delivery network (CDN) scheduling and adaptive bitrate algorithms may not fully utilize edge resources or lack a global control to optimize resource sharing. The emerging super-resolution (SR) approach can unleash the potential of leveraging computation resources to compensate for bandwidth consumption, by producing high-quality videos from low-resolution contents. Yet the uncertain SR resource sensitivity and its interplay with bitrate adaptation are under-explored. In this work, we propose *Rosevin*, the first resource scheduler that jointly decides the bitrates and fine-grained resource allocation to perform SR at the edge, which can learn to optimize the long-term QoE for distributed end users. To handle the time-varying and complex space of decisions as well as a non-smooth objective function, *Rosevin* realizes a novel online combinatorial learning algorithm, which nicely integrates convex optimization theories and online learning techniques. In addition to theoretically analyzing its performance, we implement an SR-assisted video streaming prototype of *Rosevin* and demonstrate its advantages over several video delivery benchmarks.**

I. Introduction

Video streaming applications including Netflix [1], Youtube [2] and Tiktok [3] have generated over 65% of the traffic across today's Internet [4]. To enhance users' quality of experience (QoE) (low request latency and high video quality), video streaming service providers have adopted cloud-edge collaborative networks to serve geographical end users [5]–[8]. Leveraging the storage capacity of edge clusters, video chunks can be cached at the edge and delivered to users with lower latency and potentially higher QoE, compared with frequently fetching videos directly from the cloud.

To adapt to volatile and heterogeneous downlink bandwidth, video streaming systems resort to adaptive bitrate (ABR) algorithms such as BOLA [9] and MPC [10] and video caching at edge clusters [6] [11]. These ABR methods are client-based and thus lack a global view in balancing the bitrates across users who share the same network links. In addition, given the cached videos, controlling bitrates alone merely adjusts the content delivery to adapt to bandwidth limitations, which affects user QoE. For instance, decreasing the bitrate of video delivery when encountering bandwidth deficiency generally means choosing a smaller resolution of the video frames, which could compromise the user's QoE.

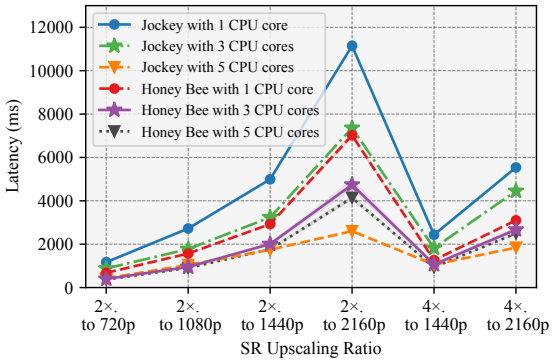


Fig. 1: Latency varying SR upscaling ratios and the number of CPU cores for two types of video chunks (2 seconds each). E.g., 2x to 720p means enhancing a chunk from 360p to 720p.

The emerging deep neural network (DNN) based super-resolution (SR) technology can convert low-resolution video chunks into high-quality ones that recover the original sharpness and texture details of a video via DNN inferences [12]. A natural solution for fully exploiting edge resources and guaranteeing user QoE is then to simultaneously control bitrates and enable SR services at the edge, exchanging computational resources for improved video quality and communication efficiency. Intuitively, with SR enabled at the edge, a high-resolution video request can be addressed by fetching a low-resolution content from the source to the edge, in proximity to the end user, and using SR at the edge to enhance it to the desired resolution with minimum quality degradation, before

This work was supported by Key-Area Research and Development Program of Guangdong Province (2021B0101400001), NSFC grants (62102460, 62201244, U20A20159), Guangdong Basic and Applied Basic Research Foundation (2023A1515012982, 2021B151520008), Guangzhou Science and Technology Plan Project (202201011392, 2024A04J6367), the Young Outstanding Award under the Zhujiang Talent Plan of Guangdong Province, and Hong Kong RGC under the contracts HKU 17208920 and C7004-22G (CRF).
§Co-corresponding authors: Jingpu Duan and Xu Chen.

sending it to the user. However, the choice of base resolution (and equivalently base bitrate) of the video fetched from the source to the edge impacts both the SR inference time and cloud-to-edge transmission time. It is thus critical for reducing the end-to-end latency and maximizing user QoE.

Balancing computation resources (e.g., CPU cores used for SR at the edge) and communication efficiency (latency of serving user-requested video chunks) for multi-user SR-assisted video delivery is still under-explored. Existing video streaming systems either perform bandwidth or request scheduling [13] [5] [14] without considering the use of SR, or ignore elastic computation resource allocation for SR with bitrates separately determined [15]–[18]. We observe that, fine-grained and adaptive allocation of CPU cores for different SR processes in tandem with bitrate selection can improve aggregate user QoE. For instance, using *more resources* to upscale a video chunk from a base to a target bitrate yields a *smaller* SR processing time and thus a higher QoE, as end-to-end latency is another component in the QoE [19]. However, given the limited resource capacity in the edge cluster, maximizing total QoE requires careful allocation of resources and selection of input bitrates for the SR model. This work designs the first online self-optimized and scalable framework that jointly optimizes bitrates and resource allocation for edge SR. This co-optimization is complex due to the following **key challenges**:

- *Uncertain impact of resource allocation on SR performance, varying video types and bitrates.* Our testbed results (Figure 1) suggest different functions of SR runtime with respect to CPU core numbers and upscaling ratios for different videos. This variability impedes proactive resource allocation, which existing rule-based schedulers [20] cannot cope with.
- *Heterogeneous and fluctuating streaming latency across geo-locations that necessitates adaptive, fine-grained selection of bitrates and resources*. Unfortunately, the high-dimensional temporal-spacial complexity greatly expands the decision space of our optimization problem and inhibits the convergence rate of classic learning algorithms such as multi-armed bandits [19] [21] [22].
- *Correlation between bitrates and resource demands for SR under time-coupled objective function and constraints.* While learning-based approaches can make sequential decisions under uncertainty, they generally rely on stationary decision spaces and independent objective values over different times, falling short of handling our time-dependent resource constraints and the non-smooth objective function.

To tackle the aforementioned challenges, we propose the first online cloud-edge collaborative video streaming system that co-optimizes bitrates and elastic resource allocation for SR at the edge. We offer both a working prototype and theoretical guarantees, through the following **technical contributions**.

First, we propose an optimization-based video streaming system with edge-assisted super-resolution technologies. It enables adaptive, fine-grained, and chunk-specific resource allocation for super resolution, which elastically exploits computational resources in proximity to end users to reduce the latency of streaming high-resolution contents. Driven by our testbed measurements that reveal heterogeneous and uncertain performance relationships with resources, we also realize personalized bitrates and resource allocations for performing SR of different types of requested videos and user geographical regions. This framework can therefore yield higher aggregate video streaming QoE than simply adopting existing ABR algorithms or SR with fixed amounts of resources.

Second, leveraging the structure of the optimization problem, we tackle two main algorithm design challenges: exponential growth of the solution space with streaming requests and temporal dependency in both the objective function and constraints. We first identify a property of the optimal source location-bitrate combinations. Based on this, we reduce the solution space without compromising the algorithm optimality incurred in the problem transformation. We then design a novel window-based combinatorial learning algorithm to manage over-time dependencies in the optimization constraints and non-smoothness in the objective function. Employing a linearly-expanded time window, during which we use the same decisions, mitigates the non-smoothness penalty. Moreover, we integrate primal-dual theories into our learning framework, effectively addressing the global resource constraint over time. Rigorous theoretical analysis is also performed to upper-bound our performance gap against the expected offline optimum.

Third, we implement a prototype of video streaming system with bitrate adaptation and elastic resource supports for real-time SR processes. A light-weight FSRCNN-enabled [23] upscaling scheme of only key-frames is developed at CPU edge servers for SR acceleration. To further realize real-time SR-assisted chunk-level video streaming in the wild, we modify the H.265 encoder based on the HEVC Test Mode [24] and integrate the neural network inference capability of OpenCV [25]. Our prototype provides real-time video SR with various resolutions. It achieves higher streaming quality and aggregate QoE than state-of-the-art systems.

II. Related Work

Cloud-edge collaborative video streaming. The resource efficiency and user experience in video streaming are largely improved by leveraging cloud-based content delivery networks (CDN) [26] and edge caching strategies [6] [27]. Existing works have proposed various algorithms for associations between viewers and content sources [14] [28] [29], and optimized solutions for caching, video transcoding, and content sharing in cloud and edge based CDNs [8] [27] [6] [7]. On the client side, adaptive bitrate (ABR) algorithms [9] [21] [6] are one of the core deployments to improve video quality by adapting to bandwidth fluctuations, with both rule-based [9] [30] [6] [15] and learning-based [21] [19] methods. All the above works do not consider the utilization of SR technologies. Our work capitalizes on the computational resources at the edge to perform SR processes, further improving the QoE of end users especially when encountering bandwidth bottlenecks, which adjusting bitrates solely is insufficient to achieve.

SR-assisted video delivery. Super resolution (SR) especially deep neural network (DNN)-based SR [16] [12] has demonstrated its remarkable performance in image restoration and video enhancement. SR decodes a low-resolution image/video, up-samples it to achieve higher resolutions through DNN inference, and then encodes it before streaming to the end users. Prominent video delivery frameworks NAS [17] and LiveNAS [16] have applied SR on top of video delivery, demonstrating the effectiveness of using pre-trained SR models and performing online SR model training, respectively. Other inspiring systems such as VISCA [6], FlexSRVC [15], and SRAVS [31] integrate SR with new bitrate adaptation or video coding strategies for QoE maximization, assuming SR is performed under fixed and uniform resources for all users. This work instead strikes a balance between SR computation and communication efficiency, considering both fine-grained resource allocation and bitrate adaption for edge-based SR.

Cloud and edge resource schedulers such as YARN [32], Kubernetes [33] and KubeEdge [34] have been extensively studied. In particular, KubeEdge extends the capabilities of Kubernetes to address edge-specific requirements, enabling holistic resource management in heterogeneous edge environments. However, it adopts pre-set rules without capturing unique properties of DNN inference tasks such as DNN-based SR. A number of scheduling systems have been proposed for elastic resource provisioning to machine learning jobs, e.g., Optimus [20], Tiresias [35] and MArk [36]. SR-assisted video streaming systems, such as NEMO [37] and NeuroScaler [18], propose novel video codec design and select the most beneficial anchor frames for SR-based video enhancement. They do not deal with fine-grained resource allocation to perform SR.

Online learning for video streaming. Video streaming systems have adopted reinforcement learning to adjust bitrates not considering SR [21] [19], or select base [17] or target resolutions [31] of videos constructed by SR, balancing bandwidth usage and SR quality. Multi-armed bandits have also been leveraged to select the portion of the panoramic scene delivered to users in interactive video delivery applications to maximize system throughput [38] or choose the network in a multi-home streaming scenario [39]. Due to the time-varying solution space and non-smooth objective function, our problem cannot be solved by existing online learning methods. We augment the learning framework with expanding-window control and integration with online primal-dual optimization.

III. Problem Model

We consider a three-tier video streaming system (Figure 2) with a cloud, an edge cluster comprising multiple servers deployed in one geo-location, and geo-distributed users sending a total of N video requests. Owing to the commoditization of transcoding technologies [7], replicas in multiple resolutions can be generated for an uploaded video and stored in the cloud.[1] For brevity, we define $[Z] \triangleq \{1, 2, \cdots, Z\}$. A set $\mathcal{R}_{i,j}$ of video bitrates, directly mapped to the video resolutions (according to the function that bitrate equals frame rate multiplied by a constant compression ratio and the number of bits per frame determined by resolution), are accessible from the cloud for each chunk (i, j), aka the jth chunk of video request $i \in [N]$. Within the edge cluster, video contents can be dynamically stored into an edge cache, and SR services are applied when needed. In practice, an edge cluster is responsible for serving users' video requests from different geographical regions (e.g., a region can be a city's area).

Rosevin serves users' video requests in a time-slotted fashion, with a total of arbitrarily large and possibly unknown T time intervals. The length of each interval $t \in [T]$ can be tens to hundreds of milliseconds. We define $\Phi(t)$ as the set of video chunk replicas stored in the edge cache in each timeslot $t \in [T]$. The video of request i is divided into a set $\mathcal{J}_i$ of chunks, each of which usually consists of 1.5-4 seconds of successive frames. According to the DASH protocol [40], each chunk is of a uniform time duration. Given a codec and compression format, the size in bits of a chunk equals the chunk duration times bitrate. The user client predetermines a customized target bitrate $r_{i,j}^{target}$ to retrieve for video chunk (i, j) (as supported by Netflix [1] and YouTube [2]). The video requests are handled by the edge cluster in *Rosevin*: if the edge cache stores video chunk (i, j) of target birate $r_{i,j}^{target}$, it is served directly to the user from the edge; otherwise, SR is performed at the edge on the chunk of a lower bitrate retrieved from the edge cache or the cloud (if not cached, and the chunk retrieved from the cloud will be cached at the edge), enhancing its resolution to meet $r_{i,j}^{target}$ and then sending it to the user, e.g., increasing the resolution from 540p to 1080p using one or more CPU cores if $r_{i,j}^{target}$ is the bitrate that matches the 1080p resolution. Specifically, *Rosevin* dynamically controls the following variables in the edge cluster.

Video chunk bitrates and SR resource allocation. For each chunk (i, j) with target bitrate $r_{i,j}^{target}$, a base bitrate $r_{i,j}$, with $r_{i,j} \leq r_{i,j}^{target}$, is decided. Then the video chunk of $r_{i,j}$ will be pulled from the edge cache or the cloud, and SR will be performed to enhance the chunk from $r_{i,j}$ to $r_{i,j}^{target}$. We define $s_{i,j}$ as the number of CPU cores we choose to allocate for performing SR on chunk (i, j) in the edge cluster[2].

Video delivery approach. We define $p_{i,j}$ as the decision of the source location (either the cloud or edge cache) of chunk (i, j) and whether to apply SR to enhance the chunk from $r_{i,j}$ to $r_{i,j}^{target}$, formalized as a two-digit representation:

$$p_{i,j} = \begin{cases} 00 & \text{Cloud to edge} \rightarrow \text{edge to client, no SR} \\ 01 & \text{Cloud to edge} \rightarrow \text{SR} \rightarrow \text{edge to client} \\ 10 & \text{Edge cache to client, no SR} \\ 11 & \text{Edge cache with SR} \rightarrow \text{edge to client} \end{cases}$$

[1] We assume that videos in all feasible bitrates are stored in the cloud; but our method works if the cloud only has a subset of bitrates.

[2] Currently our prototype only schedules CPU cores and already significantly improves user QoE. Commodity edge servers in practice use CPUs for decoding, inference, and (re)encoding in SR processes. Although the inference part can be run on GPUs, the single-GPU inference latency is already very small, negating the need of allocating multiple GPUs for an SR process.

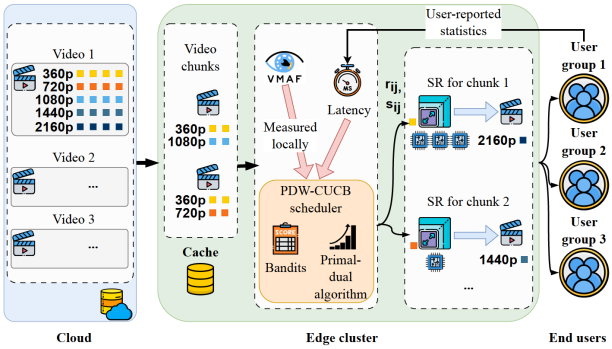


Fig. 2: System architecture of *Rosevin*.

The first bit of $p_{i,j}$ represents whether chunk (i,j) is fetched from the edge cache (=1) or from the cloud (=0), and the second bit denotes whether SR is applied (=1) or not (=0). For instance, $p_{i,j} = 01$ indicates that chunk (i,j) is obtained from the cloud in bitrate $r_{i,j}$ to the edge cache and then enhanced to $r_{i,j}^{target}$ through SR before transmitted to the client. $p_{i,j} = 00$ means that chunk (i,j) is fetched from the cloud, cached at the edge for potential future reuse by other users, and then transmitted to the client, without using SR.

Let $q(r_{i,j}, s_{i,j})$ represent the video quality of chunk (i,j) achieved by applying SR on the chunk in base bitrate $r_{i,j}$, supported by a total of $s_{i,j}$ CPU cores. An illustration of SR processes concurrently performed in the edge cluster for two video chunks is shown in our system architecture diagram Figure 2. We use VMAF [41] as the quality metric, which can estimate user perceptual video quality levels. Since no study has provided a concrete model on the effectiveness of SR under varying numbers of CPU cores, $q(r_{i,j}, s_{i,j})$ is an unknown function that may vary over different i and j. We define the QoE of each chunk (i,j) as the weighted sum of video *quality* $q(r_{i,j}, s_{i,j})$, *latency* $l(p_{i,j}, r_{i,j}, s_{i,j})$ (from the time when the edge cluster receives the video chunk request to when the chunk that meets the target rate is successfully delivered to the user), and *non-smoothness* between chunks [15] [19] (i,j) and $(i,j-1)$ for penalizing the abrupt changes of qualities between successive chunks of the same requested video. The QoE of each user i is formalized as follows, and the goal of *Rosevin* is to maximize the total QoE of all users.

$$QoE_i(\mathbf{p}_i, \mathbf{r}_i, \mathbf{s}_i) = \sum_{j \in \mathcal{J}_i} \big(\alpha q(r_{i,j}, s_{i,j}) - \beta l_{i,j}\left(p_{i,j}, r_{i,j}, s_{i,j}\right) - \gamma \left|q(r_{i,j}, s_{i,j}) - q(r_{i,(j-1)}, s_{i,(j-1)})\right|\big) \quad (1)$$

In (1), α, β, and γ are tunable importance coefficients chosen by the video streaming service provider. Latency $l(\cdot,\cdot,\cdot)$ consists of the delays of transmission and propagation, and SR execution time if SR is used. Let $\tau(r_{i,j}, s_{i,j})$ be the total SR processing time due to waiting in the queue, decoding, SR inference, and (re)encoding. More resources or a higher base bitrate could lead to a smaller $\tau(r_{i,j}, s_{i,j})$ (Figure 1), but the actual function forms of $\tau(r_{i,j}, s_{i,j})$ and $l_{i,j}$ are uncertain. We define $B_{t_{i,j}}^{edge}$ (and $B_{t_{i,j}}^{cloud}$) as the average bandwidth from the edge to client i (and that on the bottleneck link from the cloud to the edge) starting from time $t_{i,j}$ to the time that the delivery of chunk (i,j) is completed. We use $RTT_{t_{i,j}}^{oc}$ and $RTT_{t_{i,j}}^{ec}$ to denote the round-trip time (propagation delay) between the cloud and the client via the edge, and between the edge and the client, respectively. Formally, with $\pi_{i,j}(r)$ denoting the file size of chunk (i,j) at rate r, the latency function is:

$$l_{i,j} = \begin{cases} \frac{\pi_{i,j}(r_{i,j}^{target})}{B_{t_{i,j}}^{cloud}} + RTT_{i,t}^{oc}, & \text{if } p_{i,j} = 00 \\ h_\chi\left(trans_{oc}^{i,j}, \tau(r_{i,j}, s_{i,j})\right) + RTT_{i,t}^{oc}, & \text{if } p_{i,j} = 01 \\ \frac{\pi_{i,j}(r_{i,j}^{target})}{B_{t_{i,j}}^{edge}} + RTT_{i,t}^{ec}, & \text{if } p_{i,j} = 10 \\ h_\chi(trans_{ec}^{i,j}, \tau(r_{i,j}, s_{i,j})) + RTT_{i,t}^{ec}, & \text{if } p_{i,j} = 11 \end{cases}$$

Here, $h_\chi(\cdot)$ is a χ-norm function with an unknown factor $\chi \geq 1$, which captures the time of performing SR once on each group of pictures (GOP) within a chunk that have arrived at the edge and then transmitting the enhanced portion to the client (rather than waiting for the entire chunk). $trans_{ec}^{i,j}$ denotes the transmission time of delivering chunk (i,j) from the edge to the client. $trans_{oc}^{i,j}$ is the transmission time from the cloud to the client via the edge, which is more complex if using SR, due to chunk size changes at the edge after being processed through SR. *Note that this formulation is just for rigorous presentation and showing the uncertainty of latency, but not to complicate the problem. Our algorithm is agnostic to the latency formulation, achieving the design goal of bypassing this complexity and uncertainty of different terms in* $l_{i,j}$.

Our optimization goal is to maximize the total QoE of serving N video requests subject to the constraints characterizing the dependency between our three sets of control variables.

$$\underset{\mathbf{p},\mathbf{r},\mathbf{s}}{\text{maximize}}: \sum_{i \in [N]} QoE_i(\mathbf{p}_i, \mathbf{r}_i, \mathbf{s}_i) \quad (2)$$

$$\text{s.t.: } p_{i,j} \in \{00, 01, 10, 11\}, \quad \forall i,j \quad (3)$$

$$p_{i,j} \in \{00, 01\}, \text{ if: } r_{i,j} \notin \Phi(t_{i,j}), \quad \forall i,j \quad (4)$$

$$r_{i,j} = r_{i,j}^{target}, \text{if: } p_{i,j} \in \{00, 10\}, \quad \forall i,j \quad (5)$$

$$s_{i,j} = 0, \text{ if: } p_{i,j} \in \{00, 10\}, \quad \forall i,j \quad (6)$$

$$r_{i,j} \in \mathcal{R}_{i,j}, \quad \forall i,j \quad (7)$$

$$s_{i,j} \in \{0, 1, 2, \cdots, C\}, \quad \forall i,j \quad (8)$$

$$\sum_{\substack{i \in [N], j \in \mathcal{J}_i: \\ t_{i,j} \leq t \leq l_{i,j} + t_{i,j}}} s_{i,j} \leq C, \quad \forall t \quad (9)$$

Here, $\forall i,j$ represents $\forall i \in [N], j \in \mathcal{J}_i$, for brevity. Constraint (4) indicates that any chunk in our chosen base bitrate $r_{i,j}$ must be fetched from the cloud, if it is unavailable in $\Phi(t_{i,j})$. We must also choose $r_{i,j} = r_{i,j}^{target}$ (constraint (5)) if selecting a $p_{i,j}$ that indicates no SR, and thus no SR resource is allocated (constraint (6)). Further, (7) and (8) define the full sets of $r_{i,j}$ and $s_{i,j}$. Constraint (9) requires that the total number of allocated CPU cores for all the *alive* SR processes cannot exceed the resource capacity C. The duration of occupying the CPUs to preform SR for each (i,j) starts from $t_{i,j}$ until its completion time $l_{i,j} + t_{i,j}$, both depending on the bitrate $r_{i,j}$, streaming approach $p_{i,j}$, and resources $s_{i,j}$.

When each time slot t starts, *Rosevin* decides the base bitrates of video chunks that need to be served in the interval $[t, t+1)$, where to get the chunks, and how many CPU cores to use if performing SR to enhance the chunks to meet user's target bitrate. Since there might be thousands of concurrent requests from end users, it is exceptionally hard to simultaneously co-optimize the bitrates for individual users. For better scalability, we categorize the users into $\tilde{N}$ co-existing groups, where users in the same group might have similar communication quality towards the edge cluster and the same requested video currently. Grouping users can be achieved by existing methods, e.g., clustering or graph matching algorithms [14], [28], which is not the focus of this paper. We introduce in Section IV the core design of *Rosevin*, a new combinatorial learning framework to solve online optimization with a non-smooth objective function that penalizes the changing decisions over time and the complex dependencies between different variables.

IV. Design Logic and Algorithm Framework

Solving (2)–(9) on the fly is harder than handling combinatorial optimizations such as Multiple Choice Knapsack Problem due to: (i) unknown functions of $q(r_{i,j}, s_{i,j})$ and $l_{i,j}(p_{i,j}, r_{i,j}, s_{i,j})$; (ii) high-dimensionality in the decision space; (iii) the temporal correlation of decisions in both the objective function (non-smoothness part) and the resource capacity constraint in (9). To address challenges (i) and (ii), we propose to leverage the combinatorial multi-armed bandits (CMAB) framework to learn unknown functions/parameters through sequential online feedbacks and cope with combinatorial decisions (arms). However, CMAB is still prone to slow convergence under complex decision space [22], e.g., with multiple interrelated variables $p_{i,j}$, $r_{i,j}$, and $s_{i,j}$. Therefore, we conduct a problem transformation in Section IV-A for solution space reduction. To address challenge (iii), we build a novel framework, which greatly extends the capability of CMAB by integrating online primal-dual theories and a window-based control strategy into the algorithm design and theoretical analysis. We prove that *Rosevin* achieves a sub-linear performance gap against the expected optimum. Our theoretical framework and analysis can also be generalized to multi-resource cases, which will be shown in our future work.

A. Problem Transformation

We first show that our decision space can be simplified by removing the decisions $\{p_{ij}\}_{i,j}$ and then map the problem into the combinatorial bandit framework.

Mapping $p_{i,j}$ from $r_{i,j}$. Revisiting the original latency function $l_{i,j}$, we observe that: given a decided base bitrate $r_{i,j}$, if the chunk (i,j) in $r_{i,j}$ is available in the edge cache, we must use the chunk in the edge rather than the cloud, no matter if we apply SR (if $r_{i,j} < r_{i,j}^{target}$) or not (if $r_{i,j} = r_{i,j}^{target}$). The reason is that transmitting video chunks from the cloud to the client needs to traverse the link from the cloud to the edge first and thus incurs a latency that is at least that of transmitting the same chunk from the edge to the client; SR is also performed at the edge. Moreover, if $r_{i,j} = r_{i,j}^{target}$, indicating no SR needed, then we have $\tau_{i,j} = 0$. Let $\tau_{i,j}(r, r) \triangleq 0, \forall r$, indicating zero SR time incurred if $r_{i,j} = r_{i,j}^{target}$. Our latency function can be simplified as: $l_{i,j} = h_\chi\left(trans_{ec}, \tau_{i,j}(r_{i,j}, r_{i,j}^{target})\right) + \text{RTT}_{ec}$ if $r_{i,j} \in \Phi(t_{i,j})$, and it equals $h_\chi\left(trans_{oc}, \tau_{i,j}(r_{i,j}, r_{i,j}^{target}\right) + \text{RTT}_{oc}$ otherwise.

Hence, we do not need to *jointly* optimize $p_{i,j}$ with $r_{i,j}$; instead, the optimal $p_{i,j}$ can be decoded given a chosen $r_{i,j}$:

$$p_{i,j} = \begin{cases} 00 & \text{if } r_{i,j} = r_{i,j}^{target} \text{ and } r_{i,j} \notin \Phi(t_{i,j}) \\ 01 & \text{if } r_{i,j} < r_{i,j}^{target} \text{ and } r_{i,j} \notin \Phi(t_{i,j}) \\ 10 & \text{if } r_{i,j} = r_{i,j}^{target} \text{ and } r_{i,j} \in \Phi(t_{i,j}) \\ 11 & \text{if } r_{i,j} < r_{i,j}^{target} \text{ and } r_{i,j} \in \Phi(t_{i,j}) \end{cases} \tag{10}$$

Next, we map the problem with our reduced decision space to the CMAB framework and address the temporal dependencies.

Decision space mapping. We define $R \triangleq \max_{i,j} |\mathcal{R}_{i,j}|$ and C to denote the maximum number of possible discrete values of the bitrates and resource units, respectively. We then formalize the decision space as $\{\Delta_i^{R\times C\times \tilde{N}}\}_{i\in[\tilde{N}]}$ (recall that $\tilde{N}$ represents the number of user groups). Here, $\{\Delta_i^{R\times C}\}$ is an $(R\times C)$-dimensional one-hot vector, e.g., $\Delta_i = (1, 0, \cdots, 0)$ represents the first combination of $r_{i,j}$ and $s_{i,j}$ values (bitrate mapped from resolution of 360p and 1 CPU core) for the chunk of request i that arrives at the current time. We then construct a total of $2RC\tilde{N}$ *base-arms*, each of which represents a combination of a feasible value of $r_{i,j}$ and that of $s_{i,j}$, *as well as whether chunk (i,j) is in the cache ($r_{i,j} \in \Phi(t)$) or in the cloud ($r_{i,j} \notin \Phi(t)$)*. We call each base-arm a **rate-resource configuration**. We define $M \triangleq 2RC$ and use $m \in [M]$ to index the base-arms for each group $i \in [\tilde{N}]$. For instance, in Figure 3, there are a total of 2×2 rate-resource configurations for the chunk of group 1 that needs to be processed in the current timeslot, with (360p, 1 CPU), (360p, 2 CPUs) available if fetching the chunk from the edge cache, while (360p, 1 CPU), (360p, 2 CPUs), (720p, 1 CPU), (720p, 2 CPUs) are available if the scheduler takes the chunk from the cloud. The idea of our algorithm is to maintain an estimated QoE for each of the $2RC$ configurations and leverage these estimates to solve our optimization problem (Section IV-B).

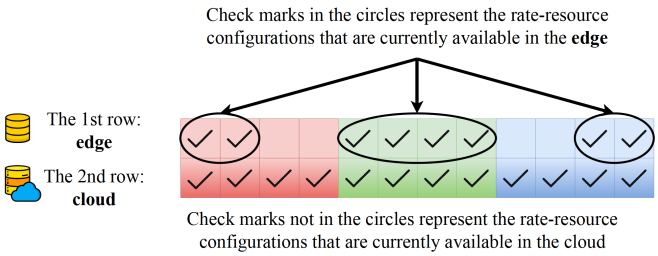


Fig. 3: An illustration of base-arms of three user groups (shown in three colors) and four rate-resource configurations. A configuration $(r_{i,j}, s_{i,j})$ labeled by a check mark in the circle means that a chunk in bitrate $r_{i,j}$ is in the cache, and a total of $s_{i,j}$ CPU cores are allocatable on the edge servers.

Reward design. Revisiting (1), using our problem transformation and the CMAB framework, we need not measure or predict different terms in $l_{i,j}$ (such as SR processing times and RTTs) separately. Let $r_{i,j,m}$ and $s_{i,j,m}$ denote the base bitrate and the allocated resource number in the mth rate-resource configuration. We have $s_{i,j,m} = 0$ if and only if $r_{i,j,m} = r_{i,j}^{target}$, meaning no SR needed with configuration m. For each m of group i, we construct a reward as:

$$\mu_{i,j,m} = \alpha q(r_{i,j,m}, s_{i,j,m}) - \beta l_{i,j}(r_{i,j,m}, s_{i,j,m}) \tag{11}$$

We also define the expected reward and latency of each request group i, when choosing the mth configuration, as $\mu_{i,m}$ and $l_{i,m}$, respectively. Note that we have excluded the non-smoothness term (see (1)) from (11). This is due to the potential of introducing estimation biases when a term, dependent on decisions at two successive time slots, is included in (11). In order to account for the non-smoothness, we instead propose a window-based algorithm with a consistent configuration within each window. But how to set the window size affects our algorithm performance and will be elaborated in Section IV-B.

B. A New Online Algorithm PDW-CUCB

Recall that our problem entails two types of temporal dependency across decisions, i.e., a global resource constraint over time and the non-smoothness penalty, which online learning algorithms (e.g., bandits, FTRL [42]) do not capture. We propose a novel online learning algorithm that can (i) regulate non-smoothness using an expanding time window with stable rate-resource configurations within each window (see Section IV-B2), and (ii) solve an online combinatorial problem that involves global resource constraints, using a primal-dual based algorithm facilitated by learned parameters (Section IV-B1).

1) Managing the long-term capacity constraint: Let $x_{i,j,m}$ be a variable that equals 1 if the mth base-arm is chosen for chunk (i,j) and equals 0 otherwise. Our goal is to maintain estimated rewards defined in (11) of each mth base-arm and solve (2)–(8). We further identify that, if we know the true expectations of reward and latency of each base-arm and exclude the non-smoothness term, the online Multiple Choice Knapsack Problem (MCKP) can be reduced to our problem (2)–(8) (the proof on this reduction will be shown in our future long version of this work). We formulate the MCKP problem by lifting the resource constraint to the objective function ($-f(\cdot)$). Let $\mathcal{A}_t$ denote the set of alive chunks at t, satisfying $t_{i,j} \le t \le t_{i,j} + l_{i,j,m}$, our optimization problem with expected rewards $\mu_{i,j,m}$ defined in (11) is:

$$\text{maximize}_{\mathbf{x}} \sum_{i\in[\tilde{N}]} \sum_{j\in\mathcal{J}_i} \sum_{m\in[M]} \mu_{i,j,m} x_{i,j,m} - f(y_t) \tag{12}$$

$$\text{s.t.} \sum_{m\in[M]} x_{i,j,m} = 1, \forall i \in [\tilde{N}], j \in \mathcal{J}_i \tag{13}$$

$$\sum_{(i,j)\in\mathcal{A}_t} \sum_{m\in[M]} s_{i,j,m} x_{i,j,m} \le y_t, \forall t \in [T] \tag{14}$$

$$x_{i,j,m} \in \{0,1\}, \forall i \in [\tilde{N}], j \in \mathcal{J}_i, m \in [M] \tag{15}$$

Here, function $f(y_t)$ is a virtual penalty function defined as $f(y_t) = \begin{cases} 0 & \textbf{if } y_t \in [0, C] \\ +\infty & \textbf{if } y_t > C \end{cases}$, where y_t is the total number of CPU cores used by SR processes in t, such that we prohibit exceeding the resource capacity by adding an infinitely large penalty in the objective function. Besides, $l^+_{i,j,m} \triangleq t_{i,j} + l_{i,j,m}$ denotes the expected time that the SR process of chunk (i,j) is completed. To solve (12)–(15), we adopt the primal-dual theories. We define dual variables $\lambda_{i,j}$ and σ_t associated with (13) and (14), respectively. $f^*(\sigma_t)$ is the conjugate of $f(\cdot)$, which equals $\sup_{y_t\ge 0}\big(\sum_t y_t\sigma_t - \sum_t f(y_t)\big)$. We further relax $x_{i,j,m}$ to be $x_{i,j,m} \ge 0$ and derive the dual problem as follows:

$$\text{minimize}_{\lambda,\sigma} \sum_{t\in[T]} \sum_{i\in[\tilde{N}], j\in\mathcal{J}_i} \lambda_{i,j} + f^*(\sigma_t) \tag{16}$$

$$\text{s.t. } \lambda_{i,j} \ge \mu_{i,j,m} - \sum_{t=t_{i,j}}^{l^+_{i,j,m}} s_{i,j,m}\sigma_t, \forall i,j,m \tag{17}$$

$$\lambda_{i,j} \ge 0, \sigma_t \ge 0, \forall i \in [\tilde{N}], j \in \mathcal{J}_i, t \in [T] \tag{18}$$

Arm selection. To solve for the integer programming problem with binary variables $x_{i,j,m}$, we leverage the KKT condition [43] which implies to choose the configuration $m^*_{i,j}$ for (i,j) that yields the largest right hand side of constraint (17) among all m. Formally, $x_{i,j,m^*_{i,j}} = 1$ for each (i,j), where $m^*_{i,j} = \text{argmax}_m \left\{ \mu_{i,j,m} - \sum_{t=t_{i,j}}^{l^+_{i,j,m}} s_{i,j,m}\sigma_t \right\}$, and $x_{i,j,m} = 0$ for all $m \ne m^*_{i,j}$ (line 8 of Alg. 1). To realize this, we also need a feasible solution of σ_t, for which we use $\sigma_t = \frac{U_{min}}{2}\left(2U_{max}/U_{min}\right)^{\frac{y_t}{C}}$ (line 4), inspired by [44], [45]. Here, U_{max} and U_{min} are the maximum and minimum rewards when allocating one CPU core to support SR, and we suppose that we can estimate these from history.

2) Regulating the non-smoothness: Recall that the non-smoothness term $-\lambda\left|q(r_{i,j}, s_{i,j}) - q(r_{i,(j-1)}, s_{i,(j-1)})\right|$ in (1) was not captured in (12). But we need to reduce the penalty of choosing different configurations for successive chunks of the same request. To achieve bounded long-term non-smoothness, we maintain a window consisting of one or more timeslots and only update $r_{i,j}$, σ_t, and $s_{i,j}$ using the approach specified in Section IV-B1 when each new window starts. Our scheme is agnostic of the potentially unknown total system timespan T, by increasing the length of the current window to be the number of windows that have passed. Formally, we define $w \in \{0, \cdots, W\}$ to index the window. We design the length of each window w, denoted by η_w, to be $\eta_w = \max\{1, w\}$, and we will explain the rationale in Lemma 1.

Algorithm workflow. To learn the expected rewards (11), we define $\bar{u}_{i,m,t}$ and $\bar{l}_{i,m,t}$ as the empirical means of $\mu_{i,m}$ and $l_{i,m}$, under base-arm $m \in \mathcal{M}$ for group i, updated in timeslot t. Let $\Theta_{i,m,t}$ denote the number of times that base-arm m is chosen for i until timeslot t. For ease of presentation, we use the subscription t in $\Theta_{i,m,t}$ and $\bar{u}_{i,m,t}$, but our Algorithm 1 only needs to maintain a single value of $\Theta_{i,m}$, $\bar{u}_{i,m}$, and $\bar{l}_{i,m}$ for each base-arm m of group i at any timeslot, so we omit the t subscription in Algorithm 1. Let $\hat{u}_{i,m,t}$ and $\hat{l}_{i,m,t}$ denote

Algorithm 1: A Primal-dual and Window Control aided Combinatorial UCB Algorithm (*PDW-CUCB*)

Input : $\tilde{N}, \{\mathcal{R}_i\}_{i\in[\tilde{N}]}, \{\mathcal{C}_k\}_{k\in[K]},\ U_{max},\ U_{min}$
Output : $\mathbf{x}_i,\ \forall i \in [\tilde{N}]$
Initialize: $\bar{\mathbf{u}} = \mathbf{0}, \mathbf{\Theta} = \mathbf{0}, t = 0, w = 0, t_w = 0$
1 **while** *the video streaming service is running* **do**
2 $t \mathrel{+}= 1$;
3 Update the total allocated resource y_t;
4 Update the dual variable: $\sigma_t = \frac{U_{min}}{2}\left(2U_{max}/U_{min}\right)^{\frac{y_t}{C}}$;
5 **if** $t > t_w + w$ **then**
6 $t_w = t$;
7 $w \mathrel{+}= 1$;
8 For each $i \in \tilde{N}$, choose configuration $m^*_{i,j}$: $x_{i,j,m^*_{i,j}} = 1$, where $m^*_{i,j} = \text{argmax}_m \left\{\hat{\mu}_{i,m} - \sum_{t=t_{i,j}}^{\hat{l}_{i,j,m}} s_{i,j,m}\sigma_t\right\}$, and set $x_{i,j,m} = 0$ for all $m \neq m^*_{i,j}$;
9 Obtain instantaneous reward $\mu_{i,m^*_{i,j}}, \forall i$ defined in (11) by measuring the quality and latency;
10 Update averaged reward $\bar{u}_{i,m}$ and latency $\bar{l}_{i,m}$;
11 Update $\hat{\mu}_{i,m^*_{i,j}} = \bar{\mu}_{i,m^*_{i,j}} + U_{max}\sqrt{\frac{3\ln w}{2\Theta_{i,m^*_{i,j}}}}$;
12 Update $\hat{l}_{i,m^*_{i,j}} = \bar{l}_{i,m^*_{i,j}} + L_{max}\sqrt{\frac{3\ln w}{2\Theta_{i,m^*_{i,j}}}}$;
13 Update $\Theta_{i,m^*_{i,j}} \mathrel{+}= 1,\ \forall i$;

the upper confidence bounds (UCBs) [22] of $u_{i,m}$ and $l_{i,m}$ updated in t. The logrithmic regularizer balances the exploration-exploitation tradeoff, encouraging a higher UCB for under-explored base-arms. U_{max} and L_{max} are maximum values of reward and latency, serving as scalars of the regularizers in UCB. We solve for the bitrate and resource allocation using the strategy elaborated in **Arm selection** of this section at the beginning of each window w (lines 3–8), and the scheduler continuously updates the estimated reward and latency under each base-arm for future timeslots (lines 11, 12).

Intuition of our adaptive window scheme. We show the idea of designing the window length through the following lemma and theorem. Basically, we wish to construct a small number of windows (W) in order to guarantee a non-smoothness sublinear with T, as the total non-smoothness is upper-bounded by $O(W)$. However, there's a trade-off: *a small number of windows comes with a large window length, and wrong decision made at the beginning of a window will be used for the entire window.* To navigate this trade-off, we first set $\eta_w = \max\{1, w^\alpha\},\ \alpha > 0$, which expands as times goes on to cope with the unknown T, and then we tune α in the analysis. Other choices such as using window sizes exponentially growing with the time-slot t is too aggressive for our form of objective function, based on our analysis. Our final choice $\eta_w = w$ for Algorithm 1 is a special case with $\alpha = 1$, which is optimized through our theoretical performance analysis by minimizing the aggregate errors per window over all windows and non-smoothness. Here, we provide the proof of Lemma 1 with $\eta_w = \max\{1, w^\alpha\}$ rather than $\eta_w = w$, to better indicate how to apply our window scheme into other scenarios that share the similar optimization structure with ours. We finally tune the factor α in proving Theorem 1.

Lemma 1. *If using $\eta_w = \max\{1, w^\alpha\},\ \alpha > 0$, the total number of windows that our Algorithm 1 maintains is at most $(T(\alpha+1))^{\frac{1}{\alpha+1}}$, while T can be oblivious to the algorithm.*

Proof. Let $W = (T(\alpha+1))^{\frac{1}{\alpha+1}}$. Taking the integral of the number of timeslots over all windows up to W, we have:

$$\int_{w=0}^{(T(\alpha+1))^{\frac{1}{\alpha+1}}} w^\alpha \mathbf{d}w \geq T \longrightarrow \sum_{w=0}^{W} \eta_w \geq T, \qquad (19)$$

meaning that $W = (T(\alpha+1))^{\frac{1}{\alpha+1}}$ windows sufficiently cover the entire system span T if using window size w^α. □

Lemma 1 upper-bounds the total number of windows W with respect to T without knowing T in each t (we omit the time slots without request arrivals), which is crucial in bounding the overall QoE gap (shown in Theorem 1). Given Lemma 1 with $\alpha = 1$ (lines 5–7), we provide the following performance guarantee of Algorithm 1 in Theorem 1. Let ϵ represent the competitive ratio [45], i.e., the QoE achieved by PDW-CUCB under perfectly estimated rewards, against the offline optimum. We adopt the performance metric of CUCB, by defining the *Regret* as the gap between an ϵ-fraction ($\epsilon \in (0, 1]$) [22] of the expected total reward using the optimal decisions and the expected total reward achieved with our chosen $\mathbf{x}_i, \forall i$ (under imperfectly estimated rewards):

$$\text{Regret} = \epsilon \max_{\{\mathbf{x}'_\mathbf{i}\}_{i\in[N]}} \mathbb{E}\left[\sum_{i\in[N]} QoE_i(\mathbf{x}'_\mathbf{i})\right] - \mathbb{E}\left[\sum_{i\in[N]} QoE_i(\mathbf{x}_i)\right]$$

Theorem 1. *If $q(r_{i,j,m},\ s_{i,j,m})$ and $l_{i,j}(r_{i,j,m}, s_{i,j,m})$ are identically and independently distributed over j (allowed to be different across i and m), the regret incurred by PDW-CUCB is at most $O((\tilde{N}RC\Delta_{max} + \lambda_{max})T^{\frac{1}{2}})$, where Δ_{max} is the maximum reward per time, λ_{max} is the maximum penalty $\lambda|q(r_{i,j}, s_{i,j}) - q(r_{i,(j-1)}, s_{i,(j-1)})|$ over all possible choices of $\mathbf{s}_i, \mathbf{s}_{i,(j-1)}, r_{i,j}$, and $r_{i,(j-1)}$ for all i, j. Besides, PDW-CUCB has a time complexity at most $O(RC\tilde{N})$ per timeslot.*

Theorem 1 shows that our *Regret* is sub-linear with the system makespan T, even when T is oblivious to PDW-CUCB before the streaming service ends.

Proof Sketch. Recall $M \triangleq \tilde{N}RC$. Our achieved expected QoE of N users requesting video chunks in T timeslots is at least:

$$\epsilon Q^* - \left(M\sum_{w=1}^{\tilde{W}} \eta_w + \sum_{w=1}^{W} \frac{2M}{w^2}\right)\Delta_{max} - W\lambda_{\max} \qquad (20)$$

The first term ϵQ^*, guaranteed by a $\epsilon \leq O(log(\tilde{N}))$ competitive ratio [45], is not our main concern. The last term

$W\lambda_{max}$ is upper-bounded by $\lambda_{max}(T(\alpha+1))^{\frac{1}{\alpha+1}}$ (Lemma 1). The middle part with $\tilde{W} = \frac{6\ln W}{(f^{-1}(\Delta_{min}))^2}$, arises from error accrued before window $\tilde{W}$, capped by $M\eta_w$ per window, and with a maximum error incidence probability of $\frac{1}{w^2}$ (Hoeffding-Azuma inequality [42]). Minimizing the orders of these terms by tuning $\alpha = 1$ yields the regret bound. As PDW-CUCB is model-free and threshold-based, its time-complexity is linear in the number of base-arms per timeslot. □

V. Implementation and Experiments

We implement a prototype of *Rosevin* on a local testbed and evaluate its performance with real-world experiments.

A. Prototype implementation

1) Testbed: Our end-to-end SR-assisted video streaming system implements the cloud-edge-user architecture in Figure 2. **The cloud servers** are built with 3 VM instances rented from a public cloud, which use FFmpeg to stream different video chunks to the edge. **The edge cluster** is hosted on 4 servers, each equipped with 32 CPU cores and 128GB RAM. The physical resources of these edge servers are managed with Kubernetes [46]. The average RTT and bandwidth between the cloud and the edge cluster are measured to be 43.4ms and 186.04Mbps, respectively.

We separate the SR-assisted video service on the edge cluster into 3 collaborating services and deploy each service as containers. The cache service receives video chunks streamed from the cloud servers and uses LRU caching policy to update the cached chunks. The cache capacity is set to 8GB. A scheduler service performs QoE measurement of the streamed chunks (Sec. V-A3) and runs the PDW-CUCB algorithm. At the beginning of each time slot, the scheduler generates new decisions for all the incoming chunk requests collected from the users. In case that SR is needed to boost the chunk resolution, the scheduler launches a container occupying a certain amount of CPU resources to execute the real-time SR task (Sec. V-A2). The streaming service pushes the video chunks with the target resolutions back to the users.

Five additional servers are used to emulate the **end users**. On each server, we launch 100 DASH-based video players to emulate users from the same group, generate the streaming requests with varying target resolutions for 50 available video clips. The RTT values between these servers and the edge cluster are enforced using Linux TC according to the real-world measurements from [47].

2) Real-time super-resolution system: We implement our real-time super-resolution system in the edge cluster by upgrading the state-of-the-art key-frame-based SR pipeline [15], [37] via extra implementation with libavcodec and libavformat in FFmpeg (i.e., version=5.0). In order to accelerate the key-frame-based SR using neural network inference, we integrate the interfaces of DnnSuperReslmpl modules of OpenCV [25] (i.e., version=4.5.2) that can invoke the TensorFlow inference function efficiently. We also exploit the light-weight pre-trained SR model FSRCNN [23]. Our real-time SR system is developed as container images with different resource configurations. Generally, up-scaling to 4K resolution, i.e., $\times 4$ of 540p requires only 5-8 CPU cores to achieve real-time chunk-level decoding, inference, and (re)encoding. So the resource configurations of the container images are limited to 1 CPU core, 3 CPU cores, and 5 CPU cores in our evaluation.

3) QoE measurement and evaluation set-up: To calculate the QoE in (1), the edge scheduler first measures the VMAF [41] of each video chunk before transmitting it to the users. The emulated users are then instrumented to report their latency measurements to the edge through remote RPCs. The coefficients α, β, and γ of the QoE function (1) are set to 3.0, 4.0, and 5.0, respectively. Each timeslot lasts 100ms.

We source raw videos from the UVG 4K Video dataset [48] and a public dataset [49], and encode them to resolutions {360, 540, 720, 1080, 1440, 2160}p, or the corresponding bitrates $\mathcal{R}$={0.5, 1, 1.5, 2, 15, 60}Mbps. Each chunk is 2 seconds long, and the target bitrate of each request is randomly sampled from $\mathcal{R}$. The number of video chunks per requested video is randomly generated from [30, 150], and the video is played on repeat [48], [49]. Streaming requests generated from the emulated users follow a Poisson process with a default (varying in Fig. 4c) expected inter-arrival time of 1 second.

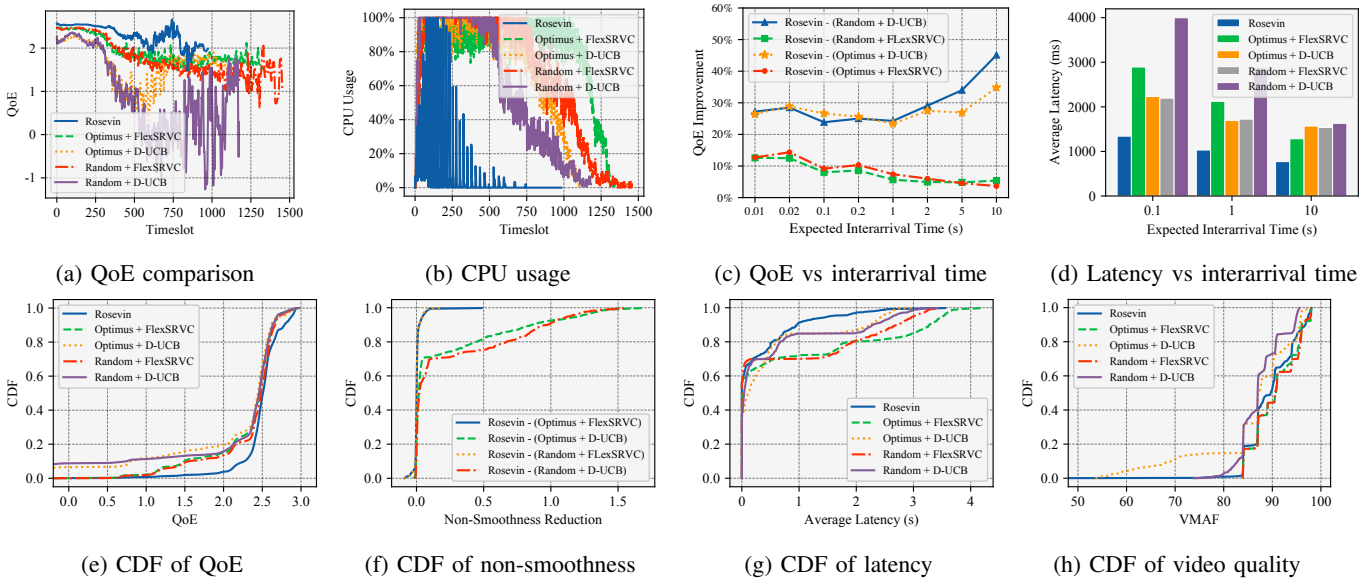
B. Baselines

We combine representative bitrate and resource allocation methods for DNN-based tasks as our baselines, as there's no existing co-optimization of computational resource allocation and bitrate adaptation. These baselines under-perform *Rosevin*, demonstrating the value of joint optimization and online learning leveraged by *Rosevin*. Especially, we use combinations of the following, i.e., each paring a resource allocation method with a bitrate adaptation strategy, as baselines that jointly select the numbers of CPU cores for each chunk of SR process and the base bitrate of the chunk.

- **Resource allocation method I: Optimus [20]** represents prediction-model-based scheduling for ML tasks, requiring precise estimated task execution speeds.
- **Resource allocation method II: Random** uniformly at random chooses CPU core number for each SR process.
- **Bitrate adaptation method I: FlexSRVC [15]** extends the MPC algorithm [10] by solving the best bitrates using predicted throughputs.
- **Bitrate adaptation method II: D-UCB [50]** uses discounted UCB scores to learn the best bitrates for each group.

C. Performance evaluation

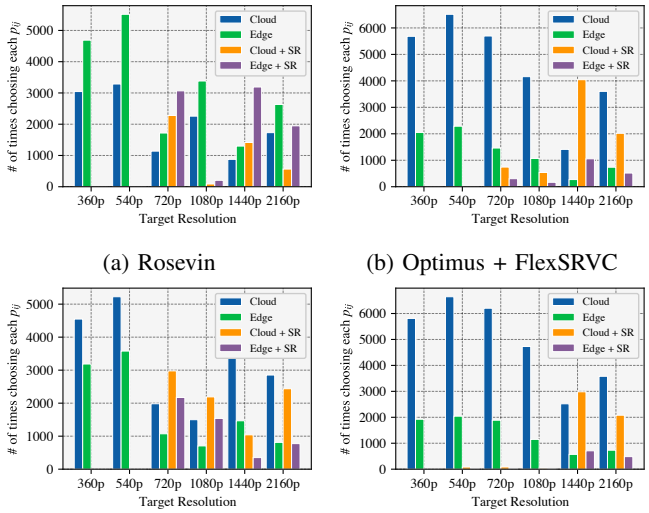
In Figure 4, we examine the performance of *Rosevin* on various metrics. Figure 4a shows that *Rosevin* achieves 7.95%, 20.29%, 6.17%, and 21.71% higher QoE than **Optimus + FlexSRVC**, **Optimus + D-UCB**, **Random + FlexSRVC**, and **Random + D-UCB**, respectively. Figure 4e further demonstrates that Rosevin also effectively reduces the percentages of video streaming requests with lower QoEs. Diving deeper into other performance metrics, we observe a significantly lower CPU usage of *Rosevin* as shown in Figure 4b, mainly



(a) QoE comparison (b) CPU usage (c) QoE vs interarrival time (d) Latency vs interarrival time

(e) CDF of QoE (f) CDF of non-smoothness (g) CDF of latency (h) CDF of video quality

Fig. 4: Rosevin achieves higher average QoE than baselines that jointly optimize bitrates and the allocation of CPU cores



(a) Rosevin (b) Optimus + FlexSRVC

(c) Optimus + D-UCB (d) Random + FlexSRVC

Fig. 5: Fine-grained evaluation of Rosevin and baselines

due to the reduced latencies of transmission, queuing, and SR decoding, which constitute the CPU occupation duration of alive video transmissions. Figure 4c shows a persistently higher QoE of *Rosevin*. Figure 4d also depicts that with shorter request interarrival times (e.g., higher arrival rates), that lead to more intensive competition in edge CPU cores and cloud-edge network bandwidth, *Rosevin* provides higher latency reduction compared to the baselines. Combining the results shown in Figure 4f and Figure 4g, we observe that *Rosevin* not only ensures better smoothness in video playback but also achieves a lower end-to-end latency. Its final video quality is also comparable to the baselines (Fig. 4h), indicating that applying SR to enhance videos to desired bitrates does not undermine the average video quality over all the requests.

Fine-grained comparison of algorithms' decisions. We further conduct in-depth examination of the advantages of *Rosevin* by comparing the numbers of times of choosing different $p_{i,j}$ configurations (indicating source location of each chunk and whether to apply SR) against the baselines. In Fig. 5, Cloud, Edge, Cloud+SR, Edge+SR correspond to $p_{i,j} = 00, 10, 01, 11$, respectively. *Rosevin* makes more proactive use of edge caching and SR than all the baselines (Random + D-UCB is relatively similar to Optimus + D-UCB and omitted), reducing the potentially high latencies caused by limited bandwidth. The higher CPU usage of baselines shown in Figure 4b can be explained by that the baselines use cloud and cloud + SR more often, incurring a high latency as well (Fig. 4g). Furthermore, we observe that D-UCB tends to select cloud + SR more frequently even for medium resolutions (Fig. 5d), resulting in a large end-to-end latency.

VI. Conclusion and Future Work

This work proposes *Rosevin*, the first system designed to jointly optimize bitrate and fine-grained resource allocation for edge super-resolution to enhance video streaming. By enabling SR in an edge cluster, *Rosevin* can mitigate bandwidth insufficiency by generating high-resolution videos using edge computation resources. To achieve the online algorithm for *Rosevin*, we introduce multiple techniques into the combinatorial multi-armed bandits framework. We design an adaptive window-based strategy with primal-dual theories integrated, to address the temporal dependencies in the resource constraint and QoE non-smoothness. In the future, we would explore the impact of caching strategies and multi-cluster correlations to further generalize our proposed algorithm. We would also study the resource allocation for multiple types of resources and augment our prototype accordingly.

REFERENCES

[1] "Netflix," http://www.netflix.com, 2023.
[2] "Youtube," http://www.youtube.com, 2023.
[3] "Tiktok," http://www.tiktok.com, 2023.
[4] "Global internet phenomena report," https://www.sandvine.com/global-internet-phenomena-report-2023, 2023.
[5] P. K. Mu, J. Zheng, T. H. Luan, L. Zhu, Z. Su, and M. Dong, "Amismu: Edge computing based adaptive video streaming for multiple mobile users," *IEEE Transactions on Mobile Computing*, 2022.
[6] A. Zhang, Q. Li, Y. Chen, X. Ma, L. Zou, Y. Jiang, Z. Xu, and G.-M. Muntean, "Video super-resolution and caching—an edge-assisted adaptive video streaming solution," *IEEE Transactions on Broadcasting*, vol. 67, no. 4, pp. 799–812, 2021.
[7] G. Gao and Y. Wen, "Video transcoding for adaptive bitrate streaming over edge-cloud continuum," *Digital Communications and Networks*, vol. 7, no. 4, 2021.
[8] X. Zhang, C. Wu, Z. Li, and F. C. Lau, "Online cost minimization for operating geo-distributed cloud cdns," in *Proc. of IEEE/ACM IWQoS*, 2015.
[9] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.
[10] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.
[11] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–6.
[12] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1874–1883.
[13] 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-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.
[14] R.-X. Zhang, C. Yang, X. Wang, T. Huang, C. Wu, J. Liu, and L. Sun, "Aggcast: Practical cost-effective scheduling for large-scale cloud-edge crowdsourced live streaming," in *ACM MM*, 2022.
[15] Q. Li, Y. Chen, A. Zhang, Y. Jiang, L. Zou, Z. Xu, and G.-M. Muntean, "A super-resolution flexible video coding solution for improving live streaming quality," *IEEE Transactions on Multimedia*, vol. early access, 2022.
[16] J. Kim, Y. Jung, H. Yeo, J. Ye, and D. Han, "Neural-enhanced live streaming: Improving live video ingest via online learning," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 107–125.
[17] H. Yeo, Y. Jung, J. Kim, J. Shin, and D. Han, "Neural adaptive content-aware internet video delivery," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 645–661.
[18] H. Yeo, H. Lim, J. Kim, Y. Jung, J. Ye, and D. Han, "Neuroscaler: Neural video enhancement at scale," in *Proc. of ACM SIGCOMM*, 2022.
[19] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the conference of the ACM special interest group on data communication*, 2017, pp. 197–210.
[20] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: An efficient dynamic resource scheduler for deep learning clusters," in *ACM EuroSys*, 2018.
[21] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: Auto-tuning video abr algorithms to network conditions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 44–58.
[22] W. Chen, "Combinatorial multi-armed bandit: General framework, results and applications," in *In Proc. of ICML*, 2013.
[23] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Computer Vision – ECCV 2016*, 2016.
[24] "Hevc," https://hevc.hhi.fraunhofer.de/, 2023.
[25] "Opencv," https://docs.opencv.org/3.4/d6/d0f/group__dnn.html, 2023.
[26] G. Peng, "Cdn: Content distribution network," *arXiv preprint cs/0411069*, 2004.
[27] A. Asheralieva and D. Niyato, "Combining contract theory and lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Transactions on Networking*, vol. 3, no. 28, 2020.
[28] R.-X. Zhang, M. Ma, T. Huang, H. Li, J. Liu, and L. Sun, "Leveraging qoe heterogeneity for large-scale livecast scheduling," in *ACM MM*, 2020.
[29] Y. Niu, B. Luo, F. Liu, J. Liu, and BoLi, "When hybrid cloud meets flash crowd: Towards cost-effective service provisioning," in *Proc. of IEEE INFOCOM*, 2015.
[30] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.
[31] Y. Zhang, Y. Zhang, Y. Wu, Y. Tao, K. Bian, P. Zhou, L. Song, and H. Tuo, "Improving quality of experience by adaptive video streaming with super-resolution," in *Proc. of IEEE INFOCOM*, 2020.
[32] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth *et al.*, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th annual Symposium on Cloud Computing*, 2013, pp. 1–16.
[33] "Kubernetes," http://kubernetes.io/, 2023.
[34] "Kubeedge," http://kubeedge.io/, 2023.
[35] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. H. Liu, and C. Guo, "Tiresias: A gpu cluster manager for distributed deep learning," in *Proc. of NSDI*, 2019.
[36] C. Zhang, M. Yu, W. Wang, and F. Yan, "{MArk}: Exploiting cloud services for {Cost-Effective},{SLO-Aware} machine learning inference serving," in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 1049–1062.
[37] H. Yeo, C. J. Chong, Y. Jung, J. Ye, and D. Han, "Nemo: enabling neural-enhanced video streaming on commodity mobile devices," in *Proc. of MobiCom*, 2020.
[38] H. Gupta, J. Chen, B. Li, and R. Srikant, "Online learning-based rate selection for wireless interactive panoramic scene delivery," in *IEEE INFOCOM*, 2022.
[39] A. Hodroj, M. Ibrahim, Y. Hadjadj-Aoul, and B. Sericola, "Enhancing dynamic adaptive streaming over http for multi-homed users using a multi-armed bandit algorithm," in *IWCMC*, 2019.
[40] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 133–144.
[41] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a practical perceptual video quality metric," *The Netflix Tech Blog*, vol. 6, 2016.
[42] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.
[43] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[44] D. Chakrabarty, Y. Zhou, and R. Lukose, "Online knapsack problems," in *Workshop on internet and network economics (WINE)*, 2008.
[45] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. Lau, "Online auctions in iaas clouds: Welfare and profit maximization with server costs," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, 2017.
[46] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
[47] B. Charyyev, E. Arslan, and M. H. Gunes, "Latency comparison of cloud datacenters and edge servers," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.
[48] "Uvg 4k video dataset," https://github.com/ultravideo/UVG-4K-Dataset, 2023.
[49] 2023. [Online]. Available: https://openi.pcl.ac.cn/OpenDatasets/PP8K-CRFV/datasets
[50] H. Wang, K. Wu, J. Wang, and G. Tang, "Rldish: Edgeassisted qoe optimization of http live streaming with reinforcement learning," in *Proc. of IEEE INFOCOM*, 2020.