BONES: Near-Optimal Neural-Enhanced Video Streaming

LINGDONG WANG, University of Massachusetts Amherst, USA
SIMRAN SINGH, New Jersey Institute of Technology, USA
JACOB CHAKARESKI, New Jersey Institute of Technology, USA
MOHAMMAD HAJIESMAILI, University of Massachusetts Amherst, USA
RAMESH K. SITARAMAN, University of Massachusetts Amherst, USA

Accessing high-quality video content can be challenging due to insufficient and unstable network bandwidth. Recent advances in neural enhancement have shown promising results in improving the quality of degraded videos through deep learning. Neural-Enhanced Streaming (NES) incorporates this new approach into video streaming, allowing users to download low-quality video segments and then enhance them to obtain high-quality content without violating the playback of the video stream. We introduce BONES, an NES control algorithm that jointly manages the network and computational resources to maximize the quality of experience (QoE) of the user. BONES formulates NES as a Lyapunov optimization problem and solves it in an online manner with near-optimal performance, making it the first NES algorithm to provide a theoretical performance guarantee. Comprehensive experimental results indicate that BONES increases QoE by 5% to 20% over state-of-the-art algorithms with minimal overhead. Our code is available at https://github.com/UMass-LIDS/bones.

CCS Concepts: • **Networks** → **Network resources allocation**; • **Information systems** → **Multimedia streaming**.

Additional Key Words and Phrases: adaptive bitrate streaming, Lyapunov optimization, neural enhancement, super-resolution

ACM Reference Format:
Lingdong Wang, Simran Singh, Jacob Chakareski, Mohammad Hajiesmaili, and Ramesh K. Sitaraman. 2024. BONES: Near-Optimal Neural-Enhanced Video Streaming. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 2, Article 19 (June 2024), 28 pages. https://doi.org/10.1145/3656014

1 INTRODUCTION

Video content dominates the Internet, accounting for more than 65% of its traffic volume [35]. However, accessing high-quality video content is often hindered by insufficient and unstable network bandwidth between the video server and video player (i.e., client). This challenge of high-quality video streaming is even more significant when delivering higher-resolution or immersive videos, such as 4K/8K videos, 360-degree videos, and volumetric videos.

The traditional approach to maximizing the user's quality of experience (QoE) is to use an adaptive bitrate (ABR) algorithm. An ABR algorithm typically runs within the video player and ensures that the video plays back continuously (i.e., without rebuffering) at the highest possible quality

Authors' addresses: Lingdong Wang, lingdongwang@umass.edu, University of Massachusetts Amherst, Amherst, Massachusetts, USA; Simran Singh, s.singh.xzy@gmail.com, New Jersey Institute of Technology, Newark, New Jersey, USA; Jacob Chakareski, jacobcha@njit.edu, New Jersey Institute of Technology, Newark, New Jersey, USA; Mohammad Hajiesmaili, hajiesmaili@cs.umass.edu, University of Massachusetts Amherst, Amherst, Massachusetts, USA; Ramesh K. Sitaraman, ramesh@cs.umass.edu, University of Massachusetts Amherst, Amherst, Massachusetts, USA.

ACM 2476-1249/2024/6-ART19
https://doi.org/10.1145/3656014

(i.e., bitrate). To achieve this goal, the ABR algorithm downloads lower-quality video segments when the available network bandwidth is low to prevent rebuffering and downloads higher-quality segments when the bandwidth is high. There has been extensive research on developing ABR algorithms for decades, with many known algorithms such as BOLA, Dynamic, Festive, MPC, Pensieve, etc [19, 22, 29, 38, 39, 50].

Neural-Enhanced Streaming (NES). Traditional video streaming relies on transmitting videos from the server to the client at the highest possible quality while ensuring continuous playback [6, 7, 43]. However, recent advances in machine learning open up a new possibility of transmitting videos from the server to the client at low quality, and then *neurally enhancing* the video quality via deep-learning techniques at the client. Some examples of such neural enhancement include super-resolution [26, 27], frame interpolation[37, 52], video inpainting [41, 53], video denoising [13, 42], point cloud upsampling [24, 51], and point cloud completion [20]. NES methods incorporate this new approach into video streaming, enabling a tradeoff between communication resources for transmitting high-quality videos and computational resources for neural enhancement. Unlike ABR algorithms that only decide the quality of the video segment to download, an NES control algorithm decides on *both* the download quality and the enhancement option, for each video segment. NES is particularly advantageous in improving the worst-case user experience under poor network conditions. Additionally, it allows video providers or clients to save bandwidth by lowering the transmission bitrate and performing enhancement afterward.

Prior work on NES. While not as extensively studied as ABR algorithms, recent works on NES use reinforcement learning (RL) [12, 36, 48, 54, 56] or heuristic approaches [55, 57]. However, these methods do not have theoretical guarantees for their performance. Further, prior works only consider one enhancement option during inference, usually the one bringing the highest quality gain in real-time. This restricts the possible design space of enhancement options and disregards the broader benefits of utilizing diverse enhancements over a longer time horizon. Finally, existing NES methods are complex to deploy and slower to converge. For example, RL-based methods have high training costs but may still not adapt well to real-world scenarios [46]. Other approaches that rely on model predictive control (MPC) compute a large rigid decision table or heuristically solve an NP-hard problem, leading to an intractable solution [50, 57].

Our NES algorithm. To rectify the above shortcomings, we propose **B**uffer-**O**ccupancy-based **N**eural-**E**nhanced **S**treaming (`BONES`), a client-side NES algorithm for on-demand video streaming. Specifically, `BONES` downloads video segments from a video provider's server to a client device and then enhances these segments opportunistically using local computational resources. To ensure efficient scheduling of the available bandwidth and computational resources, `BONES` operates within a novel parallel-buffer system model and solves a Lyapunov optimization problem online. It has a provable near-optimal performance and exploits all available enhancement methods during inference, resulting in superior performance. Besides, `BONES` has a simple control algorithm with explainable parameters, making it easier to deploy in production systems.

Our Contributions. We make the following contributions in our work:

(1) *Joint optimization of download and enhancement decisions.* `BONES` generalizes the Lyapunov optimization approach of the ABR algorithm BOLA [38, 39] to the NES problem of scheduling both bandwidth and computational resources. Our proposed parallel-buffer system model and online control algorithm allow for optimizing the download and enhancement decisions jointly while conducting the actual respective operations asynchronously, which largely contributes to the performance enhancements enabled by our framework.
(2) *Theoretical guarantees and simple algorithm design.* `BONES` is the first NES algorithm with a provable guarantee on its performance. Specifically, `BONES` achieves QoE that is provably

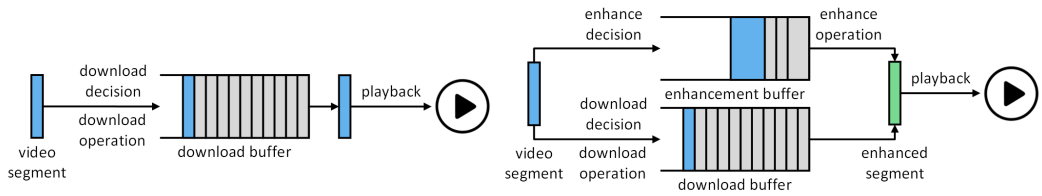


Fig. 1. System models of the traditional ABR algorithm (left) and BONES (right).

within an additive factor of the offline optimal solution. Besides enhanced performance, BONES has the advantages of linear time complexity and simplicity in deployment.

(3) *Simulation environment and prototype system implementation.* We implement an efficient simulation environment for large-scale evaluation of NES algorithms, along with a prototype system for real-world examination. Our code is publicly released to advance research in related fields.

(4) *Superior experimental performance with low overhead.* Using extensive experiments, we compare BONES with existing methods under six enhancement settings and four network trace datasets. Our experimental results demonstrate that BONES increase QoE by 3.56% to 13.20% in our simulation evaluation and 4.66% to 20.43% in our prototype evaluation. In comparison to the default ABR algorithm of the *dash.js* video player [40], BONES can improve its QoE by 7.33%, which is equivalent to increasing the average visual quality by 5.22 in the VMAF score. BONES only incurs minimal costs and offers three trade-off options between performance and overhead. The options range from less QoE improvement with zero overhead to maximum benefit with an additional 310-KB download size and 0.68-second startup latency.

2 PROBLEM FORMULATION

We now propose a system model and pose the optimization solved by BONES. The main notations used in this paper are summarized in Tab. 1.

2.1 System Model

Modern video streaming works by temporally partitioning the video into *segments*, where each segment plays for a fixed amount of time (say, 4 seconds). The video player sequentially downloads each segment from a video server and renders the segment on the viewer's device. To reduce the chance of rebuffering (i.e., freezing), each segment is downloaded and stored in a *download buffer* ahead of when it needs to be rendered. Each segment is encoded in multiple qualities. And an ABR algorithm chooses the quality to download in an online fashion with the goal of reducing rebuffering events and optimizing the QoE. As noted earlier, an NES algorithm also performs neural enhancements for downloaded segments prior to their rendering.

We show the system model of a traditional ABR algorithm within the video player and contrast that with the system model of BONES in Fig. 1. Relative to a traditional ABR system that solely schedules bandwidth resources with one download buffer, our NES system also incorporates an extra buffer and control flow to manage computational resources. In the BONES system model, the *download buffer* stores video segments that have been downloaded, and the *enhancement buffer* stores enhancement tasks waiting for computational resources. We first introduce the download process in the lower branch and then the enhancement process in the upper branch.

Suppose the video segment is indexed by $n \in [N]$, and the time slot is indexed by $k \in [K_N]$, where K_N represents the index of the slot where the N-th (last) segment of the video content is

Table 1. Summary of main notations.

Notation	Description
t_k	the k-th time slot
K_n	index of the time slot where the n-th segment is downloaded
T_k	duration of time slot t_k
d_i	binary indicator to select the i-th download bitrate
e_j	binary indicator to select the j-th enhancement method
$Q^d(t_k)$	download buffer level at time slot t_k
$Q^e(t_k)$	enhancement buffer level at time slot t_k
$u^d(i, t_k)$	download utility of the i-th bitrate in time slot t_k
$u^e(i, j, t_k)$	enhancement utility of the j-th method for the i-th bitrate in time slot t_k
$\tilde{u}^e(i, j, t_k)$	timely enhancement utility of the j-th method for the i-th bitrate in time slot t_k
p	duration of a video segment
$B_i(t_k)$	the i-th download bitrate in time slot t_k
$S_i(t_k)$	size of the video segment with the i-th bitrate in time slot t_k
$\omega(t_k)$	average bandwidth in time slot t_k
$t^e(i, j)$	processing time to enhance a segment with the i-th bitrate using the j-th method
T_{end}	playback finishing time
V	parameter to control the trade-off between Lyapunov drift and penalty
$Q^d_{\max}$	maximum download buffer capacity
$u_{\max}$	maximum utility of a video segment
$T_{\min}$	minimum time slot duration
$T_{\max}$	maximum time slot duration
u_n	total utility of the n-th video segment
ϕ_n	rebuffering time to download the n-th video segment
γ	hyper-parameter to control the trade-off between utility and smoothness
β	hyper-parameter to linearly control V

downloaded. At the beginning of each time slot t_k, BONES makes both download and enhancement decisions for the next video segment.

The download decision determines whether to download the next segment in this time slot and at which bitrate to download it. Formally, we represent the possible choices for this decision variable using a vector $\mathbf{d} = (d_1, \cdots, d_I)$, where the indicator $d_i \in \{0, 1\}$, $\sum_{i=1}^{I} d_i \leq 1$, where I denotes the number of possible download bitrate options. In particular, $\sum_i d_i = 0$ indicates no download, while $d_i = 1$ indicates that the next segment will be downloaded at the i-th available bitrate $B_i(t_k)$ at time slot t_k. A high download bitrate improves the video quality but consumes more network bandwidth, leading to a longer download time.

If the decision is to download, the segment will be retrieved at the desired bitrate and then added to the end of the download buffer, as illustrated in Fig. 1. The next time slot will commence immediately after the completion of the download. If there is no download to be performed in the present time slot, BONES waits for Δ milliseconds and starts the next slot. The video content is played back at a constant rate from the download buffer. However, if the download buffer is empty, the playback will freeze until new content is available, which results in rebuffering and negatively impacts the quality of experience (QoE) of the user.

Let p be the time duration for a video segment to be played back, expressed in milliseconds. Assume that the i-th download option has been selected, i.e., $d_i = 1$. Then, the segment's size in

bits can be expressed as $S_i(t_k) = B_i(t_k)p$. Next, let $\omega(t_k)$ denote the average network bandwidth during time slot t_k in Kbps. We can then compute the duration of time slot t_k, i.e., the download time of the segment as $T_k = S_i(t_k)/\omega(t_k)$ milliseconds. Note that T_k also captures the length of downloaded content consumed by the client due to video playback during time slot t_k. Finally, let $Q^d(t_k)$ denote the download buffer level, i.e., the total length of segments remaining in the buffer. We can formulate the download buffer dynamic as follows:

$$Q^d(t_{k+1}) = \max\left[Q^d(t_k) - T_k, 0\right] + \sum_i d_i p. \tag{1}$$

At the start of time slott_k, we also decide whether to enhance the video segment and which enhancement method to apply. The enhancement decision is defined as a vector $\mathbf{e} = (e_1, \cdots, e_J)$, where the indicator $e_j \in \{0, 1\}$. But slightly different from the download decision, we have $\sum_{j=1}^{J} e_j = 1$, where J denotes the number of possible enhancement methods. We set the first enhancement method (e_1) as "no enhancement" with zero finishing time and zero effect and $e_j = 1$ indicates that the j-th available enhancement method will be applied to the video segment. An enhancement method using a larger deep-learning model typically offers greater quality improvement but takes more time to complete due to higher computational complexity.

A video segment will be pushed into the enhancement buffer at the same time as it is pushed into the download buffer. When doing so, we are actually registering a new computation task to enhance this segment with the chosen enhancement method, and letting this task wait for resources in the enhancement buffer. The length of the video segment in the enhancement buffer is set to be the expected time required to finish its enhancement, denoted by $t^e(i, j)$. This computation time $t^e(i, j)$ is a function of the download option i and the enhancement option j, as both the downloaded video quality and enhancement method can affect the computation speed. The estimated computation time is derived by profiling the enhancement method on the client device. Since the playback duration of a segment is usually different from the processing time of its enhancement task, a video segment often has different lengths in the two buffers, as illustrated in Fig. 1.

It is important to note that, though the enhancement decision is made before a segment enters the enhancement buffer, the actual computation for the enhancement takes place when the video segment leaves the buffer. This enables BONES to optimize download and enhancement decisions jointly but perform the actual operations asynchronously in a non-blocking manner, which leads to superior performance. This approach also allows us to use the Lyapunov optimization framework [32] which can synchronously control multiple queues in a renewal process.

The departure of a segment from the enhancement buffer indicates the completion of its enhancement task. The enhancement buffer has the same departure rate as the download buffer, because in 1 second, we will playback 1-second video content and complete a 1-second computational task. Every video segment must go through both download and enhancement buffers to become an enhanced segment before playback. But the enhanced segment could be the same as the original if "no enhancement" is selected. Let the enhancement buffer level $Q^e(t_k)$ capture the aggregate length of computational tasks waiting in the buffer measured in milliseconds. Now, we can formulate the temporal evolution of the enhancement buffer as:

$$Q^e(t_{k+1}) = \max\left[Q^e(t_k) - T_k, 0\right] + \sum_{ij} d_i e_j t^e(i, j). \tag{2}$$

2.2 Optimization Objective

There are three main goals in video streaming - improve video quality, reduce the rebuffering ratio, and avoid buffer overflow. Similarly to BOLA [38, 39], we incorporate these three goals into a

Lyapunov optimization problem. To improve the video quality, we aim to maximize a time-average utility function defined as

$$\begin{aligned}\bar{u} &\triangleq \lim_{N\to\infty} \frac{\mathbb{E}\left\{\sum_{kij} d_i u^d(i,t_k) + d_i e_j \tilde{u}^e(i,j,t_k)\right\}}{\mathbb{E}\left\{T_{\text{end}}\right\}} \\ &= \frac{\lim_{K_N\to\infty} \frac{1}{K_N}\mathbb{E}\left\{\sum_{kij} d_i u^d(i,t_k) + d_i e_j \tilde{u}^e(i,j,t_k)\right\}}{\lim_{K_N\to\infty} \frac{1}{K_N}\mathbb{E}\left\{\sum_{k=1}^{K_N} T_k\right\}},\end{aligned} \tag{3}$$

where $u^d(i,t_k)$ is the base utility of a video segment, $\tilde{u}^e(i,j,t_k)$ is the extra utility obtained by neural enhancement, and T_{end} is the playback finishing time. While the proposed algorithm works for any size of the video sequences, in our theoretical analysis, we further assume there are infinite video segments, i.e., $N \to \infty$. Because the gap between the total playback time and total download time is the time to drain all the content out of the download buffer, we have $\mathbb{E}\left\{T_{\text{end}}\right\} - \mathbb{E}\left\{\sum_{k=1}^{K_N} T_k\right\} \le Q^d_{\max}$, where $Q^d_{\max}$ is the maximum download buffer capacity. Since $Q^d_{\max}$ is finite, we further have $\lim_{K_N\to\infty} \frac{\mathbb{E}\{T_{\text{end}}\}}{\mathbb{E}\left\{\sum_{k=1}^{K_N} T_k\right\}} = 1$. Based on this equation and the theory of renewal processes [16], we derive the last formula in Eq. (3). The final utility function is the expected sum of the download and enhancement utility in each time slot, averaged by the expected time slot duration.

Different from existing myopic NES methods that only consider the present time slot, we allow for selecting non-real-time enhancements and buffering of computation tasks over a longer time horizon. However, we must make sure that the enhancement of a segment is finished before its playback. Utilizing the property that the two buffers have the same departure rate, we know that an enhancement requiring $t^e(i,j)$ to finish will miss the playback deadline if $Q^e(t_k) + \sum_{ij} d_i e_j t^e(i,j) > Q^d(t_k)$. So we abandon such options by assigning them a negative infinity utility. The final enhancement utility function is expressed in Eq. (4), where $u^e(i,j)$ represents an enhancement method's utility improvement pre-measured by the video provider and transmitted to the client via metadata. Specially, we have $t^e(i,1) = 0, u^e(i,1,t_k) = 0,\ \forall i,k$ for the "no enhancement" option.

$$\tilde{u}^e(i,j,t_k) = \begin{cases} -\infty,\ Q^e(t_k) + \sum_{ij} d_i e_j t^e(i,j) > Q^d(t_k), \\ u^e(i,j),\ \text{otherwise.} \end{cases} \tag{4}$$

In order to reduce the rebuffering ratio, we aim to maximize the time-average playback smoothness function defined in Eq. (5). In particular, by maximizing the ratio of the total video length $\sum_{kij} d_i e_j p$ and the total playback time T_{end}, we are minimizing their difference, i.e., the total rebuffering time.

$$\begin{aligned}\bar{s} &\triangleq \lim_{N\to\infty} \frac{\mathbb{E}\left\{\sum_{kij} d_i e_j p\right\}}{\mathbb{E}\{T_{\text{end}}\}} \\ &= \frac{\lim_{K_N\to\infty} \frac{1}{K_N}\mathbb{E}\left\{\sum_{kij} d_i e_j p\right\}}{\lim_{K_N\to\infty} \frac{1}{K_N}\mathbb{E}\left\{\sum_{k=1}^{K_N} T_k\right\}}.\end{aligned} \tag{5}$$

To prevent buffer overflow, we require both buffers to be rate stable, which is a relaxation of the strict buffer constraint. Rate stability ensures that the expected input rate is not greater than the

expected output rate. We then establish the download buffer constraint as

$$\lim_{K_N\to\infty}\frac{1}{K_N}\mathbb{E}\left\{\sum_{ki} d_i p\right\} \le \lim_{K_N\to\infty}\frac{1}{K_N}\mathbb{E}\left\{\sum_{k} T_k\right\}, \tag{6}$$

and the enhancement buffer constraint as

$$\lim_{K_N\to\infty}\frac{1}{K_N}\mathbb{E}\left\{\sum_{k} d_i e_j t^e(i,j)\right\} \le \lim_{K_N\to\infty}\frac{1}{K_N}\mathbb{E}\left\{\sum_{k} T_k\right\}. \tag{7}$$

Finally, we formulate the neural enhancement problem as

$$\max_{\mathbf{d},\mathbf{e}} \ \bar{u}+\gamma\bar{s}, \quad \text{s.t.,} \quad \text{Constraints (6), (7)}, \tag{8}$$

where the goal is to maximize the streaming session's utility and smoothness under buffer constraints. The hyper-parameter γ controls the trade-off between the smoothness and utility objectives. Note that our optimization target can be reduced to that of BOLA by setting $e_1 = 1$ (always choosing "no enhancement"). This implies that our method can function as a neural enhancement streaming algorithm when enhancement is available or a conventional ABR algorithm, otherwise.

3 BONES: CONTROL ALGORITHM AND THEORETICAL ANALYSIS

3.1 Control Algorithm

We develop an efficient online algorithm called BONES to compute the download and enhancement decisions. Solving problem (8) optimally in an online fashion is not practically feasible since the future values of the network bandwidth are uncertain. However, we show that BONES is within an additive factor of the offline optimal solution to the problem (8). Inspired by the Lyapunov optimization framework for renewal frames [31, 32] and BOLA, BONES greedily minimizes the time-average drift-plus-penalty for each time slot. Specifically, the objective function involves the Lyapunov drift, defined as

$$O_D(t_k) = Q^d(t_k)\sum_i d_i p + Q^e(t_k)\sum_{ij} d_i e_j t^e(i,j), \tag{9}$$

and a penalty function defined as

$$O_P(t_k) = -\left(\sum_{ij} d_i u^d(i,t_k) + d_i e_j \tilde{u}^e(i,j,t_k) + \gamma d_i p\right). \tag{10}$$

We also need to divide our objective by the time slot duration $T_k = S_i(t_k)/\omega(t_k)$, but the random variable $\omega(t_k)$ can be omitted in the optimization.

Altogether, the optimization problem that BONES aims to solve in each time slot is given below:

$$\begin{aligned} \min_{\mathbf{d},\mathbf{e}} \ & \frac{O_D(i,t_k)+VO_P(i,j,t_k)}{\sum_i d_i S_i(t_k)} \\ \text{s.t.,} \ & d_i \in \{0,1\}\ \forall i,\ \sum d_i \le 1, \\ & e_j \in \{0,1\}\ \forall j,\ \sum e_j = 1, \end{aligned} \tag{11}$$

where V is a trade-off factor between the Lyapunov drift and the penalty. Note that BONES relies solely on the download and enhancement buffer levels $Q^d(t_k)$ and $Q^e(t_k)$ in its operation without using any information about the available network bandwidth. Thus, BONES is a "buffer-occupancy-based" method.

Algorithm 1: BONES: A joint control algorithm for download and enhancement decisions

Input: computation time matrix $T^e \in \mathbb{R}_+^{I\times J}$, total utility matrix $U \in \mathbb{R}^{N\times I\times J}$, segment size matrix $S \in \mathbb{R}_+^{N\times I}$, maximum download buffer capacity $Q^d_{\max}$, maximum utility $u_{\max}$, segment duration p, hyper-parameters γ and β

1 Initialize $O \in \mathbb{R}^{I\times J}$, $\quad V = \beta^{((Q^d_{\max}-p)p)/(u_{\max}+\gamma p)}$
2 **for** $n \in [1, N]$ **do**
3 $t \leftarrow$ current timestamp
4 **for** $i \in [1, I]$ **do**
5 **for** $j \in [1, J]$ **do**
6 $O[i, j] \leftarrow (Q^d(t)p + Q^e(t)T^e[i, j] - V(U[n, i, j] + \gamma p))/S[n, i]$
7 **if** $Q^e(t) + T^e[i, j] > Q^d(t)$ **then**
8 $O[i, j] \leftarrow -\infty$
9 **end**
10 **end**
11 **end**
12 $i', j' \leftarrow \arg\min_{i,j} O$
13 download the n-th segment with the i'-th bitrate, wait until the download finishes at t'
14 **if** $Q^e(t') + T^e[i', j'] > Q^d(t')$ **then**
15 $j' \leftarrow 0$
16 **end**
17 push the n-th segment into the enhancement buffer to enhance it with the j'-th method
18 push the n-th segment into the download buffer
19 $\Delta t_{\text{sleep}} \leftarrow \max[Q^d(t') + p - Q^d_{\max}, 0]$
20 wait for time Δt_{sleep}
21 **end**

We present the basic control algorithm of BONES in Algorithm 1. The algorithm receives a matrix of computation time $T^e \in \mathbb{R}_+^{I\times J}$ measured locally, a matrix of total utility $U \in \mathbb{R}^{N\times I\times J}$ from metadata, a matrix of segment size $S \in \mathbb{R}_+^{N\times I}$ from metadata, and some constant numbers and hyper-parameters as indicated in the input argument of Algorithm 1. The entries of the total utility matrix include the aggregate values of the download utility u^d_n and the enhancement utility u^e_n for each segment and enhancement option, i.e., $U[n, i, j] = u^d_n(i) + u^e_n(i, j)$. Note that BONES can also run with average utility and segment size if per-segment data is unavailable.

Concretely, the BONES algorithm operates as follows. The algorithm firstly computes V according to Eq. (13) (Line 2 in Algorithm 1). Then for each video segment, BONES solves Eq. (11) by traversing all possible combinations of download and enhancement options and choosing the one with the minimum objective score (Lines 3 - 13). BONES downloads the video segment at the bitrate given by the solution and waits for the completion of the download. Once the segment is downloaded, BONES will check whether the previously-decided enhancement option is still applicable and pushes a computation task to the enhancement buffer only if it can be finished on time (Lines 15 - 18). BONES also pushes the video segment into the download buffer. Further, BONES will pause downloading until the download buffer can hold one more segment (Lines 20 - 21). Note that the stopping criterion here is implemented differently from the theory. This is because we can know exactly when BONES should stop downloading and how long it should sleep, by assigning V a special upper bound. The impact of different values of V on the performance of BONES will be discussed in Sec. 3.3.

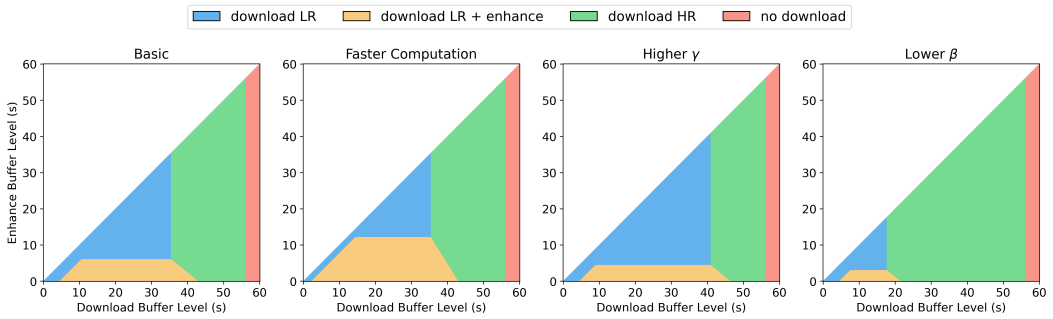


Fig. 2. The decision plane of BONES. Parameter settings from left to right are as follows. "Basic": $\gamma p = 10, \beta = 1$, 1× computation speed. "Faster Computation": 2× computation speed. "Higher γ": $\gamma p = 50$. "Lower β": $\beta = 0.5$.

BONES has a time complexity of $O(IJ)$ in each iteration, where I and J are the numbers of possible choices for bitrate and enhancement options. Besides, the computation of objective scores can be fully parallelized. As a result, BONES runs efficiently in real-time and is easy to deploy. Based on the primary control algorithm here, we propose two additional heuristics in Sec. 5.3 to further improve the practical performance of BONES.

3.2 Decision Plane of BONES

To better understand our algorithm, we depict its 2D decision plane with respect to buffer levels under a hypothetical scenario involving two types of low-resolution (LR) segments and high-resolution (HR) segments. And there is only one type of enhancement method for LR segments. We present four variants of BONES decision planes in Fig. 2 under different parameter settings.

Interpreting the decision plane of BONES. Starting from the lower left corner of a decision plane, the download buffer is empty and the viewer is in urgent need of content. In this case, BONES selects to quickly download LR segments without initiating enhancements. As the download buffer level increases, BONES still downloads LR segments but has time to enhance them and achieve higher visual quality. Once the download buffer level is high enough, BONES can pursue the highest quality by downloading HR segments since enhancement can never be perfect (enabling less quality than HR). When the download buffer is almost full, BONES suspends further downloading to prevent buffer overflow. According to Eq. (4), BONES will stop enhancing segments if the enhancement buffer level is close to the download buffer level, so it will never enter the upper left white triangle. If enhancements are unavailable on the device, the orange region vanishes and the control plane becomes a 1D function of the download buffer level, which implies BONES is reduced to BOLA. To conclude, this simple scenario demonstrates that BONES consistently exhibits reasonable behavior.

The impact of system parameters. Fig. 2 also illustrates the impact of system settings and hyper-parameter settings on the decisions of BONES. Compared with the "Basic" setting, we increase the computation speed by 2× in "Faster Computation" as if BONES runs on more powerful computational hardware. In this case, the area of downloading and enhancing LR segments increases, meaning that neural enhancement becomes more desirable with more computational resources available.

Recall that BONES has 2 hyper-parameters — γ and β. Parameter γ trades off between visual quality and playback smoothness (rebuffering ratio) in Eq. (8). In practice, we tune γ together with the constant segment duration p. Parameter β linearly controls V, the trade-off factor between Lyapunov drift and penalty in BONES optimization objective Eq. (11). In the "Higher γ" setting, we increase γp from 10 to 50, making BONES prefers a lower rebuffering ratio than higher visual quality.

As a result, BONES increases the area of downloading LR segments and requests download bitrate in a more conservative manner. In the "Lower β" setting, we decrease β from 1 to 0.5, suppressing the penalty term in the optimization objective Eq. (11). Due to the diminishing return effect of visual quality, each bit in an LR video segment provides more visual quality scores than that of an HR segment. Therefore, the penalty term in Eq. (11) prefers downloading LR segments and performing enhancements (see the structure of utility divided by segment size). On the opposite, the Lyapunov drift term prefers HR segments. This explains why BONES will download more HR segments as a consequence of lower β and V in this setting. We also provide empirical evidence of how BONES is affected by its hyper-parameters in Sec. 5.4.

3.3 Performance Bound

In this subsection, we rigorously analyze the performance of BONES and establish near-optimal guarantees. The analysis shares the same high-level logical flow with BOLA, but since BONES introduces the addition of an enhancement buffer, there are important differences in the details of the analysis. Theorem 1 bounds the size of the download buffer and shows that BONES does not violate the buffer capacity. Secondly, Theorem 2 bounds the performance of BONES with respect to that of the offline optimal solution.

Theorem 1. *Assume* $Q^d(0) = 0, Q^e(0) = 0$, *and* $0 < V \leq \frac{(Q^d_{\max}-p)p}{u_{\max}+\gamma p}$, *where* $u_{\max}$ *denotes the maximum utility. Then, the following holds:* $Q^d(t_k) \leq V\frac{u_{\max}+\gamma p}{p} + p$, *and* $Q^d(t_k) \leq Q^d_{\max}$.

We present a proof in Appendix A. Theorem 1 shows that the download buffer level has a finite bound of $O(V)$ and will never exceed the maximum capacity. The intuition here is to design a special upper bound for V, which ensures BONES to select "no download" if the download buffer level is higher than $Q^d_{\max} - p$. In this way, BONES will download only when the download buffer can accept one more segment. Since the enhancement buffer is a virtual queue without any storage space, its occupancy is not a concern.

In order to prove the performance bound of BONES, we first show that there exists an optimal stationary i.i.d. algorithm independent of the buffer occupancy for problem Eq. (8), achieving the objective $\bar{u}^* + \gamma\bar{s}^*$. Based on that, we can derive the following theorem.

Theorem 2. *Assume* $t^e \leq t^e_{\max}$, $T_{\min} \leq T_k \leq T_{\max}$, $\forall k$, *and* $t^e_{\max}, T_{\min}, T_{\max}$ *are finite. Then, we have:*

$$\bar{u}' + \gamma\bar{s}' \geq \bar{u}^* + \gamma\bar{s}^* - \frac{p^2 + (t^e_{\max})^2 + 2T_{\min}T_{\max}}{2VT_{\min}}, \tag{12}$$

where $\bar{u}' + \gamma\bar{s}'$ *is the objective score of* BONES.

Proofs for the existence of the offline optimal and Theorem 2 are given in Appendix B. While BONES can achieve an objective $\bar{u}' + \gamma\bar{s}'$ for Eq. (8), the optimal algorithm can achieve $\bar{u}^* + \gamma\bar{s}^*$. Theorem 2 shows that the performance gap of BONES toward this optimal algorithm is finitely bounded by $O(1/V)$. In other words, BONES can achieve near-optimal performance.

Results in the above theorems show that the performance of BONES depends on the value of parameter V. In the implementation of BONES, we use a hyper-parameter $\beta \in [0, 1]$ to linearly control V as follows.

$$V = \beta\frac{(Q^d_{\max} - p)p}{u_{\max} + \gamma p}. \tag{13}$$

Putting together the results in Theorems 1 and 2, one can observe that BONES has a $O(V, 1/V)$ trade-off between the download buffer level and the performance gap toward the optimal algorithm. It means one can increase V to improve the performance of BONES. Empirical results in Sec. 5.4

verify this theoretical observation, showing that the QoE of BONES increases with V. However, increasing V will increase the buffer level, thus increasing the time delay between downloading a segment and playing it back. Live streaming can be negatively affected by such behavior, but it may not be as much of an issue in on-demand video streaming. Besides, there is an upper cap for V to prevent buffer overflow.

4 EXPERIMENTAL EVALUATION

4.1 Performance Metric

To evaluate the overall performance of video streaming algorithms, we report the experimental results using a commonly-used notion of QoE that includes visual quality, quality oscillation, and rebuffering ratio. More formally, we have

$$\text{QoE} = \frac{1}{N}\sum_{n=1}^{N} u_n - \alpha_1 \frac{1}{N-1}\sum_{n=1}^{N-1} |u_{n+1} - u_n| - \alpha_2 \frac{1}{N}\sum_{n=1}^{N} \phi_n, \tag{14}$$

where u_n is the total utility of a segment summing up the download utility u_n^d and the enhancement utility u_n^e. And ϕ_n is the rebuffering time for each segment. The first term in Eq. (14) represents the average visual quality, the second term represents the average quality oscillation, and the third term is the average rebuffering time per segment. This notion of QoE is widely used in prior work, e.g., MPC-based methods [30, 46, 50, 57] and RL-based methods [29, 48].

Although ABR-only algorithms typically assume utility (visual quality) as a function of bitrate, it is not a suitable metric in NES systems. This is because bitrate only applies to compressed videos, not raw pixels after computational processing. Therefore, we choose the VMAF score [1] as the visual quality metric in our experiments, which is closer to human vision than other objective metrics like PSNR or SSIM. VMAF score ranges from 0 to 100, the higher, the better. We assign the highest-resolution video segments with a maximum score of 100 and invalid enhancement options with negative infinity scores. We measure the rebuffering time in milliseconds. And we set the trade-off factors $\alpha_1 = 1, \alpha_2 = 0.1$ as in method [57], meaning that 1 QoE score is equivalent to the average visual quality of 1 VMAF score, average quality oscillation of 1 VMAF score, or per-segment rebuffering time of 10 milliseconds (0.25% rebuffering ratio in our case).

4.2 Implementation Details

We develop a unified simulation environment for both ABR and NES algorithms to efficiently examine their performance. Our simulator extends Sabre [38], which was used to evaluate BOLA. Our simulation environment and all its algorithms are implemented using Python. And all deep-learning methods are implemented by PyTorch. We use a 636-second 30-fps video "Big Buck Bunny" for streaming. The video is chunked into 4-second segments and encoded in 5 resolutions of 240p/360p/480p/720p/1080p with bitrate of 400/800/1200/2400/4800 Kbps.

Table 2. Details of network trace datasets.

Dataset	3G	4G	FCC-SD	FCC-HD
Mean Bandwidth (Kbps)	1184	31431	6081	17127
Standard Variance of Bandwidth (Kbps)	818	14058	11615	4018
Number of Traces	86	40	1000	1000

We evaluate algorithms using four network trace datasets from Sabre, including 3G traces [34], 4G traces [44], and two subsets of FCC traces [14]. These datasets are widely used, and different

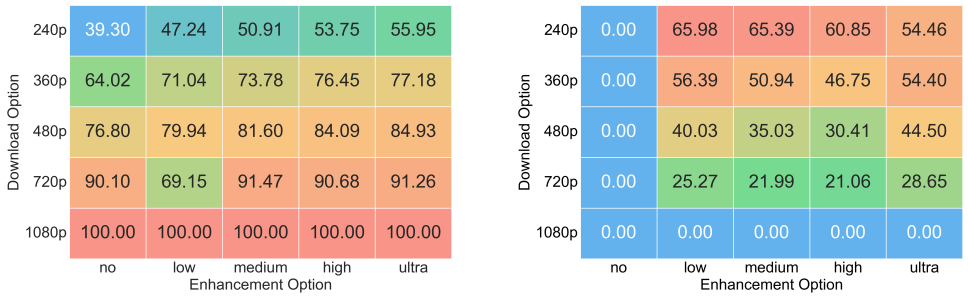
studies have adopted different subsets [48, 50, 57]. We exclude those 3G traces with an average bandwidth lower than 400 Kbps, the lowest option on our bitrate ladder. We present numerical details of our testing datasets in Tab. 2.

4.3 Enhancement Settings

Table 3. Enhancement model details. From left to right: #layers, #channels, upscale factor, model size (KB).

Model	Quality	240p	360p	480p	720p
NAS-MDSR	low	20,9,4,43	20,8,3,36	20,4,2,12	6,2,1,2
	medium	20,21,4,203	20,18,3,157	20,9,2,37	6,7,1,5
	high	20,32,4,461	20,9,3,395	20,18,2,128	6,16,1,17
	ultra	20,48,4,1026	20,42,3,819	20,26,2,259	6,26,1,41
IMDN	low	3,6,4,34	3,6,3,29	3,6,2,26	-
	medium	5,12,4,111	5,12,3,103	5,12,2,96	-
	high	6,32,4,760	6,32,3,736	6,32,2,719	-
	ultra	6,64,4,2824	6,64,3,2777	6,64,2,2743	-

We utilize five enhancement settings (six, if "no enhancement" is counted) to demonstrate that BONES can manage any enhancement method with any amount of computational power. We first adopt the pervasively-used enhancement method NAS-MDSR [15, 47, 48], a content-aware Super-Resolution (SR) model that can overfit one individual video and upscale any resolution to 1080p. We further assume that the client-side computational hardware is an Nvidia GTX 1080ti GPU card. Under this setting, there are five enhancement quality levels (no enhancement, low, medium, high, and ultra) for each of the four low-resolution download options (240p, 360p, 480p, and 720p). More details about the deep-learning model can be found in Tab. 3. Besides, the visual quality and computation speed of each enhancement option are illustrated in Fig. 3, where zero computation speed implies the enhancement is not applicable. Generally, as the download bitrate and the enhancement quality level increase, the visual quality will increase and the computation speed will decrease. However, our algorithm does not rely on this being true.



(a) Visual quality (VMAF). (b) Computation speed (fps).

Fig. 3. Enhancement performance under NAS-MDSR setting.

Beyond NAS-MDSR, we adopt IMDN [21] as another SR enhancement method. We offer 4 quality levels for 240p, 360p, and 480p videos to upsample them to 1080p. The low and medium quality levels are based on the IMDN-RTC structure, while the high and ultra levels are based on the

vanilla IMDN. Model details of IMDN can also be found in Tab. 3. As for the training technique, we explore both content-agnostic IMDN and content-aware IMDN. A content-agnostic IMDN is trained over a generic video dataset DIV2K [2]. Such a generic model can enhance any video but bring less quality improvement. In comparison, content-aware IMDN is trained over our target video to overfit its content. This strategy offers better enhancement performance but introduces overhead in training cost and startup latency. A more detailed discussion exists in Sec. 5.5. The computational hardware for IMDN is either GTX 2080ti or GTX 3060ti GPU card. Altogether, we create 4 enhancement settings for IMDN by traversing all combinations of training techniques and computational hardware.

4.4 Comparison Algorithms

We compare BONES with the following ABR-only algorithms:

(1) BOLA [39] makes download decisions by solving a Lyapunov optimization problem only related to the buffer level.
(2) Dynamic [38] switches between BOLA and a throughput-based algorithm based on carefully-designed heuristic rules.
(3) FastMPC [50] formulates bitrate adaptation as an MPC optimization problem and makes download decisions according to a pre-computed solution table.
(4) Buffer-based method [19] chooses bitrate using a piecewise linear function of buffer level.
(5) Throughput-based method [22] chooses the maximum bitrate under the estimated network bandwidth.
(6) Pensieve [29] uses deep RL to make download decisions. We train the RL agent on 10,000 randomly selected FCC traces [14] using the Asynchronous Advantage Actor-Critic (A3C) algorithm with identical training settings as in the original paper.

We further augment ABR-only algorithms with a greedy enhancement strategy, denoted by the symbol $*$ on the right of the algorithm names. Specifically, we let the ABR algorithms make download decisions while simultaneously choosing the enhancement option that brings the most utility improvement in real-time. To avoid interrupting playback, enhancements will only be applied if it can be finished on time. This greedy enhancement strategy is simple and will not affect the download performance of ABR algorithms. However, it only leads to sub-optimal performance because the download decision maker and the enhancement decision maker share no knowledge with each other. Beyond ABR algorithms, we also compare BONES with the following NES algorithms:

(1) NAS [48] controls both download and enhancement processes using an RL agent. The NAS RL agent is designed and trained similarly to Pensieve's, with the additional consideration of a greedy real-time enhancement strategy. We did not reproduce the scalable model download approach in NAS, which can be viewed as providing more enhancement options.
(2) PreSR [57] solves an MPC problem online with the heuristic to pre-fetch and enhance only "complex" segments of the video.

5 SIMULATION RESULTS

5.1 Overall Performance

We present the simulation results of BONES and other benchmark methods averaged across all network trace datasets in Tab. 4 and Fig. 4. In Tab. 4, we report the QoE score defined in Eq. (14) as well as its three components (visual quality, quality oscillation, rebuffering ratio). The results are grouped under six settings, one setting without enhancement, and five with different enhancement settings, one with NAS-MDSR, and four with IMDN as discussed in Sec. 4.3. We provided visual representations of the three QoE components for each method under two enhancement settings

Table 4. Overall performance comparison under different enhancement settings. Evaluated metrics include visual quality (VMAF), quality oscillation (VMAF), rebuffering rate (%), and composite QoE. * denotes the ABR method is augmented with greedy enhancement. BONES achieves the highest QoE under all applicable settings. The optimal results are marked in **bold**.

Method	Qual.	Osc.	Rebuf.	QoE	Method	Qual.	Osc.	Rebuf.	QoE
No Enhancement					NAS-MDSR, Content-Aware, GTX 1080ti				
BOLA	82.90	4.05	2.21	69.98	BOLA *	85.45	3.66	2.21	72.93
Dynamic	84.62	3.73	2.43	**71.14**	Dynamic*	86.62	3.43	2.43	73.44
FastMPC	87.90	**3.33**	4.22	67.65	FastMPC*	89.70	**3.22**	4.22	69.56
Buffer	85.27	4.10	2.93	69.44	Buffer*	87.47	3.66	2.93	72.07
Throughput	80.10	3.59	**1.93**	68.77	Throughput*	83.43	3.33	**1.93**	72.36
Pensieve	**89.23**	4.01	4.17	68.50	Pensieve*	**91.16**	3.47	4.17	70.96
NAS	-	-	-	-	NAS	89.20	4.40	2.92	73.08
PreSR	-	-	-	-	PreSR	89.24	4.17	4.65	66.45
BONES	-	-	-	-	BONES	89.23	3.07	2.52	**76.05**
IMDN, Content-Agnostic, GTX 2080ti					IMDN, Content-Aware, GTX 2080ti				
BOLA *	83.99	3.92	2.21	71.20	BOLA *	84.84	3.76	2.21	72.22
Dynamic*	85.62	3.63	2.43	72.25	Dynamic*	86.11	3.49	2.43	72.87
FastMPC*	88.46	**3.27**	4.22	68.26	FastMPC*	88.88	**3.33**	4.22	68.63
Buffer*	86.09	3.97	2.93	70.39	Buffer*	86.83	3.78	2.93	71.32
Throughput	81.76	3.45	**1.93**	70.57	Throughput*	82.59	3.35	**1.93**	71.50
Pensieve*	**89.90**	3.79	4.17	69.39	Pensieve*	**90.19**	3.71	4.17	69.76
NAS	88.18	4.66	2.92	71.81	NAS	88.35	4.57	2.92	72.06
PreSR	88.65	4.39	4.57	65.95	PreSR	88.79	4.20	4.51	66.53
BONES	88.40	3.52	2.80	**73.64**	BONES	88.97	3.47	2.55	**75.29**
IMDN, Content-Agnostic, GTX 3060ti					IMDN, Content-Aware, GTX 3060ti				
BOLA *	84.85	3.77	2.21	72.21	BOLA *	84.75	3.83	2.21	72.05
Dynamic*	86.39	3.49	2.43	73.14	Dynamic*	86.49	3.42	2.43	73.32
FastMPC*	89.12	**3.15**	4.22	69.06	FastMPC*	89.17	**3.21**	4.22	69.03
Buffer*	86.91	3.80	2.93	71.38	Buffer*	86.77	3.85	2.93	71.19
Throughput*	82.58	3.36	**1.93**	71.48	Throughput*	83.46	3.34	**1.93**	72.38
Pensieve*	**90.50**	3.60	4.17	70.18	Pensieve*	**91.05**	3.50	4.17	70.83
NAS	88.98	4.39	2.92	72.88	NAS	89.17	4.38	2.92	73.08
PreSR	89.18	4.11	4.63	66.53	PreSR	88.85	4.20	4.52	66.55
BONES	88.49	3.37	2.64	**74.52**	BONES	89.38	3.12	2.47	**76.36**

earlier in Fig. 4. Among ABR-only algorithms, we find that Dynamic, the default ABR algorithm of *dash.js* video player [40], reaches the best balance between the three metrics and achieves the highest QoE. Yet BONES can still improve its QoE by up to 7.33%, which is equivalent to increasing the average visual quality by 5.22 VMAF score or decreasing the rebuffering ratio by 1.30%.

By augmenting ABR-only algorithms with the greedy enhancement strategy, their QoE benefits from the additional usage of computational resources. The greedy enhancement strategy will not affect the download behavior of an ABR algorithm, thus keeping its rebuffering ratio unchanged. But by upgrading the low-quality downloaded segments, neural enhancement increases the visual quality and reduces quality oscillation, leading to an average of 2.81% QoE improvement across all kinds of augmentations. However, this enhancement strategy is decoupled from the download

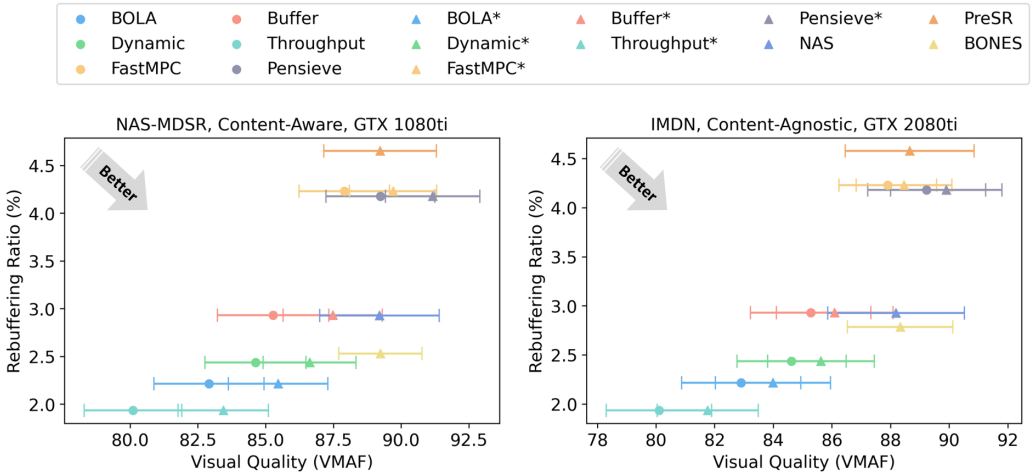


Fig. 4. Performance comparison under two enhancement settings. The error bar represents the average visual quality oscillation. Higher quality, lower oscillation, and lower rebuffering ratio are better. * denotes the ABR method is augmented with a greedy enhancement strategy.

decision and thus leads to sub-optimal performance. In contrast, BONES makes joint decisions and outperforms all the augmented ABR algorithms.

As for NES methods, we find that both NAS and PreSR perform well under high-bandwidth scenarios but poorly if the network condition is weak. In Sec. 5.2, we further scrutinize the robustness of different algorithms under different network trace datasets. Our experimental results suggest that it's hard for NAS to adapt to data distribution far from the training set. Similarly, [46] reports that RL-based methods have limited generalization ability and may not adapt to heavy-tailed real-world network traces. In comparison, BONES is a control-theoretic algorithm that does not require any learning process, making it easy and robust to deploy under versatile scenarios.

We note that the unsatisfactory performance issue of PreSR comes from its reliance on sub-optimal heuristics. PreSR formulates an NP-hard MPC optimization problem and solves it heuristically online. As a result, PreSR can perform even worse than its backbone method FastMPC, as it only explores limited solutions in the decision space. In contrast, our method BONES is guaranteed to outperform its backbone BOLA, which is theoretically shown in Sec. 3.1 and empirically verified here.

By studying the four variants of IMDN enhancement settings, we find that the content-aware model performs better than the content-agnostic one as it overfits the content of a specific video. Nevertheless, content-aware enhancement requires costly training toward an individual video and increases the startup latency due to model downloading before video streaming. We postpone the detailed overhead analysis to Sec. 5.5. Besides, we find that the performance of neural enhancement increases with better computational hardware (GTX 3060ti than GTX 2080ti), which is intuitive since more computational resources enable more powerful enhancements.

In conclusion, our method BONES consistently outperforms existing ABR and NES methods by 3.56% to 13.20% in QoE averaged across all network conditions and all enhancement settings.

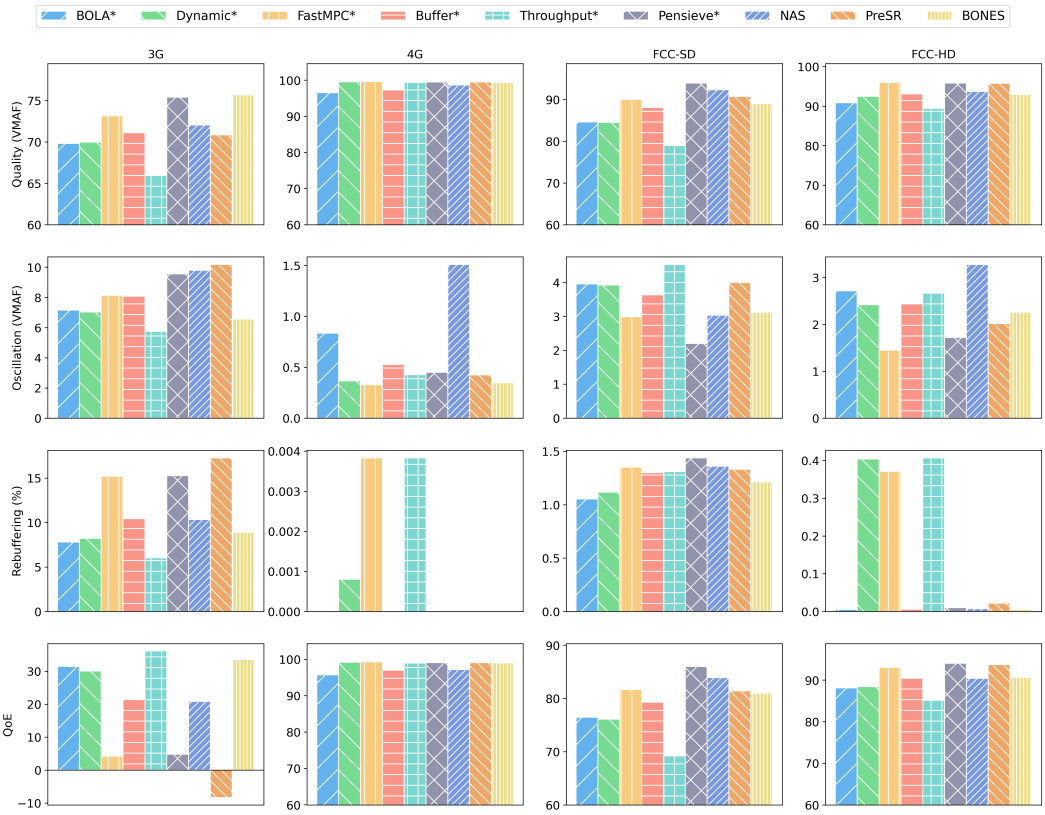


Fig. 5. Performance comparison on different network trace datasets. BONES consistently deliver high QoE across diverse network conditions, including challenging ones.

5.2 Sensitivity to Network Condition

To investigate the sensitivity of different algorithms to the network conditions, we further break down the results under the NAS-MDSR enhancement setting in Tab. 4 across four different network traces and report them in Sec. 5.2. Based on the results, we find that the performance of video streaming algorithms varies drastically with network conditions. For example, the throughput-based algorithm outperforms others on the low-bandwidth 3G dataset due to its conservative request of bitrate. However, this behavior does not generalize well to high-bandwidth conditions. Especially the throughput-based method performs worst on the FCC-SD dataset because the high variance of bandwidth leads to a misprediction of throughput. On the other hand, FastMPC, Pensieve, and PreSR perform well under good network conditions via aggressive bitrate requests. But they incur severe rebuffering events on 3G datasets. Unlike other methods, BONES behaves robustly across all network conditions, maintaining a stable QoE even under challenging low-bandwidth scenarios.

5.3 Practical Improvement

In this subsection, we introduce two additional heuristics to further advance the performance of BONES and report their empirical improvement.

Table 5. Ablation study for practical improvements.

Method	Qual. (VMAF)	Osc. (VMAF)	Rebuf. (%)	QoE
Basic	88.64	3.17	2.62	74.95
Monitor	88.76	**3.03**	**2.52**	75.62
AutoTune	89.22	3.16	2.58	75.71
Monitor+AutoTune	**89.23**	3.07	**2.52**	**76.05**

Download Process Monitoring. Since the network bandwidth can vary abruptly in practice, a decision may become out-of-date when the corresponding video segment is under download. For example, downloading a high-quality segment may incur rebuffering if the bandwidth drops in midway. In such a case, it is wise to abort the current download and select a lower-quality segment. We implement this in a similar way to BOLA, allowing our algorithm to monitor the bandwidth variations during download processes and regret its previous decision if necessary. During the download process of a segment, BONES periodically recomputes the current decision's objective score via a modified version of Eq. (11), where $S_i(t_k)$ is replaced with the remaining size to download. BONES also computes the objective scores of other options. If a better option is found, BONES will abandon the ongoing download and switch to the new solution.

Automatic Hyper-Parameter Tuning. We observe that the hyper-parameter setting of BONES is sensitive to the network condition. Under poor network conditions, it is advisable to adopt a conservative strategy with lower quality and a lower rebuffering ratio, while under good conditions, an aggressive strategy is preferred. Inspired by Oboe [3], we develop an automatic hyper-parameter tuning mechanism for BONES in accordance with different network conditions. In the offline phase, we generate synthetic network traces with different average bandwidths (from 200 to 5000 Kbps), different bandwidth variances (500 to 5000 Kbps), and different latency (20 to 100 ms). Then we search for the optimal hyper-parameter β (from 0.1 to 1) and γp (from 10 to 100) under each network condition that maximizes the QoE. During the online deployment phase, BONES estimates the network condition using the exponentially weighted moving average before each decision interval. BONES then adjusts its hyper-parameters to the offline optimal accordingly.

Empirical Results. We conduct an ablation study for practical improvement techniques and present the results under the NAS-MDSR enhancement setting in Tab. 5. Note that those versions without automatic hyper-parameter tuning are assigned with the optimal hyper-parameters found by grid search ($\beta = 1$, $\gamma p = 10$). As experimental results imply, Download Process Monitoring can reduce the rebuffering ratio by regretting previous decisions during bandwidth valley. While Automatic Hyper-Parameter Tuning can contribute to the performance of BONES mostly in visual quality. When both techniques are applied, BONES could achieve the maximum QoE improvement of 1.48%. In summary, this ablation study verifies the effectiveness of our practical improvement techniques.

5.4 Sensitivity to Hyper-Parameters

In this section, we comprehensively investigate the effect of hyper-parameters. It is worth noting that, in practice, one may leverage the Automatic Hyper-Parameter Tuning scheme in Sec. 5.3 instead of manually adjusting parameters. From the experimental results under NAS-MDSR enhancement settings in Fig. 6, we find that increasing γ generally leads to lower visual quality, higher quality oscillation, and a lower rebuffering ratio. As explained in Sec. 3.2, a higher γ makes BONES download more low-bitrate segments, trading quality for playback smoothness. Besides, due to the diminishing return of visual quality, low-bitrate segments span wider in quality scores. Thus downloading more low-quality segments brings more oscillations. We also find that a higher β leads to lower

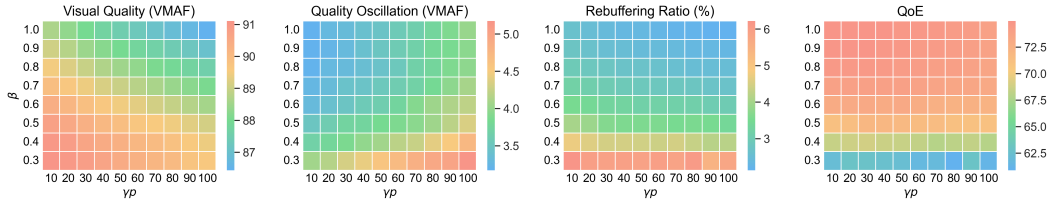


Fig. 6. Performance of BONES with respect to hyper-parameter settings.

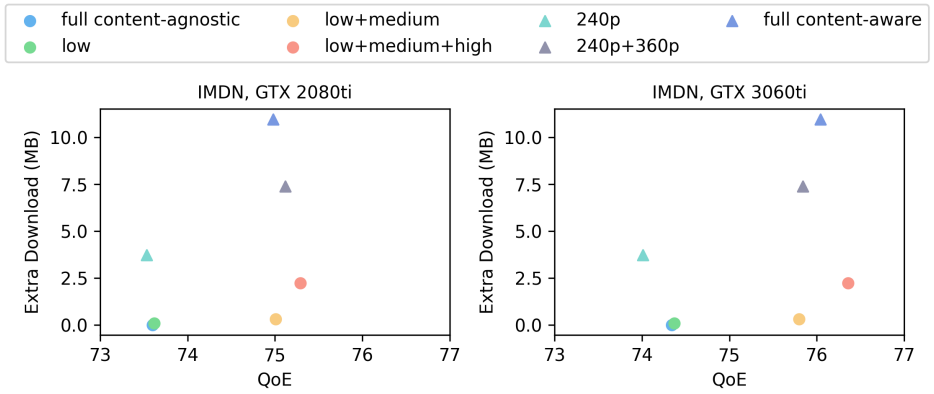


Fig. 7. Performance of BONES with respect to the additional download size.

visual quality, lower quality oscillation, and a lower rebuffering ratio. From our theoretical analysis in Sec. 3.2 we know that, by increasing the weight on the penalty term of optimization objective Eq. (11), BONES downloads more low-bitrate segments and performs more enhancements, resulting in lower quality and fewer rebuffering events. Additionally, since neural enhancement can improve visual quality and reduce the visual quality disparities between segments, increasing β also alleviates quality oscillations. Lastly, empirical results verify that higher β (so does V) positively contributes to the QoE. This conclusion roughly aligns with the observation in Theorem 2, despite the QoE here being defined in a slightly different way than the optimization objective.

5.5 Overhead Analysis

In this section, we comprehensively analyze the overhead of BONES, including offline costs of training enhancement models, online costs of downloading them during playback, and other costs.

Offline Costs. A content-agnostic model is trained over a generic video dataset and enhances all video content. So it only incurs a one-time training cost without the need for fine-tuning. In contrast, a content-aware model needs to be fine-tuned for a particular video using a content-agnostic backbone. NAS [48] reports that a typical fine-tuning will take 10 minutes for each video. The enhancement benefit will cover this additional cost after streaming the video for 30 hours. As a rule of thumb, one may apply content-aware enhancements only for popular videos.

Online Costs. Online costs of BONES include the additional bandwidth consumption and startup latency in downloading extra models. The content-agnostic enhancement has all models pre-installed on the client device. So this cost is only applicable to content-aware enhancement as it requires specific models per video. Unlike other methods, BONES allow choosing different enhancement

Table 6. Trade-off between performance, offline costs, and online costs. Offline costs include whether training or fine-tuning is required. Online costs include extra download size (KB) and additional startup latency (s).

Method	Qual.	Osc.	Rebuf	QoE	Train	FineTune	Download	Startup
Agnostic	88.49	3.37	2.64	74.52	Yes	**No**	**0**	**0**
Aware	**89.02**	**3.09**	**2.53**	**75.80**	Yes	Yes	310	0.68
Aware+QuickStart	88.80	3.14	2.56	75.40	Yes	Yes	310	**0**

methods from a pool of enhancement methods on the fly. This behavior leads to more QoE improvement but requires pre-downloading more models, which may further increase the online cost. Therefore, we study the trade-off between performance and extra download size in Fig. 7.

We plot QoE of content-agnostic IMDN and the additional download overhead (0 MB) in Fig. 7. We do the same for a full content-aware IMDN, which requires downloading 4 quality levels × 3 bitrate levels = 12 models. We then start to remove models from the enhancement method pool. For example, *low* indicates only keeping low-quality models, and *240p* indicates only keeping enhancement models for 240p. From the results, we find that low-quality content-aware models can provide slightly more QoE than content-agnostic models with a download overhead of 89 KB. And we can gain almost all the benefits by downloading the first two quality levels with only 310 KB. Surprisingly, we can outperform the full model by downloading the first three quality levels using 2215 KB. This implies the ultra-level IMDN model is dragging the performance of BONES by occupying enormous resources but providing marginal improvements.

Assume the downloading processes for models and segments occur sequentially, then downloading models before playback will cause extra startup latency. To mitigate this, we further introduce a *QuickStart* heuristic that postpones the model downloading and skips enhancement tasks until the buffer level reaches a certain threshold. To avoid rebuffering, we empirically set the threshold to be two segments (8 seconds in our case). We quantitatively analyze the trade-off between performance and overhead in Tab. 6 using IMDN with its low and medium options on a 3060ti GPU. The content-agnostic enhancement provides the lowest QoE improvement with zero overhead. The content-aware enhancement has the best performance but incurs fine-tuning costs, 310-KB extra download, and 0.68-second additional startup latency. QuickStart sacrifices the possibility of enhancing the first few segments. As a result, it offers an intermediate performance but eliminates the startup latency.

Other Costs. Content-aware enhancement introduces negligible costs of delivering enhancement model addresses in the metadata. Enhancement tasks can slow down the main playback process if they consume excessive computational resources. However, BONES will not favor enhancement under limited computational resources due to the long computation time. Thus, the computational overhead is also not a concern. In summary, BONES significantly improves QoE with minimal overhead, while providing various trade-offs between performance and overhead.

6 PROTOTYPE SYSTEM

6.1 System Overview

To further validate the BONES algorithm in practice, we develop a prototype video player in addition to our simulator. Our prototype implementation is based on the iStream framework [5], which enables the flexible composition of video player modules in a publisher-listener fashion. The prototype system architecture is presented in Fig. 8. We highlight the modules significantly different from the traditional ABR pipeline, which we implemented from scratch. Besides, we implemented a graphical renderer for visual comparison, other than the headless player provided by iStream. We

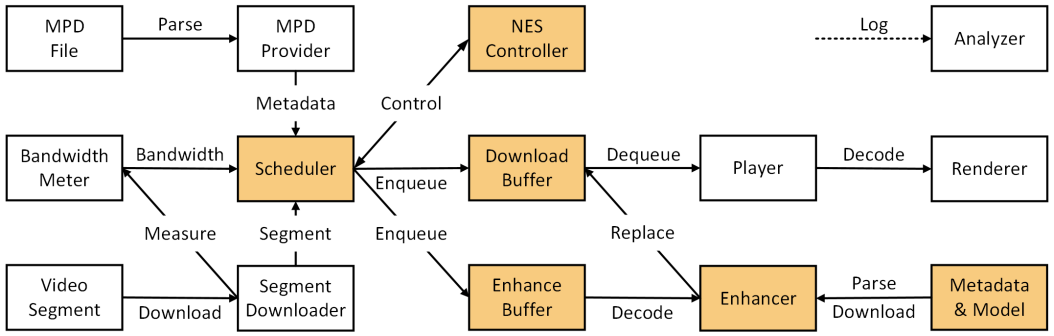


Fig. 8. Prototype system architecture. Modules largely different from the ABR pipeline are highlighted.

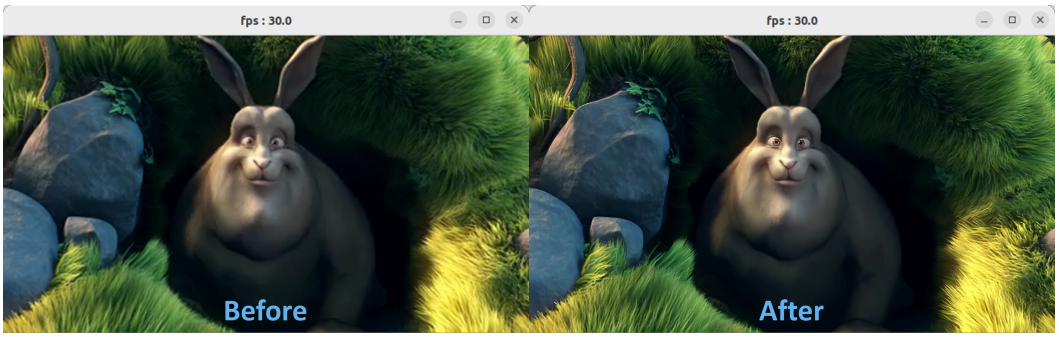


Fig. 9. Sample rendering outcome of the prototype system before and after the enhancement enabled by BONES. Compared with the original video frame, the enhanced one features clearer textures and sharper edges, especially for the eyes, grass, and rocks. It is recommended to zoom in for a closer look at the details. The frame rate is displayed in the title portion of the video frame.

implemented the entire system in Python, the deep learning model in PyTorch, the video decoder in VPF [33], and the graphical renderer in OpenGL. Our modification involves approximately 3000 lines of code. In Fig. 9, we illustrate sample frames rendered by the prototype system.

As shown in Fig. 8, the system's core component is the *Scheduler*. It first calls the *MPD Provider* to download and parse the metadata. It then interacts with the *NES Controller* to make download and enhancement decisions. Next, it notifies the *Segment Downloader* to download the corresponding segment, and the *Bandwidth Meter* to measure the bandwidth during download. After downloading the video segment, it enqueues the segment into the *Download Buffer* and the *Enhance Buffer*. Meanwhile, the enhancement models will be downloaded if needed. The *Enhancer* fetches the downloaded video segment from the *Enhance Buffer*, decodes, and enhances it. The enhanced segment replaces its counterpart in the *Download Buffer* after enhancement. The *Player* fetches the segment from the *Download Buffer*, decodes it if necessary, and gives it to the *Renderer* for display. The *Analyzer* collects logs from all components all the time.

6.2 Settings and Experiments

In our prototype implementation, the enhancement quality ladder is pre-measured and stored locally. The computation speed table is measured the first time this system is deployed. In adaptation to the

Table 7. Performance comparison in the prototype system. BONES achieves the highest QoE.

Method	Qual.	Osc.	Rebuf.	QoE	Method	Qual.	Osc.	Rebuf.	QoE
Content-Agnostic					Content-Aware with Quick Start				
BOLA *	70.44	**3.37**	**0.19**	66.29	BOLA *	70.31	**3.47**	**0.17**	66.14
Dynamic*	73.23	4.22	1.60	62.57	Dynamic*	73.45	4.16	1.58	62.93
Buffer*	71.24	5.24	0.27	64.89	Buffer*	71.35	5.24	0.17	65.41
Throughput*	69.92	3.51	1.60	59.98	Throughput*	70.15	3.68	1.58	60.10
NAS	71.65	6.92	1.45	58.92	NAS	70.95	7.72	0.24	62.26
BONES	**75.42**	3.92	0.52	**69.38**	BONES	**76.56**	3.48	**0.17**	**72.38**

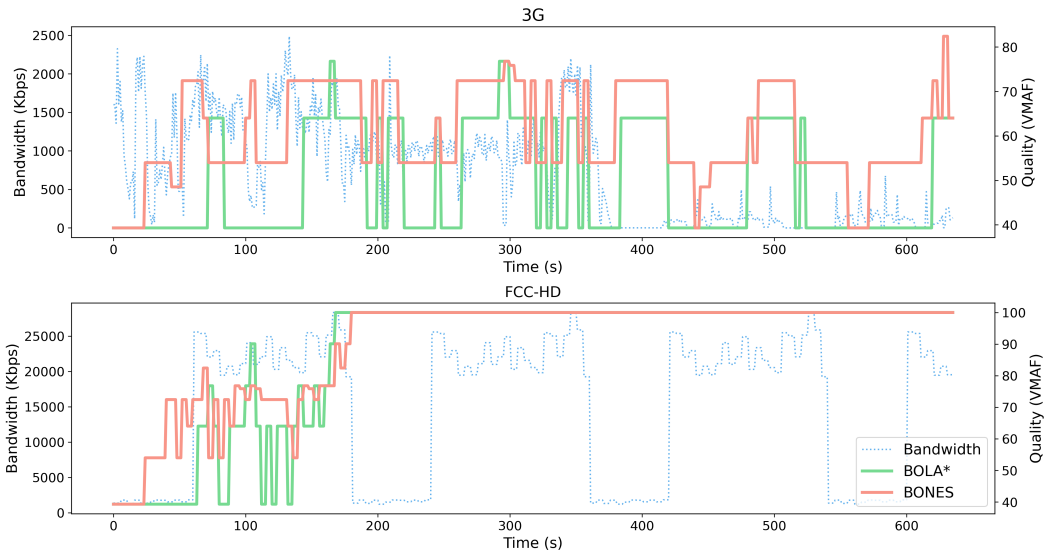


Fig. 10. Performance evolution in the prototype system over example traces.

varying computational resources, we consistently update this table using the latest measurement of enhancement speed. Outdated enhancement tasks will be immediately aborted.

We implement two variants of IMDN enhancer, one content-agnostic and one content-aware with the QuickStart heuristic introduced in 5.5. We compare our BONES algorithm with the well-performing algorithms in simulation, namely, BOLA*, Dynamic*, Buffer*, Throughput*, and NAS. For a fair comparison, we test the algorithms over 10 pre-recorded network traces sampled from the low-bandwidth 3G network trace dataset and the high-bandwidth FCC-HD dataset.

We present the overall experimental results in Tab. 7. The trend slightly differs from the simulation due to practical factors like the choice of dataset, sampling bias, varying resources, system latency, and implementation details. Under the content-agnostic setting, we observe that BONES outpeforms other baselines in QoE by 4.66% to 13.54%. While under the content-aware setting, the advantage of BONES over baselines ranges from 9.43% to 20.43%. We also illustrate the performance of BONES and the best baseline BOLA* over two example traces in Fig. 10, including the bandwidth variation over time and the video quality determined by the algorithms. We find from the figure that BONES generally achieves better quality than the baseline. In conclusion, the advantage of BONES not only holds in large-scale simulations but also translates to practice.

7 RELATED WORK

Adaptive Bitrate Streaming. The ABR algorithm selects the download bitrate for video segments in adaptation to the varying network bandwidth. ABR algorithms can be generally classified into three categories. First, *buffer-based methods* adapt the bitrate purely based on the client's buffer level. For example, the method in [19] chooses the bitrate according to a piecewise linear function of the buffer level. BOLA [39] makes download decisions by solving a Lyapunov optimization problem with respect to the buffer level. Our method could be viewed as an extension of BOLA from ABR to the new NES scenario. Second, *throughput-based methods* utilize the prediction of future network bandwidth. FESTIVE [22] estimates bandwidth using harmonic mean and chooses the maximum bitrate under that estimation. Fugu [46] proposes a more accurate download time prediction module based on fully-connected neural networks, while Xatu [46] employs a long short-term memory (LSTM) network for prediction. Third, *mixed-input algorithms* consider both buffer level and estimated throughput inputs. Pensieve [29] takes these states as input and trains a reinforcement-learning (RL) agent. FastMPC [50] formulates bitrate adaptation as an online control problem and solves it via model predictive control (MPC). Dynamic [38] switches between a throughput-based method and BOLA to take advantage of both.

Neural Enhancement. Neural enhancement refers to any method that improves video quality via deep learning. It typically encompasses the following areas. Super-resolution (SR) aims at improving the resolution of images [21, 26, 27] and videos [10, 11]. To reduce the training and inference cost of SR models, Li et al. [25] overfit the first video segment and then fine-tune the model toward the following segments. Wang et al. [45] introduce heuristics like training with smaller patches and decreasing the update frequency. DeepStream [4] further introduces scene grouping, frame sub-sampling, and model compression. Frame interpolation aims at improving the frame rate of videos [37, 52]. Inpainting methods can complete the missing region of images [41] or videos [53]. Denoising methods effectively reduce noise in videos [13, 42]. Apart from 2D videos, point cloud upsampling [24, 51] increases the density of 3D point cloud objects in volumetric videos, and point cloud completion [20] makes them whole. In this paper, we used NAS-MDSR [48] and IMDN [21] SR models as our enhancement methods, but our control algorithm `BONES` can be integrated with any subset of the above algorithms.

Neural-Enhanced Streaming NES incorporates neural enhancement methods into video streaming algorithms and enables high-quality content delivery by leveraging both bandwidth and computational resources. NAS [48] is the first method to integrate SR models into on-demand video streaming. It develops a content-aware SR model called NAS-MDSR and utilizes a Pensieve-like RL agent to manage both download and enhancement processes. SRAVS [56] adopts RL controllers and a lightweight SR model while proposing a double-buffer system model. PreSR [57] pre-fetches and enhances "complex" segments that bring the most quality improvement and bandwidth reduction. It formulates the problem as MPC and solves it heuristically. Our method is most similar to these approaches.

Neural enhancement can be applied beyond the scope of client-side on-demand video streaming. LiveNAS [23] upsamples low-resolution videos at the ingest server for live streaming. NEMO [47] increases the computational efficiency of SR by only enhancing selected "anchor" frames. Based on LiveNAS and NEMO, NeuroScaler [49] can enhance live streams at scale. Dejavu [18] improves the quality of the current frame in live video conferencing using historical knowledge. And VISCA [54] deploys SR models on edge-based cache servers. Beyond the regular 2D videos, recent work applied NES to 360-degree and volumetric videos. The substantial bandwidth requirement of these videos makes the assistance of neural enhancement even more appealing. SR360 [12] uses RL to make viewport prediction and enhancement decisions for delivering 360 videos. Sophon [36] pre-fetches

and enhances semantically salient tiles in a 360 video. Madarasingha et al. [28] enhance 360 videos with both SR and frame interpolation at edge servers. YuZu [55] streams volumetric videos and enhances them using point cloud upsampling. Emerging studies of virtual reality streaming systems also pursue optimal policies for joint allocation of the available computational and communication resources[8, 9, 17], where the concept of enhancement is not limited to deep-learning methods.

While the above methods are based on RL or heuristic optimization, we formulate and near-optimally solve a Lyapunov optimization problem. Our method BONES is the first NES algorithm with a theoretical performance bound. Furthermore, BONES has been proven to exhibit exceptional performance through numerous experiments, while also possessing the benefits of being simple and robust in deployment.

8 CONCLUSION

This paper proposed BONES, an NES algorithm incorporating neural enhancement into video streaming, allowing users to download low-quality video segments and then enhance them via deep-learning models. Residing in a parallel-buffer system model, BONES jointly optimizes download and enhancement decisions to maximize the QoE by solving a Lyapunov optimization problem. BONES achieves the performance within an additive factor toward offline optimal, making it the first NES algorithm with a theoretical performance guarantee. Experiments verify that BONES can outperform existing ABR and NES methods by 3.56% to 13.20% in simulation and 4.66% to 20.43% in practice. BONES maintains stable performance under all network conditions, has explainable hyper-parameters, and offers different trade-offs between performance and overhead. We publicly release our source code, expecting BONES to become a new baseline for the NES problem.

There are some limitations to the current work, which open future research directions. Firstly, we assume the computation time of enhancement is static or can be accurately predicted. This may not hold in reality especially when other applications are competing for computational resources. Secondly, we only consider scheduling one type of enhancement (super-resolution). In practice, it is possible to simultaneously apply multiple enhancements, which may require the scheduling of multiple enhancement buffers. Thirdly, we only consider a client-side on-demand video streaming scenario. Deploying a streaming algorithm in the cloud or for live streaming may raise other interesting challenges. Given the simplicity of algorithm design, bounded theoretical performance, and superior empirical performance of BONES, in future work, we will explore the possibility of generalizing BONES to more dynamic, multi-buffer, multi-point, and low-latency systems.

ACKNOWLEDGMENTS

The work has been supported by the National Science Foundation (NSF) under awards CNS-1763617, CNS-1901137, CNS-2106463, CNS-2102963, CAREER-2045641, CNS-2106299, CCF-2031881, ECCS-2032387, CNS-2040088, CNS-2032033, and CNS-2106150; by the National Institutes of Health (NIH) under award R01EY030470; and by the Panasonic Chair of Sustainability at the New Jersey Institute for Technology.

REFERENCES

[1] Anne Aaron, Zhi Li, Megha Manohara, Joe Yuchieh Lin, Eddy Chi-Hao Wu, and C.-C Jay Kuo. 2015. Challenges in cloud based ingest and encoding for high quality streaming media. In *2015 IEEE International Conference on Image Processing (ICIP)*. 1732–1736. https://doi.org/10.1109/ICIP.2015.7351097

[2] Eirikur Agustsson and Radu Timofte. 2017. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[3] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-Tuning Video ABR Algorithms to Network Conditions. In *Proceedings of*

the 2018 Conference of the ACM Special Interest Group on Data Communication (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 44–58. https://doi.org/10.1145/3230543.3230558

[4] Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer. 2022. DeepStream: Video Streaming Enhancements using Compressed Deep Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology* (2022), 1–1. https://doi.org/10.1109/TCSVT.2022.3229079

[5] Akram Ansari and Mea Wang. 2023. IStream Player: A Versatile Video Player Framework. In *Proceedings of the 33rd Workshop on Network and Operating System Support for Digital Audio and Video* (Vancouver, BC, Canada) *(NOSSDAV '23)*. Association for Computing Machinery, New York, NY, USA, 65–71. https://doi.org/10.1145/3592473.3592569

[6] J. Chakareski, J. Apostolopoulos, S. Wee, W.-T. Tan, and B. Girod. 2005. Rate-Distortion Hint Tracks for Adaptive Video Streaming. *IEEE Trans. Circuits and Systems for Video Technology* 15, 10 (Oct. 2005), 1257–1269.

[7] J. Chakareski and P.A. Chou. 2006. RaDiO Edge: Rate-Distortion Optimized Proxy-Driven Streaming from the Network Edge. *IEEE/ACM Trans. Networking* 14, 6 (Dec. 2006), 1302–1312.

[8] J. Chakareski, M. Khan, T. Ropitault, and S. Blandino. 2023. Millimeter Wave and Free-Space-Optics for Future Dual-Connectivity 6DOF Mobile Multi-User VR Streaming. *ACM Transactions on Multimedia Computing Communications and Applications* 19, 2(15) (feb 2023), 1–25.

[9] J. Chakareski, M. Khan, and M. Yuksel. 2022. Towards Enabling Next Generation Societal Virtual Reality Applications for Virtual Human Teleportation. *IEEE Signal Processing Magazine* 39, 5 (2022), 22–41.

[10] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. 2021. BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4947–4956.

[11] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. 2022. BasicVSR++: Improving Video Super-Resolution With Enhanced Propagation and Alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5972–5981.

[12] Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang, and Di Wu. 2020. SR360: Boosting 360-Degree Video Streaming with Super-Resolution. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (Istanbul, Turkey) *(NOSSDAV '20)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3386290.3396929

[13] Michele Claus and Jan van Gemert. 2019. ViDeNN: Deep Blind Video Denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[14] Federal Communications Commission. 2016. Raw Data - Measuring Broadband America 2016. https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016 https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016.

[15] Mallesham Dasari, Arani Bhattacharya, Santiago Vargas, Pranjal Sahu, Aruna Balasubramanian, and Samir R. Das. 2020. Streaming 360-Degree Videos Using Super-Resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 1977–1986. https://doi.org/10.1109/INFOCOM41043.2020.9155477

[16] Robert G. Gallager. 2012. *Discrete Stochastic Processes*. Springer New York, NY. https://doi.org/10.1007/978-1-4615-2329-1

[17] S. Gupta, J. Chakareski, and P. Popovski. 2023. mmWave Networking and Edge Computing for Scalable 360-Degree Video Multi-User Virtual Reality. *IEEE Trans. Image Processing* 32 (2023), 377–391.

[18] Pan Hu, Rakesh Misra, and Sachin Katti. 2019. Dejavu: Enhancing Videoconferencing with Prior Knowledge. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, USA) *(HotMobile '19)*. Association for Computing Machinery, New York, NY, USA, 63–68. https://doi.org/10.1145/3301293.3302373

[19] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *SIGCOMM Comput. Commun. Rev.* 44, 4 (aug 2014), 187–198. https://doi.org/10.1145/2740070.2626296

[20] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. 2020. PF-Net: Point Fractal Network for 3D Point Cloud Completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. 2019. Lightweight Image Super-Resolution with Information Multi-Distillation Network. In *Proceedings of the 27th ACM International Conference on Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 2024–2032. https://doi.org/10.1145/3343031.3351084

[22] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies* (Nice, France) *(CoNEXT '12)*. Association for Computing Machinery, New York, NY, USA, 97–108. https://doi.org/10.1145/2413176.2413189

[23] Jaehong Kim, Youngmok Jung, Hyunho Yeo, Juncheol Ye, and Dongsu Han. 2020. 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* (Virtual Event, USA) *(SIGCOMM '20)*. Association for Computing Machinery, New York, NY, USA, 107–125. https://doi.org/10.1145/3387514.3405856

[24] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2019. PU-GAN: A Point Cloud Upsampling Adversarial Network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[25] Xiaoqi Li, Jiaming Liu, Shizun Wang, Cheng Lyu, Ming Lu, Yurong Chen, Anbang Yao, Yandong Guo, and Shanghang Zhang. 2022. Efficient Meta-Tuning for Content-Aware Neural Video Delivery. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 308–324.

[26] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. 2021. SwinIR: Image Restoration Using Swin Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 1833–1844.

[27] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[28] Chamara Madarasingha and Kanchana Thilakarathna. 2022. Edge Assisted Frame Interpolation and Super Resolution for Efficient 360-Degree Video Delivery. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking* (Sydney, NSW, Australia) *(MobiCom '22)*. Association for Computing Machinery, New York, NY, USA, 856–858. https://doi.org/10.1145/3495243.3558261

[29] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[30] Yun Seong Nam, Jianfei Gao, Chandan Bothra, Ehab Ghabashneh, Sanjay Rao, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2021. Xatu: Richer Neural Network Based Prediction for Video Streaming. *Proc. ACM Meas. Anal. Comput. Syst.* 5, 3, Article 44 (dec 2021), 26 pages. https://doi.org/10.1145/3491056

[31] M. Neely. 2010. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool Publishers. https://books.google.com/books?id=sZpeAQAAQBAJ

[32] Michael J. Neely. 2013. Dynamic Optimization and Learning for Renewal Systems. *IEEE Trans. Automat. Control* 58, 1 (2013), 32–46. https://doi.org/10.1109/TAC.2012.2204831

[33] NVIDIA. 2023. Video Processing Framework. https://github.com/NVIDIA/VideoProcessingFramework

[34] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications. In *Proceedings of the 4th ACM Multimedia Systems Conference* (Oslo, Norway) *(MMSys '13)*. Association for Computing Machinery, New York, NY, USA, 114–118. https://doi.org/10.1145/2483977.2483991

[35] Sandvine. 2023. 2023 Global Internet Phenomena Report. https://www.sandvine.com/global-internet-phenomena-report-2023 https://www.sandvine.com/global-internet-phenomena-report-2023.

[36] Jianxin Shi, Lingjun Pu, Xinjing Yuan, Qianyun Gong, and Jingdong Xu. 2022. Sophon: Super-Resolution Enhanced 360° Video Streaming with Visual Saliency-Aware Prefetch. In *Proceedings of the 30th ACM International Conference on Multimedia* (Lisboa, Portugal) *(MM '22)*. Association for Computing Machinery, New York, NY, USA, 3124–3133. https://doi.org/10.1145/3503161.3547750

[37] Zhihao Shi, Xiaohong Liu, Chengqi Li, Linhui Dai, Jun Chen, Timothy N. Davidson, and Jiying Zhao. 2022. Learning for Unconstrained Space-Time Video Super-Resolution. *IEEE Transactions on Broadcasting* 68, 2 (2022), 345–358. https://doi.org/10.1109/TBC.2021.3131875

[38] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2019. From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player. *ACM Trans. Multimedia Comput. Commun. Appl.* 15, 2s, Article 67 (jul 2019), 29 pages. https://doi.org/10.1145/3336497

[39] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711. https://doi.org/10.1109/TNET.2020.2996964

[40] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (San Jose, CA, USA) *(MMSys '11)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1943552.1943572

[41] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. 2022. Resolution-Robust Large Mask Inpainting With Fourier Convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*

(WACV). 2149–2159.

[42] Matias Tassano, Julie Delon, and Thomas Veit. 2020. FastDVDnet: Towards Real-Time Deep Video Denoising Without Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[43] N. Thomos, J. Chakareski, and P. Frossard. 2011. Prioritized Distributed Video Delivery with Randomized Network Coding. *IEEE Trans. Multimedia* 13, 4 (Aug. 2011), 776–787.

[44] Jeroen van der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters* 20, 11 (2016), 2177–2180. https://doi.org/10.1109/LCOMM.2016.2601087

[45] Zelong Wang, Zhenxiao Luo, Miao Hu, Di Wu, Youlong Cao, and Yi Qin. 2022. Revisiting Super-Resolution for Internet Video Streaming. In *Proceedings of the 32nd Workshop on Network and Operating Systems Support for Digital Audio and Video* (Athlone, Ireland) *(NOSSDAV '22)*. Association for Computing Machinery, New York, NY, USA, 8–14. https://doi.org/10.1145/3534088.3534344

[46] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. [n. d.]. Learning in situ: a randomized experiment in video streaming. *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI '20)* ([n. d.]). https://par.nsf.gov/biblio/10186616

[47] Hyunho Yeo, Chan Ju Chong, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2020. NEMO: Enabling Neural-Enhanced Video Streaming on Commodity Mobile Devices. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages. https://doi.org/10.1145/3372224.3419185

[48] Hyunho Yeo, Youngmok Jung, Jaehong Kim, Jinwoo Shin, and Dongsu Han. 2018. Neural Adaptive Content-Aware Internet Video Delivery. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation* (Carlsbad, CA, USA) *(OSDI'18)*. USENIX Association, USA, 645–661.

[49] Hyunho Yeo, Hwijoon Lim, Jaehong Kim, Youngmok Jung, Juncheol Ye, and Dongsu Han. 2022. NeuroScaler: Neural Video Enhancement at Scale *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 795–811. https://doi.org/10.1145/3544216.3544218

[50] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *SIGCOMM Comput. Commun. Rev.* 45, 4 (aug 2015), 325–338. https://doi.org/10.1145/2829988.2787486

[51] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. PU-Net: Point Cloud Upsampling Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[52] Zijie Yue and Miaojing Shi. 2023. Enhancing Space-time Video Super-resolution via Spatial-temporal Feature Interaction. arXiv:2207.08960 [cs.CV]

[53] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. 2020. Learning Joint Spatial-Temporal Transformations for Video Inpainting. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 528–543.

[54] Aoyang Zhang, Qing Li, Ying Chen, Xiaoteng Ma, Longhao Zou, Yong Jiang, Zhimin Xu, and Gabriel-Miro Muntean. 2021. Video Super-Resolution and Caching—An Edge-Assisted Adaptive Video Streaming Solution. *IEEE Transactions on Broadcasting* 67, 4 (2021), 799–812. https://doi.org/10.1109/TBC.2021.3071010

[55] Anlan Zhang, Chendong Wang, Bo Han, and Feng Qian. 2022. YuZu: Neural-Enhanced Volumetric Video Streaming. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. USENIX Association, Renton, WA, 137–154. https://www.usenix.org/conference/nsdi22/presentation/zhang-anlan

[56] Yinjie Zhang, Yuanxing Zhang, Yi Wu, Yu Tao, Kaigui Bian, Pan Zhou, Lingyang Song, and Hu Tuo. 2020. Improving Quality of Experience by Adaptive Video Streaming with Super-Resolution. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 1957–1966. https://doi.org/10.1109/INFOCOM41043.2020.9155384

[57] Gangqiang Zhou, Zhenxiao Luo, Miao Hu, and Di Wu. 2023. PreSR: Neural-Enhanced Adaptive Streaming of VBR-Encoded Videos With Selective Prefetching. *IEEE Transactions on Broadcasting* 69, 1 (2023), 49–61. https://doi.org/10.1109/TBC.2022.3227419

A PROOF OF THEOREM 1

Proof. We prove the inequality $Q^d(t_k) \le V\frac{u_{\max}+\gamma p}{p} + p$ by induction. The bound holds for $k = 1$ as $Q^d(t_1) = Q^d(0) = 0$. Suppose it also holds for some k. Then there are two cases for $k+1$.

(1) $Q^d(t_k) \le V\frac{u_{\max}+\gamma p}{p}$. From Eq. (1) we know $Q^d(t_k)$ can increase by at most p in one single time slot. Hence, we have $Q^d(t_{k+1}) \le V\frac{u_{\max}+\gamma p}{p} + p$.

(2) $V\frac{u_{\max}+\gamma p}{p} < Q^d(t_k) \le V\frac{u_{\max}+\gamma p}{p} + p$. In this case, we have

$$V < \frac{Q^d(t_k)p}{u_{\max}+\gamma p} \le \frac{Q^d(t_k)p + Q^e(t_k)t^e(i,j)}{u^d(i,t_k) + \tilde{u}^e(i,j,t_k) + \gamma p},\ \forall i,j.$$

This makes $O_D(t_k)+VO_P(t_k) > 0,\ \forall \mathbf{d}, \mathbf{e}$, in Eq. (11) and forces the control algorithm not to download. As a result, $Q^d(t_{k+1}) < Q^d(t_k)$.

Till now, we have proven $Q^d(t_k) \le V\frac{u_{\max}+\gamma p}{p} + p$. Combining it with $V \le \frac{(Q^d_{\max}-p)p}{u_{\max}+\gamma p}$ from assumption, we have $Q_d(t_k) \le Q^d_{\max}$ and the proof is complete. □

B PROOF OF THEOREM 2

Lemma 1. *Denote the optimal objective of problem Eq. (8) as $\bar{u}^{opt} + \gamma\bar{s}^{opt}$. There exists a stationary i.i.d. algorithm for this problem that has the following properties for any $\delta > 0$.*

(1) $\bar{u}^* + \gamma\bar{s}^* \le \bar{u}^{opt} + \gamma\bar{s}^{opt} + \delta$.
(2) $\frac{\mathbb{E}\{\sum_{ki} d_i^* p\}}{\mathbb{E}\{\sum_k T_k^*\}} \le 1 + \delta$.
(3) $\frac{\mathbb{E}\{\sum_{kij} d_i^* e_j^* t^e(i,j)\}}{\mathbb{E}\{\sum_k T_k^*\}} \le 1 + \delta$.

Proof. Problem Eq. (8) is formulated as maximizing time-averaged objective under time-averaged constraints in a system with variable-size renewal frames. This lemma can be directly derived from Lemma 1 in [32]. □

Based on Lemma 1, we prove Theorem 2 as follows.

Proof. In the following proof, we will use superscript $'$ to denote the decisions of BONES, and $*$ for those of the optimal i.i.d. algorithm.

Following the framework of Lyapunov optimization over renewable frames [32], we define the Lyapunov function $L(t_k)$ as

$$L(t_k) \triangleq \frac{1}{2}(Q^d(t_k)^2 + Q^e(t_k)^2). \tag{15}$$

Define $\mathbf{Q}(t_k) = [Q^d(t_k), Q^e(t_k)]$, then the conditional Lyapunov drift is formulated as

$$D(t_k) \triangleq \mathbb{E}\{L(t_{k+1}) - L(t_k)|\mathbf{Q}(t_k)\}. \tag{16}$$

The drift $D(t_k)$ is bounded by

$$\begin{aligned} D(t_k) \le\ & \frac{1}{2}p^2 + \frac{1}{2}(t^e_{\max})^2 + T_{\max}\mathbb{E}\{T_k|\mathbf{Q}(t_k)\} \\ & + Q^d(t_k)\mathbb{E}\{\sum_i d_i p - T_k|\mathbf{Q}(t_k)\} \\ & + Q^e(t_k)\mathbb{E}\{\sum_{ij} d_i e_j t^e(i,j) - T_k|\mathbf{Q}(t_k)\}. \end{aligned} \tag{17}$$

For the sake of simplicity, we denote $\frac{1}{2}p^2 + \frac{1}{2}(t^e_{\max})^2$ as C. Now we reuse $O_D(t_k)$ in Eq. (9) and $O_P(t_k)$ in Eq. (10). Adding V times $E\{O_P(t_k)|\mathbf{Q}(t_k)\}$ on both sides and simplifying the formula using $E\{O_D(t_k)|\mathbf{Q}(t_k)\}$, we have

$$\begin{aligned}
&D(t_k) + V\mathbb{E}\{O_P(t_k)|\mathbf{Q}(t_k)\} \\
&\leq C + (T_{\max} - Q^d(t_k) - Q^e(t_k))\mathbb{E}\{T_k|\mathbf{Q}(t_k)\} \\
&+ \mathbb{E}\{O_D(t_k) + VO_P(t_k)|\mathbf{Q}(t_k)\}.
\end{aligned} \tag{18}$$

Recall that the time slot duration can be computed by $T_k = (\sum_i d_i S_i)/\omega(t_k)$. As our algorithm greedily minimizes the objective defined in Eq. (11), we can derive

$$\begin{aligned}
&\frac{O'_D(t_k) + VO'_P(t_k)}{\sum_i d'_i S_i} \leq \frac{O^*_D(t_k) + VO^*_P(t_k)}{\sum_i d^*_i S_i} \\
\Rightarrow &\frac{O'_D(t_k) + VO'_P(t_k)}{(\sum_i d'_i S_i)/\omega(t_k)} \leq \frac{O^*_D(t_k) + VO^*_P(t_k)}{(\sum_i d^*_i S_i)/\omega(t_k)} \\
\Rightarrow &\mathbb{E}\{O'_D(t_k) + VO'_P(t_k)|\mathbf{Q}(t_k)\} \\
&\leq \frac{\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\}}{\mathbb{E}\{T^*_k|\mathbf{Q}(t_k)\}}\mathbb{E}\{O^*_D(t_k) + VO^*_P(t_k)|\mathbf{Q}(t_k)\}.
\end{aligned} \tag{19}$$

We then apply Eq. (19) to Eq. (18) and bring in the definition of Eq. (3), Eq. (5). Since the total video length or total enhancement time is less or equal to the total playback time, we have

$$\begin{aligned}
&D'(t_k) + V\mathbb{E}\{O'_P(t_k)|\mathbf{Q}(t_k)\} \\
&\leq C + T_{\max}\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\} - V(\bar{u}^* + \gamma\bar{s}^*)\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\} \\
&+ Q^d(t_k)(\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\}(\frac{\mathbb{E}\{\sum_i d^*_i p|\mathbf{Q}(t_k)\}}{\mathbb{E}\{T^*_k|\mathbf{Q}(t_k)\}}) - 1) \\
&+ Q^e(t_k)(\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\}(\frac{\mathbb{E}\{\sum_{ij} d^*_i e^*_j t^e(i,j)|\mathbf{Q}(t_k)\}}{\mathbb{E}\{T^*_k|\mathbf{Q}(t_k)\}}) - 1) \\
&\leq C + T_{\max}\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\} + V(\bar{u}^* + \gamma\bar{s}^*)\mathbb{E}\{T'_k|\mathbf{Q}(t_k)\}.
\end{aligned} \tag{20}$$

Next, sum both sides over $k = 1, \cdots, K_N$ and we get

$$\begin{aligned}
&\mathbb{E}\{L(t_{k+1})\} + V\mathbb{E}\{\sum_k O'_P(t_k)\} \\
&\leq K_N C + T_{\max}\mathbb{E}\{\sum_k T'_k\} - V(\bar{u}^* + \gamma\bar{s}^*)\mathbb{E}\{\sum_k T'_k\}.
\end{aligned} \tag{21}$$

We divide both sides by $\mathbb{E}\{\sum_k T'_k\}$ and take the limit $K_N \to \infty$. Since $L(k_{k+1}) \geq 0$, we have

$$-V(\bar{u}' + \gamma\bar{s}') \leq \lim_{K_N\to\infty} \frac{K_N C}{\mathbb{E}\{\sum_k T'_k\}} + T_{\max} - V(\bar{u}^* + \gamma\bar{s}^*). \tag{22}$$

Because the lower bound of time slot duration is $T_{\min}$, we can transform the above formula into Eq. (12).

□

Received August 2023; revised January 2024; accepted March 2024