

## RESEARCH ARTICLE

# Joint Preloading and Bitrate Adaptation for Short Video Streaming

NGUYEN TIEN PHONG<sup>1</sup>, TRUONG THU HUONG<sup>1</sup>, (Member, IEEE),  
PHAM NGOC NAM<sup>2</sup>, (Member, IEEE), TRUONG CONG THANG<sup>3</sup>, (Senior Member, IEEE),  
AND DUC NGUYEN<sup>4</sup>, (Member, IEEE)

<sup>1</sup>School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi 100000, Vietnam

<sup>2</sup>College of Engineering and Computer Science, VinUniversity, Hanoi 10000, Vietnam

<sup>3</sup>Department of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan

<sup>4</sup>Department of Information and Communication Engineering, Tohoku Institute of Technology, Sendai 982-0846, Japan

Corresponding author: Truong Thu Huong (huong.truongthu@hust.edu.vn)


This work was supported by the Hanoi University of Science and Technology (HUST) under Project T2022-PC-012.

**ABSTRACT** Short videos have become one of the most popular content mobile users consume nowadays. However, unlike traditional videos, users watch many short videos each time and frequently skip those not of their interest. Not taking into account this factor, conventional video streaming methods result in a large amount of data wastage. Moreover, short videos are usually streamed over time-varying network conditions, which requires preloading to provide satisfactory Quality of Experience (QoE). In this paper, we formulate the QoE-Wastage trade-off optimization problem in short video streaming as a multi-objective optimization problem. We then develop a lightweight joint preloading and bitrate adaptation algorithm that utilizes cross-user behaviors to decide a suitable amount of preloaded data for each video. In addition, our method dynamically adjusts video bitrate per chunk basis to achieve high user QoE. Experimental results show that the proposed method improves QoE by approximately 12% to 45% and reduces data wastage by up to 58% compared to existing methods.

**INDEX TERMS** Short video streaming, quality of experience, data wastage, preloading, bitrate adaptation.

## I. INTRODUCTION

Short videos have become one of the most popular multimedia content consumed by mobile users these days. According to [1], over a billion monthly users are watching short videos. The global short video platform market is predicted to double in size during 2022-2030 [2]. Short videos are typically viewed on smartphone apps and are displayed in full-screen mode. Users scroll up/down to go to the previous/next video in a playlist containing a list of videos, which are recommended to users. Short videos are stored on a streaming server in which each video is divided into multiple chunks with the same playback duration. A video chunk might be further encoded into multiple versions of different quality levels to support network adaptation. The

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaoqing Pan .

user agent (i.e., mobile apps) needs to decide which chunk should be downloaded at which quality. Then, it downloads the video data from the streaming server chunk-by-chunk using standard protocols such as HTTP [3].

There is a trade-off between users' Quality of Experience (QoE) and data wastage in short video streaming. To achieve a high QoE under time-varying network conditions, the user agent should preload a certain amount of video data before starting the playback of a video. Using a high amount of preloaded data (i.e., buffer size) can help reduce re-buffering events when the network bandwidth suddenly drops [4]. Since re-buffering causes negative effects on the user's QoE, a large buffer size can help improve the user's QoE [5]. A large buffer size, however, might result in a significant amount of data wastage. For example, if the user agent buffers all segments of a video but the user only watches half of the video, then half of the downloaded data becomes wasted. In fact, a recent

investigation found that most short videos are early skipped by users, leading to a significant amount of data wastage [6]. Therefore, how to design an effective video chunk download strategy for the best trade-off between the user's QoE and data wastage is the primary problem in short video streaming. A chunk download strategy must decide 1) when to download and 2) which version to download for every video chunk. The chunk to download at each step is not necessarily belonging to the video currently being viewed by the user. Instead, video chunks of the next videos in the playlist can also be preloaded to allow users to view the next video instantly.

Chunk download strategies for short video streaming have been previously proposed. Early works focus on designing appropriate chunk preloading strategies based on various factors such as predicted user viewing time [7], and network throughput [8]. Preloading and bitrate adaptation are jointly considered in [9]. However, this method results in high variations in video quality, causing negative effects on user experience. More recent works employ a learning-based approach in which user viewing data is collected and used to train deep reinforcement learning models that decide chunk download strategy [10], [11], [12], [13], [14], [15]. The primary problem with the learning-based approach is that the user agent needs to run complex machine learning models on the user's device. Running a machine learning model consumes a significant amount of computational resources, thus the learning-based approach is not suitable for resource-constrained devices.

In this paper, we first present the system model for short video streaming and formulate the QoE-Wastage trade-off optimization problem in short video streaming as a multi-objective optimization problem. To the best of our knowledge, our work is the first that present a detailed analysis of the QoE-Wastage trade-off in short video streaming. We then propose a joint preloading and bitrate adaptation method to improve the trade-off between the user's QoE and data wastage. Compared to learning-based methods, the proposed method is much simpler, requiring only statistics of the user's viewing duration. The data can be collected by the service providers and sent to the user agent prior to a streaming session with insignificant overhead. The proposed method dynamically adjusts the buffer size based on the user's retention rate to minimize data wastage. Also, a Model Predictive Control (MPC)-based bitrate adaptation method is proposed to decide the quality level on a per-chunk basis to maximize user's QoE. Comparison with five existing methods shows that the proposed method achieves state-of-the-art performance in terms of QoE while achieving the smallest amount of data wastage.

The remainder of the paper is organized as follows. Related work is presented in Section II. The QoE-Wastage trade-off optimization problem in short video streaming is formulated in Section III. The proposed method is described in Section IV, followed by an evaluation in Section V. Finally, the paper is concluded in Section VI.

## II. RELATED WORK

Recently, measurement studies have been conducted to understand the characteristics of commercial short video services. In [6], the authors analyze a top-10 short video service in China and found that video content is extremely short with a median video length of 22s, and about 77% of the short video sessions are less than 40s. The study also reveals that users of the considered short video service have a very limited attention span of approximately 25s with nearly 70% of videos being skipped early. Since the service uses a simple download-and-play adaptation scheme, a significant amount of downloaded video data is discarded without being viewed, causing data wastage. While the short video services studied in [6] and [10] support only a single quality level (version) of each video, other services such as that in [3] support multiple video quality levels. The preloading strategies also vary between different services. While some services such as Douyin preload only the next video [11] in the playlist, services such as YouTube Shorts, and Instagram Reel preload several videos ahead [3]. A short video can be preloaded in full (e.g., Douyin) or in part, (e.g., Facebook Watch) [3].

Two main approaches have been used to develop preloading strategies for short video streaming: Learning-based approach [10], [11], [12], [13], [14], [15], and conventional approach [7], [8], [9], [17]. In the learning-based approach, user data such as viewing duration, and swipe times are collected and used to train machine learning models that decide the chunk download strategy. In [11], the authors propose *LiveClip*, a deep reinforcement learning-based method. The method collects the users' staying time in the past as input to predict when the user will scroll, then decides the next action which downloads the next chunk of the current video or the next 2 videos or pauses to save bandwidth usage based on predicting user staying time. In [14], the adaptation is made at the session level instead of the chunk level. In general, the learning-based methods are found effective in reducing data wastage. In [10], Wastage Aware Streaming (WAS) method is developed based on viewing behaviors and network conditions to minimize data wastage while maintaining QoE. WAS tunes two wastage-aware parameters, one for controlling video bitrate and another for controlling buffer size. Both parameters are learned based on past user behaviors and network conditions using greedy search. In [12], *DUASVS* is introduced as an enhanced iteration of WAS. This approach employs the Asynchronous Advantage Actor Critic (A3C) algorithm with an Actor-Critic Network to optimize wastage-aware parameters. In [13], the reinforcement learning-based agent decides which chunk to download at which bitrate, as well as the pause time. In addition, domain knowledge is incorporated via action masking to improve transparency as well as accelerate the training process. Moreover, in [15], the utilization of imitation learning and multi-agent proximal policy optimization (MAPPO) is observed. These techniques are employed to

TABLE 1. Overview of the state of the art.

Ref.	Year	Dataset	Method	Contribution		Evaluation metric				
				Adaptive Preloading	Adaptive Bitrate	QoE			Data Wastage	Bandwidth Usage
						Bitrate Quality	Re-buffer. Time	Bitrate Variation		
<b>Learning-based approach</b>										
[11]	2020	Douyin	Actor-Critic	✓	×	×	✓	×	✓	×
[14]	2021	N.A	Actor-Critic	×	✓	✓	×	×	×	×
[10]	2021	Anonymous	Greedy search	✓	✓	✓	✓	✓	✓	×
[12]	2022	Anonymous	A3C	✓	✓	✓	✓	✓	✓	×
[13]	2022	MMGC2022 [16]	Actor-Critic, Action Masking	✓	✓	✓	✓	✓	×	✓
[15]	2023	MMGC2022 [16], DUASVS [12]	Imitation learning, MAPPO	✓	✓	✓	✓	✓	×	✓
<b>Conventional approach</b>										
[7]	2020	Tiktok	Lyapunov Opt.	✓	×	×	✓	×	✓	×
[8]	2022	N.A	Heuristic	✓	×	×	✓	×	✓	×
[9]	2022	MMGC2022 [16]	RobustMPC	✓	✓	✓	✓	✓	×	✓
[17]	2023	Mturk, College Campus	RobustMPC	✓	✓	✓	✓	✓	✓	×
<b>This paper</b>		MMGC2022 [16]	MPC	✓	✓	✓	✓	✓	✓	×

acquire a chunk download strategy that involves separate execution of preloading and bitrate adaptation.

In the conventional approach, the buffer size and bitrates of video chunks are decided using various factors such as the user's past viewing duration [7], and network throughputs [8]. In [7], the authors design an adaptive preloading mechanism for short-form videos which is based on the Lyapunov optimization. The method predicts the user's viewing duration of the next video and decides the number of next videos that can be downloaded and the buffer length of all videos to maximize the playback smoothness (minimize the re-buffering) while causing less waste of bandwidth. Next, in [8], a network-based preloading method is proposed to dynamically adjust the buffer size and a number of prefetched videos based on network conditions. The above methods, however, do not consider bitrate adaptation. In [9], the authors propose PDAS, a probability-based adaptation method for short video streaming, in which the maximum buffer size is chosen based on chunk-level viewing probability and estimated network bandwidth. Furthermore, in [17], the authors present Dashlet, a system that predicts the re-buffering time for each potential video chunk to determine the optimal chunk buffering sequence for preloading. To select the appropriate bitrate level for this buffering sequence, the authors employ RobustMPC, a control algorithm with a moving horizon based on time. Table 1 provides a quick overview of the state of the art.

### III. SHORT VIDEO STREAMING

#### A. GENERAL SYSTEM ARCHITECTURE

In this section, we describe the general architecture of a short video streaming system. As shown in Figure 1, the system consists of a streaming server and a user agent

(i.e., Mobile App). The streaming server stores short videos in which each video is temporally divided into multiple chunks with the same playback duration. Each chunk is encoded into multiple versions of different bitrates. The higher the bitrate is, the better the quality becomes. Description information of individual videos such as chunk duration, chunk bit rates, and a number of versions are also stored in the server in advance.

The user agent is located on the user device and is composed of five main components: A recommender, a download scheduler, a player, a downloader, and a buffer. At the beginning of a session, the recommender engine generates a list of recommended videos based on information such as user preference, user's past viewing history, etc. The downloader then requests the metadata of the recommended videos from the server. Such information is forwarded to the scheduler that makes a decision on which chunk of the video should be downloaded. The downloader fetches the video chunks from the streaming server and stores them in the buffer. In our system, chunks of each video in the playlist are stored in a separate buffer. In short video applications, videos are displayed in full-screen mode. The user can scroll up/down to view the previous/next video in the playlist. When a switching event occurs, the player will reset their current status and start playing the new video from the beginning. In this paper, we assume that users only switch to the next video in the playlist.

The recommender and the download scheduler are the two key components in short video applications. In this paper, we design and develop an effective download scheduler, highlighted in Figure 1. Recommending suitable videos to users is an interesting issue and is reserved for our future work.

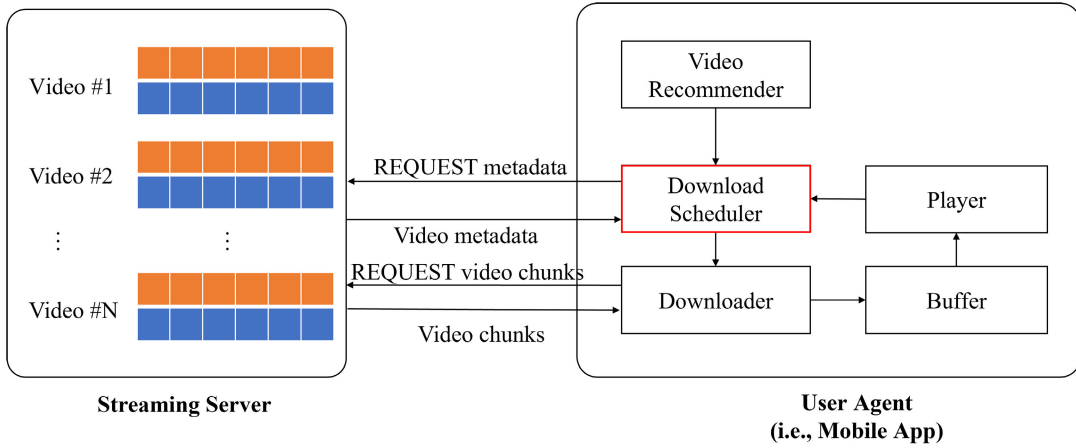


FIGURE 1. Short video streaming system.

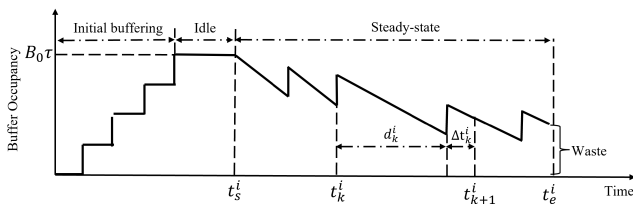


FIGURE 2. Buffer dynamic of individual videos.

In the following section, the operation of our proposed algorithm developed inside the download scheduler will be described.

**B. SHORT VIDEO STREAMING MODEL**

Assume that a user watches a total of  $N$  short videos over a streaming session. Video  $i$  ( $1 \leq i \leq N$ ) consists of  $K_i$  consecutive video chunks or chunks, each has a playback duration of  $\tau$  seconds. Each chunk is encoded into different bitrates. Let  $\mathcal{R}$  be the set of all available bitrate levels. The user agent can decide to download chunk  $k$  of video  $i$  at bitrate  $R_k^i \in \mathcal{R}$ . Downloaded video chunks of each video are stored in a dedicated playback buffer.

For a video in the playlist, the user agent first enters the initial buffering stage where it downloads the first  $B_0$  chunks of the video. If the initial buffering has not been completed at the time the user switches to a video, then the user will experience a start-up delay. In video streaming, long start-up delay is known to cause negative effects on user experience [5]. After initial buffering, the video enters an idle stage if the user has not switched to the video. Otherwise, the playback of the video is started and the user agent enters the steady-state stage in which the player gets downloaded chunks out of the playback buffer while the downloader adds new chunks to it.

The buffer dynamic of individual videos is shown in Figure 2. Let  $B^i(t) \in [0, B_{max}^i]$  be the buffer occupancy of video  $i$  at time  $t$ . The buffer size  $B_{max}^i$  should depend on not only network conditions but also video start times. At the time

$t_k^i$ , the user agent starts to download chunk  $k$  ( $1 \leq k \leq K_i$ ). The download time of chunk  $k$ , denoted  $d_k^i$ , is:

$$d_k^i = \frac{S(R_k^i)}{C_k^i}, \tag{1}$$

in which  $C_k^i$  is the average download speed experienced during the download process and  $S(R_k^i)$  is the chunk's data size. If we denote  $C_t$  as the network throughput at time  $t$ , then we have:

$$C_k^i = \frac{1}{d_k^i} \int_{t_k^i}^{t_k^i + d_k^i} C_t dt. \tag{2}$$

Once chunk  $k$  is completely downloaded, the user agent waits for a time  $\Delta t_k^i$  and starts to download the next chunk  $k + 1$  at time  $t_{k+1}^i$ .

$$t_{k+1}^i = t_k^i + d_k^i + \Delta t_k^i. \tag{3}$$

The waiting time  $\Delta t_k^i$  is a decision variable, depending on the playback status as well as the stage in which the player is.

The buffer occupancy changes according to chunk download time and video playback status. In particular, in the initial buffering stage, the buffer occupancy increases by  $\tau$  seconds for every chunk download. In idle stage, the buffer occupancy remains unchanged. In steady-state stage, the buffer occupancy decreases by chunk download time and increases by a chunk's playback duration. Thus, if the download time of a chunk is higher than the buffer size when the downloader starts downloading the chunk, the buffer is depleted before the chunk is completely downloaded, causing a re-buffering event. Let  $B_k^i = B^i(t_k^i)$  be the buffer occupancy when the downloader starts to download chunk  $k$  of the video  $i^c$ . The buffer dynamic of the current video can be formulated as follows:

$$B_{k+1}^{i^c} = \begin{cases} \left( (B_k^{i^c} - d_k^{i^c})_+ + \tau - \Delta t_k^{i^c} \right)_+, & k \geq B_0 \\ B_k^{i^c} + \tau, & k < B_0. \end{cases} \tag{4}$$

Here,  $(x)_+$  denotes  $\max(x, 0)$ ,  $i^c$  is the current video, and  $B_0$  is the number of chunks to download in the *initial buffering* stage. In addition, the downloader can download chunks of the next videos to avoid re-buffering events when the user skips the current video, therefore, we have the buffer dynamic when downloading chunk  $k'$  of a next video  $i'$  (video  $i'$  can be any of the next videos that are considered to preload):

$$B_{k'+1}^{i'} = B_{k'}^{i'} + \tau. \quad (5)$$

This results in the current video having to wait until the chunk  $k'$  is downloaded completely to continue preloading the next chunk:

$$\Delta t_k^{i^c} = d_{k'}^{i'}. \quad (6)$$

The *re-buffering* time at chunk  $k$  is dependent on not only chunk download time but also video start times. If the initial buffering has been completed by the time the user switches to video  $i$ , then the player can start the video immediately without delay. As a result, there is no re-buffering occurred at the first chunk. Otherwise, the player has to wait until the first  $B_0$  chunks are completely downloaded, resulting in a re-buffering event. In the *steady-state* stage, re-buffering events occur when the chunk download time is higher than the buffer size at the start of the download. Let  $t_s^i$  be the time when user switches to video  $i$ ,  $RB_k^i$  be the re-buffering time at chunk  $k$ , we have:

$$RB_k^i = \begin{cases} (t_{B_0}^i + d_{B_0}^i - t_s^i)_+ & , k = 1 \\ 0 & , 1 < k \leq B_0 \\ ((d_k^i + \Delta t_k^i) - B_k^i)_+ & , k > B_0 \end{cases} \quad (7)$$

From Eq. (7), we can see that re-buffering at the first chunk (i.e., start-up delay) can be eliminated by preloading  $B_0$  first chunks prior to switching time  $t_s^i$ . For later chunks, re-buffering can be mitigated by maintaining a high buffer occupancy  $B_k^i$ .

When the user switches to the next video in the playlist, the buffered but not-yet-played data of the current video will be discarded. The amount of discarded data is thus equal to the total amount of buffered video data minus the total amount of played data. Let  $k_i^{**}$  be the last downloaded chunk of video  $i$ ,  $t_e^i$  be the time where user stops watching video  $i$ , we have:

$$k_i^{**} = \operatorname{argmax}_k (tp_k^i | tp_k^i < t_e^i). \quad (8)$$

The total buffered data of video  $i$  can be calculated as the sum of all downloaded chunks:

$$Buffered = \sum_{k=1}^{k_i^{**}} S(R_k^i). \quad (9)$$

Let  $tp_k^i$  denote the play time of chunk  $k$ , then we have:

$$tp_k^i = \begin{cases} t_s^i + RB_k^i & , k = 1 \\ tp_{k-1}^i + \tau + RB_k^i & , k > 1 \end{cases} \quad (10)$$

Let  $k_i^{**}$  denote the last played chunk before video  $i$  stops, we have:

$$k_i^{**} = \operatorname{argmax}_k (tp_k^i | tp_k^i < t_e^i). \quad (11)$$

The total played data of video  $i$  is calculated as follows.

$$Played = \sum_{k=1}^{k_i^{**}-1} S(R_k^i) + \frac{t_e^i - tp_{k_i^{**}}^i}{\tau} S(R_{k_i^{**}}^i). \quad (12)$$

The amount of discarded data of video  $i$ , denoted  $Wastage_i$ , is thus:

$$\begin{aligned} Wastage_i &= Buffered - Played \\ &= \sum_{k=1}^{k_i^*} S(R_k^i) - \left( \sum_{k=1}^{k_i^{**}-1} S(R_k^i) + \frac{t_e^i - tp_{k_i^{**}}^i}{\tau} S(R_{k_i^{**}}^i) \right). \end{aligned} \quad (13)$$

From Eq. (7) and Eq. (13), we can see the trade-off between QoE and data wastage in short video streaming. To increase QoE, one should maintain a high buffer occupancy ( $B_k^i$ ) to mitigate re-buffering. However, a high buffer size would result in  $k_i^* \gg k_i^{**}$ , leading to more data wastage.

### C. QOE-WASTAGE TRADE-OFF OPTIMIZATION PROBLEM

The goal of the download scheduler is to decide which video chunk to download at each time to maximize the user's hIQoE while minimizing data wastage. The data wastage can be measured as the total discarded data over all videos in the playlist.

$$Wastage = \sum_{i=1}^N Wastage_i. \quad (14)$$

In general, the QoE of a video streaming session depends on start-up delay, chunk bitrate (quality), chunk bitrate variations, and re-buffering times [5]. In this paper, we adopt the QoE model proposed in [16]. Specifically, the QoE is modeled as the sum of individual videos' QoE values, which is calculated as a weighted sum of average chunk bitrate, re-buffering duration, and chunk bitrate variation as follows.

$$QoE = \sum_{i=1}^N QoE_i, \quad (15)$$

$$\begin{aligned} QoE_i &= \sum_{k=1}^{K_i} \left( w_1 \times R_k^i - w_2 \times RB_k^i \right) \\ &\quad - \sum_{k=1}^{K_i-1} \left( w_3 \times \left| R_k^i - R_{k+1}^i \right| \right). \end{aligned} \quad (16)$$

The parameters  $w_1, w_2, w_3$  in Eq. (16) reflect the significance of individual factors which is provided by [16].

The QoE-Wastage trade-off optimization problem in short-form video streaming can be formulated as a multi-objective optimization problem as follows.



Given chunk sizes  $S(R_k^i)$ , network throughput  $\{C_t\}_{t>0}$ , video switch times  $\{t_s^i\}_{1 \leq i \leq N}$ , find a chunk download schedule  $\{(R_k^i, t_k^i), 1 \leq i \leq N, 1 \leq k \leq K_i\}$  to maximize QoE while minimizing data wastage.

$$\begin{aligned} & \max_{(R_k^i, t_k^i)} (QoE - Wastage). \\ & \text{Subject to : Eq.(1) } \sim \text{(16)}. \end{aligned} \quad (17)$$

If the network throughput and video switching times are known in advance, it is possible to find the optimal solution to the above problem using methods such as  $\varepsilon$ -constraints [18]. However, both network throughput and video switch times are difficult to predict in practice. Thus, it is important to take into consideration errors in estimations when designing adaptation solutions for short video streaming.

#### IV. PROPOSED ALGORITHM

In this section, we present a novel algorithm that jointly performs preloading and bitrate adaptation for short video streaming that can reduce the amount of data wastage while achieving high QoE. Our proposed algorithm consists of three steps: 1) *Bandwidth Prediction*, 2) *Adaptive Preloading*, and 3) *Bitrate Adaptation*. In the first step, we predict future bandwidth from the past bandwidth samples that we collect. In the second step, our algorithm considers cross-user information and buffer occupancy to choose a video to download the next chunk. In the third step, our algorithm selects a bitrate level for the chunk selected in Step 2. In the following subsections, we will describe the proposed solution step by step.

##### A. BANDWIDTH PREDICTION

Our algorithm utilizes two bandwidth measures: average bandwidth, and smooth bandwidth. To obtain these two values, our algorithm calculates an average bandwidth sample for every chunk download based on Eq. (2). Following that, the average bandwidth  $C_{avg}$  over the latest  $M$  samples is calculated:

$$C_{avg} = \frac{1}{M} \sum_{m=1}^M C^S(m), \quad (18)$$

where:

- $M$ : the number of the latest past bandwidth samples.
- $C^S(m)$ : the average bandwidth of sample  $m$ -th ( $1 \leq m \leq M$ ).

The smooth bandwidth  $C_{smooth}(m)$  is calculated as a sum of its previous value and the newest bandwidth sample with a smooth weight parameter  $\eta$ :

$$C_{smooth}(m) = \eta \times C_{smooth}(m-1) + (1-\eta) \times C^S(m), \quad (19)$$

and its initial value is set as  $C_{smooth}(m) = C^S(m)$  with  $m = 1$ .

Given the smooth bandwidth across the latest  $M$  bandwidth samples, the estimation of future bandwidth is determined as the smooth bandwidth of the  $M$ -th sample, as shown below:

$$C_{future} = C_{smooth}(M). \quad (20)$$

In the following step, our proposed method employs future bandwidth prediction to estimate the download time of the next chunk, thereby establishing a dynamic buffer threshold.

##### B. ADAPTIVE PRELOADING

Our algorithm dynamically adjusts the amount of preloaded data (i.e., buffer size) for each video based on cross-user behavior. In particular, our algorithm utilizes chunk-level user retention rate, which is defined as the percentage of users who watch until chunk  $k$  of video  $i$ , denoted  $RET_k^i$ . The user retention rate can be estimated by collecting viewing information of past users and be distributed to the user agent at the beginning of a streaming session. When the user retention rate is high, the user will likely continue watching the current video, thus a high buffer level can be maintained for the video. When the user retention rate is low, there is a high probability that the user will switch to the next video in the playlist. Thus, it is better to maintain a low buffer occupancy to minimize data wastage.

We can calculate the user's probability of staying until chunk  $k$ -th ( $k > k^c$ ) of video  $i$  from the current playing chunk  $k^c$  as follows:

$$p_{st}^i(k) = \frac{RET_k^i}{RET_{k^c}^i}. \quad (21)$$

The probability of leaving the current video, denoted by  $p_{lea}^i(k)$ , is then:

$$p_{lea}^i(k) = 1 - p_{st}^i(k). \quad (22)$$

If the video  $i$  has not been played, then  $RET_{k^c}^i = 1$  and  $k^c = 0$ .

Next, we estimate the maximum download time  $d_{max}^i(k)$  of the next chunk  $k$  of video  $i$  based on the maximal next chunks' data size and the future bandwidth as follows:

$$d_{max}^i(k) = \frac{\max(S(R_k^i), S(R_{k+1}^i), \dots, S(R_{k+(\mu-1)}^i))}{C_{future}}. \quad (23)$$

Here,  $\mu$  ( $\mu \geq 1$ ) denotes the number of the next chunks ahead that we consider. This parameter is also referred to as a moving horizon utilized for MPC, and its details can be found in Section IV-C.

Similarly, the minimum download time of chunk  $k$  of video  $i$ , denoted by  $d_{min}^i(k)$ , can be estimated as follows:

$$d_{min}^i(k) = \frac{\min(S(R_k^i), S(R_{k+1}^i), \dots, S(R_{k+(\mu-1)}^i))}{C_{avg}}. \quad (24)$$

Our proposed method presents a dynamic buffer threshold for the current and the next video based on the information on the probability of staying and the download time of the next chunk as follows.

$$B_{th}^{ic}(k) = \begin{cases} p_{st}^{ic}(k)d_{max}^{ic}(k) + B_{th}^{i'}(k') + \tau, & d_{min}^{ic}(k) < \tau \\ p_{st}^{ic}(k)d_{max}^{ic}(k), & d_{min}^{ic}(k) \geq \tau, \end{cases} \quad (25)$$

$$B_{th}^{i'}(k') = p_{st}^{i'}(k') \times d_{max}^{i'}(k'). \quad (26)$$

Here,  $B_{th}^{ic}(k)$  and  $B_{th}^{i'}(k)$  denote the dynamic buffer threshold for the current and the next video, respectively. We also set an upper bound  $B_{th}^{iup}$  and a lower bound buffer threshold  $B_{th}^{ilow}$  to prevent the dynamic buffer threshold  $B_{th}^i(k)$  becomes too high or too low as follows:

$$B_{th}^{iup} = 4\tau, \quad (27)$$

$$B_{th}^{ilow} = \tau + \sigma, \quad (28)$$

where:

- $\tau$ : the playback duration of each video chunk.
- $\sigma$ : the sleep time for the downloader.

If the buffer occupancy of the current and all the next videos (the next videos that are considered to preload) exceeds the buffer threshold, we will initiate a sleep mode for the downloader lasting  $\sigma$  seconds, aimed at reducing potential data wastage resulting from preloading extra chunks. By incorporating upper bound and lower bound buffer thresholds, along with a dynamic buffer threshold, our proposed method effectively minimizes data wastage and reduces re-buffering events.

### C. BITRATE ADAPTATION

#### 1) MODEL PREDICTIVE CONTROL (MPC)

In an ideal scenario, if we had accurate information about future network throughput for the entire duration of user viewing, we could calculate the optimal bitrate level for every video's chunks and maximize QoE in a single optimization process. However, in reality, obtaining such perfect information is impossible. Nevertheless, it is possible to obtain reasonably accurate throughput predictions for a short-term future chunk horizon  $[k, k + (\mu - 1)]$ . This is because network conditions tend to be relatively stable within a short time interval. Building on this understanding, we employ MPC to decide the bitrate level for the next preloading chunk [19]. The concept behind MPC is to find optimal bitrate selection within a short moving horizon  $\mu$  using the predicted throughput and the chunk level's user retention rate. The optimal bitrate selection is found by searching for the one that maximizes a reward function. The user agent then downloads the next chunk at the selected bitrate. The horizon is then forwarded to  $[k+1, k+\mu]$  and the process is repeated for every video chunk. In video streaming, it is important to note that the reward function is usually taken into account information from past downloaded chunks.

#### 2) REWARD FUNCTION

To implement MPC in short video streaming, we define our reward function based on the QoE-Wastage trade-off as described in Eq.(17). In this paper, the expected reward of downloading chunk  $k$  of video  $i$  at bitrate  $R_k^i$  is defined as follows:

$$Reward(R_k^i) = \Phi(R_k^i) - \Psi(R_k^i) - \Gamma(R_k^i) - \Theta(R_k^i). \quad (29)$$

The reward is a weighted sum of four factors. The first factor of the reward, i.e.,  $\Phi(R_k^i)$ , is the bitrate quality of downloading chunk  $k$  of video  $i$ . It should be noted that higher bitrate quality directly corresponds to an enhanced QoE for the user. The bitrate quality factor is calculated by the bitrate  $R_k^i$  and the weight parameter  $w_1$ :

$$\Phi(R_k^i) = w_1 \times R_k^i. \quad (30)$$

The second factor, denoted by  $\Psi(R_k^i)$ , is the bitrate variation of downloading chunk  $k$  of video  $i$  with bitrate  $R_k^i$ . Abrupt changes in bitrate quality between chunks lead to a decrease in the user's QoE. It depends on the weight parameter  $w_3$  and the difference with the bitrate level of the previous chunk:

$$\Psi(R_k^i) = w_3 \times |R_k^i - R_{k-1}^i|. \quad (31)$$

When the downloader makes the decision to download a chunk, the process must continue until the download is successfully completed. Let  $\varphi$  be the number of chunks which is expected to play during the process of downloading chunk  $k$  of video  $i$ :

$$\varphi = \frac{d_{max}^i(k)}{\tau}. \quad (32)$$

The third factor of the reward is the expected re-buffering time. A re-buffering event can occur if the download time of chunk  $k$  at video  $i$  exceeds the buffer occupancy of the current video. Furthermore, the user might switch to the next videos while downloading and cause a re-buffering event on that video if it is not preloaded enough. As a result, the expected re-buffering time can be determined as follows:

$$\Gamma(R_k^i) = w_2 \times \left( p_{st}^{ic}(k^c + \varphi) \times (d_{max}^i(k) - B_{k_i}^{ic})_+ + p_{lea}^{ic}(k^c + \varphi) \times p_{st}^{ic+1}(\varphi) \times (d_{max}^i(k) - B_{k_i}^{ic+1})_+ \right). \quad (33)$$

Here,  $B_{k_i}^{ic}$  is the buffer occupancy of the current video when starting to download chunk  $k$  of video  $i$ . The probability  $p_{st}^{ic}(k^c + \varphi)$ ,  $p_{lea}^{ic}(k^c + \varphi)$ , and  $p_{st}^{ic+1}(\varphi)$  are calculated using Eq.(21)(22). In practice, during the downloading process, users may switch between multiple videos. However, in our proposed algorithm, we assume that users have the probability of switching only one to the next video during the downloading process. It is important to note that if a user decides to switch to the next video while the downloader is preloading a chunk of the current video, all of the data downloaded for that particular chunk will be considered data wastage. The fourth factor of the reward is the amount of

**Algorithm 1** Preloading and Bitrate Adaptation

**Data:** The latest M bandwidth samples  $C^S(m)$ ; buffer occupancy  $B_k^i$ ; User retention rate  $RET_k^i$ ; the next  $\mu$  chunks' data size  $S(R_k^i)$ ; and the current playing chunk  $k^c$  at video  $i^c$ .

**Result:** The optimal chunk's bitrate  $R_{opt}$  to download or go to sleep for  $\sigma$  seconds.

- 1 Calculate future bandwidth  $C_{future}$  using Eq. (18)~(20).
- 2 Calculate buffer threshold  $B_{th}^i(k)$  using Eq. (21)~(28).
- 3 Initialize  $Reward_{max} \leftarrow -\infty$ .
- 4 **for**  $i \leftarrow i^c$  **to**  $i^c + \gamma$  **do**
- 5     **if**  $B_k^i \leq B_{th}^i(k)$  **then**
- 6         **if**  $i == i^c$  **then**
- 7             Run MPC with the moving horizon  $\mu = 5$ .
- 8         **else**
- 9             Run MPC with the moving horizon  $\mu = 2$ .
- 10         **end**
- 11         Save the bitrate  $R_k^i$  corresponding to the score  $Reward(R_k^i)$ .
- 12         **if**  $Reward(R_k^i) \geq Reward_{max}$  **then**
- 13              $Reward_{max} \leftarrow Reward(R_k^i)$ ,
- 14              $R_{opt} \leftarrow R_k^i$ .
- 15         **end**
- 16         **Break.**
- 17     **end**
- 18 **end**
- 19 **if**  $Reward_{max} \neq -\infty$  **then**
- 20     **Return**  $R_{opt}$  to download.
- 21 **else**
- 22     **Sleep** for  $\sigma$  seconds.
- 23 **end**

data wastage incurred by switch events as follows:

$$\Theta(R_k^i) = \begin{cases} p_{lea}^i(k^c + \varphi) \times S(R_k^i), & R_k^i = R_k^c \\ 0, & R_k^i \neq R_k^c \end{cases} \quad (34)$$

While the quality bitrate has positive effects on the rewards, the bitrate variations, re-buffering, and data wastage factors have negative effects on the rewards. The objective of the proposed algorithm is to decide the bitrates of video chunks to maximize the total reward.

**D. PRELOADING AND BITRATE ADAPTATION ALGORITHM**

Our proposed algorithm, described in Algorithm 1, takes inspiration from the PDAS study [9]. Both our proposed algorithm and PDAS share a fundamental strategy for managing short video streaming. The initial step involves establishing a buffer threshold, followed by determining the optimal bitrate level for the next chunk to download. However, the difference between our method and the PDAS method lies in details. In the initial step, a buffer threshold is computed using the predictions of future bandwidth with

**TABLE 2.** Video characteristics [16].

Video	Time(s)	Video Type
1	17	Study
2	26	Entertainment
3	37	Life
4	40	Life
5	47	Life
6	6	Entertainment
7	125	Game

the data of user retention rate, and size of the next chunks. In the subsequent step, MPC is utilized with a varying moving horizon for the current and next videos, accompanied by a reward function that considers factors such as quality bitrate, bitrate variation, re-buffering time, and data wastage. By integrating these components, our proposed method can outperform PDAS in terms of QoE and data wastage.

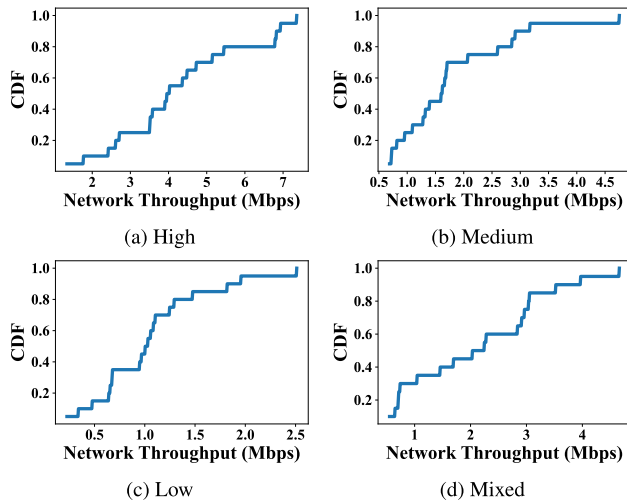
Algorithm 1 runs iteratively each time a chunk download has been completed. Its detailed operation can be elaborated as follows: Firstly, the algorithm calculates the future bandwidth  $C_{future}$  and the buffer threshold  $B_{th}^i(k)$ . Next, the algorithm checks the buffer level across videos from the current one to the subsequent  $\gamma$  videos. If a video has the current buffer occupancy  $B_k^i$  less than or equal to the buffer threshold  $B_{th}^i(k)$ , the proposed algorithm runs MPC with the moving horizon  $\mu$  to determine the bitrate level  $R_k^i$  and the corresponding reward  $Reward(R_k^i)$  (i.e., lines 6-10). In other words, within the range of the current video to the next  $\gamma$  videos, our algorithm selects the first video that meets the condition of having a buffer occupancy below the buffer threshold to run MPC. It is important to note that different moving horizons of the MPC are used for the current video and the next videos. In particular, we set the moving horizon of the current video to 5 and of the next videos to 2. Subsequent to the MPC phase, the algorithm selects the bitrate level with the highest reward as the optimal choice for downloading the next chunk (i.e., lines 12-15), and terminates the examination (line 16). If the buffer occupancy of the current and all the next  $\gamma$  videos exceed the buffer threshold, we will put the downloader to sleep for  $\sigma$  seconds (line 22).

**V. EVALUATION**

**A. EXPERIMENTAL SETTINGS**

For the experiment, we implement and evaluate the proposed method under different network conditions and user behavior using a simulator, and datasets provided by [16] and [20]. In each particular scenario, we compute the performance metrics for both QoE and Data Wastage. Subsequently, we calculate and present the average results across all cases in Figure 5, Figure 6, and Figure 7 in the following subsection. Related studies, including [9], [13], and [15], have also used the same evaluation approach to assess the performance of their proposed methods. This approach has allowed us to





**FIGURE 3.** CDFs of throughput of four network trace types: (a) High, (b) Medium, (c) Low, (d) Mixed.

draw valuable insights and demonstrate the effectiveness of our proposed method.

The dataset consists of seven short videos with different content characteristics as described in Table 2. The dataset also contains network traces and user retention rate information. The network traces are categorized into four types, namely high, medium, low, and mixed. There are twenty network traces for each type, each trace lasts for 2900 seconds. The network throughput is updated every 500ms to simulate real network conditions as accurately as possible. The CDF of four network throughput types is shown in Figure 3. Each video is encoded into three bitrate levels in Variable Bitrate (VBR) mode, and the videos are divided into multiple chunks with a duration of one second each ( $\tau = 1$ ). The user retention rate data provides information about the probability of users switching to the next video at different points in time during video playback. For instance, consider the following retention rate data: [(0, 1), (1, 0.95), (2, 0.81), (3, 0.74), (4, 0.69), (5, 0.57), (6, 0.43), (7, 0)], which represents the user retention rate of a 6-second video. This data indicates that at the end of the first second, 95% of users are still watching, and only 43% of users watch the entire video [16]. In addition, the simulator supports five video players, therefore, it allows the user agent to preload chunks of the current and the next four videos in the playlist. Therefore, in our algorithm, the parameter  $\gamma$  is set to 4 (line 4 in Algorithm 1).

In the bandwidth prediction step, the parameter settings for achieving accurate results are crucial. Specifically, we set the number of past bandwidth samples, denoted as  $M$  in Eq.(18)~(20), to 15, while configuring the smooth weight parameter, represented as  $\eta$  in Eq.(19), to a value of 0.8. It is noteworthy that increasing the value of  $M$  can cause inaccuracies in predictions, attributed to the inclusion of outdated samples. Conversely, a smaller value could result in an insufficient representation of the network's past behavior.

For  $\eta$ , opting for a lower value prioritizes the influence of recent samples in bandwidth prediction, while still considering past trends. This balanced selection effectively captures both past and recent bandwidth trends, enhancing the precision of bandwidth predictions. Additionally, our proposed method includes the sleep time  $\sigma$  for the downloader, which is set to 0.5 seconds. This strategy lets the downloader temporarily pause when no chunk downloads are necessary at a particular step, preventing data wastage from redundant preloaded chunks and reducing computational efforts during inactivity. The impacts of these parameters on the performance of the proposed method are investigated in Section V-B.

In this study, we compare our proposed method with five existing methods. The reference methods used in our evaluation are as follows:

- 1) **Next-One**: This method preloads only the current and the next video, without considering bitrate adaptation. The next video will be preloaded once the preloading of the current video is completed. This method is applied by the Douyin app [11].
- 2) **Network-Based**: This is the proposed method in [8]. The user agent is allowed to preload chunks of the current and the next  $K$  videos. However, only  $B$  chunks ahead can be preloaded. The parameters  $B$  and  $K$  are calculated based on the past network throughput. This method also does not encompass bitrate adaptation.
- 3) **PDAS**: This method is rank first at MMGC2022 Short Video Streaming which is presented in [9]. The authors proposed a dynamic maximum buffer size scheme that utilizes the estimated harmonic mean network throughput and the retention probability of the user viewing each video. If the buffer occupancy reaches the buffer size threshold for all videos, the user agent will sleep for 50 milliseconds. RobustMPC is used to decide bitrate levels for each chunk.
- 4) **Fixed-Preload**: This method is a baseline algorithm that is provided by MMGC2022 [16]. The downloader only downloads the next video when the current video is downloaded completely. The next videos are preloaded up to 4 chunks ahead which depends on the user retention rate. After that, the bitrate level is decided by the buffer occupancy of the video.
- 5) **No-Save**: This is also a default baseline of MMGC2022. It is quite similar to the Fixed-Preload method except that the next videos have a preloaded threshold of 800KBytes. RobustMPC [19] is also used to calculate the optimal bitrate level.

It should be noted that for those methods where bitrate adaptation is not considered, we use the highest quality level for all video chunks. To evaluate the performance of each method, 50 different traces of user viewing behaviors are generated based on user retention rate data. The CDF of the user retention time for each video in the dataset is shown in Figure 4. Consequently, with 80 network traces and

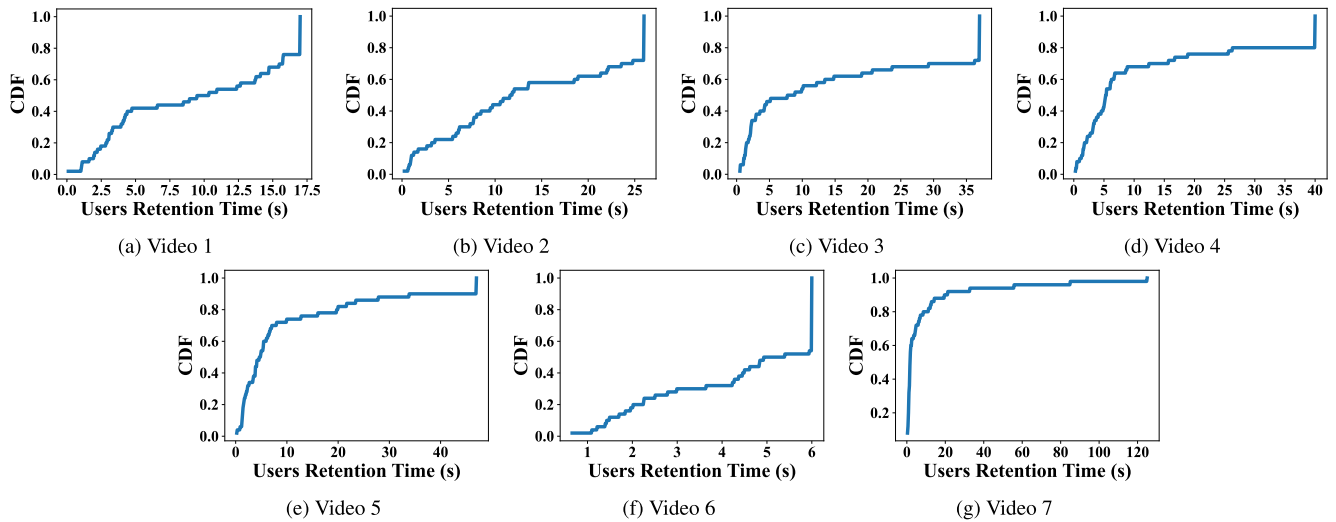


FIGURE 4. CDFs of users retention time for each video used in the evaluation.

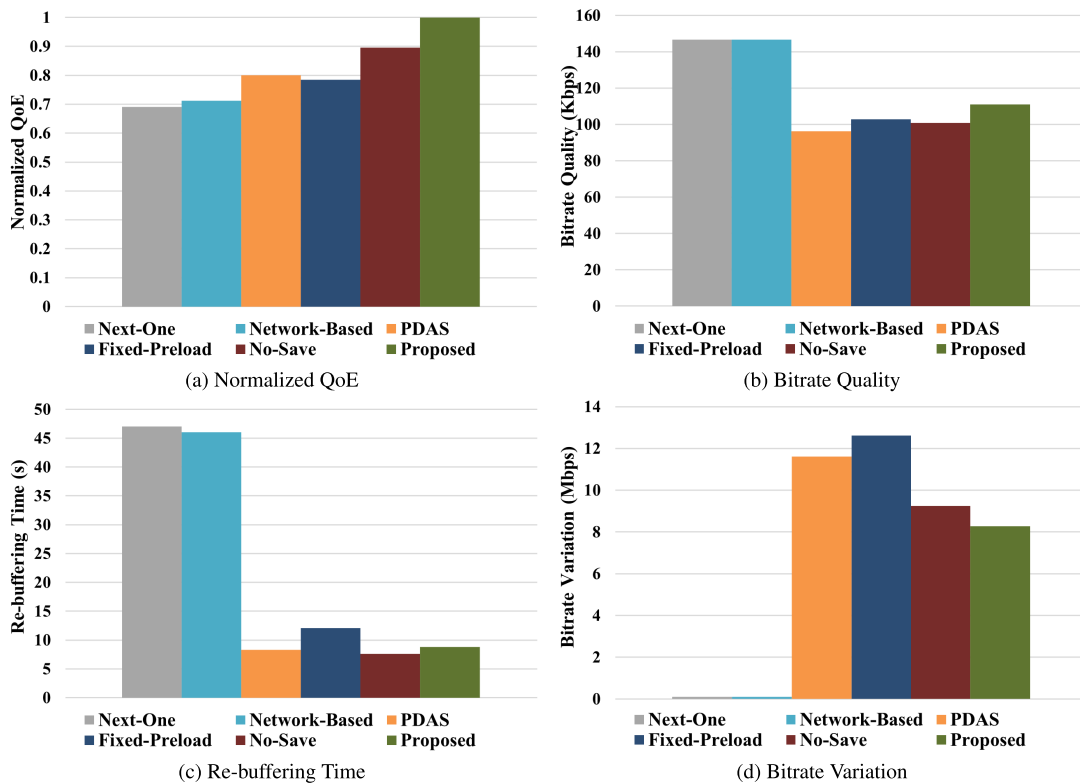


FIGURE 5. Average QoE and its components of the proposed method and the reference methods.

50 user traces, our experimental results ensure fairness in the performance comparison between the methods.

## B. EXPERIMENTAL RESULTS

### 1) QOE-WASTAGE TRADE-OFF EVALUATION

Figure 5 and Figure 6 present the average QoE and the average data wastage results for our proposed method

and five reference methods over 50 user traces under four network conditions. Overall, our method demonstrates a significant reduction in data wastage compared to the reference methods, while achieving the highest QoE score. Furthermore, our proposed method consistently outperforms the reference methods across all network scenarios, including High, Medium, Low, and Mixed network conditions.

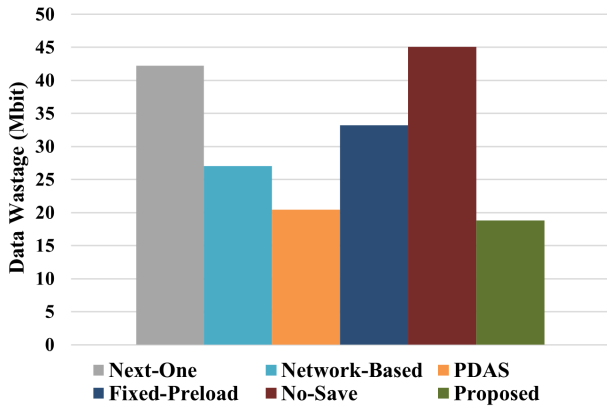


FIGURE 6. Average data wastage of the proposed method and the reference methods.

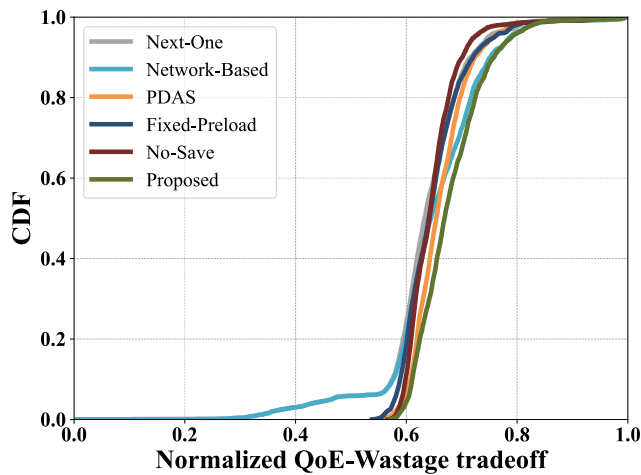


FIGURE 7. CDF normalized QoE-Wastage trade-off of the proposed method and the reference methods across all scenarios.

Figure 5 presents the average QoE value along with its components, namely bitrate quality, re-buffering time, and bitrate variation. In scenarios characterized by low and fluctuating network throughput, bitrate adaptation plays a crucial role. Despite the Next-One and Network-based methods exhibiting the highest bitrate quality and no bitrate variation (constant bitrate level), they demonstrate significantly low QoE values due to a high amount of re-buffering time, resulting in a poor user experience. On the other hand, our proposed method achieves higher bitrate quality and lower bitrate variation compared to the remaining methods, albeit with slightly increased re-buffering time. Overall, our method presents a noticeable enhancement in the QoE score, showing a 25% increase compared to the PDAS method. Furthermore, it significantly outperforms the other remaining methods, improving QoE scores by 12% compared to No-Save, and 27% compared to Fixed-Preload. Our method also reaches a more satisfactory QoE than the Next-One and Network-Based methods by 45% and 40%, respectively.

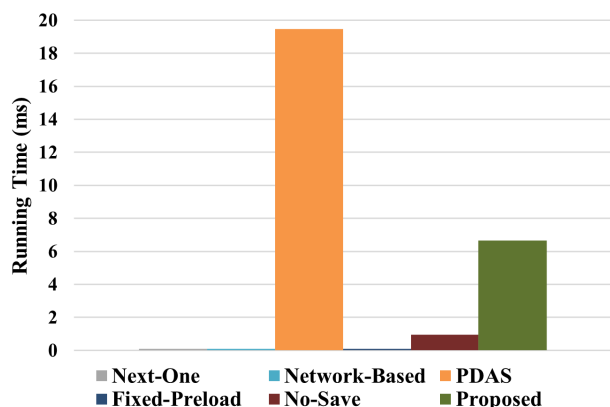
Regarding the data wastage, Figure 6 demonstrates the comparison of our proposed method to the reference methods.

TABLE 3. Effects of four parameters on QoE-Wastage trade-off of the proposed method.

(a) Number of bandwidth samples ( $M$ )					
$M$	5	10	15	20	25
QoE-Wastage	66.09	67.48	67.66	67.58	67.62
(b) Weight parameter ( $\eta$ )					
$\eta$	0.9	0.8	0.6	0.4	0.2
QoE-Wastage	66.91	67.66	66.94	65.41	63.72
(c) Moving horizon ( $\mu$ )					
$\mu$	1	2	3	4	5
QoE-Wastage	68.18	67.66	66.98	66.57	66.32
(d) Sleep time ( $\sigma$ )					
$\sigma$	0.1	0.25	0.5	0.75	1
QoE-Wastage	67.66	67.65	67.66	67.68	67.05

It can be seen clearly that the proposed method achieves the lowest amount of data wastage. Following close by is the PDAS method, which causes approximately 9% higher data wastage than our method. Moreover, Network-Based and Fixed-Preload engender even more data wastage, 44~76% more than our method. Compared to the No-Save and the Next-One method, the proposed method can reduce data wastage by 55% and 58%, respectively. Figure 7 illustrates the Cumulative Distribution Function (CDF) of QoE-Wastage trade-off across all scenarios, as it is observed, our method consistently outperforms all reference methods, providing a noticeable improvement in both QoE and data wastage reduction.

Next, we investigate the impacts of the key parameters on the performance of the proposed method. In particular, we consider four parameters: 1) the number of bandwidth samples  $M$  that are required in Eq. (18)~(20), 2) the smooth weight parameter  $\eta$  in Eq. (19), 3) the moving horizon  $\mu$  of the MPC used for the next videos, and 4) the sleep time  $\sigma$ . These four parameters are pre-determined and have been empirically selected, with their value settings detailed in Section IV-D and Section V-A. To reiterate for clarity, the specific parameter values are as follows:  $M = 15$ ,  $\eta = 0.8$ ,  $\mu = 2$ , and  $\sigma = 0.5$  seconds. By modifying a single parameter value while keeping the others constant, we examine the impact of each parameter on the QoE-Wastage trade-off score. Table 3 shows the QoE-Wastage trade-off of the proposed method at five different values of the key parameters. It can be seen that the performance of the proposed method increases as  $M$  increases from 5 to 10. As the value of  $M$  increases further, the performance of the proposed method is mostly stable. As for the weight parameter  $\eta$ , the highest performance is observed when  $\eta = 0.8$ . In addition, the performance at  $\eta = 0.4$  and  $\eta = 0.2$  are lower than those at  $\eta = \{0.6, 0.8, 0.9\}$ . This result indicates that the weight should not be smaller than 0.5 for good performance. Regarding the moving horizon of the MPC, it can be seen that the shorter the moving horizon is, the



**FIGURE 8.** Average running time of the proposed method and the reference methods.

better the performance would become. Thus, a short moving horizon should be used for the next videos. For the sleep time, it is shown that there is a small difference in the performance of the proposed method across different values of  $\sigma$ . This result implies that the effects of sleep time are small.

## 2) RESOURCE USAGE EVALUATION

In this part, we analyze and evaluate the resource usage of the proposed method in terms of memory usage and time complexity.

### a: MEMORY USAGE

Regarding memory usage, the operation of our proposed method requires the availability of certain data elements, including the most recent bandwidth samples, user retention rate, data size of the following chunks, buffer occupancy, and the current playing chunk. These data components are essential for the computation of the reward function, which is a crucial aspect of our proposed algorithm during the MPC phase, enabling estimation of each future step. The total required memory for the whole process is measured approximately 56 KBytes.

### b: TIME COMPLEXITY

The computational complexity of our algorithm can be evaluated through both analytical and empirical approaches. The complexity primarily originates from the MPC phase. Within this phase of our proposed solution - Algorithm 1, it calculates the reward function for all possible scenarios of future chunks  $[k, k + (\mu - 1)]$  while taking into account a range of bitrate levels  $\mathcal{R}$ . This calculation involves considering each potential download step. As a result, the complexity of our proposed algorithm can be described as  $O(\mu \times \mathcal{R}^\mu)$ .

Next, we carry out experiments to measure the running time of our proposed method and the reference methods. All experiments are conducted on a 64-bit Windows 11 Laptop

equipped with a 2.3GHz AMD Ryzen 7 CPU and 16GB Memory. As illustrated in Figure 8, the results show that our proposed method has an average running time of 6.7 ms, whilst PDAS takes approximately 19.5 ms to execute. Besides, the No-Save method has an average running time of approximately 1ms, and the remaining methods complete their executions in under 0.01 ms. Although our proposed method does not achieve the fastest average running time, it offers a significant improvement in terms of QoE and data wastage. Moreover, the running time of 6.7 ms is negligible when considering its application in the context of a real-time adaptive short video streaming system, where the priority lies in optimizing user experience and minimizing data wastage.

From the above results, it can be concluded that the proposed algorithm is relatively lightweight in both terms of memory usage and time complexity. This characteristic makes it well-suited for implementation on resource-constrained mobile devices and for real-time adaptation.

## VI. CONCLUSION

In this paper, we present an analytical model for short video streaming and formulate the QoE-Wastage trade-off optimization problem as a multi-objective optimization problem. We then propose a simple yet effective chunk preloading and bitrate adaptation algorithm. The proposed method uses user data and network status statistics to decide on a suitable video chunk to download at each step. Experiment results show that the proposed algorithm outperforms the reference methods by approximately 12% to 45% in terms of QoE and reduces data wastage by up to 58%. In addition, the impacts of key parameters on the proposed method are also investigated. Furthermore, our proposed method is suitable for real-time adaptation due to its reasonably low memory usage of 56 Kbytes and running time of 6.7 ms. In future work, we will investigate advanced adaptation techniques such as layered video coding to further improve the system's performance. Moreover, modeling users' perception of short video streaming is an important research topic and will be addressed in our future work. Also, we will develop mathematical models for predicting the performance of the proposed method.

## REFERENCES

- [1] (Jun. 2022). *YouTube Press*. [Online]. Available: <https://blog.youtube/press/>
- [2] (Jun. 2022). *Short Video Platforms Market Report Global Forecast From 2022 to 2030*. [Online]. Available: <https://dataintelo.com/report/short-video-platforms-market/>
- [3] S. Zhu, T. Karagioules, E. Halepovic, A. Mohammed, and A. D. Striegel, "Swipe along: A measurement study of short video services," in *Proc. 13th ACM Multimedia Syst. Conf.* New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 123–135, doi: 10.1145/3524273.3528186.
- [4] H. T. Le, D. V. Nguyen, N. P. Ngoc, A. T. Pham, and T. C. Thang, "Buffer-based bitrate adaptation for adaptive HTTP streaming," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2013, pp. 33–38.



- [5] H. T. T. Tran, N. P. Ngoc, A. T. Pham, and T. C. Thang, "A multi-factor QoE model for adaptive streaming over mobile networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Washington, DC, USA, Dec. 2016, pp. 1–6.
- [6] Y. Zhang, Y. Liu, L. Guo, and J. Y. B. Lee, "Measurement of a large-scale short-video service over mobile and wireless networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3472–3488, Jun. 2023.
- [7] H. Zhang, Y. Ban, X. Zhang, Z. Guo, Z. Xu, S. Meng, J. Li, and Y. Wang, "APL: Adaptive preloading of short video with Lyapunov optimization," in *Proc. IEEE Int. Conf. Vis. Commun. Image Process. (VCIP)*, Dec. 2020, pp. 13–16.
- [8] D. Nguyen, P. Nguyen, V. Long, T. T. Huong, and P. N. Nam, "Network-aware prefetching method for short-form video streaming," in *Proc. IEEE 24th Int. Workshop Multimedia Signal Process. (MMSP)*, Sep. 2022, pp. 1–5.
- [9] C. Zhou, Y. Ban, Y. Zhao, L. Guo, and B. Yu, "PDAS: Probability-driven adaptive streaming for short video," in *Proc. 30th ACM Int. Conf. Multimedia*, Lisboa, Portugal, Oct. 2022, pp. 7021–7025.
- [10] G. Zhang, K. Liu, H. Hu, and J. Guo, "Short video streaming with data wastage awareness," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Shenzhen, China, Jul. 2021, pp. 1–6.
- [11] J. He, M. Hu, Y. Zhou, and D. Wu, "LiveClip: Towards intelligent mobile short-form video streaming with deep reinforcement learning," in *Proc. 30th ACM Workshop Netw. Operating Syst. Support Digit. Audio Video*, Istanbul, Turkey, Jun. 2020, pp. 54–59.
- [12] G. Zhang, J. Zhang, K. Liu, J. Guo, J. Y. B. Lee, H. Hu, and V. Aggarwal, "DUASVS: A mobile data saving strategy in short-form video streaming," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1066–1078, Mar. 2023.
- [13] S.-Z. Qian, Y. Xie, Z. Pan, Y. Zhang, and T. Lin, "DAM: Deep reinforcement learning based preload algorithm with action masking for short video streaming," in *Proc. 30th ACM Int. Conf. Multimedia*, Lisboa, Portugal, Oct. 2022, pp. 7030–7034.
- [14] J. Guo and G. Zhang, "A video-quality driven strategy in short video streaming," in *Proc. 24th Int. ACM Conf. Model., Anal. Simul. Wireless Mobile Syst.*, Alicante, Spain, Nov. 2021, pp. 221–228.
- [15] Y. Li, Q. Zheng, Z. Zhang, H. Chen, and Z. Ma, "Improving ABR performance for short video streaming using multi-agent reinforcement learning with expert guidance," in *Proc. 33rd Workshop Netw. Operating Syst. Support Digit. Audio Video*, Vancouver, BC, Canada, Jun. 2023, pp. 58–64.
- [16] X. Zuo, Y. Li, M. Xu, W. T. Ooi, J. Liu, J. Jiang, X. Zhang, K. Zheng, and Y. Cui, "Bandwidth-efficient multi-video prefetching for short video streaming," in *Proc. 30th ACM Int. Conf. Multimedia*, New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 7084–7088, doi: [10.1145/3503161.3551584](https://doi.org/10.1145/3503161.3551584).
- [17] Z. Li, Y. Xie, R. Netravali, and K. Jamieson, "Dashlet: Taming swipe uncertainty for robust short video streaming," in *Proc. 20th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Boston, MA, USA: USENIX Association, Apr. 2023, pp. 1583–1599. [Online]. Available: <https://www.usenix.org/conference/nsdi23/presentation/li-zhuqi>
- [18] G. Mavrotas, "Effective implementation of the  $\epsilon$ -constraint method in multi-objective mathematical programming problems," *Appl. Math. Comput.*, vol. 213, no. 2, pp. 455–465, Jul. 2009.
- [19] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 325–338, Sep. 2015.
- [20] (Mar. 2022). *Acm Multimedia 2022 Grand Challenge Official GitHub*. [Online]. Available: <https://github.com/AItransCompetition/Short-Video-Streaming-Challenge>



**TRUONG THU HUONG** (Member, IEEE) received the B.Sc. degree in electronics and telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2001, the M.Sc. degree in information and communication systems from the Hamburg University of Technology, Germany, in 2004, and the Ph.D. degree in telecommunications from the University of Trento, Italy, in 2007. Her research interests include oriented toward network security, artificial intelligence, traffic engineering in next-generation networks, QoS/QoE guarantee for network services, green networking, and development of the Internet of Things ecosystems and applications.



**PHAM NGOC NAM** (Member, IEEE) received the bachelor's degree in electronics engineering from the Hanoi University of Science and Technology (HUST), in 1997, and the M.S. degree in artificial intelligence and the Ph.D. degree in electrical engineering from KU Leuven, Belgium, in 1999 and 2004, respectively. He is currently the Vice Dean of the College of Engineering and Computer Science, VinUniversity, Vingroup. He is also a Visiting Scholar with Cornell University.

He has been the PI of one key national project and three ministerial-level projects. He is the author or coauthor of 100 scientific articles, including more than 30 ISI and Scopus publications. His research interests include artificial intelligence, QoS/QoE management for multimedia applications, reconfigurable computing, and low-power embedded system design. He has been a key member of four other national projects.



**TRUONG CONG THANG** (Senior Member, IEEE) received the B.E. degree from the Hanoi University of Science and Technology, Vietnam, in 1997, and the Ph.D. degree from KAIST, South Korea, in 2006. From 1997 to 2000, he was a Network Engineer with the Vietnam Posts and Telecommunications Group (VNPT). From 2007 to 2011, he was a member of Research Staff with the Electronics and Telecommunications Research Institute (ETRI), South Korea.

Since 2011, he has been an Associate Professor with The University of Aizu, Japan. His research interests include multimedia networking, image/video processing, content adaptation, IPTV, and MPEG/ITU standards. He was an active member of Korean and Japanese delegations to standard meetings of ISO/IEC and ITU-T, from 2002 to 2014.



**DUC NGUYEN** (Member, IEEE) received the M.E. and Ph.D. degrees in computer science and engineering from The University of Aizu, Japan, in 2016 and 2019, respectively. From 2019 to 2021, he was a Researcher with KDDI Research Inc., Japan. Since 2021, he has been a Lecturer with the Tohoku Institute of Technology, Sendai, Japan. His research interests include multimedia communications, video streaming, multimedia quality assessment, virtual reality/augmented reality, data

mining, and recommendation systems.

...



**NGUYEN TIEN PHONG** is currently pursuing the bachelor's degree with the Hanoi University of Science and Technology. He is also a Research Assistant with the Future Internet Laboratory, School of Electronics and Telecommunications. His research interests include video streaming and multimedia processing.