



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공학석사학위논문

세그먼트 교체 기법을 활용한 심층
강화학습 기반의 ABR 알고리즘

Deep Reinforcement Learning Based
ABR Algorithms
Using Segment Replacement Technique

2021년 2월

서울대학교 대학원

컴퓨터공학부

배형호

세그먼트 교체 기법을 활용한 심층 강화학습 기반의 ABR 알고리즘

Deep Reinforcement Learning Based
ABR Algorithms
Using Segment Replacement Technique

지도교수 김 종 권

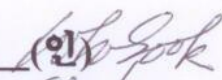
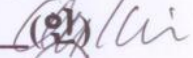

이 논문을 공학석사 학위논문으로 제출함

2020년 10월

서울대학교 대학원
컴퓨터공학부
배 형 호

배형호의 공학석사 학위논문을 인준함

2020년 12월

위원장 : 전 화 숙 (인) 
부위원장 : 김 종 권 (인) 
위원 : 권 태 경 (인) 

Abstract

Deep Reinforcement Learning Based ABR Algorithm Using Segment Replacement Technique

Hyeongho Bae

Department of Computer Science & Engineering

The Graduate School

Seoul National University

Adaptive bitrate (ABR) algorithm is one of the representative techniques used to optimize the playback quality of online video services, namely Quality of Experience (QoE). So far, ABR algorithms based on various optimization techniques have optimized QoE. However, most of the ABR algorithms proposed to date have common limitations; the range of options for optimization. Currently, most ABR algorithms only determine the bit rate of the next segment for QoE optimization. This type of ABR algorithm can optimize

the bit rate of a segment to be downloaded in the future in a dynamic network environment. However, it is not possible to optimize any segment previously downloaded, so the changed network environment cannot be utilized to the maximum.

To overcome this limitation, we propose LAWS, learning based ABR algorithm with segment replacement. LAWS can be replaced with a better bit rate, even for previously downloaded segments, in conditions such as an improved network environment. First for this, we design a novel form of reward for optimization, including segment replacement. Through this, QoE, the optimization objective of the ABR algorithm, can be optimized in the form of segment replacement. In addition, we propose a rule-based learning method to solve the challenges arising in the model learning process. We finally propose an ABR algorithm with segment replacement based on deep reinforcement learning. Experiments based on network traces show that the newly proposed technique has a QoE improvement of 13.1% compared to the existing ABR techniques.

Keywords:

video streaming, adaptive bitrate algorithm, deep reinforcement learning, rate control, segment replacement, DASH, optimization

Student number: 2019-20993

Contents

Abstract	i
Contents	iii
List of Figures	v
List of Tables	vi
Chapter I Introduction	1
Chapter II Related Work.....	4
2.1 DASH.....	4
2.2 Adaptive BitRate Algorithm.....	6
Chapter III Motivation and Approach	9
3.1 Motivation.....	9
3.2 Approach	11
Chapter IV NeuralABR algorithm with Segment Replacement	13
4.1 Action.....	15
4.2 State.....	15
4.3 Reward	18
4.4 Rule based learning	26
4.5 Implementation	27
Chapter V Experiments	28
5.1 Experiment Setup	28

5.2 Baselines	29
5.3 Comparison with Existing ABR algorithms	33
5.4 Analyze Replacement Characteristics	35
5.5 Comparison Between Learning Based Algorithms	35
Chapter VI Conclusion.....	37
Bibliography.....	38
국문초록.....	43

List of Figures

Figure 1. DASH end to end operation	5
Figure 2. Profiling the pilot experiment results on selected network traces .	10
Figure 3. Overview of LAWS	14
Figure 4. Comparing LAWS with existing ABR algorithms in three different QoE metrics	31
Figure 5. Comparing LAWS with existing ABR algorithms on the individual components in the general QoE definition	32
Figure 6. Comparing LAWS-single and LAWS in terms of average QoE of each test trace	34
Figure 7. Final played bitrate of Pensieve and LAWS on selected network traces	36

List of Tables

Table 1. Notation25

Table 2. QoE metrics28

Chapter I

Introduction

In recent years, the demand for video services has skyrocketed and the amount of video traffic has grown to account for the majority of total mobile network traffic [1]. According to [2], video content traffic is expected to grow to 80 percent of the total mobile traffic by 2022. To meet the demand for exploding video services, network operators and service providers have been studying various communication techniques to transmit video content [3, 4, 5]. In addition to research on the transmission of video content, studies have been actively conducted to improve video playback quality in the user's network environment [6, 7, 8, 9, 10]. Researches about the impact of video quality on user behavior, such as video interruptions, further highlight the importance of providing high video quality. [11, 12, 13, 14]

ABR algorithm is one of the representative techniques used to improve video playback quality. To date, the proposed ABR algorithms dynamically adjust the bitrate of the video segment to be downloaded using a number of information, including the user's buffer level and network situation [6, 7, 8, 9, 10, 15, 16]. This allows users to download video segments with the quality of video optimized for their network environment. To this end, algorithms based on learning techniques [15, 16], or optimization control techniques such as MPC [6], PID-like [17] have been proposed. However, existing ABR studies

have focused a lot on optimization methods, but options for optimization are limited. A technique such as [7] has been proposed to overcome the existing limitations, but has a limitation in that it is a fixed rule type algorithm rather than an optimization type.

In this study, we propose Learning based ABR algorithms With Segment replacement, namely LAWS, to overcome the limitations of existing ABR algorithms. First, we design a novel type of reward. reflecting segment replacement. We also show that even if we transform problems into extended forms, we can maintain the same purpose of the QoE maximization problem, which is the goal of the ABR algorithm of existing video streaming. As the segment replacement option is added, the complexity of the QoE problem increases. Simply applying the deep reinforcement learning technique to these problems raises many challenges, such as convergence and learning time. We solve this by using a rule-based action limiting technique.

We evaluated the proposed method through network trace-based experiments. In the experiment, we demonstrated that the performance improved up to 13.1% in overall QoE than the existing DRL-based ABR algorithm.

The key contributions of this paper are as follows:

- We propose a novel QoE objective function including the segment replacement option. Through this, options for optimization can be expanded while maintaining the existing objectives.
- We provide a solution to solve the problem of increased complexity in DRL.

It is possible to induce learning of the DRL model through the solution.

- We provide diversified performance evaluation. This provides insight into how segment replacement affects QoE optimization.

This paper is organized as follows. At the very beginning, we introduce the background of ABR streaming and related research (§2). Next, we briefly deal with the motivation of our idea and conduct a simple experiment to support it. Along with this, we briefly mention the approach to solving the proposed idea (§3). Next, we will discuss the detail of our proposed technique (§4). We then evaluate the proposed algorithm in detail (§5). Finally, the conclusion of the paper is described (§6).

Chapter II

Related Work

ABR streaming is a de facto standard being used to optimize QoE [18, 19]. ABR algorithms are designed for different purposes. Even if the ABR algorithm works on the client side, several tasks must be performed on the server side. Section 2.1 addresses how servers and video contents are organized to enable the ABR algorithm to operate. Section 2.2 then outlines the recently proposed representative ABR algorithms, along with the objectives of the ABR algorithm.

2.1 DASH

There are many implementations that implement ABR streaming. Examples include Adobe's HTTP dynamic streaming [31], Apple's HTTP live streaming [32] and Microsoft's smooth streaming [33]. One of the most representative is Dynamic Adaptive Streaming over HTTP (DASH) [21, 22, 23]. DASH is the first ABR streaming technique in which international standardization took place. Currently, many video streaming platforms use DASH, and it is also a standard for research. Figure 1 shows the end to end of DASH. First of all, video content is encoded in multiple bitrates and stored on the server side. At this point, one video is stored divided into short-length (2s to 10s) videos called segments.

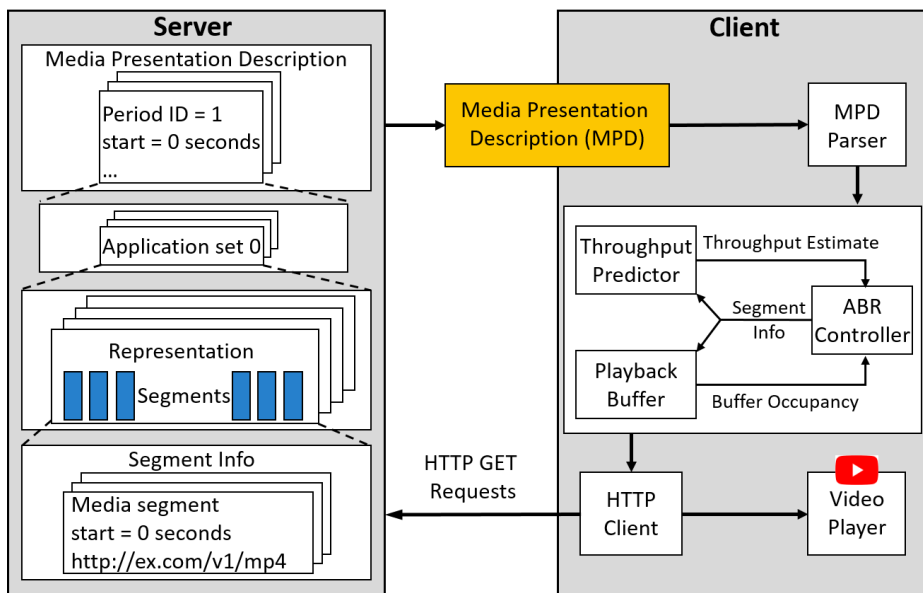


Figure 1. DASH end to end operation

Video content stored in this form provides users with the option to download segments of bitrate optimized according to their network environment, buffer level, etc. In the process of the user starting streaming the video, the user obtains the video's information (segment length, capacity, etc.) through the Media Presentation Description (MPD) file. Subsequently, the segments are downloaded by determining the appropriate bitrate according to the ABR algorithm that operates. This allows users to download segments optimized for their network environments.

2.2 Adaptive BitRate Algorithm

2.2.1 Quality of Experience

Many ABR algorithms are designed with each objective. Although there are various purposes, such as latency, bitrate, buffer level, etc., the most representative object is QoE [18, 19]. QoE is made up of three elements: bitrate, rebuffering time and smoothness (Equation 1). These are conflicting factors that make the QoE optimization problem difficult. Downloading high bitrate segment increases the likelihood of increasing the rebuffering time as it requires a lot of network resources. Also, if bitrate differences from the previous segment are caused by simply downloading bitrate only high, smoothness penalty will increase. Also, if the bitrate of the segment to be downloaded this time is too high, it may be difficult to reduce smoothness with the bitrate of the segment to be downloaded next time. Also, downloading segments in the direction of reducing rebuffering time does not always reduce smoothness penalty, so optimizing these two factors at the same time becomes a difficult

problem.

2.2.2 Adaptive BitRate algorithm

ABR algorithms optimize QoE through their own optimization techniques for each purpose. The ABR algorithm should choose the optimal bitrate for the dynamically changing user's network environment. Generally, ABR algorithms are classified as follows, depending on the information and methods used to determine the bitrate [23].

- Rate-based: ABR algorithms belonging to this classification determine bitrates based on network throughput information [24, 25, 26]. ABR algorithms in this category have the advantage of being able to respond to network changes faster than buffer-based algorithms by directly utilizing network information. However, the network situation is characterized by dynamic change, which has the disadvantage of showing poor performance if the reasoning in the network environment is wrong.
- Buffer-based: ABR algorithms belonging to this classification determine the bitrate of the next segment through the buffer level and segment bitrate information inside the buffer [7, 27, 28]. The logical background of these techniques is the assumption that the network environment can be deduced through the bitrate, segment size, etc. of the downloaded segment. Algorithms in this category are generally simple to operate and have the advantage of being able to use certain information. However, in the beginning stage of video streaming, or if the buffer remains empty due to video jump, there is a limitation that it cannot perform normal operation.

- Hybrid: Each of the above two categories has pros and cons. This led to the emergence of ABR algorithms that combine the two categories to take advantage of both categories [6, 29, 30]. Recently, ABR algorithms using machine learning techniques have emerged to improve performance for QoE optimization [15]. In addition, techniques that can be used in combination with the existing ABR algorithm have been proposed beyond the simple bitrate determination algorithm. A prime example of this is the fast switch [7] technique. After determining the bitrate to download, if there is a segment with a lower bitrate in the buffer, it is replaced with the bitrate you determined.

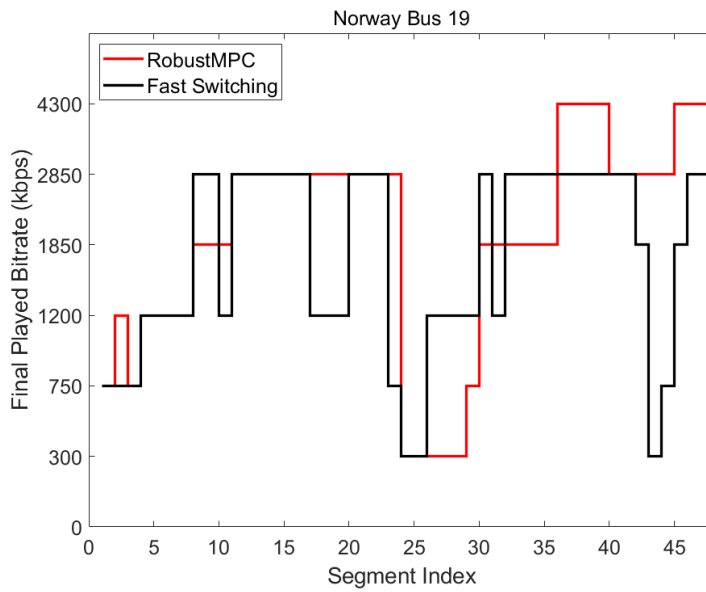
Chapter III

Motivation and Approach

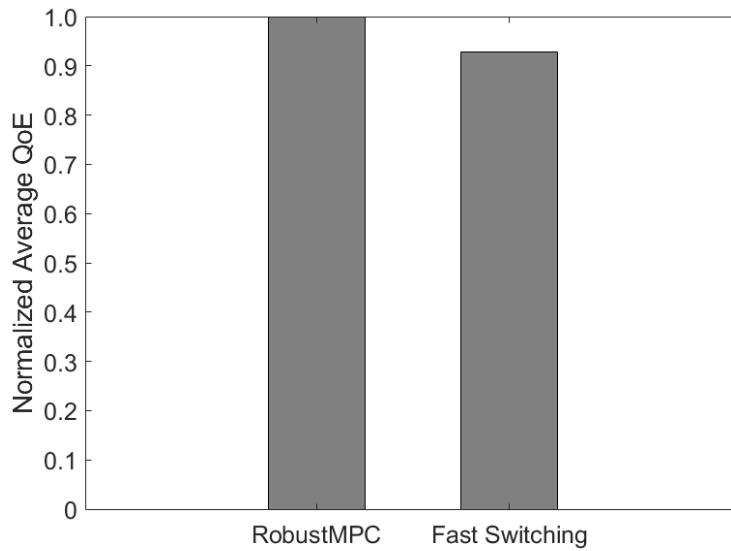
In this section we deal with the motives of our ideas and the approach to the realization of our ideas. First we propose the motivation of the proposed idea. Also, a pilot experiment is performed on selected network traces to provide an experimental basis for motivation. Next, the approach to realizing the proposed idea is briefly discussed.

3.1 Motivation

The final target that the ABR algorithm should optimize is QoE, which consists of a combination of bitrate, rebuffering time and smoothness (Equation 1). QoE is determined by the segment played. Most ABR algorithms optimize QoE by determining only the next bitrate to download. However, in the case of such an optimization, if the network environment changes suddenly, there is a limitation that it is not possible to modify the previous selection. ABR techniques such as [7] provide an option to replace previously downloaded segments (Replace the lower bitrate segment than the bitrate you want to download if it is in the buffer.). However, since it is operated by a fixed rule, not a result of optimization, it can show a result of reducing QoE.



(a) Final played bitrate of RobustMPC and Fast Switching



(b) Normalized Average QoE of RobustMPC and Fast Switching

Figure 2. Profiling the pilot experiment results on selected network traces

Figure 2 shows a pilot experiment conducted to support this claim. As shown in Figure 2, the experimental result performed on the selected network trace suggests that the QoE may decrease by applying the fast switch technique.

To solve these problems, segment replacements should be performed as a result of optimization by designing ABR algorithms, including the replacement option in. Accordingly, we propose a new type of ABR algorithm, LAWS, Learning based ABR algorithm with segment replacement. LAWS perform segment replacement as a result of optimization. This ensures that segment replacement is carried out in a way that maximizes QoE.

3.2 Approach

Most existing algorithms do not replace already downloaded segments with better ones (which increase the QoS). This makes it possible to optimize QoE simply by optimizing the bitrate of the next segment to be downloaded, because the segments downloaded and played are the same.

However, if segment replacement is possible as we suggest in this paper, challenges arise. First, it is necessary to design an objective function that optimizes QoE while including the segment replacement option. Second, it is a problem of finding an optimization technique to be applied to the designed objective function. As the complexity of the problem increases compared to the existing QoE optimization problem, the existing optimization techniques cannot be used due to a time problem or a performance problem.

We solve this challenge by proposing an ABR algorithm based on deep reinforcement learning that includes a segment replacement option in the action.

The proposed ABR algorithm is learned in the form of optimizing QoE while including segment replacement options in the optimization problem through novelly designed rewards. In order to reduce the increased complexity of the QoE optimization problem, an action limiting technique based on logically constructed rules is used.

Chapter IV

Neural ABR algorithm with Segment Replacement

In this section we LAWS, Learning based ABR algorithm With Segment replacement. As mentioned earlier, we design the ABR algorithm, including segment replacement options, by applying deep reinforcement learning techniques. Figure 3 shows the overview of LAWS. We describe the action of LAWS in section 4.1. Unlike the existing DRL based ABR algorithm, segment replacement is included as an action. Sufficient information must be given through the state in order for a deep reinforcement learning model to take proper action. We describe the states in Section 4.2. It also deals with the reasons set as a state through analysis of their characteristics. Section 4.3 describes the newly defined Rewards. This enables LAWS to ultimately optimize existing QoE, even though it includes segment replacement options. Section 4.4 proposes a solution for solving the model convergence problem and the increase in learning time caused by the increased problem complexity. Section 4.5 covers brief parameter settings with the DRL models used.

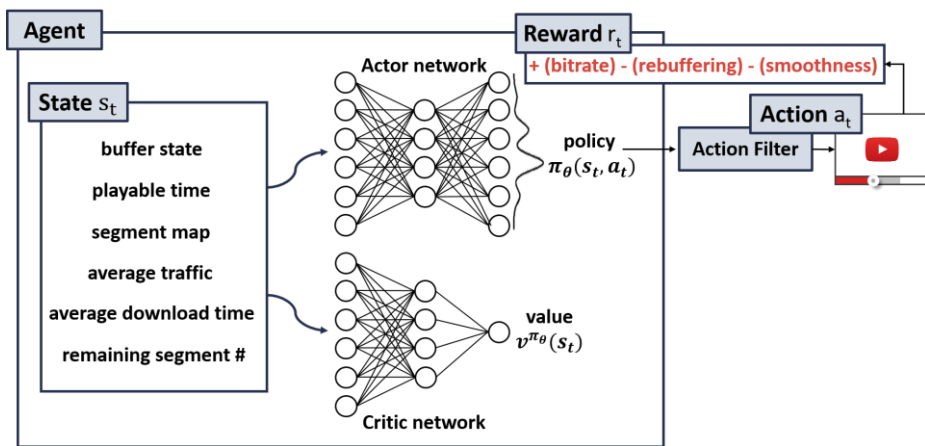


Figure 3. Overview of LAWS

4.1 Action

Action is one of the key elements of the DRL model. The action in the DRL model is consistent with the object of optimization. Existing ABR algorithms decide only the bitrate of the segment to download next (as an action at existing DRL based ABR algorithm) to whether there is a difference in the optimization method. As a result, ABR Algorithm simply has as many choices as the number of bitrates, determines which bitrate it is thought to derive the optimal QoE, and downloads segments. However, as described earlier, this has limitations in dynamically responding to changes in the network environment. We thus define an action in which the option to replace segments in buffers with segments with better bitrate is added. The number of actions is equal to *total # of bitrate quality × buffer size*.

4.2 State

We apply the DRL technique as a method of realizing the proposed idea. Models get information about their current situation through State. Therefore, state, one of the key elements in DRL, is a factor that directly affects the model's reasoning ability. Providing all information that can be known from the client side as a state of the model guarantees the maximum amount of information. However, in our case, setting all information to state leads to the increased state space, and an adverse effect can be generated such as exploration and convergence of the model. In addition, a situation in which an inference relationship with irrelevant information is learned may occur. Thus,

we consider following six states as an essential information for learning our model.

1). Current Buffer State

Because most of existing ABR algorithms do not consider replacement of downloaded segments, these algorithms use buffer level and the quality of last downloaded segment when making bitrate decision. However, LAWS can use segment replacement to improve the quality of previously downloaded segments by replacing them with higher bitrate segments. For this reason, it is impossible to infer the smoothness if only last downloaded bitrate and buffer level are provided as states. It is also important to know the bit rate of the downloaded segment in the buffer so that our model can decide whether to replacement the segment with a low bitrate or download a new segment. To provide the quality of neighboring segment of last downloaded segment and buffer information, we take the buffer itself (e.g., [300, 750, 1350, ..., 4300]) as a state. This allows our model to obtain the information needed to decide action and to infer expected reward.

2). Current buffer level

Providing the status of the buffer provide rough information about the buffer level. However, by the buffer level itself, it is not possible to know how much the first segment in the buffer has been played (Because only bitrate is provided as information in). in order to provide complementary information of this, we provide a playable time in the current buffer state.

3). Segment Map

For simple download schemes (existing ABR algorithm), the decision can be made through the information of the next segment. However, our model can also download segments already in the buffer, so we have to provide a manifest of them. Thus, we use segment map as a state. At this time, providing information on all segments is a great burden on the inference and learning of the model. we provide only segment information in buffer for replacement and information on the future five segments for planning in a state.

4). Average Traffic

Network resources are one of the biggest limitations in video steaming. The buffer based ABR algorithms make rough guesses about the network situation with buffer status alone. However, segment replacement can create the same buffer state in a completely different environment, and vice versa. In addition, providing separate information about the network environment, even if a buffer level is given, allows the model to better respond to the network environment. To solve this problem, we use the average traffic per segment as state.

5). Average Download Time

Average traffic enables inference about the approximation network conditions. However, because the download time is different for each segment, each average shows a difference in the degree of quantization. In order to provide information on this, we provide additional information on how many seconds

each average was made, that is, the segment download time.

6). Number of remaining segments

The number of segments left in the future is one of the important factors that tells the model how much to consider for the future. By providing this information directly, we enable the model to consider how to plan for the future.

4.3 Reward

In DRL, the final objective of the learning agent is to maximize the expected cumulative reward, $E \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$ [34]. Therefore, in ABR algorithms using DRL, QoE per segment is generally used as a reward. In general, QoE metric used by MPC can be defined:

$$QoE = \sum_{n=1}^N q(R_n) - \alpha \sum_{n=1}^N T_n - \beta \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| \quad (1)$$

As a result, the rewards of existing DRL based ABR algorithms such as [15] take the following form. Where N represents the total segment number. R_n , $q(\cdot)$, T_n denotes the bitrate of n -th segment, the utility function, and the rebuffering time when playing the n th segment, respectively. The last term in the above equation denotes the changes in video quality between two continuous segments. α and β penalizes the rebuffering time and the smoothness, respectively.

$$\text{Reward}_n = q(R_n) - \alpha T_n - \beta k_n \quad (2)$$

$$k_n = \begin{cases} 0 & , n = 1 \\ |q(R_{n+1}) - q(R_n)| & , n \neq 1 \end{cases} \quad (3)$$

However, this type of reward cannot be used if replacement action is added to action space. In order to accommodate the final optimization objective with the existing QoE while considering the replacement option in the Rewards, we have designed a new type of reward.

To match the final QoE with the cumulative reward of the replacement model, we redefine a reward of d -th action as below:

$$\text{Reward}_d = q_{\text{our}}(R_d) - \alpha T_d - \beta k_{\text{our}}(R_d) \quad (4)$$

Unlike LAWS, general ABR algorithms that only consider sequential single download action per segment, the segment index is as same as the action index. With the new type of Rewards, LAWS will eventually lead in the form of optimizing QoE while including the replacement option. The basic idea is to eliminate the QoE increment and decrease caused by previous downloads and to reflect the QoE increment and decrease caused by segments of the new bitrate. The notes used to prove this are organized in table 1.

1). Bitrate term, $q_{\text{our}}(R_d)$

Our newly defined quality function is as follows:

$$q_{\text{our}}(R_d) = \begin{cases} q(R_{\text{count}(d),\text{trans}(d)}) - q(R_{\text{count}(d)-1,\text{trans}(d)}), & \text{count}(d)=1 \\ q(R_{\text{count}(d),\text{trans}(d)}) & , \text{count}(d) \neq 1 \end{cases} \quad (5)$$

It is demonstrated that the cumulative sum for $q_{\text{our}}(R_d)$ is equal to the $\sum_{n=1}^N q(R_n)$ of QoE:

$$\begin{aligned}
\sum_{d=1}^D q_{our}(R_d) &= q_{our}(R_1) + q_{our}(R_2) + \dots + q_{our}(R_D) \\
&= q(R_{1,1}) + q(R_{1,2}) + \dots + q(R_{1,total(1)}) + \dots + q(R_{N,1}) \\
&\quad + q(R_{N,total(N)}) \\
&= \sum_{n=1}^N \sum_{re=1}^{total(n)} q(R_{re,n}) \\
&= \sum_{n=1}^N q(R_{1,n}) + (-q(R_{1,n}) + q(R_{2,n})) + \dots \\
&\quad + (-q(R_{total(n)-1,n}) + q(R_{total(n),n})) \\
&= \sum_{n=1}^N q(R_{total(n),n})
\end{aligned} \tag{6}$$

2). Rebuffering Time, T_d

We define T_d as the rebuffering time that occurred during the d th action. This definition allows for the consideration of all the rebuffering times that occurred during video playback due to two assumptions. The first assumption is that the segment must be downloaded in order for it to play. The second is that streaming is finished only after all segments have been downloaded. Because of this, the rebuffering time that occurs while all segments are downloaded is reflected in the reward.

$$\alpha \sum_{n=1}^N T_n = \alpha \sum_{d=1}^D T_d \tag{7}$$

3). Smoothness term, $k_{our}(R_d)$

Where segments are downloaded from existing ABR algorithms is always the last part of the buffer. Therefore, in order for users to calculate smoothness when downloading segments, they only need to calculate smoothness between the segments downloaded this time and the segments downloaded just before. But our algorithm is that the segment located in the middle of the segments can be retransmission. Accordingly, careful form of reward design is required depending on the replacement situation.

The Smoothness term of the Reward is defined as follows:

- Case 1. $trans(d) = 1$

This is the case for downloading the first segment in the process of starting streaming. In this case, no smoothness penalty occurs.

$$k_{our}(R_d) = 0 \quad (8)$$

- Case 2. $trans(d) \neq 1$ and $count(d) = 1$

In this case, segments after the first are downloaded for the first time. In this case, to calculate the smoothness penalty, we just need to calculate the bit rate difference from the previous segment, so it will be defined as follows:

$$k_{our}(R_d) = |q(R_{1,n}) - q(R_{n-1}^d)| \quad (9)$$

- Case 3. $trans(d) \neq 1$ and $count(d) \neq 1$ and $i(d) = 1$

This is the first of the cases in which the downloaded segment is not the first segment but has been replaced. However, this case deals with the case in which the segment is located at the end of the buffer. Accordingly,

smoothness is calculated only with the previous segment. In this case, the smoothness penalty already calculated in Case 2 must be removed for the calculation of new smoothness. The new calculation of smoothness with the previous segment then results in the correct smoothness penalty.

$$k_{our}(R_d) = |q(R_{count(d),trans(d)}) - q(R_{trans(d)-1}^d)| - |q(R_{count(d)-1,trans(d)}) - q(R_{trans(d)-1}^d)| \quad (10)$$

- Case 4. $trans(d) \neq 1$ and $count(d) \neq 1$ and $i(d) \neq 1$

This case is similar to case 3, but the segment located in the middle of the buffer rather than the end of the buffer is replaced. Accordingly, the smoothness with the immediately preceding segment as well as with the immediately preceding segment must be updated together.

$$k_{our}(R_d) = (|q(R_{count(d),trans(d)}) - q(R_{trans(d)-1}^d)| - |q(R_{count(d)-1,trans(d)}) - q(R_{trans(d)-1}^d)|) + (|q(R_{count(d),trans(d)}) - q(R_{trans(d)+1}^d)| - |q(R_{count(d)-1,trans(d)}) - q(R_{trans(d)+1}^d)|) \quad (11)$$

We show that our smoothness term is the same as the existing smoothness term in the following ways. First, case 1 and case 2 are the same definitions as the existing smoothness terms. For the sake of proof, case 3 occurs first and case 4 occurs first can be considered separately.

- Case 3 occurred before case 4

Let's consider the first case 3 to occur. If case 3 occurred at the s-th

action point, it is self-evident that it is the same as the existing smoothness term of QoE because no replacement occurred until s-1 point. The s-th action will be given a reward of case 3.

$$k_{our}(R_s) = |q(R_{count(s),trans(s)}) - q(R_{trans(s)-1}^d)| - |q(R_{count(s)-1,trans(s)}) - q(R_{trans(s)-1}^s)| \quad (12)$$

At this time, $|q(R_{count(s)-1,trans(s)}) - q(R_{trans(s)-1}^s)|$ is the smoothness penalty assigned when the currently replaced segment was previously downloaded. On the other hand, $|q(R_{count(s),trans(s)}) - q(R_{trans(s)-1}^d)|$ is a smoothness penalty caused by this replacement. Accordingly, the action up to the point s has the same value as the existing QoE smoothness term. Accordingly, even if case 3 or case 4 occurs again next, the smoothness penalty can be calculated as if case 3 or case 4 first occurred.

- Case 4 occurred before case 3

This can be also proved as in case 3. When case 4 first occurs, $|q(R_{count(d)-1,trans(d)}) - q(R_{trans(d)-1}^d)|$ is a penalty imposed for smoothness with the segment immediately preceding it, and $|q(R_{count(d)-1,trans(d)}) - q(R_{trans(d)+1}^d)|$ is a penalty imposed for smoothness with the immediately preceding segment. The terms $|q(R_{count(d),trans(d)}) - q(R_{trans(d)-1}^d)|$ and $|q(R_{count(d),trans(d)}) - q(R_{trans(d)+1}^d)|$ are newly calculated smoothness penalties, and as in case 3, the same result as the

smoothness term of the existing QoE is calculated.

$$\sum_{d=1}^D k_{our}(R_d) = \sum_{n=1}^{N-1} |q(R_{n+1}) - q(R_n)| \quad (13)$$

Table 1: Notation

Notation	Explanation
N	Total number of segments
n	The position of the segment being played
D	Total number of download actions
d	Sequence of download actions
R_n	Bitrate of the d -th played segment
R_d	Bitrate of d -th downloaded segment
R_n^d	bitrate of n -th segment during d -th action ($n \leq d$)
$R_{s,n}$	Bit rate when n -th segment is downloaded s times
T_d	Rebuffering generated when the d -th download action is performed
T_n	Rebuffering time that occurs when the n -th segment is played
$q(\cdot)$	Utility function
α	Penalty for rebuffering
β	Penalty for smoothness
$i(d)$	Function indicating whether the segment downloaded at d -th action is at the end of the buffer
$trans(d)$	Mapping function from action number to segment position
$count(d)$	A function that maps how many times the segment downloaded in the d -th action has been replaced
$total(n)$	Total number of times the n -th segment was downloaded

4.4 Rule based learning

Our model can also re-download segments in buffers in the form of replacement. However, enabling replacements in all cases without any limitations significantly increases the complexity of the problem, causing problems in performance and convergence. As a solution to these problems, we apply some rules in learning and action.

- Do not replace with low bitrate.

Clearly, there may be cases where QoE can be improved by replacing segments with low bitrate. However, this is because the model was taught to download high bitrate first and then replace it with low bitrate. We limit replacement to low bitrate to induce segment of low bitrate to download first.

- If the model downloads segments that are not in the buffer, the latest segments must be downloaded.

Our model doesn't skip segment downloads. That is, if the current buffer contains up to the n th segment, the $(n+1)$ -th segment must be downloaded before the $(n+2)$ -th segment.

- The two segments in front of the buffer are not replaced.

Finally, our model does not conduct replacement for the first two segments of the buffer. This is a rule to induce learning in the form of preventing rebuffering. If there is only one segment in the buffer, rebuffering occurs when the download speed is later than the playback speed of the segment. For the $(n+1)$ -th segment to play, the n th segment must be completed. In this situation, if the segment at the

front of the buffer is replaced in the situation, rebuffering must occur. If you replace the second segment in front of the buffer, the download must be completed before the first chunk has finished playing, so rebuffering will not occur. To do this, the download rate must be faster than the playback rate, causing a case that adversely affects smoothness and bitrate. Because of this, we limit the replacement of the first two segments in the buffer.

4.5 Implementation

Recently, a number of DRL models have been proposed. As the DRL model improves, it is possible to solve more complex and difficult problems. In the case of Pensieve [15], an existing DRL-based ABR algorithm, the A3C [36] model was used. Since there is no reason not to utilize the performance of the recently developed DRL model, we train the ABR algorithm using the PPO model [37]. Accordingly, it is necessary to additionally check whether the increase or decrease in performance is caused by the improvement of the model or by segment replacement (section 5.4).

Chapter V

Experiments

In this section we evaluate LAWS and compare its performance to existing ABR algorithms.

5.1 Experiment Setup

We evaluate our model using trace-driven simulation similar to Sabre [19]. We used the ‘Envivio-Dash3’ video from DASH-264 JavaScript reference client test page [35] for our experiments. This video is encoded using the H.264/MPEG-4 codec at bitrates in $\{300, 750, 1200, 1850, 2850, 4300\}$ kbps. It has a total length of 193 seconds and is divided into 48 segments each consisting of 4 seconds. We set the buffer size to 40s. We used a general QoE metric used by MPC [6] described in equation 1. Also, we consider QoE_{HD} and QoE_{log} by varying the utility function and rebuffer penalty. The exact values used in QoE metrics are described in Table 2.

Table 2: QoE metrics

QoE Metric	Utility Function $q(\cdot)$	Rebuffer Penalty α
QoE_{linear}	R	4.3
QoE_{HD}	$0.3 \rightarrow 1, 0.75 \rightarrow 2, 1.2 \rightarrow 3,$ $1.85 \rightarrow 12, 2.85 \rightarrow 15, 4.3 \rightarrow 20$	8
QoE_{log}	$\log(R/R_{min})$	2.66

We use 3G/HSDPA mobile dataset to evaluate our model. We choose 141 traces from HSDPA dataset, which contain various mobility environments such as bus, car, ferry, metro, train and tram. We did not consider the traces whose average throughput is less than 0.2Mbps, and above 6Mbps. We use 65% of randomly divided traces as training set and the remaining 35% of traces as test set.

5.2 Baselines

We compare the performance of LAWS with the following ABR algorithms.

- Buffer-Based [27]: Buffer-Based algorithm selects bitrates only based on the buffer level. If the buffer level is below 5 seconds, it selects the lowest bitrate. On the contrary, if the buffer level is above 15 seconds, it selects the highest bitrate. In between, the linear function is used to select the bitrate. We set a reservoir value as 5 seconds, and a cushion value as 10 seconds, as described in [27].
- Rated-Based [15]: Rate-Based algorithm predicts the future throughput using the harmonic mean of the throughput for the past 5 segments. It selects the highest bitrate that is below the predicted bandwidth.
- Robust-MPC [6]: MPC uses the classical model predictive model to determine the next bitrate. In MPC, both buffer level and future throughput prediction results are used for bitrate decision. Like RB, future throughput is predicted using the harmonic mean of the past 5 segments. After it obtains a predicted throughput, it selects the next bitrate that maximizes a QoE over a horizon of 5 future segments. In our

evaluation, we use RobustMPC that reflects the prediction error.

- Pensieve [15]: Pensieve is a learning based ABR algorithm, which selects bitrates through deep reinforcement learning.

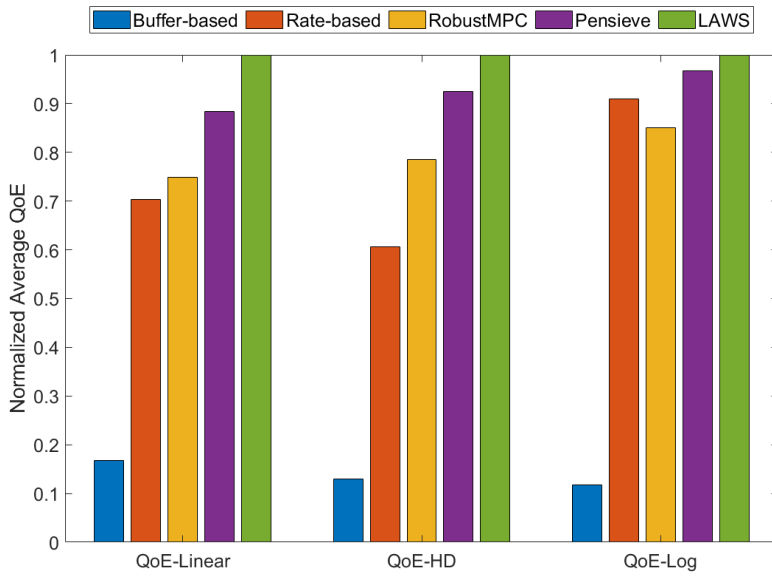


Figure 4. Comparing LAWS with existing ABR algorithms in three different QoE metrics

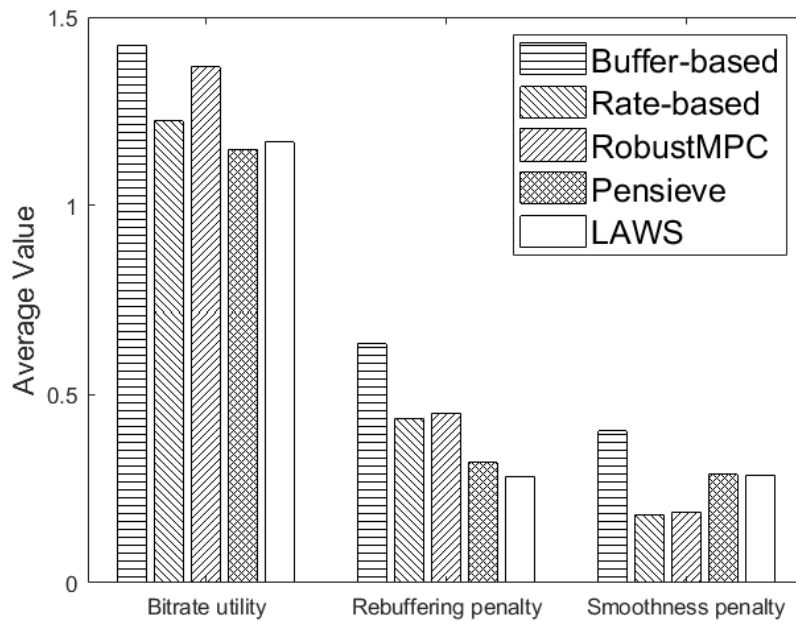


Figure 5. Comparing LAWS with existing ABR algorithms on the individual components in the general QoE definition

5.3 Comparison with Existing ABR algorithms

To evaluate the overall performance of LAWS, we first compared the normalized average QoE of LAWS with existing ABR algorithms. We use the average QoE per segment, thus the total QoE value is divided by the number of segments in the video. As shown in figure 4, LAWS performs better than existing ABR algorithms in all three different QoE metrics. For QoE_{linear} , the average QoE for LAWS is 33.6%, 13.1% higher than robustMPC and Pensieve, respectively. In the case of QoE_{HD} and QoE_{log} , the gap in performance between LAWS and other ABR algorithms has narrowed. For QoE_{HD} which prefers High Definition (HD) video, the average QoE for LAWS is 27.4%, 9.6% higher than robustMPC and Pensieve, respectively. For QoE_{log} which was used by BOLA [28], the average QoE for LAWS is 17.6%, 3.3% higher than robustMPC and Pensieve, respectively.

Next, we analyzed the impact of individual terms in general QoE definitions. Figure 5 shows the average bitrate utility, rebuffering time, and smoothness of each ABR algorithm. As shown, LAWS does not always achieve best performance on every QoE terms. However, LAWS balance between QoE terms through optimization.

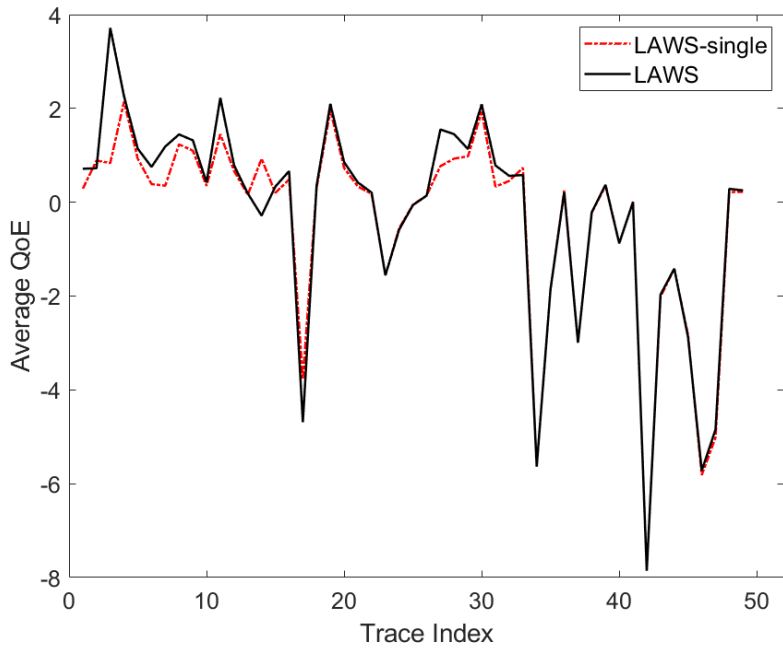


Figure 6. Comparing LAWS-single and LAWS in terms of average QoE of each test trace

5.4 Analyze Replacement Characteristics

Replacement has a positive effect on QoE due to multi-download chance of each segment, but it also has an adverse effect that it increases the action space and thus makes it difficult to explore and convergence of DRL model. To analyze the replacement characteristics, we compared the average QoE of LAWS and LAWS with no replacement (LAWS-single). Figure 6 shows the average QoE of LAWS and LAWS-single on each test trace. As shown, LAWS does not outperform LAWS-single on every test trace. Nevertheless, normalized average QoE of LAWS shows 6.4% of improvement compared to LAWS-single.

5.5 Comparison Between Learning Based Algorithms

To give additional perspective on the value of DRL model with segment replacement action, we compared final played bitrate of two learning based ABR algorithms, LAWS and Pensieve. As shown in figure 7, after replacement happens at segment 5, the final bitrate of LAWS tends to follow the final bitrate of Pensieve. However, if the network throughput increased after downloading all segments with Pensieve algorithm, LAWS perform higher final played bitrates of the segments at the end of the video than Pensieve, resulting performance gain through bitrate utility.

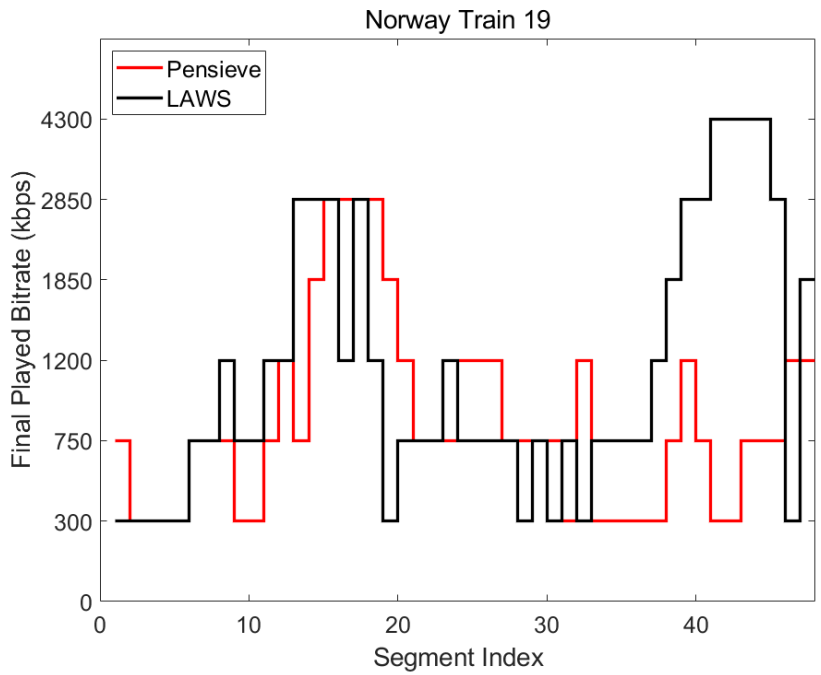


Figure 7. Final played bitrate of Pensieve and LAWS on selected network traces

Chapter VI

Conclusion

In this paper, we studied how to apply segment replacement option in ABR algorithm for QoE optimization. It has been proven that even if the segment replacement option is included in the ABR algorithm, the same results as the existing QoE optimization problem can be achieved. We proposed an ABR algorithm based on deep reinforcement learning, LAWS, by defining rewards and actions including segment replacement options. The challenges that exist in applying the deep reinforcement learning technique were solved using novel techniques. We evaluated the proposed method by experimenting based on the network trace. In the experimental results, it was confirmed that LAWS showed better performance in QoE optimization when compared to the existing ABR techniques.

We believe that the extension of the QoE problem that we have found can be applied to better optimization solutions, thereby providing the basis for research on improved ABR algorithms.

Bibliography

- [1] Live Streaming Statistics about Twitch and Facebook.
<https://www.theverge.com/2020/1/9/21058907/twitch-youtube-mixer-facebook-live-streaming-numbers-growth-q4>
- [2] Cisco, “Cisco visual networking index: global mobile data traffic forecast update, 2017–2022,” 2019.
- [3] H. Xie, A. Boukerche, and A. A. Loureiro, “Mervs: A novel multi channel error recovery video streaming protocol for vehicle ad hoc networks,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 2, pp. 923–935, 2015.
- [4] S. Swetha and D. Raj, “Optimized video content delivery over 5g networks,” in *2017 2nd International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2017, pp. 1000–1002
- [5] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas, “Caching and operator cooperation policies for layered video content delivery,” in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.
- [6] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over http,” in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.
- [7] K. Spiteri, R. Sitaraman, and D. Sparacio, “From theory to practice: Improving bitrate adaptation in the dash reference player,” *ACM Transactions on*

Multimedia Computing, Communications, and Applications (TOMM), vol. 15, no. 2s, pp. 1–29, 2019.

[8] J. Jiang, V. Sekar, and H. Zhang, “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, 2012, pp. 97–108.

[9] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 272–285

[10] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, “A buffer-based approach to rate adaptation: Evidence from a large video streaming service,” in Proceedings of the 2014 ACM conference on SIGCOMM, 2014, pp. 187–198

[11] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, “Understanding the impact of video quality on user engagement,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[12] F. Dobrian, A. Awan, D. Joseph, A. Ganjam, J. Zhan, V. Sekar, I. Stoica, and H. Zhang, “Understanding the impact of video quality on user engagement,” *Communications of the ACM*, vol. 56, no. 3, pp. 91–99, 2013.

[13] S. S. Krishnan and R. K. Sitaraman, “Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs,” *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

- [14] M. Z. Shafiq, J. Erman, L. Ji, A. X. Liu, J. Pang, and J. Wang, “Understanding the impact of network dynamics on mobile video user engagement,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp.367–379, 2014.
- [15] H. Mao, R. Netravali, and M. Alizadeh, “Neural adaptive video streaming with pensieve,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 197–210.
- [16] T. Huang, X. Yao, C. Wu, R.-X. Zhang, Z. Pang, and L. Sun, “Tiyuntsong: A self-play reinforcement learning approach for abr video streaming,” in *2019 IEEE International Conference on Multimedia and Expo (ICME)*.
- [17] Y. Qin, R. Jin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, C. Yue, and B. Wang, “A control theoretic approach to abr video streaming: A fresh look at pid-based rate adaptation,” *IEEE Transactions on Mobile Computing*, 2019.
- [18] Y. Xu, Y. Zhou, and D.-M. Chiu, “Analytical qoe model for bit-rate switching in dynamic adaptive streaming systems,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2734–2748, 2014.
- [19] N. Barman and M. G. Martini, “Qoe modeling for http adaptive video streaming—a survey and open challenges,” *IEEE Access*, vol. 7, pp. 30 831–30 859, 2019
- [20] DASH Industry Forum Official Website. <https://dashif.org/>.
- [21] Dash-Industry-Forum/dash.js,
<https://github.com/Dash-Industry-Forum/dash.js>.
- [22] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP—Standards and design principles. In *Proceedings of the 2nd Annual ACM*

Conference on Multimedia Systems. ACM, 133–144.

[23] Z. Meng, J. Chen, Y. Guo, C. Sun, H. Hu, and M. Xu, “Pitree: Practical implementation of abr algorithms using decision trees,” in Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 2431–2439.

[24] Jiang, J., Sekar, V., and Zhang, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In ACM CoNEXT (2012), pp. 97–108.

[25] Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C., and Oran, D. Probe and adapt: Rate adaptation for http video streaming at scale. IEEE Journal on Selected Areas in Communications 32, 4 (2014), 719–733.

[26] Wang, C., Rizk, A., and Zink, M. Squad: A spectrum-based quality adaptation for dynamic adaptive streaming over http. In ACM MMSys (2016)

[27] Huang, T.-Y., Johari, R., McKeown, N., Trunnell, M., and Watson, M. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In ACM SIGCOMM (2014).

[28] Spiteri, K., Urgaonkar, R., and Sitaraman, R. K. Bola: Near-optimal bitrate adaptation for online videos. In IEEE INFOCOM (2016).

[29] De Cicco, L., Caldaralo, V., Palmisano, V., and Mascolo, S. Elastic: A client side controller for dynamic adaptive streaming over http (dash). In IEEE International Packet Video Workshop (2013)

[30] Yadav, P. K., Shafiei, A., and Ooi, W. T. Quetra: A queuing theory approach to dash rate adaptation. In ACM MM (2017)

[31] Adobe HTTP Dynamic Streaming,

<http://www.adobe.com/products/httpdynamicstreaming/>

- [32] R. Pantos, W. May, “HTTP Live Streaming”, IETF draft,
<http://tools.ietf.org/html/draft-pantos-http-live-streaming-07>
- [33] Microsoft Smooth Streaming,
<http://www.iis.net/download/smoothstreaming>
- [34] R. S. Sutton, A. G. Barto. Reinforcement learning: An introduction. MIT press. 2018.
- [35] DASH Industry Form. 2016. Reference Client 2.4.0.
<http://mediapm.edgesuite.net/dash/public/nightly/samples/dash-if-reference-player/index.html>. (2016).
- [36] Mnih, Volodymyr, et al. “Asynchronous methods for deep reinforcement learning.” International conference on machine learning. (2016)
- [37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, “Proximal Policy Optimization Algorithms”, ArXiv 2017, arXiv:1707.06347v2, 2017.8

국문초록

적응형 비트레이트 알고리즘은 온라인 비디오 서비스의 재생 품질, 즉 사용자 체감 품질을 올리기 위하여 사용되는 대표적 기술 중 하나이다. 지금까지 적응형 비트레이트 알고리즘은 다양한 최적화 기법에 기반하여 사용자 체감 품질을 최적화하였다. 그러나 대부분의 적응형 비트레이트 알고리즘은 공통된 한계점을 지닌다. 사용자 체감 품질을 최적화하기 위해 단순히 다음으로 다운로드 해야 하는 세그먼트의 비트레이트만을 결정한다는 점이 그 한계점으로, 이러한 유형에 속하는 적응형 비트레이트 알고리즘들은 변화하는 네트워크 환경에 맞춰 앞으로 다운로드할 세그먼트의 비트레이트는 최적으로 조정할 수 있지만 이미 다운로드한 세그먼트에 대해선 어떠한 최적화도 진행할 수 없다. 그렇기에 사용자의 네트워크 환경이 극단적으로 개선되더라도 이에 대한 활용도가 떨어진다.

이러한 한계점을 극복하기 위해 우리는 LAWS 기법, 학습 기반의 세그먼트 교체 전략을 포함한 적응형 비트레이트 알고리즘, 을 제안한다. 제안 모델은 사용자의 네트워크 환경 등에 따라서 더 나은 비트레이트로 세그먼트를 교체할 수 있다. 제안 기법을 실현하기 위해 우리는 새로운 형태의 리워드를 디자인한다. 이를 통해 제안 기법은 세그먼트 교체 전략을 포함한 형태로 사용자 체감 품질을 최적화할 수 있다. 또한 세그먼트 교체 전략을 포함함에 따라 증가

하는 문제의 복잡도에 대응하기 위해 규칙 기반 행동 제약 기법을 사용하여 모델의 학습을 원하는 방향으로 유도한다. 우리는 최종적으로 심층 강화학습 기반의 적응형 비트레이트 알고리즘을 제안한다. 네트워크 트레이스를 기반으로 실시한 실험에서는 제안 기법이 기존의 기법들에 비해 사용자 체감 품질을 13.1%까지 개선시키는 것으로 확인됐다.

주요어: 비디오 스트리밍, 적응형 비트레이트 알고리즘, 심층 강화학습, 세그먼트 교체, DASH, 최적화, 전송 제어

학번: 2019-20993