M.S. THESIS

Hybrid Adaptive Bitrate for Video Streaming

비디오 스트리밍을 위한 혼합 적응 비트레이트

2023 년 8 월

서울대학교 대학원

협공과정 인공지능전공

맹 재 우

M.S. THESIS

Hybrid Adaptive Bitrate for Video Streaming

비디오 스트리밍을 위한 혼합 적응 비트레이트

2023 년 8 월

서울대학교 대학원

협공과정 인공지능전공

맹 재 우

Hybrid Adaptive Bitrate for Video Streaming

비디오 스트리밍을 위한 혼합 적응 비트레이트

지도교수 전 병 곤

이 논문을 공학석사학위논문으로 제출함

2023 년 6 월

서울대학교 대학원

협공과정 인공지능전공

맹 재 우

맹재우의 공학석사 학위논문을 인준함

2023 년 7 월

위 원 장	유 승 주	(인)
부위원장	전 병 곤	(인)
위 원	이 영 기	(인)

Abstract

A good quality-of-experience (QoE) percieved by the end user is the most important goal of network systems for video streaming. Client-side video players adopt adaptive bitrate (ABR) to achieve this goal over varying network conditions. In optimizing the performance of ABR, researchers and engineers have proposed several rule-based algorithms and more recently, deep reinforcement learning based ABR algorithm has been proposed, reporting a better overall QoE over real network traces. However as RL-based bitrate control gained more attention, recent research also report corner cases of RL, such as occasional bitrate overshoots or underperformance in network conditions far from the trained dataset. In this paper we analyze when the RL-based ABR alogorithm makes poor bitrate decisions compared to rule-based algorithms and propose a desgin to replace these decisions with the better decisions of a rule-based algorithm. We implemented Hybrid, the ABR algorithm that gets the best of the two worlds and validated it over real network traces.

Keywords: Reinforcement Learning, Adaptive Bitrate, Video Streaming

Student Number: 2021-22299

Contents

Chapter 1

Introduction

As quality-of-experience (QoE) is the most important goal of video streaming systems, abundant algorithms have been proposed to better perform the adaptive bitrate (ABR) in the client-side video player. As traditional rule-based algorithms [1, 2, 3] are still leveraged today, an approach to use deep reinforcement learning [4] has shown great performance.

As the RL-based approach performs the best in average QoE. We find that this does not mean the RL-based approach excels in all conditions. We find similar number of traces in the entire trace data [5] where a rule-based state-of-the-art, RobustMPC performs the best in terms of trace-level average QoE. As most content providers aim to provide quality video streaming experience for all end users, not on average, this can be a problem to address.

Since RL is a black-box, it is hard to fix the RL model to act in well in some wanted scenarios. So in the paper, we propose decision-level multiplexing, using RL model's decision as default, but using the rule-based algorithms as fallback.

Yet, due to the black-box nature of RL, it is challenging to find scenar-

ios where an RL model's bitrate selection is worse than that of a rule-based decision. We assumed that there will be patterns of conditions where the RL underperforms and decided to use the patterns as a solution.

We divide the solution into two stages. In the first stage, which is offline, we detect the fallback points and map patterns from the detected fallback points. In the online stage, we leverage the pre-computed pattern information from the offline stage and identify conditions that are adequate to fallback.

We implemented this design with the name of Hybrid. Hybrid uses Pensieve [4] and RobustMPC [3] as the RL approach and the rule-based algorithm. We show through evaluation that Hybrid performs better of average QoE compared to Pensieve ro RobustMPC, with a QoE gain up to 17.9%. We also test Hybrid in network traces that Hybrid's offline stage did not see and showed QoE improvement. Our contributions are as follows:

- We find that RL-based ABR algorithm and rule-based algorithms have different strength.
- We analyze when the RL-based approach makes poor decisions compared to the traditional rule-based algorithms.
- We design and implement Hybrid, an approach that uses RL as default, but detects RL model's weak points and enhances performance by replacing them with rule-based decisions.

Chapter 2

Background

2.1 Video Streaming over HTTP

Dynamic adaptive streaming over HTTP, which is standardized as DASH [6], is the dominant model for delivering videos today. In a DASH system, the main idea is that a video content is split into multiple segments typically corresponding to a few seconds long. As shown in Figure 2.1, the bitrate controller collects information from components such as throughput predictor or playback buffer. Based on the network conditions perceived, the ABR algorithm in the bitrate controller makes the next bitrate decision and the video player will request the next video chunk over HTTP with the decided bitrate.

To deliver a good QoE, video players and their bitrate controllers consider multiple aspects that can influence the QoE. These aspects include minimizing the startup delay of a video, providing the highest level of bitrate as possible as end users prefer high resolution, minimizing rebuffering, and keeping the bitrates as smooth as possible since large bitrate jumps are disturbing to end

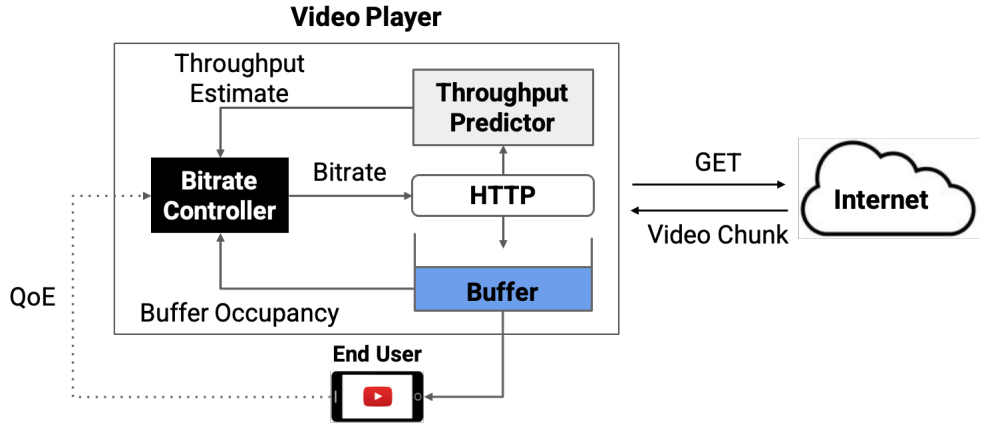

Figure 2.1 Overview of HTTP-based adaptive video streaming.

users. Considering the multiple aspects are challenging for ABR algorithms because these features are often conflicting. For example, providing a smooth experience may result in the video player not being able to achieve the highest bitrate since pursuing the highest bitrate will result in the end user experiencing all the fluctuations of network conditions. Existing algorithms propose many methods to address this problem with slightly different focus on aspects influencing QoE.

2.2 Adaptive Bitrate Algorithms for Video Streaming

We can categorize ABR algorithms used in bitrate controllers into traditional rule-based algorithms and the relatively recent deep reinforcement learning based algorithm.

2.2.1 Rule-based Algorithms

Rule-based algorithms take network conditions as input and calculate the next request bitrate as the output of a fixed rule. We can sort the rule-based algo-

rithms into two types, (1) algorithms that primarily decide bitrates on playback buffer occupancy observations and (2) algorithms that look at both the playback buffer occupancy and the estimated network throughput.

The buffer-based approach proposed by Huang et al [1] is an example of the first type. In this approach, the algorithm considers a playback buffer occupancy of 5 seconds as an untouchable minimum. If the buffer occupancy gets low near 5 seconds, the algorithm always decides on the lowest bitrate. It also considers an additional 10 seconds of buffer occupancy as a cushion. If the buffer occupancy is over the total of 15 seconds, the algorithm decides on the highest bitrate possible. BOLA [2] is also an algorithm primarily considering the playback buffer occupancy. BOLA sees the bitrate decision as an optimization problem with a minimum buffer occupancy as with [1] and a target buffer occupancy.

Model predictive control (MPC) [3] and its variant RobustMPC [3] are examples of the second type where the algorithms take network throughput estimation into account as well. In MPC, instead of trying to reach for the features contributing to the final QoE, since QoE is established as an equation of network inputs, it solves the optimization problem with the QoE as the objective itself. It exploits the given throughput estimator of the video player, which takes recent video chunk throughputs to predict future throughput. Also taking the current buffer occupancy and the previous bitrate into account, MPC utilizes an off-the-shelf solver to decide on the optimal bitrate. A variant of MPC, the RobustMPC is a conservative version where it takes into account maximum throughput error prediction in the last five video chunk estimations. It uses the maximum error as a discount factor to the throughput predictions, resulting in considering lower throughput estimates. Besides the RL-based approach, RobustMPC is considered the state-of-the-art ABR algorithm.

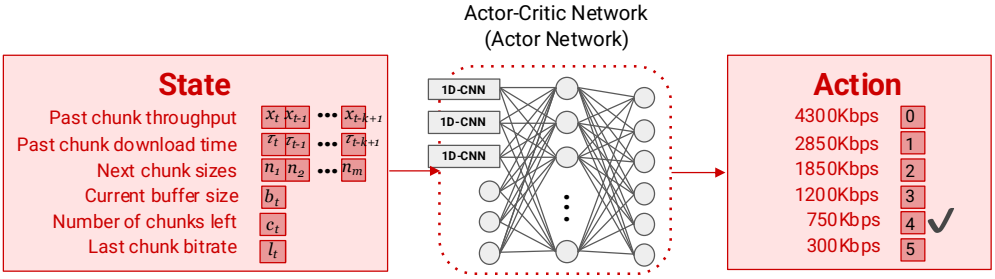

Figure 2.2 Overall design of Pensieve [4], an RL-based ABR algorithm.

2.2.2 RL-based Algorithm

With the rule-based algorithms using a fixed control rule, they require an accurate model of system dynamics and do not generalize over different environments. To address this, Pensieve [4], a deep reinforcement learning based ABR algorithm is proposed, not relying on any assumptions about the network environment. Pensieve trains a neural network to select a bitrate when the observations of the network is given as an input. As shown in Figure 2.2, Pensieve takes raw observations such as previous throughputs, download times, playback buffer occupancy, and last bitrate. As output, the neural network presents the probabilities of each action and the bitrate with the highest probability is chosen to be the next bitrate. Pensieve exploits the state-of-the-art actor-critic RL algorithm A3C [7] and is trained on various real network traces. As shown in section 3.1, Pensieve generally outperforms the rule-based ABR algorithms.

Chapter 3

Observation

3.1 Performance of ABR Algorithms

We analyzed performance of different ABR algorithms in simulations using 142 Norway 3G/HSDPA mobile traces [5]. For the simulation environment we used the chunk-level simulator available at [4]. When the ABR algorithms decide on the next bitrate, the simulator faithfully returns metrics such as throughput, QoE, and buffer occupancy according to the trace, bitrate decision, and the current network state.

For the average QoE over the traces, we can see from Table 3.1 that Pensieve shows the best result with 17.7% better average QoE than the rule-based state-of-the-art, RobustMPC. However, this did not mean that Pensieve shows best performance in all traces. As we examined trace-level average QoE, we could find that Pensieve shows best average QoE on only 73 out of 142 traces, which is only 51.4% of the entire set. RobustMPC showed best average QoE on 68 of the 142 traces, which is 47.9%, not much different from that of Pensieve.

	Algorithm	Average QoE
	Buffer-based (BB) [1]	0.279
Rule-based	Model Predictive Control (MPC) [3]	- 0.197
	RobustMPC [3]	0.485
RL-based	Pensieve [4]	0.571

Table 3.1 **Average quality of experience.** Evaluation on simulations over 142 Norway 3G/HSDPA mobile traces [5].

	Buffer-based (BB)	MPC	RobustMPC	Pensieve
Number of best QoE trace	0	1	68	73

Table 3.2 **Number traces where each algorithm showed best average QoE.** Although Pensieve [4] shows best average QoE throughout the entire trace set, Pensieve does not deliver best QoE in all scenarios.

To get a hint of the behaviors and strong suits of each ABR algorithms, we analyzed the characteristic of top-10 traces where Pensieve and RobustMPC outperformed the second best algorithm regarding average QoE. Figure 3.1 shows the network bandwidth distributions of the traces. As shown in Figure 3.1, the environments where each algorithm excel were fairly different. Average bandwidth of RobustMPC top traces were 2.64 Mbps, while the average of Pensieve top traces were only 1.63 Mbps. Also, the average bandwidth fluctuations were 0.28 Mbps for RobustMPC top traces and 0.21 Mbps for Pensieve top traces, with fluctuations calculated as the gap between the current and the last network bandwidth. As video content providers are interested in providing quality video streaming experience to all the end users, not only on average, the lack of an algorithm that suits all scenarios can be a problem.

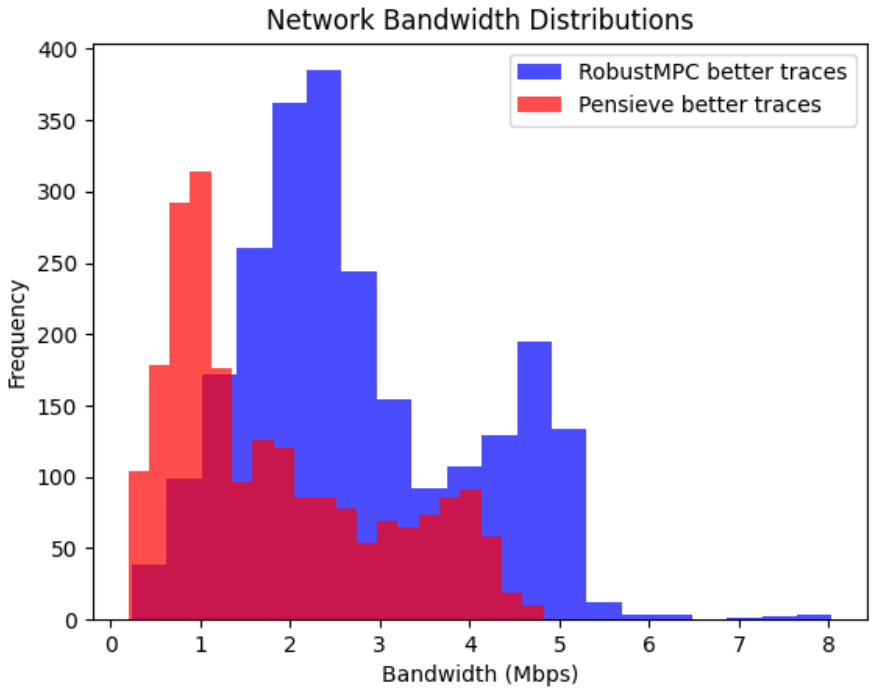

Figure 3.1 Network bandwidth distributions of top-10 traces where Pensieve and RobustMPC outperforms the second best algorithm.

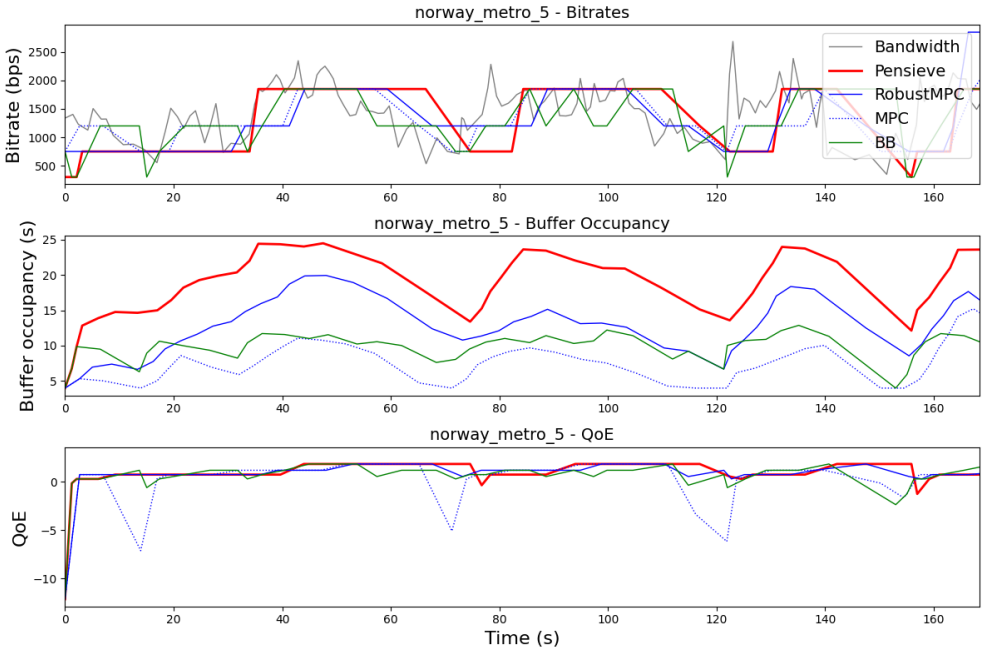

Figure 3.2 Example trace where RobustMPC achieved the best average QoE.

3.2 Behaviors of ABR Algorithms

Figure 3.2 and Figure 3.3 show how each algorithm behaves along the same traces. Looking at the bahviors of the buffer-based approach (BB) [1], we can see much fluctuations in terms of bitrate. This is because as described in subsection 2.2.1, the buffer-based approach only takes the playback buffer occupancy. We can also find the buffer-based approach keeping the buffer occupancy at a stable level. MPC [3] shows smaller fluctuations compared to the buffer-based approach, with the algorithm considering more features comprehensively, and with RobustMPC being the conservative version of MPC, it exhibits a more smoother bitrate. Figure 3.2 is a scenario where RobustMPC achieved the best average QoE. With the bandwidth fluctuations high in the trace, the approach

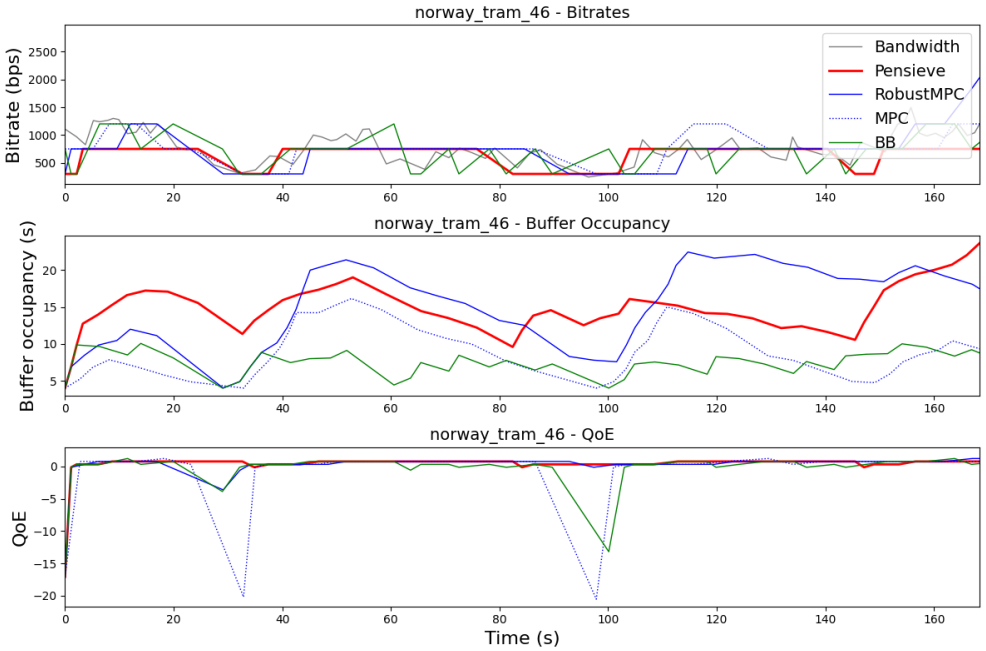

Figure 3.3 Example trace where Pensieve achieved the best average QoE.

of RobustMPC which is designed to cope with situations where throughput estimation has errors looks right to perform well.

Unlike the case of the three rule-based algorithms, it is hard to analyze the behaviors of Pensieve [4] as it depends on a black-box model. Looking at recent research for hints, some research on RL on network systems [8, 9] mention bitrate overshoot of RL as a problem and this pattern could also be found in our observations as in Figure 3.2. Also in works that analyze RL-based ABR algorithm using interpretation tools [10, 11] point out that the ABR algorithm considers the last bitrate as the most important decision factor. As shown in Figure 3.3, Pensieve displayed many cases where it achieves the best smoothness, aligning with the intuition that the black-box model considers the last bitrate importantly. This trait of Pensieve usually led to better QoE,

but in some cases, considering smoothness too much led to under-utilization of network bandwidths and poor QoE compared to other algorithms.

Chapter 4

Hybrid: Decision Level Multiplexing

In chapter 3 we observed that rule-based algorithms and the RL-based algorithm excels in different scenarios. From the observation, we propose Hybrid, a ABR algorithm that aims to achieve the best of two worlds. Figure 4 shows the overview of Hybrid. As default, Hybrid utilizes the generally good performing RL-based ABR algorithm, but falls back to decisions of rule-based algorithms when rule-based decisions are better. To achieve the goal of Hybrid, it uses a two-stage design. The first stage is an offline stage, where the calculation of patterns of adequate fallback points is done in advance. The second stage is online, where Hybrid determines whether the current network condition is adequate for fallback based on the pre-computed results of the offline stage.

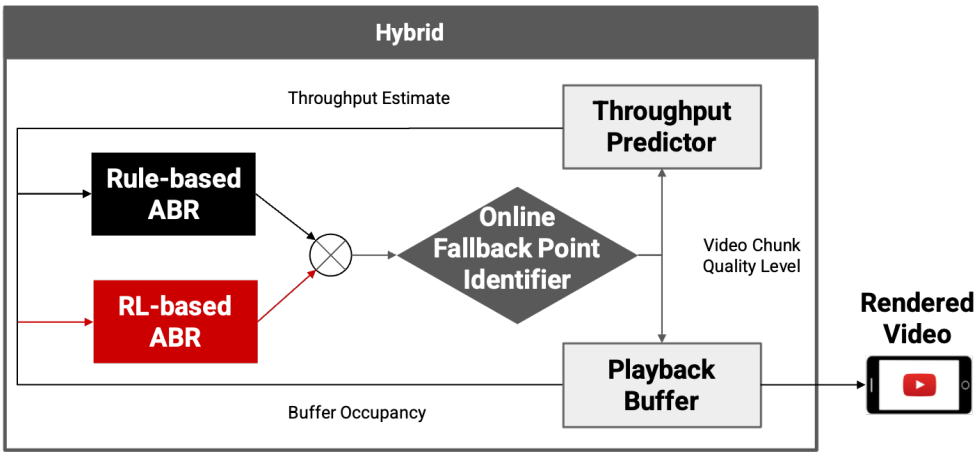

Figure 4.1 Overview of Hybrid.

4.1 Offline Mapping of Fallback Patterns

The challenge in achieving Hybrid's goal is in deciding when to fallback to the rule-based algorithm. This is complicated further by the black-box nature of RL. As mentioned, it is hard to comprehend how the RL model makes its decision and the agent is not targeted to achieve certain aspects contributing to the QoE as some rule-based algorithms are. Because of this black-box nature of RL, it is difficult even for domain experts to decide on when the RL model underperforms and so the fallback is needed. We decided to use the past network traces as a solution, by examining when it would have been better to replace the RL agent's decision with a rule-based bitrate selection, and mapping the characteristics of the incidents into patterns for later use.

4.1.1 Fallback Point Detection

The first part of the offline mapping is to detect when an RL based decision should have been replaced by a rule-based decision. Hybrid leverages Pen-

sieve [4] and RobustMPC [3] as its RL and rule-based ABR algorithms. To begin with, we list up the candidate points of RL algorithm underperformance. The candidate points are selected as moments in trace where the QoE of Pensieve was lower than that of RobustMPC along with 5 past moments from the lower QoE observed moment. Past moments are considered as well because an underperformance at one point can be a result from a bad decision in the past. For all the candidate points, we replay the trace with *(trace name, timestamp)* as input. The replay mechanism will run the trace with the decision at the given timestamp replaced with the bitrate selection of RobustMPC. If the replaced trace average QoE is higher than the original RL-only average QoE, we classify the point as a *fallback point*. From 142 Norway 3G/HSDPA mobile traces [5], we could detect 203 fallback points. A detailed analysis of the fallback points will be illustrated in section 5.1.

4.1.2 Pattern Mapping of Fallback Points

From the detected fallback points, we need to find patterns the *online fallback point identifier* can look for. From our analysis(section 5.1), observations(section 3.2), and from the our domain knowledge that rebuffering is treated critically we categorized the fallback points into five types. Detailed explanation can be found in Table 4.1.

The fallback points are classified according to the decision tree in Figure 4.2. First, we check whether the rebuffering was observed at the fallback point. All fallback points with rebuffering are classified as *REBUFFERING* because in ABR, the behaviors of ABR algorithms are drastically different from non-rebuffering cases. Then, we compare the RL-based bitrate decision with the bitrate decision of RobustMPC. We know for a fact that the rule-based decisions are better decisions for the fallback points. So if the rule-based decision is lower

Type	Description
UP	Need to increase bitrate, but not enough shown.
UP-OVERSHOOT	Need to increase bitrate, but too much shown.
DOWN	Need to decrease bitrate, but not enough shown.
DOWN-OVERSHOOT	Need to decrease bitrate, but too much shown.
REBUFFERING	Poor rebuffering handling.

Table 4.1 Fallback types.

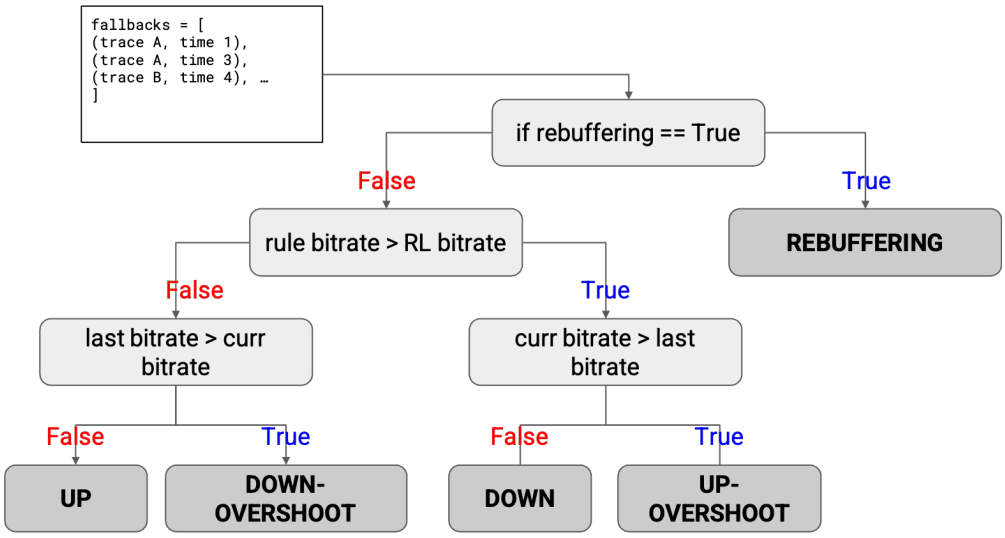

Figure 4.2 Decision tree to classify fallback points.

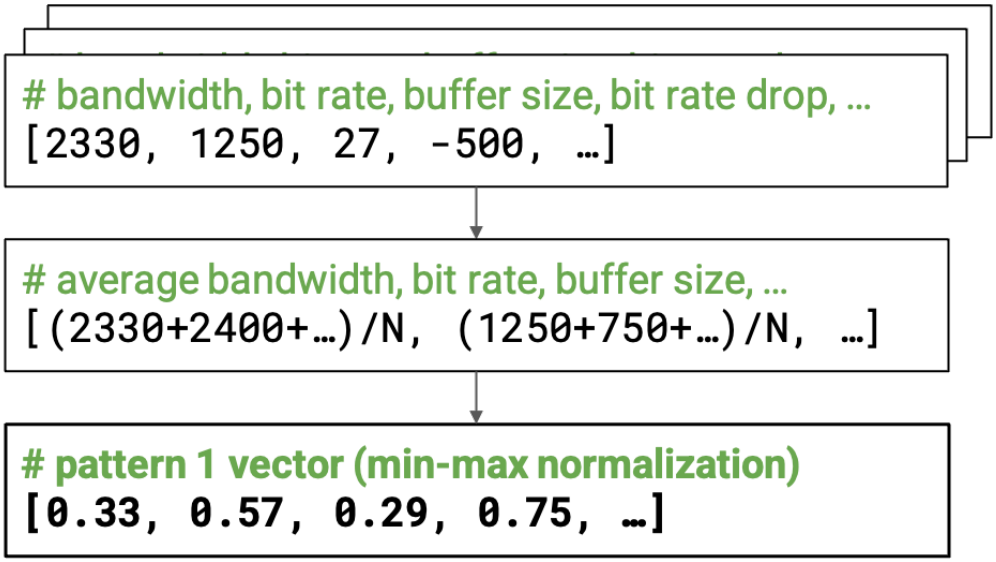

Figure 4.3 Aggregation of fallback point observations converted to a pattern.

than the RL-based bitrate selection, we can classify the fallback points to one of *DOWN* and *UP-OVERSHOOT*, meaning the bitrate should be lower than the current decision. The rest can be classified into *UP* or *DOWN-OVERSHOOT* in the same way. Lastly, we compare the last bitrate with the current bitrate decision of RL. If a fallback point is one of *DOWN* and *UP-OVERSHOOT* and the bitrate has increased, it can be classified as *UP-OVERSHOOT*. The same can be applied to *DOWN-OVERSHOOT*.

With the fallback points classified into five types, now we can map these points to fallback patterns for online use. Bitrate controller of a DASH [6] system makes bitrate decisions on observations of the present and the past as these are all the information we can get in the online stage. This is also the case for fallback decision. So Hybrid needs to identify a fallback point by finding similar patterns of the observed data of the five fallback types. For accurate pattern detection, we take 8 types of observations into consideration. These include

bitrate, bandwidth, buffer occupancy, extra bandwidth, delay time, rebuffering time, smoothness, and the difference between RL and rule-based decision. As we identified fallback points into five types, we aggregate the observations of each patterns. By averaging the observations and applying min-max normalization, we can retrieve five vectors of observation patterns as in Figure 4.3 for each type.

4.2 Online Fallback Point Detection

In the online stage, Hybrid leverages the pre-computed pattern vectors to identify a fallback point. Listing 4.1 describes how the online identifier detects fallback points. hybrid collects observations online, including the 8 features of observations used in offline mapping and converts it into a vector like one of the pattern vectors. Then the observed vector will be calculated cosine similarity with the offline pattern vectors. In the case any of the pattern similarity is over the threshold, the moment is identified as a fallback pattern like moment and rule-based decision of bitrate will be used.

```
# Fallback point identifier
# patterns from offline mapping
PATTERNS = [[0.33, 0.57, 0.29, 0.75, ...], [0.78,
0.19, 0.35, 0.58, ...], ...]
# inputs are normalized
def identifier(bandwidth, bitrate, buffer_size,
    bitrate_drop, ...):
    observed_state = [bandwidth, bitrate,
    buffer_size, bitrate_drop, ...]
    pattern_match = [0, 0, 0, 0, 0]
    similarities =
    cosine_similarity(PATTERNS, [observed_state])
    for i, sim in enum(similarities):
        if sim > THRESHOLD:
            pattern_match[i] = 1
    if sum(pattern_match) > 0:
        return True
    return False
```

Listing 4.1 Fallback point identifier.

Chapter 5

Evaluation

In this section, we present the analysis over the result of fallback point detection of subsection 4.1.1 and the performance measures of Hybrid, developed on top of the detected fallback points.

5.1 Fallback Point Detection

Figure 5.1 and Figure 5.2 are examples of fallback points detected over real network traces. As shown, when rebuffering occurs Pensieve tends to over-react and assign a unnecessarily low bitrate. The rebuffering incidents could be found mostly at trace start when the playback buffer is not ready with a low buffer occupancy. As in Figure 5.2, *UP* type fallback points could be detected with Pensieve not increasing bitrate in situations where extra-bandwidth are constantly observed. Actually of 79 *UP* fallback points, 78 fallback points showed the same bitrate as the previous bitrate. This aligns with the observation of section 3.2 that the RL black-box tends to take the previous bitrate as the most

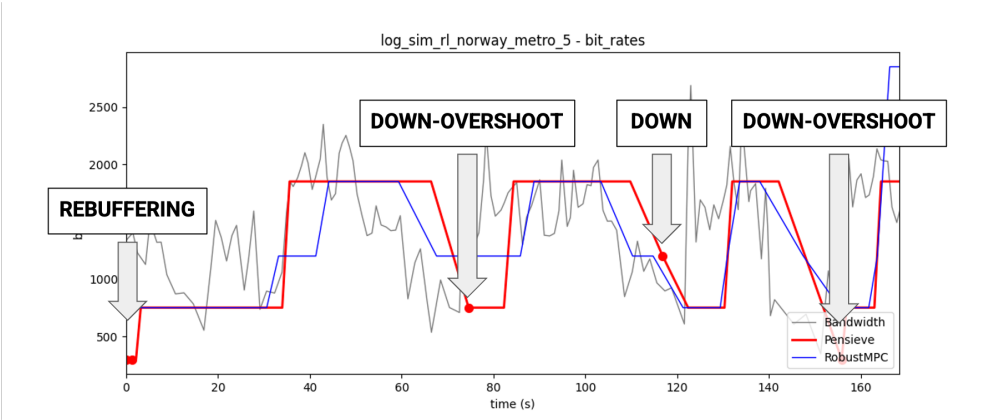

Figure 5.1 Fallback points detected over a network trace 1.

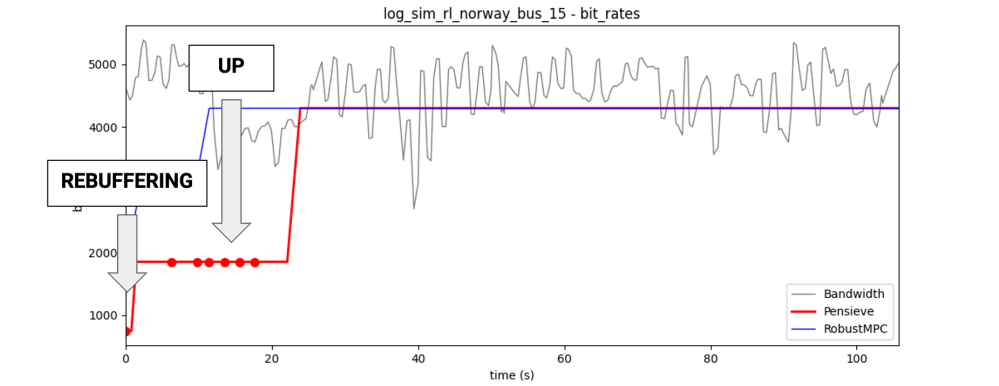

Figure 5.2 Fallback points detected over a network trace 2.

UP	DOWN-OVERSHOOT	DOWN	UP-OVERSHOOT	REBUFFERING
79	71	15	10	29
(38.7%)	(34.8%)	(7.35%)	(4.9%)	(14.22%)

Table 5.1 The number of each fallback point types.

important decision factor. The same applies to *DOWN*. Overshoots of RL leading to poor bitrate decisions took 39.7% of all the fallback points supporting the claim in section 3.2 where recent research pointed at RL overshoot as one of the problems. As shown in Table 5.1, out of 203 fallback points, more than 70% even without counting *REBUFFERING* types in, account as fallback points that required a higher bitrate decision. We assume this is because the reward function of Pensieve has a rebuffering penalty, causing decisions to slightly be conservative against increasing the bitrate compared to decreasing it.

5.2 Hybrid Performance

5.2.1 Evaluation Setup

Environment

We evaluated Hybrid in a chunk-level simulator available at [4]. For other environments, we utilized Ubuntu 18.04, 72 CPU, 252G memory with Python version of 3.7 and Tensorflow 1.14. As for QoE metric, we utilized *QoE-lin* from [3].

Workloads

As workloads, we used 142 network traces of Norway 3G/HSDPA mobile traces [5] of 37.1k seconds and 85 FCC broadband traces [12] of 105.8k seconds.

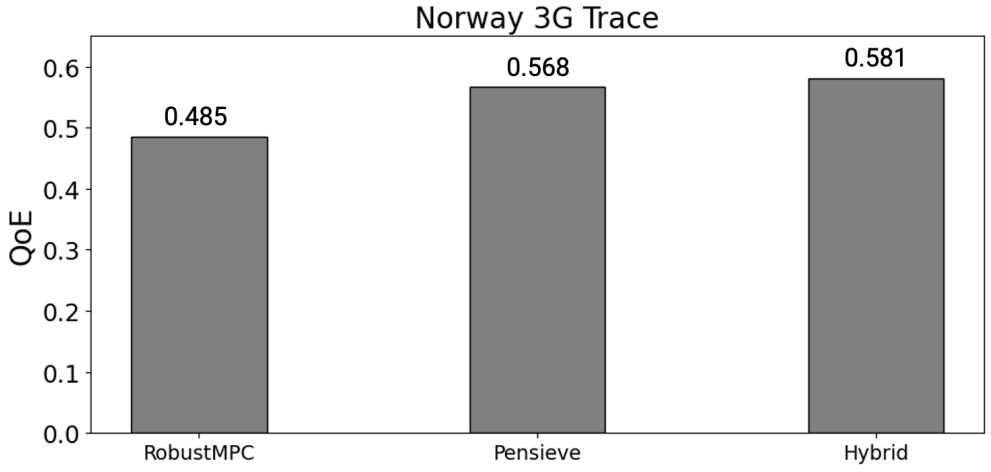

Figure 5.3 Average QoE over Norway 3G traces.

5.2.2 Quality of Experience

Figure 5.3 shows the average QoE over 142 Norway 3G traces. The average QoE of Hybrid outperform that of Pensieve and RobustMPC with 2.3% gain in average QoE compared to Pensieve. Out of 142 traces, 107 traces encountered online fallback point detection and in 60 of the traces, the average QoE increased.

5.2.3 Generalization

Generalization is also an important factor we considered in designing Hybrid. To evaluate generalization we evaluated Hybrid over unseen network traces in during the offline pattern mapping process. We used FCC broadband traces for this use and Figure 5.4 shows that Hybrid also provides the best average QoE in unseen environment with a gain of 17.9% gain in average QoE compared to Pensieve. Surprisingly the gain is even higher in FCC. We assume that since all the ABR algorithms struggle in the FCC trace, it is easier for Hybrid to find

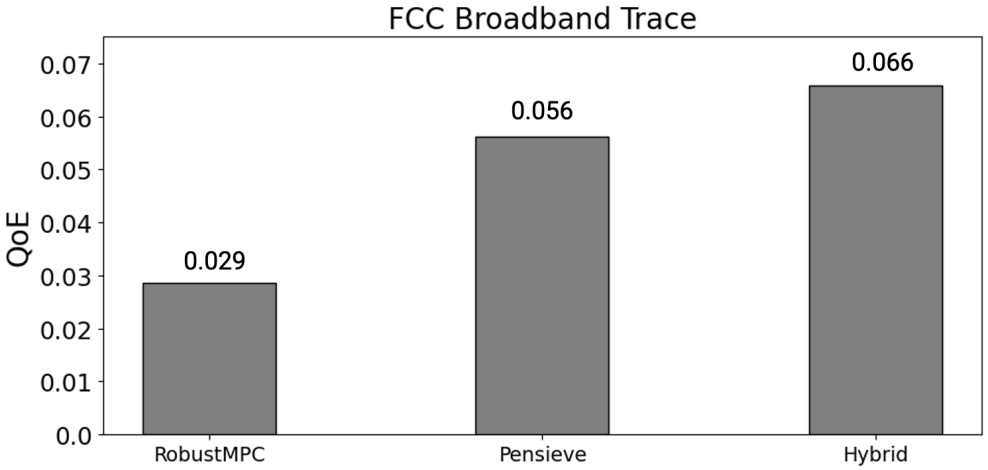

Figure 5.4 Average QoE over unseen FCC broadband traces.

fallback points for improvement.

Chapter 6

Related Work

6.1 RL for Networking Systems

RL has been used for several network systems and many have been showing notable performance. Especially, RL is currently actively looked upon in the domain of WebRTC [13]. Recent research such as Concerto [14], OnRL [8], and Loki [9] are work that apply reinforcement learning to bitrate control in RTC. In works like Loki, although WebRTC is a different domain to ABR, sees an opportunity to leverage rule-based bitrate control to enhance RL performance. Loki introduces a dual fusion attention that converts a rule-based model to a neural network model and fuses it with the existing RL to reach the best of two worlds. This kind of methodology may also be applied to ABR and further we may be able to apply Hybrid to the fused model itself.

In the context of ABR, several works have presented methods to improve the performance of RL. For example, Genet [15] that uses curriculum learning to improve the training process of the RL-based ABR algorithm. Another work [16]

focused on the learning of throughput prediction, which is then used by the MPC [3]. The techniques of RL for network systems are not directly applicable to Hybrid, but used together, some of the works seem to be able to gain even better performance.

6.2 Interpretability of RL in Network Systems

Some research has been proposed regarding the interpretabiliy of RL in network systems. Metis [10] and Trustee [11] are two works that are in this direction. Through imitation learning, these works provide decisions trees that corresponds to the input black-box model. Since the main challenge of Hybrid was about figuring out when the RL model would underperform, these works were helpful in designing Hybrid. Future works on the interpretability of RL are expected to aid in designing systems and algorithms like Hybrid.

Chapter 7

Discussion and Conclusion

7.1 Other Approaches

To solve the problem stated in chapter 3, there can be other solutions. We have tried some of the solutions and have seen some potential. Although these solution are not dealt in depth in this paper, we present the potential solutions and our explorations on them.

Simple multiplexing according to bandwidth

Section 3.1 points out that RobustMPC [3] excelling network conditions are high bandwidth. Using the observation, the first idea would be using rule-based bitrate decisions in high bandwidth situations. Using the Norway traces [5], we have prototyped this idea with the bandwidth threshold 2.5 Mbps. As shown in Table 7.1, this bandwidth-based approach showed higher average QoE compared to RobustMPC but still lower than Pensieve. This is because this simple design does not consider the detailed behaviors of each algorithm resulting in many

	RobustMPC	Pensieve	Bandwidth-based
Average QoE	0.485	0.571	0.54

Table 7.1 Average quality-of-experience of the simple bandwidth-based multiplexing approach compared to RobustMPC and Pensieve.

false positives into where to fallback.

Reward engineering

As we analyzed the behaviors in section 3.2 and section 5.1, we can exploit the information about the behavior patterns of the RL. From analysis we can find that Pensieve has a tendency to overrate smoothness, resulting in *UP* and *DOWN* patterns, not changing bitrates when necessary. Through reward engineering, we attempted to address this. The following is the reward function used for training Pensieve with the REBUF_PENALTY 4.3 and the SMOOTH_PENALTY 1.

$$\begin{aligned} reward = bitrate - REBUF\_PENALTY \times rebuffering\_time \\ - SMOOTH\_PENALTY \times smoothness \end{aligned} \tag{7.1}$$

We have tried different SMOOTH_PENALTYs from 0.9 1.05 and the result is shown in Figure 7.1. The model has been trained with the dataset provided at [4] using simulations as done in the original Pensieve, but since not all the dataset used for training the pretrained Pensieve is provided, the average QoE is not the same even with the original coefficient. We found that 0.975 shows the best performance and 0.95 also showing better performance compared to the original 1.

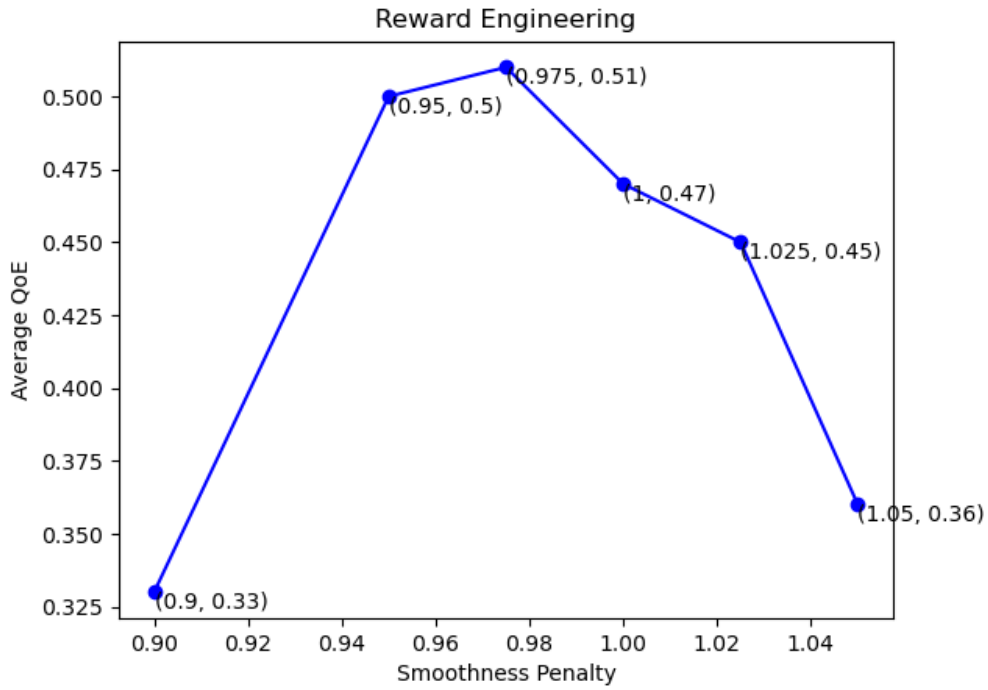

Figure 7.1 **Average QoE according to different smoothness penalty.** Soothness penalty of 0.975 and 0.95 show better performance than the original 1.

	Pensieve	Pensieve-plus
Average QoE	0.47	0.51

Table 7.2 Average quality-of-experience of the original Pensieve and Pensieve with extra-bandwidth added to states over Norway 3G traces [5].

RL agent tuning

One of the reasons that enabled the RL-based algorithm to perform better than the rule-based algorithm in the first place would be that the RL-based model can take various types of network conditions into consideration, while designing a rule that includes multiple conditions is difficult. Also, Hybrid could detect the faulty network conditions exploiting the many network conditions in the pattern detection process. From this observation, another possible approach would be to improve the RL agent itself to use more network conditions as state. Table 7.2 is the result of average QoE of original Pensieve and Pensieve with *extra-bandwidth* added to states. Extra-bandwidth is the value of bandwidth subtracted by the current bitrate. Both RL agents are trained using the dataset provided at [4] under simulations as in the original Pensieve paper. We can see that Pensieve-plus outperforms the original Pensieve leading to hopes that using this technique along with Hybrid would result in even better results.

7.2 Conclusion

In this paper, we have observed that rule-based and reinforcement learning-based ABR algorithms have different strengths and proposed an ABR design that can exploit the best decisions of both worlds. In achieving this goal, we also provide analysis on conditions where RL-based ABR made poor choices. On top of our observations, we have implemented Hybrid a ABR algorithm

that takes offline mapped fallback pattern characteristics and identifies when to utilized the rule-based decision. We have evaluated Hybrid on real network traces and also in unseen traces for generalization.

Although performance gain has been detected, a marginal gain is reported in some scenarios. This is due to the false-negatives and false-positives of the pattern identification. As future work, engineering about observation characteristics to take into account may be a good direction. Also, learning the patterns through deep learning would also be a good exploration route.

Bibliography

[1] 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," *SIGCOMM Comput. Commun. Rev.*, vol. 44, p. 187–198, aug 2014.

[2] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions on Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[3] 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*, SIGCOMM '15, (New York, NY, USA), p. 325–338, Association for Computing Machinery, 2015.

[4] 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*, SIGCOMM '17, (New York, NY, USA), p. 197–210, Association for Computing Machinery, 2017.

[5] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys '13, (New York, NY, USA), p. 114–118, Association for Computing Machinery, 2013.

[6] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, 2011.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[8] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "Onrl: Improving mobile video telephony via online reinforcement learning," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, MobiCom '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[9] H. Zhang, A. Zhou, Y. Hu, C. Li, G. Wang, X. Zhang, H. Ma, L. Wu, A. Chen, and C. Wu, "Loki: Improving long tail performance of learning-based real-time video adaptation by fusing rule-based models," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, (New York, NY, USA), p. 775–788, Association for Computing Machinery, 2021.

[10] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer*

Communication, SIGCOMM '20, (New York, NY, USA), p. 154–171, Association for Computing Machinery, 2020.

[11] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, (New York, NY, USA), p. 1537–1551, Association for Computing Machinery, 2022.

[12] "Federal communications commission. measuring broadband america.." `https://www.fcc.gov/general/measuring-broadband-america`.

[13] N. Blum, S. Lachapelle, and H. Alvestrand, "Webrtc: Real-time communication for the open web platform," *Commun. ACM*, vol. 64, p. 50–54, jul 2021.

[14] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen, "Learning to coordinate video codec with transport protocol for mobile video telephony," in *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[15] Z. Xia, Y. Zhou, F. Y. Yan, and J. Jiang, "Genet: Automatic curriculum generation for learning adaptation in networking," in *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, (New York, NY, USA), p. 397–413, Association for Computing Machinery, 2022.

[16] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: A randomized experiment in video streaming," in *Proceedings of the 17th Usenix Conference on Networked Systems*

Design and Implementation, NSDI'20, (USA), p. 495–512, USENIX Association, 2020.

초록

사용자에게 전달되는 영상의 체감 품질을 높이는 것은 비디오 스트리밍에 있어 중요한 목표이다. 사용자쪽의 동영상 플레이어들은 적응 비트레이트를 도입해 변화하는 네트워크 환경에도 좋은 체감 품질을 전달하고자 했고, 관련해서 많은 규칙 기반 알고리즘들이 개발되어 왔다. 최근에는 강화학습 기반 적응 비트레이트가 제안되어 좋은 성능을 보였는데, 간혹 발생하는 지나친 비트레이트 조정이나 학습된 것과 거리가 먼 환경에서의 좋지 않은 체감 품질 등 이의 단점 또한 부각되고 있다. 본 논문에서는 강화 학습 기반 적응 비트레이트가 잘 동작하지 못하는 경우들을 분석하고, 이를 기반으로 이런 경우들에만 규칙 기반 비트레이트의 결정을 사용하여 보완하는 적응 비트에이트를 제안한다. 또한, 이 디자인을 구현해 Hybrid를 만들고, 이를 실제 네트워크 기록을 활용해 검증한다.

주요어: 강화학습, 적응 비트레이트, 비디오 스트리밍
학번: 2021-22299