360SRL: A SEQUENTIAL REINFORCEMENT LEARNING APPROACH FOR ABR TILE-BASED 360 VIDEO STREAMING

Article · June 2019

CITATIONS
0

READS
50

5 authors, including:

Jun Fu
University of Science and Technology of China
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Xiaoming Chen
University of Science and Technology of China
51 PUBLICATIONS **141** CITATIONS
SEE PROFILE

Zhizheng Zhang
University of Science and Technology of China
11 PUBLICATIONS **4** CITATIONS
SEE PROFILE

Shilin Wu
University of Science and Technology of China
3 PUBLICATIONS **0** CITATIONS
SEE PROFILE

360SRL: A SEQUENTIAL REINFORCEMENT LEARNING APPROACH FOR ABR TILE-BASED 360 VIDEO STREAMING

Jun Fu[1]*, Xiaoming Chen*[1]*, Zhizheng Zhang*[1]*, Shilin Wu*[1]*, Zhibo Chen*[1]

[1] CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System
University of Science and Technology of China, Hefei 230027, China

fujun@mail.ustc.edu.cn, xiaoming_chen29@163.com, {zhizheng, shilinwu, chenzhibo}@ustc.edu.cn

ABSTRACT

Tile-based 360-degree video (360 video) streaming, employed with adaptive bitrate (ABR) algorithms, is a promising approach to offer high video quality of experience (QoE) within limited network bandwidth. Existing ABR algorithms, however, fail to achieve optimal performance in real-world fluctuated network conditions as they heavily rely on unbiased bandwidth predictions. Recently, reinforcement learning (RL) has shown promising potential in generating better ABR algorithms in 2D video streaming. However, unlike existed work in 2D video streaming, directly applying RL in the tile-based 360 video streaming is infeasible due to the resulting exponential decision space. To overcome these limitations, we propose in this paper 360SRL, an improved ABR algorithm employing Sequential RL (360SRL). Firstly, we reduce the decision space of 360SRL from exponential to linear by introducing a sequential ABR decision structure, thus making it feasible to be employed with RL. Secondly, instead of relying on accurate bandwidth predictions, 360SRL learns to make ABR decisions solely through observations of the resulting QoE performance of past decisions. Finally, we compare 360SRL to state-of-the-art ABR algorithms using trace-driven experiments. The experiment results demonstrate that 360SRL outperforms state-of-the-art algorithms with around 12% improvement in average QoE.

Index Terms— adaptive bitrate decision, tile-based streaming , sequential reinforcement learning

1. INTRODUCTION

In recent years, we have witnessed the rapid development and increasing applications of virtual reality (VR) technology. Among the VR technological advancements, 360 video streaming plays an indispensable role in VR applications. Currently, many video service providers, such as YouTube, simply deliver entire 360 video with constant spatial quality to clients, regardless of the fact viewers solely go through a small portion of 360 video at a time, which inevitably results in a waste of bandwidth resources. Using dynamic adaptive streaming over HTTP (DASH) [1], tile-based viewport adaptive streaming [2–6] becomes the mainstream method to deliver 360 video through internet due to its bandwidth-efficient nature. As depicted in Fig. 1, in such a framework, the server encodes the original 360 video constituted of several tiles at multiple bitrate levels and then packages the encoded bitstreams based on DASH protocol. On the other hand, for adaptive viewport transmission, the client first forecasts which parts of 360 video will be viewed by users in the future seconds and determines the priority of each tile according to certain criterion, e.g. the number of pixels within users' field of view (FoV). And then each tile's bitrate level is subsequently decided by adaptive rate (ABR) algorithms. Although the precision of FoV prediction is improved by cross-users based algorithms [3, 4] or content and single-user based algorithms [7, 8], making bitrate decisions can still be challenging owing to (1) the time-varying network condition; (2) the contradictory video QoE requirements (e.g. high quality, minimal rebuffering, spatial and temporal smoothness, etc.); (3) the mutual influence of bitrate decisions (e.g. choosing high bitrates may endanger the playback buffer size and lead to rebuffering and decreasing temporal video quality smoothness in the future); and (4) the exponential solution space (e.g. given that original 360 video is divided into N tiles and each tile has M bitrate levels, the number of the possible bandwidth distribution solutions is M^N).

The majority of relevant ABR algorithms start with combining playback buffer size and throughput signals to estimate the upper bound of available bandwidth resources [2] and then formulate bandwidth distribution as a QoE-driven optimization problem, which can be solved by either greedy algorithms [2–4] or divide and conquer algorithms [6]. However, the bandwidth predictions of these schemes are based on traditional rate-based algorithms [9] and buffer-based algorithms [10]. Therefore, these predictions can be easily biased, which causes inaccurately estimated bandwidth distribution thus severe performance drop of ABR algorithms. To improve the performance of ABR algorithms, one promising

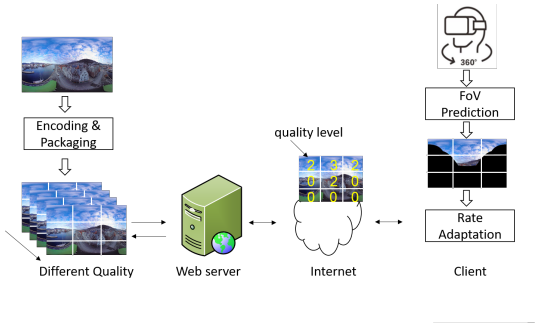


Fig. 1. The general framework of tile-based video streaming

approach is to employ deep reinforcement learning (DRL) in making better ABR decisions. However, the ABR process in tile-based streaming owes exponential solution space, where deep reinforcement (DRL) algorithms [11–13] have difficulty in learning to find an optimal solution over such a huge decision space.

To address these challenges, in this paper, we propose 360SRL, an improved ABR algorithm based on sequential deep reinforcement learning. Inspired by the recent success of sequence-to-sequence model for tackling high-dimension continuous tasks [14], we deploy 360SRL to determine sequentially the bitrate level for each tile rather than predicting bitrate levels for all tiles at one step like traditional RL algorithms [13]. With this formulation, we can get rid of the hindrance of the exponential solution space. Then, we are able to utilize DRL algorithm to learn a policy that directly maps diverse observations concerning the operating environment into the bandwidth distribution solution space without requiring unbiased bandwidth prediction.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first paper to present a highly effective and practical sequential DRL-based algorithm to tackle the ABR problem in the tile-based 360 video streaming system.
- We construct a tile-based streaming simulator, which significantly speeds up the training phase.
- Based on trace-driven experiments, we compare our method with the existing ABR algorithms. The results demonstrate 360SRL achieves the state-of-the-art performance.

The rest of this paper is organized as follows. Section 2 discusses the literature. The proposed method is shown in Section 3 and the experimental results are shown in Section 4. Finally, we conclude the paper and outline future research directions in Section 5.

2. RELATED WORK

In this section, we first introduce conventional ABR algorithms in tile-based 360 video streaming systems, followed by an analysis of applying RL in the ABR tile-based 360 video streaming framework.

2.1. Conventional ABR Algorithms

According to the existing literature, ABR process can be divided into two consecutive steps: 1) predicting the upper bound of available bandwidth, and 2) distributing the predicted bandwidth among tiles. Xie [2] firstly puts forward a target-buffer-based method to predict available maximum bandwidth and then develops a greedy algorithm to acquire a solution of bandwidth distribution for the sake of maximizing QoE score concerning the spatial video quality smoothness and the average video quality. Ban [5] enriches the rule of calculating QoE score by involving the temporal video quality smoothness and conducts the same algorithm to obtain a bandwidth distribution solution. Apart from approaches in [2, 5], Hosseini [6] develops a unique divide and conquer approach for selecting bitrate level. However, the bandwidth prediction is liable to be biased due to that it heavily depends on accurate models of the operating environment. Therefore, in the above methods, the biased bandwidth may lead to a severe drop of QoE (e.g. overestimated bandwidth may drain playback buffer size to a critical level and result in rebuffering and serious jitter in temporal video quality).

2.2. Potential of Applying RL in Tile-based Streaming

Recently, supposing the ABR decision in ordinary video streaming system as a Markov Decision Process (MDP), many schemes (e.g. [11, 12]), have been proposed to learn ABR policies via RL. Unfortunately, compared with ordinary video streaming, the tile-based 360 video streaming framework has exponential solution space at each decision step, which hampers these schemes to be further applied in the tile-based streaming system. On the other hand, although the traditional deep deterministic policy gradient (DDPG) algorithm [13] is suitable for high-dimension continuous tasks, it may easily achieve low performance in high-dimension discrete tasks, as in learning ABR policies for the tile-based 360 video streaming. As a result, it is still challenging for practically utilizing RL algorithms to tackle the ABR problem in tile-based 360 video streaming.

3. PROPOSED METHOD

In this section, we firstly define the ABR problem in tile-based 360 video streaming, followed by introducing the proposed sequential ABR decision structure of 360SRL. And then we further describe the design of 360SRL. Finally, we detail the training methodology of 360SRL.

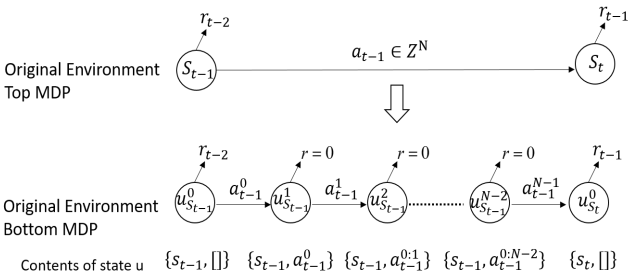


Fig. 2. Sequential ABR decision structure in 360SRL

3.1. Problem Definition

In the tile-based video streaming, the original 360 videos are divided into continous segments with fixed duration T and then each segment consists of N tiles which are independently encoded at M bitrate levels. Consequently, for each segment of 360 video, there are $N \times M$ optional encoded chunks. To ensure continuous playback, it is requisite for clients to prefetch segments from the server. Thus, we intend to find the optimal set of bitrate levels $\{a_0, ..., a_i, ..., a_{N-1}\}$ for tiles in the next segment, where $i \in \{0, ..., N-1\}$ represents z-scan order of the tile and $a_i \in \{0, ..., M-1\}$ stands for the selected bitrate level of i^{th} tile. Besides, we denote q_{i,a_i} and w_{i,a_i} as the quality and corresponding chunk size of i^{th} tile at a_i^{th} bitrate level. In addition, we let $p_i \in [0, 1]$ be the viewing probability of i^{th} tile.

3.2. Sequential ABR Decision Structure

Similar to ordinary video streaming [11], we assume that the ABR decision process in tile-based video streaming can be also regarded as a MDP illustrated by the "Top MDP" in Fig. 2. The "Top MDP" shows that, at a specific time, e.g. t-1, we derive N-dim action $a_{t-1} \in Z^N$ from observed state s_{t-1} and then perform a_{t-1} to the environment which later sends a feedback signal r_{t-1} that reflects the video QoE, where $a_{t-1} \in Z^N$ stands for the set of bitrate levels of tiles. It is to be noted that the "Top MDP" has M^N possible actions at each decision step, which hinders the application of RL. To resolve this problem, we propose a novel sequential ABR decision structure to reduce the number of action in one step from M^N to M. As shown in Fig. 2, we stretch out one transition with a N-dim action in the "Top MDP" into N cascading transitions with a 1-dim action in the "Bottom MDP", which are tied together with Bellman Equation [15]. As shown in the proposed "Bottom MDP", the state of i^{th} dimension contains the original state s_{t-1} and the previous selected action set $a_{t-1}^{0:i-1}$. It is to be noted that the original environment does not make changes until a decision is made for the last dimension. As a result, the rewards of previous N-1 dimensions are set to 0. In this paper, instead of designing a unique agent for each tile, we merely design a common agent to sequentially make bitrate decisions for each dimension.

3.3. Agent Design

Next, we specifically introduce the state, action and reward representation.

- State: For i^{th} dimension, its input states contain original environment state s_t and signals related to the action set of previous dimensions, denoted as $u_{s_t}^i = \{Th, C_i, p_{0:i-1}, q_{0:i-1,a_t^{0:i-1}}, b_t, p_i, S_i, Q_{t-1}\}$ where Th means the past m average throughput of downloading one segment; $C_i \in R^M$ is a vector that represents the available chunk sizes of the i^{th} tile; $p_{0:i-1}$ and $q_{0:i-1,a_t^{0:i-1}}$ are the selected bitrate set and viewing probability set of previous i-1 tiles. b_t is the playback buffer size; p_i is the viewing probability of i^{th} tile; S_i is the sum of selected chunk size of previous $i-1$ tiles, which can be calculated by $S_i = \sum_{h=0}^{i-1} C_{h,a_t^h}$. The Q_{t-1}, at last, records the average video quality of N tiles in the last segment.

- Action: The action space is discrete, and the output of the agent is defined as a value function $f(u_{s_t}^i, a_t^i)$, meaning the value of selected action $a_t^i \in \{0, ..., M-1\}$ being in state $u_{s_t}^i$.

- Reward: Referring to [5, 11], as shown in Eq. 5, we define the reward as a weighted sum of the following factors: average video quality q_t^{avg}, spatial video quality variance $q_t^{s\text{-}v}$, temporal video quality variance $q_t^{t\text{-}v}$, and rebuffering time T_t^r, which are calculated by Eq. 1- 4. Additionally, in Eq. 4, T_t is expressed as the cumulative time to download all tiles.

$$q_t^{avg} = \frac{1}{\sum_{i=0}^{N-1} p_i} \cdot \sum_{i=0}^{N-1} p_i \cdot q_{i,a_i} \tag{1}$$

$$q_t^{s\text{-}v} = \frac{1}{\sum_{i=0}^{N-1} p_i} \cdot \sum_{i=0}^{N-1} p_i \cdot |q_{i,a_i} - q_t^{avg}| \tag{2}$$

$$q_t^{t\text{-}v} = |q_{t-1}^{avg} - q_t^{avg}| \tag{3}$$

$$T_t^r = max\{T_t - b_{t-1}, 0\} \tag{4}$$

$$R_t = w_1 \cdot q_t^{avg} - w_2 \cdot q_t^{s\text{-}v} - w_3 \cdot q_t^{t\text{-}v} - w_4 \cdot T_t^r \tag{5}$$

3.4. Training Methodology

We now detail the training process for 360SRL. In our work, we use Deep Q-Network (DQN) [16] as the fundamental algorithm to learn an action-value function $Q(s_t, a_t; \theta)$ parameterized by θ and a corresponding greedy-policy $\pi(s_t; \theta) = arg\max_{\theta} Q(s_t, a_t; \theta)$. The key thought of DQN network is to

update the parameter in the direction of minimize the following loss:

$$L(\theta) = E[y_t - Q(s_t, a_t; \theta)] \tag{6}$$

with

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}; \theta'); \theta') \tag{7}$$

where θ' represents the parameters of a fixed and separae target network; $r(\cdot)$ is the immediate reward function represented by Eq.5 and $\gamma \in$ [0,1] is a discount factor. To mitigate the overestimation of value function, we introduce the structure of double-DQN [17]. Therefore, Eq. 7 can be rewritten as:

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \pi(s_{t+1}; \theta); \theta') \tag{8}$$

In our work, we set the discount factor $\gamma_b = 1$ in the "Bottom MDP". Making use of the Eq. 6 and 8, the temporal loss for i^{th} dimension is:

$$l_t^i = Q_{target} - Q(u_{s_t}^i, a_t^i; \theta), \forall i \in [0, ...N-1] \tag{9}$$

with

$$Q_{target} = r_t + \gamma_u \cdot Q(u_{s_{t+1}}^0, \pi(u_{s_{t+1}}^0; \theta); \theta'). \tag{10}$$

where γ_u stands for the discount factor in the "Top MDP".

From Eq. 9 and 10, we observe the fact that each dimension has the same target value, which means we can not distinguish the contribution of individual dimension action a_t^i to r_t. Therefore, to overcome this limitation, based on the prior information that the action of certain tile a_t^i is proportional to its viewing probability p_i, we conduct a reward shaping by adding an extra item r_{extra}^i to l_t^i as seen in Eq. 11.

$$l_t^i = r_{extra}^i + Q_{target} - Q(u_{s_t}^i, a_t^i; \theta), \forall i \in [0, ...N-1] \tag{11}$$

And the r_{extra}^i is calculated by Eq. 12 where we set a threshold P and give a negative reward $-a_t^i$ whilst selecting high bitrate for tiles whose viewing probabilities are lower than P.

$$r_{extra}^i = \begin{cases} 0, & p_i > P \\ -a_t^i, & p_i \leq P \end{cases} \tag{12}$$

As a result, the total average loss is formulated as below:

$$l_t^{avg} = \frac{1}{N} \sum_{i=0}^{N-1} l_t^i \tag{13}$$

And then we utilize ordinary gradient descent methods to update our model with a learning rate α:

$$\theta \leftarrow \theta + \alpha \bigtriangledown l_t^{avg} \tag{14}$$

Besides, we apply experience replay method [13] in the training phase to improve the generalization of 360SRL. See more details in algorithm 1 and 2.

Algorithm 1 360SRL's Overall Training Procedure

Initialize action-value network parameters θ
Initialize target network parameters $\theta' \leftarrow \theta$
Initialize replay buffer D
Initialize epoch counter T = 0
for $T < T_{max}$ **do**
  Reset environment state
  repeat
    $s_t \leftarrow$ Current environment state
    $a_t \leftarrow \pi'(s_t; \theta)$
    Execute a_t in the environment receiving r_t, and s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) to D
    Sample B transitions$(s_t, a_t, r_t, s_{t+1})$from D
    Calculate average loss:
    $L^{avg} \leftarrow \frac{1}{B} \cdot \sum_{b=0}^{B-1} l_b^{avg}$
    Update action-value network:
    $\theta \leftarrow \theta + \alpha \bigtriangledown L^{avg}$
    Update the target network:
    $\theta' \leftarrow (1-\tau)\theta' + \tau\theta$
  until epoch done
  $T \leftarrow T + 1$
end for

Algorithm 2 360SRL's Squential Decision Procedure

$a \leftarrow []$
for i = 0 to N **do**
  if $rand() \leq \epsilon$ **then**
    Random sample an action $a^i \in [0, ..., M-1]$
  else
    $a^i \leftarrow \pi(u_{s_t}^i; \theta)$
  end if
  Append a^i to a: $a \leftarrow [a; a^i]$
end for
return a

4. EVALUATION

In this section, we start by explaining the details of setting up experiments and implementing the proposed 360SRL. And then we show a comprehensive quantitative evaluation by comparing 360SRL with traditional ABR methods.

4.1. Experiment Setup

- Network traces: Utilizing the Markovian-based model in [11], we separately generate 200 network traces covering a broad set of network conditions for training, validation, and testing.

- View traces: We leverage the users' orientation dataset collected by [7] and calculate the determined visibility probability of each tile with the assistance of forwarding projection [18].

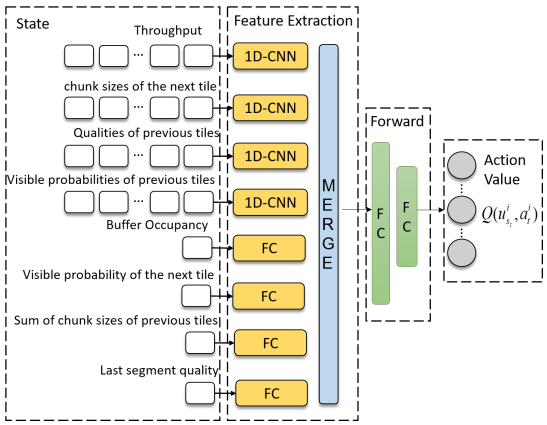


Fig. 3. Architecture of 360SRL

- Video content: We use 10 one-minute panoramic videos with various motion strength [19] as our delivering content. Each video is divided into 30 2-second segments. Each segment consists of 3×3 tiles and is encoded with kvazaar HEVC encoder [20] at 0.1Mbps, 2Mbps, 4Mbps, and 6Mbps. Then, we use MP4box [21] to package the encoded bitstreams into chunks.

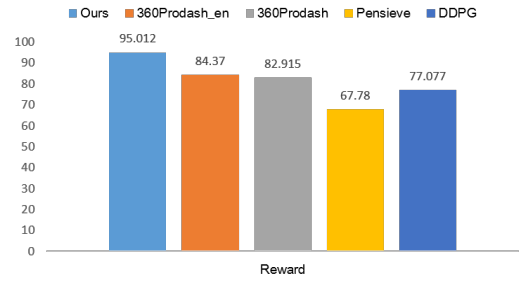
4.2. Implement Details

We use pytorch [22] to implement 360SRL. As demonstrated in Fig. 3, 360SRL is composed of 3 network architectures. Features are extracted from the input state via a feature extraction network. The first 4 inputs are passed through a 1D convolution layer with 128 filters, whose kernel size are 1xm, 1xM, 1xN, and 1xN, whilst the latter 4 inputs are feed in the full connection (FC) layer with 128 neurons. Subsequently, the feature maps are concatenated as a tensor, followed by the forward network. The forward network contains a 1024-neuron and 256-neuron FC layer. The output of the action value network is a M-dim vector. Both feature extraction and forward network use the Leaky-ReLU as the activation function followed by a layer-normalization layer. In the training phase, the capacity of experience replay buffer and the learning rate α are set to 1e6 and 0.001, respectively. Besides, we set the weights of factors in Eq. 5 to 1, 0.5, 1, and 5. Meanwhile, M, N, m, γ_u, and P are set to 4, 9, 8, 0.1 and 0.

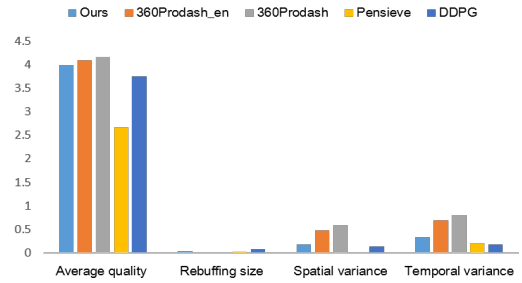
4.3. Experiments and Results

In this experiment, we compare 360SRL with the following methods:

- Pensieve: This approach performs in such a way that the client downloads the entire 360 video with constant spatial quality from the server and uses Pensieve [11] to select the bitrate level.



(a) Average QoE score



(b) Individual factors

Fig. 4. Comparing the existing ABR algorithms with 360SRL

- 360Prodash: This method is implemented on the tile-based 360 video streaming framework. It first adopts the buffer-based algorithm [10] to predict the current available maximum bandwidth and then makes bitrate decisions by solving an optimal optimization problem concerning video quality and spatial video quality variance.

- 360Prodash_en: 360Prodash_en [5] adopts a similar strategy to 360Prodash. However, it strengthens the constraint of the QoE-driven optimization problem by taking temporal video quality variance into consideration.

- DDPG: This approach trains a neural network to learn to make bitrate decisions in tile-based streaming framework utilizing traditional RL algorithm DDPG [13] on the same network traces, users' view traces and video contents as 360SRL.

Firstly, to evaluate the functionality of our proposed unique 360SRL, we compare 360SRL to DDPG. As shown in Fig. 4(a), 360SRL outperforms DDPG resulting in noticeably higher average QoE score. This confirms that the traditional DDPG is liable to diverge to the local optimum in tasks with high-dimension discrete action space. On the contrary, the proposed sequential ABR decision structure aids 360SRL in escaping local optimum in such tasks. Meanwhile, it is noted that the naive strategy of delivering the entire 360 video is worse than all methods based on the tile-based stream-

ing framework due to bandwidth-consuming nature. Moreover, Fig. 4(a) indicates the QoE improvement of 360SRL over 360Prodash_en , 360Prodash, and Pensieve are 12.6%, 14.59% and 40.18%.

Secondly, to better understand the average QoE gains obtained by 360SRL, we further analyzed 360SRL's performance on the individual factors in our QoE definition (Eq. 5). Specifically, Fig. 4(b) compares 360SRL to state-of-the-art ABR algorithms in terms of average video quality, rebuffing size, spatial video quality variance, and temporal video quality variance. As shown in the figure, a large portion of 360SRL's performance gains come from its ability to limit spatial and temporal video quality variance. In comparison with 360Prodash_en, 360SRL reduces spatial and temporal video quality variance by 50.96% and 63.03%, respectively. Additionally, Fig. 4(b) illustrates that 360SRL does not outperform all state-of-the-art schemes on every QoE factor. Instead, 360SRL is able to better reconcile various QoE related factors in a way that optimizes the overall QoE.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose 360SRL, an improved ABR algorithm using RL for the tile-based 360 video streaming framework. In 360SRL, we propose a sequential ABR decision structure to replace the original ABR decision process, which reduces the decision space from being exponential to linear. Unlike traditional QoE-driven approaches, 360SRL utilizes SRL to learn a policy for optimizing the QoE that balances various QoE factors, including average video quality, rebuffering size, spatial video quality variance, and temporal video quality variance. The experimental results prove that 360SRL noticeably outperforms the state-of-the-art ABR algorithms in terms of resulting QoE. In our future work, we plan to further accelerate the training process by employing asynchronous training methods. Moreover, we also plan to extend 360SRL to other 360 video streaming frameworks, e.g. asymmetric panorama adaptive streaming [23].

6. ACKNOWLEDGEMENTS

This work was supported in part by NSFC under Grant 61571413 and 61390514.

7. REFERENCES

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

[2] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo, "360probdash: Improving qoe of 360 video streaming using tile-based http adaptive streaming," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 315–323.

[3] Lan Xie, Xinggong Zhang, and Zongming Guo, "Cls: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming," in *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 564–572.

[4] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang, "Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[5] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yueyu Hu, "An optimal spatial-temporal smoothness approach for tile-based 360-degree video streaming," in *Visual Communications and Image Processing (VCIP), 2017 IEEE*. IEEE, 2017, pp. 1–4.

[6] Mohammad Hosseini and Viswanathan Swaminathan, "Adaptive 360 vr video streaming: Divide and conquer," in *Multimedia (ISM), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 107–110.

[7] Yanyu Xu, Yanbing Dong, Junru Wu, Zhengzhong Sun, Zhiru Shi, Jingyi Yu, and Shenghua Gao, "Gaze prediction in dynamic 360 immersive videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5333–5342.

[8] C Fan, Jean Lee, W Lo, C Huang, K Chen, and Cheng-Hsin Hsu, "Fixation prediction for 360 video streaming to head-mounted displays," *Proceedings of ACM NOSSDAV 2017*, 2017.

[9] Junchen Jiang, Vyas Sekar, and Hui Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 1, pp. 326–340, 2014.

[10] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[11] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 197–210.

[12] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, and Andrea Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[13] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson, "Discrete sequential prediction of continuous actions for deep rl," *arXiv preprint arXiv:1705.05035*, 2017.

[15] Satinder Singh, Peter Norvig, David Cohn, et al., "How to make software agents do the right thing: An introduction to reinforcement learning," *Adaptive Systems Group*, 1996.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[17] Hado Van Hasselt, Arthur Guez, and David Silver, "Deep reinforcement learning with double q-learning.," in *AAAI*. Phoenix, AZ, 2016, vol. 2, p. 5.

[18] Matt Yu, Haricharan Lakshman, and Bernd Girod, "A framework to evaluate omnidirectional video coding schemes," in *2015 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2015, pp. 31–36.

[19] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu, "360 video viewing dataset in head-mounted virtual reality," in *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017, pp. 211–216.

[20] Marko Viitanen, Ari Koivula, Ari Lemmetti, Jarno Vanne, and Timo D Hämäläinen, "Kvazaar hevc encoder for efficient intra coding," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 1662–1665.

[21] Jean Le Feuvre, Cyril Concolato, and Jean-Claude Moissinac, "Gpac: open source multimedia framework," in *Proceedings of the 15th ACM international conference on Multimedia*. ACM, 2007, pp. 1009–1012.

[22] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan, "Pytorch," 2017.

[23] Zhimin Xu, Xinggong Zhang, Kai Zhang, and Zongming Guo, "Probabilistic viewport adaptive streaming for 360-degree videos," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.