QoE-Based MEC-Assisted Predictive Adaptive Video Streaming for On-Road Driving Scenarios

Wanting Yang, *Graduate Student Member, IEEE*, Xuefen Chi, Linlin Zhao, *Member, IEEE*, and Zehui Xiong, *Member, IEEE*

***Abstract*—This letter studies a multi-access edge computing assisted predictive adaptive video streaming scheme for on-road driving scenarios based on deep reinforcement learning (DRL). By judiciously designing the state and the reward, the user movement awareness is integrated into our scheme, which enables it to make proactive decisions that can maximize the long-term quality of experience. To enhance the learning efficiency, we introduce the post-decision state into DRL. The effectiveness of the scheme has been validated by simulation results.**

***Index Terms*—Adaptive video streaming, PDS-DQN, QoE, mobility-aware, MEC.**

I. Introduction

ONLINE video applications have been booming. However, the transmission in the dynamic wireless link is still a bottleneck [1]. Especially for the vehicular users, their wireless channel gains may vary considerably over time. The high fluctuation in bandwidth share imposes a serious challenge to preserve the desired user quality of experience (QoE).

Considering that the multi-access edge computing (MEC) server has a better viewpoint of the competing users' wireless channel conditions than the individual users, some research has transferred the intelligence of adaptive video streaming (AVS) from user side to MEC side. In recent years, Chang *et al.* presented an edge-assisted AVS scheme in [2]. The superiority of the real-time reaction to dynamic channel conditions of MEC-side schemes has been demonstrated. Then, the authors in [1] and [3] further demonstrated the gain of the resource utilization brought by the centralized video quality assignment. However, the AVS scheme in both works can only provide a best-suited QoE value in the current environment.

Recently, some research has resorted to reinforcement learning (RL) paradigm to optimize the long-term QoE. In [4], Luo *et al.* formulated the joint optimization problem including video quality adaptation as a Markov decision process (MDP). Unfortunately, all these works only considered the small-scale fading in channel modeling. The finite state Markov channel model adopted in their work fails to capture the large-scale channel gain variations caused by user movement, which makes their schemes only applicable to walking-speed scenarios.

Atawia *et al.* were one of the few research teams working on video delivery strategies in fast moving scenarios [5]. In [5], Atawia *et al.* proposed a multi-user predictive AVS (PAVS) scheme based on the predicted channel variations to achieve the long-term fairness in video-quality level assignment. They addressed the issue of quality assignment lagging behind the channel variation in the fast moving scenarios. However, only two metrics, average quality and video stalling, are considered in [5], when evaluating user QoE. The most applicable metric in on-road driving scenarios, i.e., quality level switching, was not well addressed, possibly because it made the high-dimensional optimization problem more challenging.

Different from the predict-and-optimize PAVS scheme in [5], we propose an end-to-end PAVS based on deep Q-network (DQN), where the three QoE metrics[1] mentioned above are all considered. To make it work in the on-road driving scenarios, we introduce a new assisted time scale to record the large-scale channel variation in fast moving scenarios, from which the user mobility information, such as driving velocity and direction, can be extracted. In this sense, by integrating the variation of historical channel gains into the state, the channel prediction can be naturally captured by the delivery controller, which enables our scheme to make proactive decisions to maximize long-term QoE even with large fluctuations in average channel gains.

Moreover, specifically for the partially-known MDP model in our scheme, we introduce a post-decision state (PDS) into DQN and propose an amalgamated PDS-DQN algorithm. By integrating the available part of model into the design of the Q-network, the action-value function can be estimated by training a smaller neural network (NN) designed for the PDS-value function. The superiority of PDS-DQN over conventional DQN in terms of learning efficiency has been proven in simulations and the effectiveness of our scheme has been validated by the comparison with baseline algorithm in [5]. Moreover, simulation results show that transfer learning can speed up convergence significantly, which allows for the possibility of online training for refining the models.

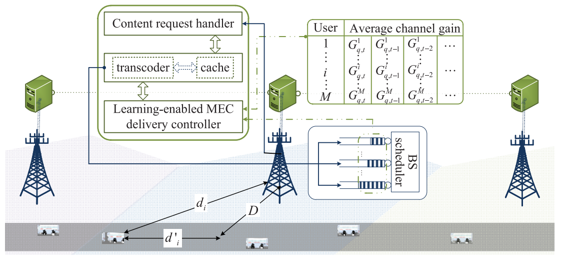
Manuscript received July 26, 2021; accepted August 19, 2021. Date of publication August 23, 2021; date of current version November 9, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61801191; in part by the Jilin Scientific and Technological Development Program under Grant 20200401147GX; and in part by the SUTD under Grant SRG-ISTD-2021-165. The associate editor coordinating the review of this article and approving it for publication was A. Alsharoa. *(Corresponding author: Xuefen Chi.)*

Wanting Yang, Xuefen Chi, and Linlin Zhao are with the Department of Communications Engineering, Jilin University, Changchun 130012, China (e-mail: yangwt18@mails.jlu.edu.cn; chixf@jlu.edu.cn; zhaoll13@mails.jlu.edu.cn).

Zehui Xiong is with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design, Singapore 487372 (e-mail: zehui_xiong@sutd.edu.sg).

Digital Object Identifier 10.1109/LWC.2021.3106945

[1] 1) *Average quality*: The average chunk quality level assigned in a video duration; 2) *Video stalling*: The duration of video stalling in a video duration; 3) *Quality switch*: The number of quality level switches in a video duration.



Content request handler
transcoder
cache
Learning-enabled MEC delivery controller
User
Average channel gain
1
i
M
$G^1_{q,t}$ $G^1_{q,t-1}$ $G^1_{q,t-2}$ …
$G^i_{q,t}$ $G^i_{q,t-1}$ $G^i_{q,t-2}$ …
$G^M_{q,t}$ $G^M_{q,t-1}$ $G^M_{q,t-2}$ …
BS scheduler
d_i
D
d'_i

Fig. 1. Scenario description.

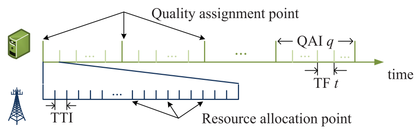


Fig. 2. Timescale illustration.

II. System Model

As shown in Fig. 1, we consider a road across multiple cells. Each base station (BS) is connected to a MEC server with transcoding technology and serves an active vehicular video user set denoted by $\mathcal{M}$, where the user is indexed by $i \in \mathcal{M}$. The MEC server is responsible for making a video quality assignment strategy and finishing transcoding, as well as delivering the video contents to the BS. The BS is responsible for scheduling and transmitting video packets to the users. Assume that the BS maintains individual queues to cache the packets waiting to be scheduled for users. Moreover, the delivery model in MEC server mainly consists of three units. The MEC server may well have cached the highest quality level video contents requested by users with the edge caching technology. If the video content is not pre-cached, the content request handler unit redirects and downloads video content to local cache [7]. Then, the cache and transcoder unit transcodes the video content to the favourable quality levels if necessary under the control of the learning-enabled delivery controller.

The video contents stored in the local cache are divided into multiple video chunks with equal duration τ_{chunk}, which is the minimal unit for transcoding. The video playback starts after the user receives the first entire video chunk. The period between the two quality level assignment time points is called the quality assignment interval (QAI) and indexed by q. We equalize the QAI duration τ_{QAI} and τ_{chunk}.[2] Each video chunk can be transcoded into several quality levels denoted by $\mathcal{L} = \{1, 2, \ldots, L\}$. Moreover, let $l^i_q \in \mathcal{L}$ denote the quality level assigned to user i in QAI q. A higher value of l^i_q means a higher video quality. Accordingly, the average bitrate of the video chunk assigned to user i in QAI q is denoted by $R(l^i_q)$.

To capture the trend of the user average channel gain, we introduce an intermediate time scale called time frame (TF) between the QAI and the transmission time interval (TTI) between the two resource allocation time points. During a TF, the large-scale channel gain is assumed to remain constant. A QAI consists of T TFs and the TF in a QAI is indexed by $t \in (0, T]$. The relationship of the three time scales is shown in Fig. 2. Assume that the average channel gain experienced by each user i within TF t in QAI q (denoted by $G^i_{q,t}$) can be obtained by the delivery controller via accessing the channel status information in the intrinsic Radio Network Information Service of MEC [6]. To facilitate the handover management for maintaining the user QoE, we assume the user historical channel gain information can be shared with the adjacent BSs.

[2]In conventional AVS schemes, the value of τ_{QAI} depends on the transmission duration of a video chunk. Whereas, in the steady state, τ_{QAI} should be approximately equal to τ_{chunk}. To facilitate the delivery controller's learning, we set $\tau_{\text{QAI}} = \tau_{\text{chunk}}$ from the beginning of training process.

III. Adaptive Video Streaming With DRL

A. Reinforcement Learning Framework

In this letter, the delivery controller acts as an agent. The time step in RL framework is specified as an AQI and Q time steps are treated as an episode. At each time step q, the agent observes the *state* $\mathbf{s}_q$ of the environment and executes an *action* $\mathbf{a}_q$. Then, the agent receives a *reward* r_q from the environment and transits into a new state $\mathbf{s}_{q+1}$. The goal of the agent is to learn a mapping from $\mathbf{s}_q$ to $\mathbf{a}_q$ (i.e., policy π) for maximizing an expected accumulated reward $\mathbb{E}[\sum_{q=1}^{Q-1} \gamma^{q-1} r_q]$ during an episode, where γ denotes the discount factor. The detailed design of the RL framework is given as follows:

1) State: In each time step q, to keep the video quality consistency, the video-quality level assigned in the last AQI should be considered and is denoted by $\mathbf{s}^{\text{L}}_q = (l^1_{q-1}, l^2_{q-1}, \ldots, l^M_{q-1})^{\text{T}}$. In addition, to reduce the video stalling events, the queue length in BS also must be considered, which is denoted by $\mathbf{s}^{\text{B}}_q = (B^1_q, B^2_q, \ldots, B^M_q)^{\text{T}}$, where B^i_q denotes the queue length for user i at the start of QAI q. Moreover, the playback smoothness depends heavily on the current channel states. On top of that, to capture the channel gain variations in the on-road driving scenarios, the long-term channel prediction should also be integrated into the RL paradigm. To this end, the average channel gains in the past TFs with the last N_{Q} QAIs should also be included into $\mathbf{s}_q$ which can help the agent capture the user mobility pattern [8]. These state values related to the channel gain can be constructed as an M-row and $(N_{\text{Q}} \times T)$-column matrix denoted by $\mathbf{s}^{\text{G}}_q = [\mathbf{G}_{q-1,T}, \ldots, \mathbf{G}_{q-1,1}, \ldots, \mathbf{G}_{q-N_{\text{Q}}+1}]$, where $\mathbf{G}_{q,t} = [G^1_{q,t}, G^2_{q,t}, \ldots, G^M_{q,t}]^{\text{T}}$. From the above analysis, the state in step q can be expressed by a matrix $\mathbf{s}_q = [\mathbf{s}^{\text{L}}_q, \mathbf{s}^{\text{B}}_q, \mathbf{s}^{\text{G}}_q]$ with M rows and $(N_{\text{Q}} \times T + 2)$ columns.

2) Action: In time step q, the agent determines the assignment of video quality levels according to $\mathbf{s}_q$, i.e., $\mathbf{a}_q = (l^1_q, l^2_q, \ldots, l^M_q)^{\text{T}} \in \mathcal{L}^M$ to achieve a desired long-term QoE.

3) Reward: The long-term goal of the agent is formalized in the immediate reward passing from the environment in each time step. In order to characterize the effect of $\mathbf{a}_q$ on the QoE, the reward r_q aims to capture a weighted combination of the three metrics in QoE evaluation. Thus, r_q is defined as

$$r_q = \sum_{i=1}^{M} l^i_q - \lambda \sum_{i=2}^{M} \left| l^i_q - l^i_{q-1} \right|^{\alpha} - \kappa \sum_{i=1}^{M} \log\left(B^i_{q+1} + 1\right), \quad (1)$$

where the parameters λ, κ, and α are weight coefficients. The positive reward of high quality level and the negative reward of quality level switching, i.e., the first two parts in (1), are two explicit functions about $\mathbf{a}_q$ and $\mathbf{s}^{\text{L}}_q$. Whereas, the playback smoothness cannot be explicitly expressed, which not

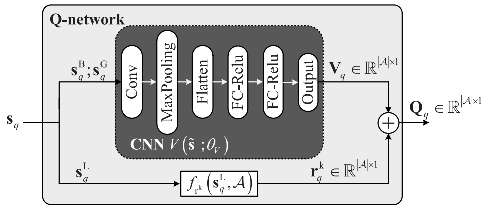

Q-network
$\mathbf{s}_q$
$\mathbf{s}_q^{\mathrm{B}};\mathbf{s}_q^{\mathrm{G}}$
Conv
MaxPooling
Flatten
FC-Relu
FC-Relu
Output
CNN $V(\tilde{\mathbf{s}};\theta_V)$
$\mathbf{V}_q \in \mathbb{R}^{|\mathcal{A}|\times 1}$
$\mathbf{Q}_q \in \mathbb{R}^{|\mathcal{A}|\times 1}$
$\mathbf{s}_q^{\mathrm{L}}$
$f_{\mathrm{r}^{\mathrm{k}}}(\mathbf{s}_q^{\mathrm{L}},\mathcal{A})$
$\mathbf{r}_q^{\mathrm{k}} \in \mathbb{R}^{|\mathcal{A}|\times 1}$

Fig. 3. Q-network structure.

only depends on $\mathbf{a}_q$ and $\mathbf{s}_q$, but relies on the scheduling mode adopted by the BS. We assume that the action in each QAI can been fully implemented. The playback smoothness of video content assigned to the users is mainly affected by the channel gains. In this sense, we use $\mathrm{s}_{q+1}^{\mathrm{B}}$ to quantitatively characterize playback smoothness. If the queue length is larger than zero, which means that the quality level assigned to the user is too high to allow the video chunk to be transmitted in QAI q, the third part will serve as a stalling penalty to the reward.

B. PDS-DQN Structure

Based on the immediate reward defined in (1), the action value, which the agent aims to learn a policy π to maximize, is defined as $Q(\mathbf{s}_q, \mathbf{a}_q) \triangleq \mathbb{E}[\sum_{i=q}^{Q-1} \gamma^i r_i | \mathbf{s}_q, \mathbf{a}_q]$. The optimal action-value function is defined as $Q^*(\mathbf{s}_q, \mathbf{a}) \triangleq \max_\pi Q_\pi(\mathbf{s}_q, \mathbf{a}_q)$, from which the optimal policy can be decided as $\pi^* = \arg\max_{\mathbf{a}} Q^*(\mathbf{s}_q, \mathbf{a})$. The Bellman equation for $Q^*(\cdot)$ can be expressed by

$$Q^*\big(\mathbf{s}_q, \mathbf{a}_q\big) = \mathbb{E}\Big[r_q + \gamma \max_{\mathbf{a}} Q^*\big(\mathbf{s}_{q+1}, \mathbf{a}\big)\big|\mathbf{s}_q, \mathbf{a}_q, \pi^*\Big]. \quad (2)$$

Considering that the state lies in continuous space and the action lies in discrete space, we resort to DQN and employ a deep neural network (NN) to learn the optimal action value.

From the RL model defined above, we can see that both the values of the first two parts in the reward function and the first state vector $\mathbf{s}_{q+1}^{\mathrm{L}}$ in the next time step can be determined immediately after the execution of $\mathbf{a}_q$. To leverage the partially known model to enhance the learning efficiency, we introduce a PDS $\tilde{\mathbf{s}}_q = [\mathbf{a}_q, \mathbf{s}_q^{\mathrm{B}}, \mathbf{s}_q^{\mathrm{G}}]$ between $\mathbf{s}_q$ and $\mathbf{s}_{q+1}$ [8]. The transition process of $\mathbf{s}_q \rightarrow \tilde{\mathbf{s}}_q$ is denoted by $\tilde{\mathbf{s}}_q = \mathscr{T}_{\mathrm{PDS}}(\mathbf{s}_q, \mathbf{a}_q)$ for short. Accordingly, the reward function is further decomposed as $r_q = r_q^{\mathrm{k}} + r_q^{\mathrm{u}}$. The reward r_q^{k} is the reward received from transition $\mathbf{s}_q \rightarrow \tilde{\mathbf{s}}_q$, i.e., $r_q^{\mathrm{k}} = \sum_{i=1}^{M} l_q^i - \lambda \sum_{i=1}^{M} |l_q^i - l_{q-1}^i|^\alpha$, which is denoted by $r_q^{\mathrm{k}} = f_{\mathrm{r}^{\mathrm{k}}}(\mathbf{s}_q, \mathbf{a}_q)$ for brevity. In this sense, the reward r_q^{u} corresponds to the video stalling penalty. As the effect on the playback smoothness of action $\mathbf{s}_q^{\mathrm{L}}$ executed in QAI $q-1$ has been incorporated into $\mathbf{s}_q^{\mathrm{B}}$, the value of r_q^{u} only relies on $\tilde{\mathbf{s}}_q$. Thereby, r_q^{u} can be regarded as the reward received form transition $\tilde{\mathbf{s}}_q \rightarrow \mathbf{s}_{q+1}$. Define the *PDS-value function* of $\tilde{\mathbf{s}}_q$ as the expected accumulated reward started from $\tilde{\mathbf{s}}_q$, i.e., $V^*(\tilde{\mathbf{s}}_q) \triangleq \mathbb{E}[r_q^{\mathrm{u}} + \sum_{i=q+1}^{Q-1} \gamma^i r_i | \tilde{\mathbf{s}}_q, \pi^*]$. Then, the Bellman equation (2) can be re-expressed as follows:

$$V^*\big(\tilde{\mathbf{s}}_q\big) = \mathbb{E}\Big[r_q^{\mathrm{u}} + \gamma \max_{\mathbf{a}} Q^*\big(\mathbf{s}_{q+1}, \mathbf{a}_q\big)\Big], \quad (3)$$

$$Q^*\big(\mathbf{s}_q, \mathbf{a}_q\big) = \mathbb{E}\Big[r_q^{\mathrm{k}} + V^*\big(\tilde{\mathbf{s}}_q\big)\Big]. \quad (4)$$

Algorithm 1 PDS-DQN Based Adaptive Video Streaming

```
1:  Initialize reply memory $\mathcal{D}$
2:  Initialize the weights of two CNNs $\boldsymbol{\theta}_V^{\mathrm{C}}$ and $\boldsymbol{\theta}_V^{\mathrm{T}}$ with random weights
3:  for episode = 1 to MAX_EPISODE do
4:      Initialize s
5:      for $q = 1$ to $Q$ do
6:          With probability $\varepsilon$ select a random $\mathbf{a}_q$
7:          otherwise select $\mathbf{a}_q = \arg\max_{\mathbf{a}} Q(\mathbf{s}_q, \mathbf{a})$
8:          Execute action $\mathbf{a}_q$
9:          for $t = 1$ to $T$ do
10:             Execute packet scheduling algorithm within QAI $q$
11:         end for
12:         Calculate reward $r_q$ based on (1)
13:         $\mathbf{s}_{q+1} \leftarrow \mathbf{s}_q$
14:         Record transition $\mathbf{e}_q = [\mathbf{s}_q, \mathbf{a}_q, r_q, \mathbf{s}_{q+1}]$ in $\mathcal{D}$
15:         Sample random $\mathcal{B}$ of transitions $[\mathbf{s}_m, \mathbf{a}_m, r_m, \mathbf{s}_{m+1}]$ from $\mathcal{D}$
16:         $y_m = r_m + \max_{\mathbf{a}}\Big(r_{m+1}^{\mathrm{k}} + V\Big(\mathscr{T}_{\mathrm{PDS}}(\mathbf{s}_{m+1}, \mathbf{a}); \boldsymbol{\theta}_V^{\mathrm{T}}\Big)\Big)$
17:         Perform a gradient descent step based on (5) with respect to $\boldsymbol{\theta}_V^{\mathrm{C}}$
18:         Every TARGET_REPLACE time steps reset $\boldsymbol{\theta}_V^{\mathrm{T}} \leftarrow \boldsymbol{\theta}_V^{\mathrm{C}}$
19:     end for
20: end for
```

As the PDS $\tilde{\mathbf{s}}_q$ and r_q^{k} become deterministic given that the agent executes action $\mathbf{a}_q$ at state $\mathbf{s}_q$, (4) can be rewritten as $Q^*(\mathbf{s}_q, \mathbf{a}_q) = r_q^{\mathrm{k}} + V^*(\tilde{\mathbf{s}}_q)$. Based on this, we use a convolution neural network (CNN) to approximate $V^*(\tilde{\mathbf{s}}_q)$ and the approximated $Q^*(\mathbf{s}_q, \mathbf{a}_q)$ can be expressed as $Q(\mathbf{s}_q, \mathbf{a}_q; \boldsymbol{\theta}_V) = f_{\mathrm{r}^{\mathrm{k}}}(\mathbf{s}_q, \mathbf{a}_q) + V(\mathscr{T}_{\mathrm{PDS}}(\mathbf{s}_q, \mathbf{a}_q); \boldsymbol{\theta}_V)$, where $\boldsymbol{\theta}_V$ is the CNN weight set to be trained. The Q-network is as shown in Fig. 3.

To help the agent find the optimal policy, the ε-greedy strategy is utilized to choose the current action. In order to keep the learning stability, we use two CNNs with the same structure and the initial weights. One named *current network* is used to choose the action, the weights of which are denoted by $\boldsymbol{\theta}_V^{\mathrm{C}}$ and update in each training step. The other named *target network* is used to calculate the Q-value, the weights of which are denoted by $\boldsymbol{\theta}_V^{\mathrm{T}}$ and update periodically according to $\boldsymbol{\theta}_V^{\mathrm{C}}$. Additionally, the experience replay strategy is adopted to break the correlations among data samples in training. The experience vector composed by $\mathbf{e}_q = [\mathbf{s}_q, \mathbf{a}_q, r_q, \mathbf{s}_{q+1}]$ is recorded in the reply memory $\mathcal{D}$. In each time step, the agent samples a mini-batch of the experiences $\mathcal{B}$ from $\mathcal{D}$ to train the Q-network. The loss function used in training is given by

$$F_{\mathrm{loss}} = \mathbb{E}\bigg[\Big(y_m - \Big(r_m^{\mathrm{k}} + V\Big(\mathscr{T}_{\mathrm{PDS}}(\mathbf{s}_m, \mathbf{a}_m); \boldsymbol{\theta}_V^{\mathrm{C}}\Big)\Big)\Big)^2\bigg], \quad (5)$$

where $y_m = r_m + \max_{\mathbf{a}}(r_{m+1}^{\mathrm{k}} + V(\mathscr{T}_{\mathrm{PDS}}(\mathbf{s}_{m+1}, \mathbf{a}); \boldsymbol{\theta}_V^{\mathrm{T}}))$. The PDS-DQN based adaptive video streaming algorithm is described in Algorithm 1.

IV. Simulation Results

A. Simulation Setup

In our simulation, we focus on a cell with the radius of 500 m, where four users move along a road. The channel gain is modeled as $35.3 + 37.6\log_{10}(d)$ in dB [8], where d is the distance between a user and BS in meters. The transmitting power and noise power are given as 46 dBm and -95 dBm. Assume that the proportional fair scheduling algorithm is applied to finish user scheduling in each TTI. The actual transmission rate during each TTI is determined by

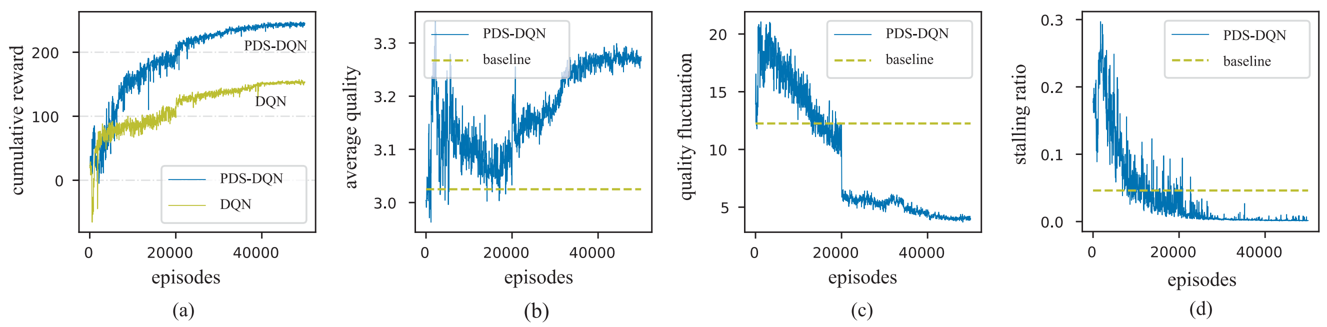


Fig. 4. Learning curves of cumulative reward and the three QoE metrics for the scenario with the first four users in Table I. The cureves are smoothed by averaging over a window of 40 episodes. The exploration probability ε linearly decreases from 0.2 to 0 within the first 40000 episodes. The penalty coefficients in (1) are set to $\lambda = 0.3$, $\alpha = 2$, and $\kappa = 1$.

TABLE I
PARAMETERS FOR USER GENERATION

User index	Initial distance d'[1]	Drive velocity	User index	Initial distance	Drive velocity	User index	Initial distance	Drive velocity
1	150 m	37.8 km/h	2	297 m	36 km/h	3	450 m	36 km/h
4	500 m	32.4 km/h	5	180 m	39.6 km/h	6	294 m	41.4 km/h
7	450 m	41.4 km/h	8	480 m	32.4 km/h			

[1] The distance d' is the distance from the user initial position to the BS along the road. The initial communication distance between the user and the BS can be calculated as $d_i = \sqrt{D^2 + d'^2}$, as shown in Fig. 1. Moreover, all users drive towards to the BS at the initial moment.

the selected Modulation and Coding Scheme [9]. Besides, the MEC server jointly considers the average channel gain in the past two QAIs, i.e., $N_{\mathrm{Q}} = 2$. The duration of QAI, TF, and TTI is set to 3 s, 1 s, and 1 ms, respectively. The videos requested by the users can be transcoded into four video-quality levels with the average bitrates 8 Mbps, 10 Mbps, 15 Mbps, and 20 Mbps, respectively. Assuming that the packet size is 1000 Bytes, we model the number of the packets generated in MEC transcoder in each TTI as independent identical distributed Poisson distribution.

Assume that the distance D between the BS and the road is 200 m. The initial positions of the users are scattered along the road in the cell. The initial velocity ranges from 30 to 45 km/h and each user's acceleration in each TF is drawn from the Gaussian distribution with zero mean and standard deviation of 0.3 m/s^2 [8]. The other generating parameters of the eight users involved in the simulations are listed in Table I.

B. Fine-Tuned Parameters in DQN

With the aim of optimizing the long-term QoE, the discount factor is set to $\gamma = 1$. Since the state elements have different ranges, the buffer states and the channel gains are scaled as their natural logarithm before going through the Q-network. The input matrix first goes through a convolution layer with 30 out-channels and kernel size of 2, and then goes through a max-pooling layer with kernel size of 2 and stride of 1. Then the output matrix is flattened into a one-dimensional vectors and put to three fully-connected layers with 400 nodes, 300 nodes, and 256 nodes respectively. The initial weights obey the normal distribution with zero mean and standard deviation of 0.1. Except for the max-pooling layer and the output layer, the other layers employ batch normalization and then are full connected with rectified linear unit as the activation function. Besides, Adam is used to adjust the learning rate during training, and the initial learning rate is 10^{-5}. The number of the time steps in an episode (i.e., Q) is set to 20. The target network is updated at a frequency of once every ten time steps. The replay memory size is $|\mathcal{D}| = 10^4$ and the mini-batch size is $|\mathcal{B}| = 64$.

C. Performance Evaluation

As it is still challenging to give a rigorous analytical proof of the convergence, the convergence of our scheme may deserve further study, which is beyond the scope of this letter. Nonetheless, we have provided the simulation results to numerically validate the convergence. As shown in Fig. 4(a), the algorithms can converge to a favorable policy after sufficient exploration and learning steps. The time complexity of the PDS-DQN algorithm is O(MAX_EPISODE $\times$ Q). With NVIDIA GeForce GTX 1650 GPU, the average computational time in each training step is 0.006 s. Moreover, the superiority of the proposed PDS-DQN algorithm in terms of both the convergence speed and the cumulative reward compared to the non-PDS counterpart has been demonstrated.[3]

For validating the effectiveness, the learning curves for the three QoE metrics[4] in the above training process are shown respectively in Fig. 4(b)–Fig. 4(d). It is to be pointed out that the exploration of agent is to select a random action within $\mathcal{A}$ in the first half of the training process, and in the second half, it is to randomly increase or decrease a quality level based on the action given by the current policy. Thereby, the quality fluctuation index decreases significantly after 20000 episodes. Moreover, we treat the PAVS scheme in [5] as the baseline scheme.[5] Since the mobility is assumed to be known in the baseline scheme, we consider the same four-user scenario, where the users keep the constant velocity of their initial velocity. It can be seen that our algorithm has a significant advantage in terms of both video quality and video fluctuation after convergence. Moreover, the channel gain variations are shown in Fig. 5(a). By comparing Fig. 5(b) and Fig. 5(c), it can be seen that both PAVS schemes can well capture the user's

[3]In non-PDS counterpart, the NN's structure and the hyper-parameters is same as that in the Section IV-B, except that the complete state matrix is treated as the NN's input.

[4]The quality fluctuation index is calculated as $\sum_{q=2}^{Q} |l_q^i - l_{q-1}^i|^2$. The stalling ratio is the ratio of the cumulative delay and the episode duration, where the delay in QAI q is calculated as $B_q^i / R(l_q^i)$.

[5]To ensure fairness, we assume that the average data rate can be predicted perfectly based on the known user traces.

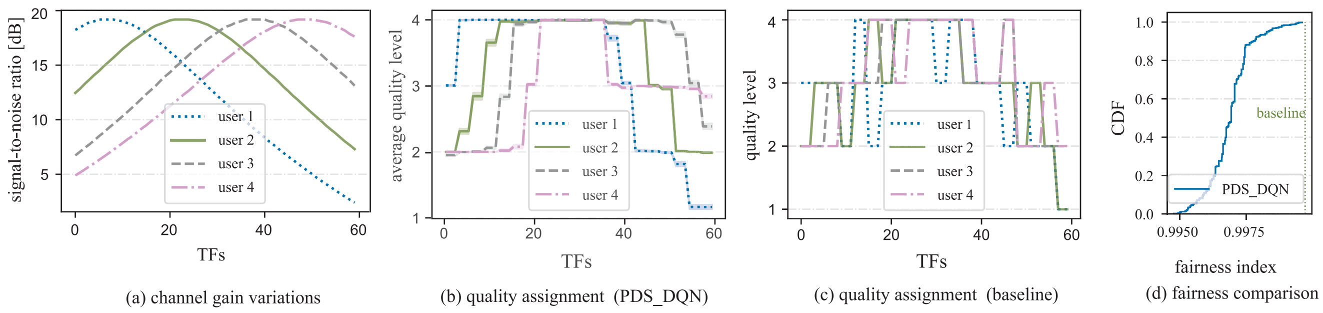


Fig. 5. Comparison of quality assignment strategies in two schemes for the scenario with the first four users with constant velocity of the initial velocity in Table I. The curves of our scheme in Fig. 5(b) and Fig. 5(d) are drawn based on results in the successive 500 episodes after convergence.

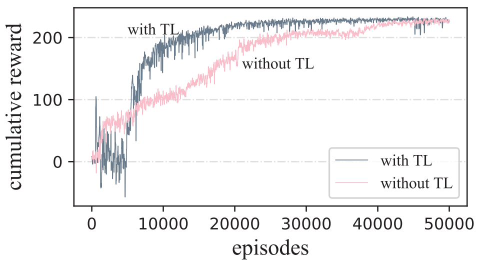


Fig. 6. Learning curves of cumulative reward with TL and without TL for the scenario with the last four users in Table I. The curves are smoothed by averaging over a window of 40 episodes. The exploration probability ε linearly decreases from 0.1 to 0 and 0.2 to 0 within the first 20000 and 40000 episodes, respectively.

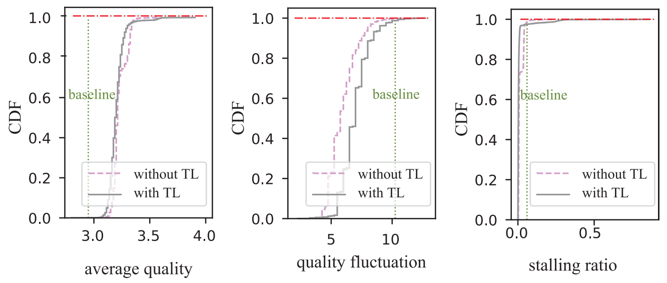


Fig. 7. The CDFs of the three QoE metrics in episode 20000 to episode 30000 in the training process with TL and episode 40000 to episode 50000 in the counterpart without TL.

channel variation. However, due to the strict fairness guarantees of the baseline scheme in each QAI, which is measured by the cumulative and current quality assigned, many unnecessary quality switches are introduced. The average Jain fairness index (i.e., $(\sum x_i)^2/n\sum {x_i}^2$) of the total quality assigned to each user are shown in Fig. 5(d). It can be seen that although the fairness index of our scheme is lower than that in the baseline, it can still achieve over 0.995.

However, when the users's initial position changes significantly, the network has to be re-trained as the MDP model changes with it. In order to explore whether the scheme can be applied in practice, we consider the transfer learning (TL) to enhance learning efficiency. We use the well-trained model in the scenario with the first four users as the initialization of a new model to be trained for the scenario with the last four users in Table I. As shown in Fig. 6, the TL is indeed effective to speed up the convergence. In Fig. 7, we present the cumulative distribution functions (CDFs) of the three QoE metrics in episode 20000 to episode 30000 in the training process with TL and episode 40000 to episode 50000 in the counterpart without TL, from which we can see that the performance of the both has already been comparable and have all outperformed the baseline scheme in terms of average quality and quality fluctuation.

V. Conclusion

In this letter, we proposed a MEC-assisted PAVS scheme for on-road driving scenarios. The effectiveness of our scheme and the potential of TL to speed up convergence have been validated by simulation results. In the future works, we will further combine TL with virtual experiences generated based on theoretical knowledge to enable the scheme to satisfy the requirement for real-world applications.

References

[1] A. Mehrabi, M. Siekkinen, and A. Yla-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 787–800, Apr. 2019.

[2] Z. Chang, X. Zhou, Z. Wang, H. Li, and X. Zhang, "Edge-assisted adaptive video streaming with deep learning in mobile edge networks," in *Proc IEEE WCNC*, 2019, pp. 1–6.

[3] W. U. Rahman, C. S. Hong, and E. Huh, "Edge computing assisted joint quality adaptation for mobile video streaming," *IEEE Access*, vol. 7, pp. 129082–129094, 2019.

[4] J. Luo, F. R. Yu, Q. Chen, and L. Tang, "Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1577–1592, Mar. 2020.

[5] R. Atawia, H. S. Hassanein, and A. Noureldin, "Robust long-term predictive adaptive video streaming under wireless network uncertainties," *IEEE Trans. Wireless Commun.*, vol. 17, no. 2, pp. 1374–1388, Feb. 2018.

[6] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 401–418, 1st Quart., 2016.

[7] L. Li, D. Shi, R. Hou, R. Chen, B. Lin, and M. Pan, "Energy-efficient proactive caching for adaptive video streaming via data-driven optimization," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5549–5561, Jun. 2020.

[8] D. Liu, J. Zhao, C. Yang, and L. Hanzo, "Accelerating deep reinforcement learning with the aid of partial model: Energy-efficient predictive video streaming," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3734–3748, Jun. 2021.

[9] H. Burchardt, S. Sinanovic, Z. Bharucha, and H. Haas, "Distributed and autonomous resource and power allocation for wireless networks," *IEEE Trans. Commun.*, vol. 61, no. 7, pp. 2758–2771, Jul. 2013.