Energy-Saving Predictive Video Streaming with Deep Reinforcement Learning

Dong Liu*, Jianyu Zhao†, and Chenyang Yang†

*University of Southampton, Southampton, UK
†Beihang University, Beijing, China
Email: d.liu@soton.ac.uk, jianyuzhao_buaa@163.com, cyyang@buaa.edu.cn

***Abstract*—In this paper, we propose a policy to optimize predictive power allocation for video streaming over mobile networks with deep reinforcement learning. The objective is to minimize the average energy consumption for video transmission under the quality of service constraint that avoids video stalling. To handle the continuous state and action spaces, we resort to deep deterministic policy gradient to solve the formulated problem. In contrast to previous predictive resource policies for video streaming, the proposed policy operates in an on-line and end-to-end manner. By judiciously designing action and state, the policy can exploit future information without explicit prediction. Simulation results show that the proposed policy can converge closely to the optimal policy with perfect prediction of future large-scale channel gains and outperforms the prediction-based optimal policy when prediction errors exist.**

I. Introduction

Mobile video traffic is expected to account for more than 75% of the global mobile data by 2021 as reported in [1]. Video streaming over cellular networks enables mobile users to watch the requested video while downloading. To avoid video stalling for a user experiencing bad channel conditions, a base station (BS) can increase the transmit power to ensure that the video segment is downloaded before playback. This, however, may cause a significant increase in energy consumption, which runs counter to energy efficiency (EE), an important design metric for cellular networks.

If the future channel gains can be predicted, say by user trajectory prediction [2] and using a radio map [3], a BS can transmit more data in advance to a user at good channel conditions and the buffered data can be consumed to support video streaming when channel conditions are poor. By harnessing future information in a minute-level time horizon, such predictive resource allocation (PRA) has been shown to provide a remarkable gain in terms of improving the EE of mobile networks supporting video streaming [4–9].

With known future instantaneous channel gains, the trade-off between required resources and video stalling time was investigated in [4]. With known future data rates in each time slot (within the duration of channel coherence time) of a prediction horizon, the number of time slots for video streaming was minimized in [5] to save energy. Assuming known future average rate in each frame (within which the large scale channel gains change a little) with bounded prediction errors, a robust PRA policy was optimized in [6]. However, since the future data rate of a user depends on resource allocation, the rate prediction is coupled with PRA, the energy saving potential of PRA cannot be fully exploited by the policies in [5, 6]. With known future large-scale channel gains, the optimal PRA was derived in [7] to minimize the energy consumption for video streaming.

Existing PRA policies for video streaming are operated in four phases, which consist of a prediction step with machine learning and a resource allocation step with optimization [8, 9]. The first phase is to train a predictor (say for large scale channel gains) in an off-line manner by using a large number of labels (say the historical locations along the trajectories of users [3]). The second phase is to gather data for making the prediction after a user initiates a request (say the locations along the trajectory of the user). The third phase is to assign radio resources to all the frames or time slots in a prediction window. Finally, the BS allocates resources and transmits to the user in each time slot according to the pre-assigned resources. Before future resources have been assigned, the BS has to serve the user in a non-predictive manner. Moreover, such a procedure is tedious. A natural question is: can we optimize PRA in an on-line and end-to-end manner? This is possible by resorting to reinforcement learning (RL), but how?

In this paper, we propose a deep reinforcement learning (DRL) policy to optimize predictive power allocation for video streaming without explicit prediction of future information. The objective is to minimize the total average energy consumption for video transmission under the quality of service (QoS) constraint that avoids video stalling. We consider a scenario where users travel across multiple cells covered by a central unit (CU) during video streaming. To reduce the communication overhead between the RL agent (acted by the CU) and each BS, we set the average data rate as the action without loss of optimality. To leverage future user trajectory for PRA without using the private-sensitive location information, we include the current and past large scale channel gains of a user with its associated BS and adjacent BSs into the state. To cope with continuous state and action spaces, we employ deep deterministic policy gradient (DDPG) [10] to solve the

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61731002.

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

RL problem. Simulation results show that the proposed DRL-based policy can converge closely to the optimal policy with perfect prediction and outperforms the optimal PRA when prediction errors exist.

II. System Model

Consider a learning-enabled network architecture where the BSs in an area are connected with a CU. The CU monitors and records the status of mobile users (say channel conditions and buffer status) via BSs. The CU learns the target average data rate a BS should provide to a user given the user's status and sends instruction to the BS. Then, the BS adjusts transmit power based on the instruction and channel information.

Each user requesting a video may travel across multiple cells during the video streaming process. Assume that each user is associated with the BS with the strongest large-scale channel gain, and each BS serves the associated users over orthogonal time-frequency resources. Since all the users in the considered network share the same network topology (e.g., BS locations), system configurations (e.g., maximal transmit power and transmission bandwidth), wireless channels (e.g., path loss and small-scaling channel distribution), and road topology, we consider a randomly chosen user and the learned policy is applicable to every user.

A. Transmission and Channel Models

Each video is divided into N_v segments, each of which is the minimal unit for video playback. The playback duration of each segment is divided into L_v frames, each with duration ΔT. Each frame is divided into N_s time slots, each with duration τ, i.e., $\tau = \Delta T/N_s$, as shown in Fig. 1. The large-scale channel gains (i.e., average channel gains) remain constant within each frame and may change from one frame to another due to user mobility. The small-scale channel gains remain constant within each time slot and are independently and identically distributed (i.i.d.) among time slots.

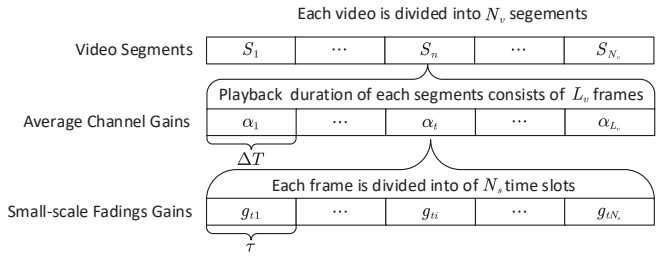

Fig. 1. Video segment playback duration and channel variation.

Denote $\alpha_t g_{ti}$ as the instantaneous channel gain between a user and its associated BS in the ith time slot of the tth frame, where α_t and g_{ti} are the large-scale channel gain and small-scale fading gain, respectively. With capacity achieving coding, the instantaneous data rate in the ith time slot of the tth frame can be expressed as

$$R_{ti} = W \log_2 \left(1 + \frac{\alpha_t}{\sigma^2} p_{ti} g_{ti}\right) \tag{1}$$

where W is the transmission bandwidth, σ^2 is the noise power, p_{ti} is the transmit power in the ith time slot of the tth frame.

B. Video Streaming and Power Consumption Model

The video playback starts after the user has received the first segment of the video. To avoid stalling, each segment should be downloaded to the user's buffer before displaying. Assume that the user's buffer is larger than the size of the video, which is reasonable for today's mobile devices. Hence, buffer overflow is not considered. Then, the following QoS constraint should be satisfied

$$\sum_{n=1}^{l} \sum_{t=(n-1)L_v+1}^{nL_v} \sum_{i=1}^{N_s} \tau R_{ti} \geq \sum_{n=2}^{l+1} S_n,\ l = 1, \cdots, N_v - 1 \tag{2}$$

where $\sum_{t=(n-1)L_v+1}^{nL_v} \sum_{i=1}^{N_s} \tau R_{ti}$ is the amount of data transmitted to the user during the playback of the nth segment, and S_n is the size of the nth segment.

The total energy consumed by a BS for video transmission during the tth frame is modeled as [11]

$$E_t = \frac{1}{\rho} \sum_{i=1}^{N_s} \tau p_{ti} + \Delta T P_c \tag{3}$$

where ρ reflects the impact of power amplifier, and P_c is the power consumed for operating the baseband and radio frequency circuits as well as the cooling and power supply.

The objective is to find power allocation among time slots that minimizes the average energy consumption subject to the QoS and maximal power constraints,

$$\min_{\{p_{ti}\}} \ \mathbb{E}\left[\sum_{t=1}^{(N_v-1)L_v} \left(\frac{1}{\rho} \sum_{i=1}^{N_s} \tau p_{ti} + \Delta T P_c\right)\right] \tag{4a}$$

$$s.t. \ \sum_{n=1}^{l} \sum_{t=(n-1)L_v+1}^{nL_v} \sum_{i=1}^{N_s} \tau R_{ti} \geq \sum_{n=2}^{l+1} S_n,\ l = 1, \cdots, N_v - 1 \tag{4b}$$

$$p_{ti} \leq P_{\max},\ \forall\ t, i \tag{4c}$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and $P_{\max}$ is the maximal transmit power of each BS.

Since the values of g_{ti} and α_t in the future are not known in advance, it is impossible to directly solve the above optimization problem at the beginning of video streaming. In the sequel, we resort to RL to find the solution.

III. Energy-Saving Power Allocation With DDPG

In this section, we formulate a RL framework for the considered problem and propose a policy learning algorithm.

A standard RL problem can be described as an agent learning from interactions with an environment in a sequence of discrete time steps $t = 1, 2, \cdots, T$ to achieve a goal [12]. At each time step t, the agent observes the state of the environment $\mathbf{s}_t$ and executes an action a_t. Then, the agent receives a reward r_t from the environment and transits into a new state $\mathbf{s}_{t+1}$. The interaction of the agent with the environment is then captured by an experience vector $\mathbf{e}_t = [\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}]$.

The agent learns a policy from its experiences to maximize an expected return, which reflects the cumulative reward received by the agent during the T-time-step episode. The policy (denoted by π) determines which action should be executed in which state. The expected return is defined as $\mathbb{E}\left[\sum_{t=1}^{T}\gamma^{t-1}r_t\right]$, where γ denotes the discount factor.

A. Reinforcement Learning Framework

In our learning-enabled network, the CU serves as the agent. A direct formulation of the RL problem is to regard the power allocated to the ith time slot p_{ti} as the action. Then, the action should depend on the instantaneous channel gain so as to adapt to the time-varying channels, and hence $\alpha_t g_{ti}$ should be included into the state. However, this incurs millisecond-level information exchange between the CU and each BS and hence causes large signaling overhead. Besides, this makes it hard for the agent to learn a good policy. This is because g_{ti} is hard to predict beyond the channel coherence time (i.e., the time slot duration in the considered model), although α_t depends on user trajectory that is possible to learn [2]. In fact, g_{ti} can be regarded as a multiplicative noise on α_t and hence $\alpha_t g_{ti}$ has a much larger dynamic range than α_t. This inspires us to find the action and state that depend on α_t.

1) Action: In practice, it is not hard to measure the distribution of small-scale fading, based on which we can derive the optimal policy that minimizes the average transmit power with given target average data rate in a frame.

The average transmit power in the tth frame can be expressed as $\bar{P}_t = \mathbb{E}_g\left[p_{ti}\right]$ since g_{ti} is i.i.d. among time slots. Then, the objective function in (4a) can be rewritten as $\mathbb{E}\left[\sum_{t=1}^{(N_v-1)L_v}\left(\frac{1}{\rho}\tau\bar{P}_t + \Delta T P_c\right)\right]$, minimizing which is equivalent to minimizing $\mathbb{E}\left[\sum_{t=1}^{(N_v-1)L_v}\bar{P}_t\right]$. Considering that the power allocation policy minimizing $\bar{P}_t$ with a target average rate $\bar{R}_t \triangleq \mathbb{E}_g[R_{ti}]$ in the tth frame is equivalent to the policy that maximizes $\bar{R}_t$ with given average power constraint $\bar{P}_t$, we first derive the power allocation policy that maximizes $\bar{R}_t$ with given $\bar{P}_t$, which has the following water-filling structure [13]

$$p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}}) = \begin{cases} \frac{\sigma^2}{\alpha_t}\left(\frac{1}{g_t^{\mathrm{th}}} - \frac{1}{g_{ti}}\right), & g_{ti} \geq g_t^{\mathrm{th}} \\ 0, & g_{ti} < g_t^{\mathrm{th}} \end{cases} \tag{5}$$

where the water level g_t^{th} satisfies

$$\int_{g_t^{\mathrm{th}}}^{\infty}\frac{\sigma^2}{\alpha_t}\left(\frac{1}{g_t^{\mathrm{th}}} - \frac{1}{g}\right)f(g)dg = \bar{P}_t \tag{6}$$

and $f(g)$ is the probability density function (PDF) of the small-scale channel, e.g., $f(g) = e^{-g}$ for Rayleigh fading.

Then, the average rate in the tth frame achieved by the policy $p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})$ can be expressed as

$$\bar{R}_t = \int_0^{\infty} W\log_2\left[1 + \frac{\alpha_t}{\sigma^2}p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})g\right]f(g)dg \tag{7}$$

By substituting (5) into (7), we can obtain

$$\bar{R}_t = W\int_{g_t^{\mathrm{th}}}^{\infty}\log_2\left(\frac{g}{g_t^{\mathrm{th}}}\right)f(g)dg \tag{8}$$

Because $p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})$ is also the policy that minimizes $\bar{P}_t$ with the target average rate $\bar{R}_t$, we can obtain the water level in $p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})$ that minimizes $\bar{P}_t$ with given $\bar{R}_t$ from (8). Denote the relation between g_t^{th} and $\bar{R}_t$ given by (8) as $g_t^{\mathrm{th}} \triangleq g^{\mathrm{opt}}(\bar{R}_t)$. For Rayleigh fading, we can obtain

$$\bar{R}_t = W\int_{g_t^{\mathrm{th}}}^{\infty}\log_2\left(\frac{g}{g_t^{\mathrm{th}}}\right)e^{-g}dg = -\frac{W}{\ln 2}\mathrm{Ei}(-g_t^{\mathrm{th}}) \tag{9}$$

where $\mathrm{Ei}(-x) \triangleq -\int_{-x}^{\infty}\frac{e^{-}t}{t}dt$. Then, $g^{\mathrm{opt}}(\bar{R}_t)$ can be expressed as

$$g^{\mathrm{opt}}(\bar{R}_t) = -\mathrm{Ei}^{-1}\left(-\frac{\bar{R}_t\ln 2}{W}\right) \tag{10}$$

For other channel distribution, $g^{\mathrm{opt}}(\bar{R}_t)$ can be obtained from (8) via a simple bisection searching.

With the relation between the target average rate $\bar{R}_t$ and the water level g_t^{th} in the optimal power allocation policy $p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})$ in the tth frame, the CU only needs to decide $\bar{R}_t$ and send $g_t^{\mathrm{th}} = g^{\mathrm{opt}}(\bar{R}_t)$ to the user's associated BS. Then, the BS can adjust the transmit power in each time slot of the tth frame according to $p^{\mathrm{opt}}(\alpha_t, g_{ti}, g_t^{\mathrm{th}})$. In this way, the original learning problem that optimizes $\{p_{ti}\}$ is equivalently converted into optimizing $\{\bar{R}_t\}$. Consequently, the **action** can be chosen as the target average rate in each frame as

$$a_t = \bar{R}_t \tag{11}$$

In this way, the agent interacts with the environment once a frame (i.e., a time step in RL parlance, in seconds) while the BS adjusts transmit power in each time slot (in milliseconds).

2) State: Since the average power consumed for video transmission depends on the large-scale channel gain, α_t should be included into the state. Due to the dependency on the user's location, the large-scale channel gain in the next time step is related to the locations of the user in current and past time steps. To help the agent to capture the user mobility pattern, the state should also include the large-scale channel gains in the past time steps. Since different user locations can result in the same large-scale channel gain between the user and its associated BS, we also include the large-scale channel gains between the user and the other $N_b - 1$ BSs with the largest large-scale channel gains. Define a vector $\boldsymbol{\alpha}_t \triangleq [\alpha_{1,t}, \cdots, \alpha_{N_b,t}]$, where $\alpha_{n,t}$ is the large-scale channel gain between the user and the BS with the nth largest large-scale channel gain. We denote $\alpha_{1,t} = \alpha_t$ as the large-scale channel gain between the user and its associated BS. To ensure the QoS, the buffer status at the user should also be considered into the state. Let B_t denote the amount of data remaining in the user's buffer at time step t. Then, the **state** vector can be expressed as

$$\mathbf{s}_t = [B_t, \boldsymbol{\alpha}_t, \boldsymbol{\alpha}_{t-1}\cdots, \boldsymbol{\alpha}_{t-N_t}] \tag{12}$$

We can see that the state vector lies in the continuous space and will have a huge number of possible states if discretizing the state space due to combinatorial elements. Tabular-based RL, such as Q-learning [12], encounters with the curse of

dimensionality. Therefore, we employ DRL to enable the agent to learn how to act in unexperienced states from experienced states in the following.

3) Reward: The **reward** for the agent is designed as

$$r_t = -\sum_{i=1}^{N_s} \tau p_{ti} - \lambda \max\{S_{l(t+1)} - B_{t+1}, 0\} \quad (13)$$

where $\sum_{i=1}^{N_s} \tau p_{ti}$ is the transmit energy consumed in the tth time step, $l(t+1)$ denotes the index of video segment to be played in the next time step, $B_{t+1} = B_t + R_t - S_t I_{\text{end}}(t)$ is the amount of data in the user's buffer at the beginning of the next time step, $I_{\text{end}}(t)$ is an indicator function with $I_{\text{end}}(t) = 1$ if the playback of the $l(t)$-th segment is completed at the end of the tth time step and $I_{\text{end}}(t) = 0$ otherwise. $-\lambda \max\{S_{l(t+1)} - B_{t+1}, 0\}$ is a term that imposes a penalty on the reward when video stalls (i.e., when $S_{l(t+1)} - B_{t+1} > 0$), λ is the penalty coefficient and $S_{l(t)} - B_t$ increases the impact of penalty when there is less amount of data in the buffer.

B. Transmission Policy Based on DDPG

Since the action lies in the continuous space, value-based DRL methods such as deep Q-network [14] are not applicable. In the following, we resort to DDPG [10] to solve the problem.

DDPG is based on the actor-critic architecture and employs deterministic policy gradient that can be estimated more efficiently than stochastic policy gradient. DDPG maintains two deep neural networks, namely actor network $\mu(\mathbf{s}; \boldsymbol{\theta}_\mu)$ and critic network $Q(\mathbf{s}, a; \boldsymbol{\theta}_Q)$ stored at the CU. The network architecture in our framework is shown in Fig. 2. The actor network specifies the current policy by deterministically mapping each state into a specific continuous action. The output of the actor network is then used to compute the water level $g_t^{\text{th}} = g_t^{\text{opt}}(\bar{R}_t)$ according to (10), based on which the BS controls the transmit power at each time slot within the tth frame according to the current channel gain $\alpha_t g_{ti}$, and the water level g_t^{th} according to the policy $p^{\text{opt}}(\alpha_t, g_t^{\text{th}}, g_{ti})$ given by (5). The critic network is used to approximate the action-value function, $Q^\pi(\mathbf{s}, a) \triangleq \mathbb{E}\left[\sum_{i=t}^{T} \gamma_{i-t} r_i | \mathbf{s}_t = \mathbf{s}, a_t = a, \pi\right]$, defined as the expected return achieved by policy π when taking action a under state $\mathbf{s}$.

During the interactions with the environment, the CU collects the experience $\mathbf{e}_t = [\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}]$ from the BSs within its coverage in the database at the server as $\mathcal{D} = \{\mathbf{e}_1, \cdots, \mathbf{e}_t\}$. In each time step, a mini-batch of the experiences $\mathcal{B}$ is sampled from $\mathcal{D}$ to update the network parameters, i.e., *experience replay* [14]. The parameters of the critic network are updated with gradient descent as

$$\boldsymbol{\theta}_Q \leftarrow \boldsymbol{\theta}_Q + \frac{\delta_Q}{|\mathcal{B}|} \nabla_{\boldsymbol{\theta}_Q} \sum_{j \in \mathcal{B}} [y_j - Q(\mathbf{s}_j, a_j; \boldsymbol{\theta}_Q)]^2 \quad (14)$$

where $y_j = r_j + \gamma Q'(\mathbf{s}_{j+1}, \mu'(\mathbf{s}_{j+1}; \boldsymbol{\theta}'_\mu); \boldsymbol{\theta}'_Q)$, and δ_Q is the learning rate of the critic network. $Q'(\mathbf{s}, a; \boldsymbol{\theta}'_Q)$ and $\mu'(\mathbf{s}; \boldsymbol{\theta}'_\mu)$ are the target critic network and target actor network, respectively, which have the same structure as $Q(\cdot)$ and $\mu(\cdot)$, and are respectively updated by $\boldsymbol{\theta}'_Q \leftarrow \omega \boldsymbol{\theta}_Q + (1-\omega)\boldsymbol{\theta}'_Q$ and $\boldsymbol{\theta}'_\mu \leftarrow \omega \boldsymbol{\theta}_\mu + (1-\omega)\boldsymbol{\theta}'_\mu$ with very small value of ω to stabilize the learning [10].

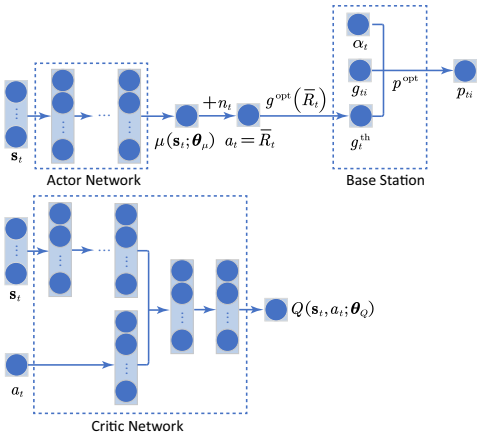

Fig. 2. Architecture of the actor and critic networks.

The parameters of the actor network are updated using the sampled policy gradient as

$$\boldsymbol{\theta}_\mu \leftarrow \boldsymbol{\theta}_\mu + \frac{\delta_\mu}{|\mathcal{B}|} \nabla_a Q(\mathbf{s}, a; \boldsymbol{\theta}_Q)|_{\mathbf{s}=\mathbf{s}_i, a=\mu(s_i;\boldsymbol{\theta}_\mu)} \nabla_{\boldsymbol{\theta}_\mu} \mu(\mathbf{s}|\boldsymbol{\theta}_\mu)|_{\mathbf{s}=\mathbf{s}_i} \quad (15)$$

where δ_μ is the learning rate of the actor network.

To find the optimal policy, the agent needs to explore in the action space during the interactions with the environment. A noise term is added to the output of the actor network, i.e., $a_t = \mu(\mathbf{s}_t; \boldsymbol{\theta}_\mu) + n_t$, to encourage exploration.

The detailed algorithm for learning the transmission policy is given in Algorithm 1.

IV. Simulation Results

In this section, we evaluate the performance of the proposed policy by comparing with baseline policies via simulations.

A. Simulation Setup

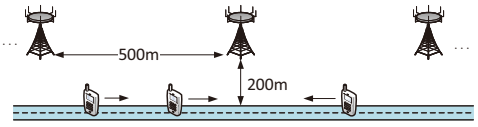

Fig. 3. Simulation scenario.

We consider a scenario where users move along a road across multiple cells as shown in Fig. 3. The distance between adjacent BSs is 500 m and the maximal transmit power of each BS is 46 dBm. The noise power is -95 dBm/Hz and the transmission bandwidth of each user is 2 MHz. Since the circuit energy consumption is the same for all the considered policies, we only evaluate transmit energy consumed for video streaming. The path loss is modeled as $35.3 + 37.6 \log_{10}(d)$ in dB where d is the distance between user and BS in meters. The small-scale fading is Rayleigh fading. The playback duration of each video is 150 s and that of each segment is 10 s. Each video segment is with size 1 MBytes. The duration of each frame is $\Delta T = 1$ s and the duration of each time slot is $\tau = 1$ ms. The user mobility is modeled by a Markov chain [8],

Algorithm 1 Transmission Policy Learning Based on DDPG

Initialize critic network $Q(\mathbf{s}, a; \boldsymbol{\theta}_Q)$ and actor network $\mu(\mathbf{s}|\boldsymbol{\theta}_\mu)$ with random weights $\boldsymbol{\theta}_Q$, $\boldsymbol{\theta}_\mu$.
Initialize target networks Q' and μ' with weights $\boldsymbol{\theta}'_Q \leftarrow \boldsymbol{\theta}_Q$, $\boldsymbol{\theta}'_\mu \leftarrow \boldsymbol{\theta}_\mu$.
Initialize replay memory $\mathcal{D}$, $done \leftarrow 0$.
for episode $= 1, 2, \cdots$ **do**
  Receive initial observation state $\mathbf{s}_1$.
  for time step (i.e., frame) $t = 1, 2, \cdots$ **do**
    Select action $a_t = \mu(\mathbf{s}_t; \boldsymbol{\theta}_\mu) + n_t$ according to the current transmission policy and exploration noise.
    Set $\bar{R}_t = a_t$ and $g_t^{\text{th}} = g^{\text{opt}}(\bar{R}_t)$.
    for time slot $i = 1, \cdots, N_s$ **do**
      Allocate transmit power in each time slot within the tth frame according to (5).
    end for
    Observe reward r_t and new state $\mathbf{s}_{t+1}$.
    if all video segments have been transmitted to the user **then**
      $done \leftarrow 1$
    end if
    Store experience $\mathbf{e}_t = [\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}]$ in $\mathcal{D}$.
    Randomly sample a mini-batch of experiences from $\mathcal{D}$ as $\mathcal{B} = \{[\mathbf{s}_j, \mathbf{a}_j, r_j, \mathbf{s}_{j+1}]\}$
    Set $y_j = \begin{cases} r_j, & \text{if } done = 1 \\ r_j + \gamma Q'(\mathbf{s}_{j+1}, \mu'(\mathbf{s}_{j+1}; \boldsymbol{\theta}'_\mu); \boldsymbol{\theta}'_Q), & \text{otherwise} \end{cases}$
    Update the actor and critic networks according to (14) and (15), respectively.
    Update the target networks:
    $\boldsymbol{\theta}'_\mu \leftarrow \omega\boldsymbol{\theta}_\mu + (1-\omega)\boldsymbol{\theta}'_\mu$, $\boldsymbol{\theta}'_Q \leftarrow \omega\boldsymbol{\theta}_Q + (1-\omega)\boldsymbol{\theta}'_Q$
    if $done = 1$ **then**
      break
    end if
  end for
end for

where the velocity of a user may increase or decrease by 1 m/s in the next frame both with probability q, and remains unchanged with probability $1-2q$. The minimal and maximal velocities of each user are 0 m/s and 30 m/s, respectively. The initial velocity of users is set as 15 m/s.

B. Fine-Tuned Parameters in Algorithm 1

The actor network has four hidden layers each with 600 nodes, respectively. The output layer of the actor network employs a modified $\tanh$ function $0.5 \times (\tanh(x) + 1)$ as the activation function, which normalizes the action within $[0, 1]$. For the critic network, the state and action first go through three hidden layers each with 600 nodes and one hidden layer with 600 nodes, respectively, and then combine together followed by two hidden layers each with 600 nodes. The output layer of the critic network has no activation function. All the hidden layers of the actor and critic networks are fully connected with rectified linear unit (ReLU) as the activation function. For the state representation, we set $N_b = 2$ and $N_t = 2$. Since the elements of the input have different ranges, we scale the large-scale channel gain as $45 + \log_2(\alpha_{b,t})$ before going through the neural networks and employ batch normalization [15] for each hidden layer.

Adam [16] is used to adjust the learning rate during training, and the initial learning rate is $\delta_\mu = \delta_Q = 10^{-4}$. The update rate for the target networks is $\omega = 2.5 \times 10^{-3}$. We also add a L2-norm regularization term with weight 10^{-4} on the loss function when training the critic network to avoid over-fitting. The replay memory size is $|\mathcal{D}| = 10^6$ and the mini-batch size for gradient descent is $|\mathcal{B}| = 64$. The discount factor and the penalty coefficient are set as $\gamma = 1$ and $\lambda = 30$, respectively. The noise term n_t follows Gaussian distribution with zero mean and the variance decreased linearly from 0.1 to 0 within 5×10^4 episodes.

C. Performance Evaluation

We compare the proposed DRL-based policy with the following baselines:

1) *Predict & Optimize*: This is the policy proposed in [7], which assumes perfect prediction of future large-scale channel gains and hence can serve as the performance upper bound. The policy is obtained by solving problem (4) at the beginning of video streaming.
2) *Non-predictive*: This is the existing policy without using any future information. The power allocation policy can only minimize the energy consumption in the current frame via water-filling given by (5). To satisfy the QoS constraint, the segment to be played in the tth frame is transmitted in the $t-1$th frame and the water-level g_t^{th} is obtained from (10) by setting $\bar{R}_t = S_{t+1}/L_v$.

We first set $q = 0$, i.e., each user moves with constant velocity. In this scenario, the future large-scale channel gains can be perfectly predicted based on the velocity of user and hence "*Predict & Optimize*" achieves the optimal performance.

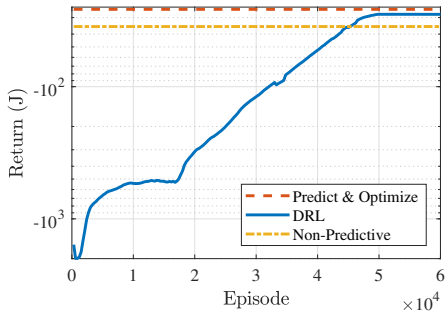

Fig. 4. Learning curve, $q = 0$. The results are averaged over 30 Monte Carlo trials with random initial user locations and over 300 successive episodes.

In Fig. 4, we show the learning curve of the proposed DRL-based policy. Since there is no video stalling after convergence (and hence no penalty on the reward), the negative of the converged return is the transmit energy consumption. We can see that the proposed policy can converge to "*Predict & Optimize*" after 5×10^4 episodes and outperform "*Non-predictive*".

In Fig. 5, we show how the proposed policy behaves. The result is obtained from one snapshot of an episode after Algorithm 1 converges. We can see that the proposed policy transmits more data when the large-scale channel gain is higher and behaves similarly to "*Predict & Optimize*".

We then set $q = 0.5$ in the sequel to show the impact of imperfect predictions, considering that future large-scale

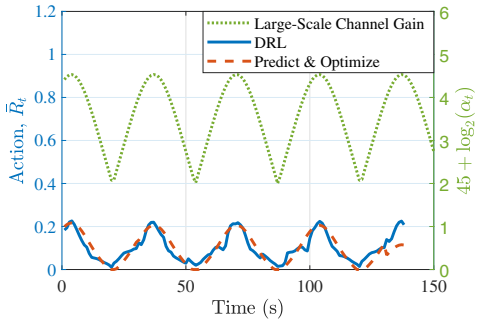

Fig. 5. Behavior of the DRL-based policy, $q = 0$.

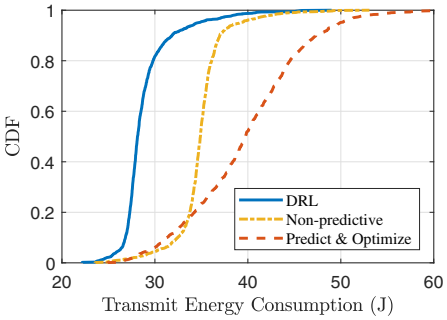

Fig. 6. CDF of transmit energy consumption, $q = 0.5$.

channel gains cannot be perfectly predicted due to various reasons, e.g., inaccurate velocity prediction. In this scenario, the velocity of the user is random, and the future average channel gains are predicted based on the average velocity for "*Predict & Optimize*".

In Fig. 6, we compare the cumulative distribution functions (CDFs) of the transmit energy consumed for each video in 1000 episodes after the DRL policy converges. In each episode, the location that a user initiates a request and the velocity of user are randomly generated. We can see that the proposed DRL-based policy outperforms both baseline policies, with the average energy consumption 30% lower than "*Predict & Optimize*". The average energy consumption of "*Predict & Optimize*" is even higher than "*Non-predictive*". The CDF curve of the proposed policy is steeper than "*Predict & Optimize*", which suggests that the proposed policy is more robust to the random user behaviors.

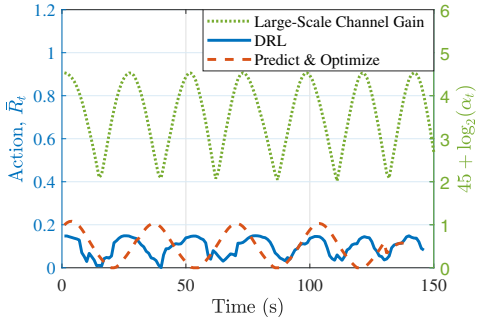

Fig. 7. Behavior of the DRL-based policy, $q = 0.5$.

In Fig. 7, we compare the behavior of the proposed policy with "*Predict & Optimize*". We can see that "*Predict & Optimize*" cannot adapt to the evolution of large-scale channel gains because the random acceleration may cause large accumulative prediction errors on the large-scale channel gains. On the contrary, the proposed DRL-based policy can adjust the transmit power on-line to adapt to the channel variation.

V. Conclusion

In this paper, we proposed a DRL-based policy to optimize predictive power allocation for video streaming over mobile networks aimed at minimizing the average energy consumption under the QoS constraint. We formulated the problem in RL framework and resorted to DDPG to learn the policy. With the judiciously designed action and state, the predictive power allocation can be implemented in two timescales, which reduces the signaling overhead between the CU and each BS. Simulation results show that the proposed policy can converge to the optimal policy with perfect future large-scale channel gains. When prediction errors exist, the proposed policy outperforms the prediction-based optimal policy.

References

[1] Cisco, "Global mobile data traffic forecast update, 2016–2021 white paper," Feb. 2017.

[2] W. Zhang, Y. Liu, T. Liu, and C. Yang, "Trajectory prediction with recurrent neural networks for predictive resource allocation," in *Proc. IEEE ICSP*, 2018.

[3] J. Chen, U. Yatnalli, and D. Gesbert, "Learning radio maps for UAV-aided wireless networks: A segmented regression approach," in *Proc. IEEE ICC*, 2017.

[4] D. Tsilimantos, A. Nogales-Gómez, and S. Valentin, "Anticipatory radio resource management for mobile video streaming with linear programming," in *Proc. IEEE ICC*, 2016.

[5] H. Abou-Zeid, H. S. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 5, pp. 2013–2026, Jun. 2014.

[6] R. Atawia, H. S. Hassanein, H. Abou-Zeid, and A. Noureldin, "Robust content delivery and uncertainty tracking in predictive wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 4, pp. 2327–2339, Apr. 2017.

[7] C. She and C. Yang, "Context aware energy efficient optimization for video on-demand service over wireless networks," in *Proc. IEEE/CIC ICCC*, 2015.

[8] N. Bui and J. Widmer, "Data-driven evaluation of anticipatory networking in LTE networks," *IEEE Trans. on Mobile Comput.*, vol. 17, no. 10, pp. 2252–2265, Oct. 2018.

[9] J. Guo, C. Yang, and C.-L. I, "Exploiting future radio resources with end-to-end prediction by deep learning," *IEEE Access*, vol. 6, Dec. 2018.

[10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. ICLR*, 2015.

[11] G. Auer, V. Giannini, C. Desset, I. Godor, P. Skillermark, M. Olsson, M. A. Imran, D. Sabella, M. J. Gonzalez, O. Blume, and A. Fehske, "How much energy is needed to run a wireless network?" *IEEE Wireless Commun.*, vol. 18, no. 5, pp. 40–49, Oct. 2011.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[13] A. Goldsmith, *Wireless Communications*. New York, NY, USA: Cambridge University Press, 2005.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, Feb. 2015.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv: 1502.03167*, 2015.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014.