

Cache-Enabled Dynamic Rate Allocation via Deep Self-Transfer Reinforcement Learning

Zhengming Zhang, Yaru Zheng, Meng Hua, Yongming Huang and Luxi Yang
 School of Information Science and Engineering, Southeast University, Nanjing 210096, China
 Email: {zmzhang, yzrheng, mhua, huangym and lxyang}@seu.edu.cn

Abstract—Caching and rate allocation are two promising approaches to support video streaming over wireless network. However, existing rate allocation designs do not fully exploit the advantages of the two approaches. This paper investigates the problem of cache-enabled QoE-driven video rate allocation problem. We establish a mathematical model for this problem, and point out that it is difficult to solve the problem with traditional dynamic programming. Then we propose a deep reinforcement learning approaches to solve it. First, we model the problem as a Markov decision problem. Then we present a deep Q-learning algorithm with a special knowledge transfer process to find out effective allocation policy. Finally, numerical results are given to demonstrate that the proposed solution can effectively maintain high-quality user experience of mobile user moving among small cells. We also investigate the impact of configuration of critical parameters on the performance of our algorithm.

Index Terms—Bit rate allocation, caching, deep reinforcement learning, transfer learning.

I. INTRODUCTION

Due to the exponential growth of the number of smart mobile equipments and innovative high-rate mobile data services (such as videos streaming for mobile gaming and road condition monitoring), the networks should accommodate the overwhelming wireless traffic demands [1], [2]. In fact it can be predicted that the mobile video traffic will be over one third of mobile data traffic by the end of 2018 [3]. People's demand for video loading speed and video clarity is endless, thus the viewers' Quality-of-Experience (QoE) is an important indicator of the mobile communication network performance.

Mobile users often encounter video lag or sudden blur when using mobile devices to watch videos. The reason for these two cases is that the video is divided into small video chunks (usually of the same length) by special algorithms [4]. If the current network is of low quality, the video sender will reduce the video resolution of the next few seconds to ensure that users can continue to watch the video [5], [6]. In this way, the quality can not be guaranteed. If the user fasts forward the video, and the paragraph has not been loaded, then the video playback will be interrupted until the corresponding paragraph is cached. The goal of using the adaptive bit rates allocation algorithm is to provide a better user experience and to reduce network bandwidth usage. We will focus on three factors that affect the user experience: quality (quantified by video rates), packet dropping and video frozenness duration.

Recently, caching at base stations has been used as a promising approach for video streaming [7]. Caching during off-peak

times can bring popular contents closer to users, and hence improves users' QoE [8]. Although caching effectiveness has been extensively explored in the literature [7], [8], [9], they do not consider the influence of bit rate allocation to the effectiveness of caching.

In this paper, we adapt the ideas underlying the success of QoE-driven dynamic video rate allocation to the cache-enabled wireless network. There are two important and practical assumption in our paper.

Assumption 1 : The bit rate received by the user is lied in a large discrete space which almost can be considered as a continuous range.

Assumption 2 : The central network has finite caching capability but knows the size and status of the buffer of the user's mobile device [10], [11].

Most of the existing studies assume that the bit rate is in discrete space, and the dimension of the space is low so that all the values can be traversed within a finite time [6], [10]–[12]. This can be achieved by uniformly quantizing the candidate regions where the bit rate is located. The main reason for doing so is to simplify the complex real-world model and to facilitate the solution of the problem. In practice, however, the bit rate received by the user is lied in a large discrete space which almost can be considered as a continuous range.

In our work, we maximize the bit rate of video chunks that are actually consumed in a period of time, and minimize packet dropping and video frozenness duration. We track the network capacity and the buffer state like [10]. However, unlike previous studies, the capacity of the network and buffer state are considered as continuous values. We formulate the problem as a Markov Decision Process (MDP), then we present a deep Q-learning approach with normalized advantage functions and special knowledge transfer process to solve it.

The main contributions of this paper are summarized as follows:

- (i) We define a more realistic QoE criteria that consider the total accumulation bit rates, packet dropping and video freeze duration for cache-enabled video streaming.
- (ii) We convert the allocation problem into a Markov decision problem which extends the network capacity and candidate bit rate from discrete space to continuous space. And we point that the states transition probability cannot be obtained directly.
- (iii) We propose a way to go beyond the traditional algorithms we call it deep self-transfer reinforcement learning. We

get (sub)optimal dynamic video rate allocation policy by using this approach.

The rest of this paper is organized as follow. In Section II, we introduce existing rate adaptation approaches. Section III the system model is presented. Section IV, we proposed the deep reinforcement learning algorithms in which the solution of bitrate allocation policy for video streaming is verified. Finally, numerical results and discussion are presented in Section V, and a conclusion is reached in Section VI.

II. RELATED WORK

Improving the QoE of adaptive video streaming has been the main focus of many researchers in recent years. Although caching methods used to optimize the video transmission, the audience's requirements for the video quality are endless, the sites providing wireless video services have to compromise between the quality and transmission speed.

Model : Numerous models have been proposed to address real-time adaptation of multimedia contents over wireless channels for a better QoE. A novel cache-enabled video rate control model that balances the needs for video rate smoothness and energy cost saving is proposed in [11]. In this work, they consider the playback rate in their QoE model, but the transmission delay and caching delay are ignored. A logarithmic QoE model derived from experimental results is used in [13] and they formulate the content cache management for video streaming into a convex optimization to give their problem an analytical framework. A QoE-aware video rate adaptation model is also proposed in [10]. In this work the user's buffer state is modeled as a vector and the value of network capacity is discretized as a finite set. To describe the dynamic evolution of the entire network, a large number of studies have modeled the problem as a Markov decision problem [10], [14]–[16]. We emphasize that the above studies ignore the impact of delay on QoE and cache performance and their MDP models are built in discrete action space and discrete state space.

Algorithm : Convex optimization algorithms are used in [11], [13] to solve their allocation problems. Their algorithms are simple and efficient but are not valid for complex and dynamic scenes. Optimal policy for dynamic MDP model in [14] is based on a fast heuristic. The state transition function is given in detail (this is not practical in the real world) and standard value iteration method [17] is used to obtain the optimal policy. In [16] the optimal problem is solved via dynamic programming [18]. In [10] similar approaches which based on value iteration and Q function are used to solve the MDP problem. It is worth emphasizing that the methods used in these studies are suitable to solve discrete dynamic programming problems and are not suitable for large scale decision space and continuous space.

Our study considers the cache-enabled video rate allocation MDP problem in continuous state space and action space, and we solve it using continuous deep Q-learning approach. This approach has three key technologies. They are continuous deep

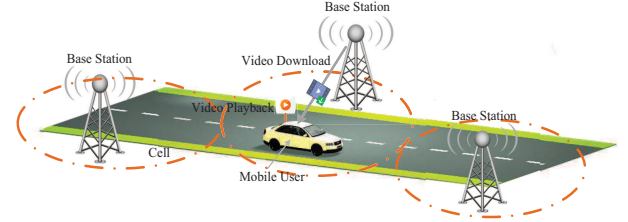


Fig. 1: Video streaming architecture.

reinforcement learning [19]–[21], imitation learning [22], [23] and transfer learning [24].

Continuous deep reinforcement learning: Deep reinforcement learning has received considerable attention in recent years and has been applied to a wide variety of real-world robotic control tasks [19] including learning policies to play go game [25]. To incorporate the benefits of value function estimation into continuous deep reinforcement learning, policy network and value network should be established [20]. A different point of view is put forward in [21], the authors complete the task by learning a single network that outputs both the value function and policy. We are inspired by this work to create a deep Q-learning approach to solve our problem without any prior knowledge about the state transition.

Imitation learning: This kind of algorithms consider the problem of acquiring skills from observing demonstrations. Behavioral cloning and inverse reinforcement learning are two main lines of work within imitation learning. There two kinds of imitation learning can improve the learning efficiency of reinforcement learning [21], [23].

Transfer learning: This kind of algorithms consider the problem of learning policies with applicability and re-use beyond a single task. They can acquire a multitude of skills faster than what it would take to acquire each of the skills independently, and let the artificial intelligence revolution from virtual to reality [24].

While there past works have led to a wide range of impressive results, they consider each skill separately, and do not fully exploit the advantages of the three key technologies. These led us to integrate these technologies with their respective advantages and get a more powerful algorithm.

III. MODEL AND PROBLEM FORMULATION

A highway high-mobility scenario with multiple base stations (BSs) deployed is shown in Fig. 1. We consider the downlink of a cache-enabled content-centric wireless network which provides full coverage of all driving sections and high data rate in each of the coverage areas. The BSs are equipped with cache storing some subsets of video contents. We assume the contents stored in the cache are given [26]. The video stream is divided into N chunks and the m th chunk has a length of ΔT_m seconds. The network consisting of F overlaid small cell base stations and each of them is connected to the video server via optical fibers. The indices of F BSs are

orderly included in a set $\mathcal{B} = \{1, 2, \dots, F\}$. The caching system contains L files $\mathcal{W} = \{W_1, W_2, \dots, W_L\}$. The user moving on a segment of highway which is covered by \mathcal{B} . Both BSs and mobile user are equipped with directional antennas. We define the following variables to describe our problem: state space, action space, state transition probabilities and reward function. The state space is a finite set of states. The action space is a finite set of actions. The state transition probabilities is the probability that action a in state s at time t will lead to state s' . And the reward function is the immediate reward received after transitioning from state s to state s' , due to action a .

A. The state space

Consider time slots indexed by $T = \{t_1, t_2, \dots, t_N\}$ where N is the number of slots. The time that the system stays at each slot is $\Delta t_x = t_{x+1} - t_x$. We assume that the network capacity is a random variable following a certain distribution and its value at time t_x is $C^{t_x} \in \mathbb{R}^+$. Let n^{t_x} as the requested content and $\mathbf{B}(t_x) \in \mathbb{R}^M$ as the buffer state where M is large enough. The element $b_i(t_x)$ of $\mathbf{B}(t_x)$ is the bitrate of the i th video chunk and it must meet two conditions [10], the first is FIFO constraint which means if $b_i = 0$ then $b_j = 0, j > i$, and the second is buffer length constraint which means $\sum_i b_i < U$, where U is the total size of the buffer.

Let \mathbf{S}^{t_x} denote the current state of the system at time t_x , the state \mathbf{S}^{t_x} is the combination of C^{t_x} , n^{t_x} and \mathbf{B}^{t_x} . The state space \mathcal{S} is composed of all the possible combination of the states.

$$\mathcal{S} = \left\{ \mathbf{S}^{t_x} = (C^{t_x}, n^{t_x}, \mathbf{B}^{t_x}) \mid C^{t_x} \in \mathbb{R}^+, n^{t_x} \in \mathcal{W}, \mathbf{B}^{t_x} \in \mathbb{R}^M \right\} \quad (1)$$

B. The action space

When the system is in state \mathbf{S}^{t_x} , the set of possible actions $\mathbf{A}_j^{t_x}$ is the set of bit rates which would be allocated to the user from the base station j . Specifically, $\mathbf{A}_j^{t_x} = (b_1, b_2, \dots, b_k)$, where $b_i = b$ is the bit rate of the i th video chunk. The bit rate b should satisfy $b_{\min} \leq b \leq b_{\max}$, and the number of video chunks that can be successfully transmitted is

$$k = \left\lfloor \frac{C^{t_x}}{b} \right\rfloor \quad (2)$$

where $\lfloor \cdot \rfloor$ indicates a round-down operation. Therefore, the action space can be defined as the possible combination of $\mathbf{A}_j^{t_x}$ i.e., $\mathcal{A} = \cup \mathbf{A}_j^{t_x}$.

C. The state transition probabilities

The state transition probability from state \mathbf{S}^{t_j} to state \mathbf{S}^{t_k} can be given by

$$p_{j,k} = p_{j,k}(C^{t_x}, n^{t_x}, \mathbf{B}^{t_x}) \quad (3)$$

Remark 1: Although the expression of the transition probability is given here, the mapping relation cannot be used directly to solve problems, because it is not available in the real world and is difficult to estimate accurately.

D. The reward function

A reward function defines the goal in a reinforcement learning problem. In the rate allocation problem, the objective is to select proper action for each system state to optimal the overall performance of the network and video streaming quality. We divide the reward function into two parts, cache miss cost and video streaming quality. Cache miss cost measures the cost of downloading data from the server. Video streaming quality is used to measure the delay caused by video wireless transmission.

Cache miss cost: First, we consider the overhead caused by data download from the remote sever. Let $c(n^{t_x})$ denote the cost for fetching content n^{t_x} via the backhaul link, depending on the content size. Then the cache miss cost is given by

$$C_{n,dl}^{t_x} = \mathbf{1}(n^{t_x} \notin \mathcal{C})c(n^{t_x}) \quad (4)$$

where \mathcal{C} is the cache set, $\mathbf{1}(\cdot)$ denotes the indicator function.

Video streaming quality: Now we consider the reward of the transmission. There three kinds of rewards of it, video playback quality u^{t_x} , packet loss cost $C_{loss}^{t_x}$ and video freeze cost $C_{freeze}^{t_x}$. The video playback quality is represented by the accumulated rate of the video chunks watched in the duration Δt_x by the viewer.

$$u^{t_x}(\mathbf{S}^{t_x}, \mathbf{A}^{t_x}) = \sum_{i=1}^{\tilde{k}} b_i \quad (5)$$

where \tilde{k} is number of video chunks pushed from the buffer in Δt_x . For the user in the cell C_j the base station B_j will transfer the video clip of size Z_j to the user, and Z_j can be given as $Z_j = \sum_{m=1}^M b_m \Delta T_m$. Assume that the buffer size of the user at that time is $U^{t_x} \leq U$, and T_{t_x} is the time spent emptying U^{t_x} . Then the packet loss cost and the video freeze cost can be given by

$$C_{loss}^{t_x} = \begin{cases} \sum_{m=\tau+1}^M \Delta T_m, & \text{if } Z_j + U^{t_x} > U \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

$$C_{freeze}^{t_x} = \begin{cases} \Delta t_x - \mathbf{1}(n^{t_x} \notin \mathcal{C})\tilde{T}_n - T_{t_x} - \sum_{m=1}^{\tau} \Delta T_m, \\ \text{if } \Delta t_x > \mathbf{1}(n^{t_x} \notin \mathcal{C})\tilde{T}_n + T_{t_x} + \sum_{m=1}^{\tau} \Delta T_m \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

where τ is the number of video blocks that can be received successfully by the user, \tilde{T}_n is the time taken to download the video n^{t_x} from the remote server.

Finally we combine cache miss cost and video streaming quality as our QoE model to get the reward of the user at t_x .

$$r_{\text{QoE}}^{t_x} = \sum_{i=1}^4 \lambda_i r_i \quad (8)$$

where $r_1 = -C_{n,dl}^{t_x}$, $r_2 = u^{t_x}$, $r_3 = -C_{loss}^{t_x}$, $r_4 = -C_{freeze}^{t_x}$ and λ_i are associated weights for the r_i . Given the cost of each slot, the total reward throughout the whole process is

$$R = \sum_{x=1}^{\aleph} \sum_{n=1}^N \gamma_j^{t_x} r_{QoE}^{t_x} \quad (9)$$

where N is the total number of requests sent by the user, $\gamma_j^{t_x}$ is the decay factor of the reward where $j \in \mathcal{B}$

E. Problem formulation

The global objective is to find the optimal bit rate allocation profile that maximizes the system reward throughout the whole process. The considered problem can be formulated as

$$\min_{A_j^{t_x}} \limsup_{L \rightarrow \infty} \frac{-1}{L\aleph} \sum_{x=1}^{\aleph} \sum_{n=1}^N \gamma_j^{t_x} r_{QoE}^{t_x} \{t_x \in T, j \in \mathcal{B}\} \quad (10)$$

where L means that the system is running L times and each time has $\aleph \in \mathbb{N}^+$ slots. Here $A_j^{t_x}$ is the optimization variable, it is also action of the agent at time t_x . The system total reward is the objective function.

Optimality analysis: The cache-enabled bit rate allocation problem (10) is an infinite horizon cost MDP and it is well known to be difficult problem due to the curse of dimensionality. While dynamic programming approaches represent a systematic approach for MDPs, there are usually not practical due to the curse of dimensionality and they generally need the state transition probabilities which are not available in practice. Existing works do not solve problem (10) and the optimal allocation solution remains unknown and is highly nontrivial.

As for the problem (10), we can obtain the optimal bit rate allocation policy π^* by solving the Bellman equation [27].

Lemma 1 (Bellman equation): There exist a value function $V(\cdot)$ and a scalar δ satisfying :

$$\delta + V(S^{t_x}) = \max_{A \in \mathcal{A}} \left\{ \pi(S^{t_x} | A^{t_x}) (r_{QoE}^{t_x} + \gamma^{t_x} E[V(S^{t_{x+1}})]) \right\} \quad (11)$$

Where the expectation $E[\cdot]$ is with respect to the probability distribution of the request arrival n^{t_x} and the network capacity C^{t_x} . $\delta = V^*$ is the optimal value to Problem and the optimal policy π^* achieving the optimal value V^* is given by

$$\mu^*(S^{t_x}) = \arg \max_{A \in \mathcal{A}} \left\{ \pi(S^{t_x} | A^{t_x}) (r_{QoE}^{t_x} + \gamma^{t_x} E[V(S^{t_{x+1}})]) \right\} \quad (12)$$

Proof: Please refer to Appendix A.

Algorithm design: Closed solution almost does not exist to the infinite horizon cost MDPs without any knowledge of state transition probabilities, and standard brute-force algorithms are usually impractical for implementation due to the curse of dimensionality. Therefore, it is of great interest to develop low-complexity suboptimal solutions, which can adapt to different initial state and complex environment. A deep neural network is built to approximate the allocation policy whose input is any possible system state and output is appropriate action. It has three advantages, first it can use historical data to training off-line, second the trained deep neural network can be used online, third it has good generalization performance.

IV. SOLVE DECISION PROCESS PROBLEM

In this section, we first introduce deep Q-learning that can work in continuous space. Then we embed a special imitation learning algorithm and a transfer learning algorithm into it to make our deep Q-learning more efficient.

Continuous deep Q-learning: The Q function $Q^\pi(s_t, \mathbf{a}_t)$ is used in many reinforcement learning algorithms, it corresponds to a policy π and is defined as the expected return after taking an action \mathbf{a}_t in state s_t and following the policy π thereafter.

$$Q^\pi(s_t, \mathbf{a}_t) = \mathbb{E}_{r_t \geq t, s_t \geq t \sim E, \mathbf{a}_t \geq t \sim \pi} [R_t | s_t, \mathbf{a}_t] \quad (13)$$

Q-learning uses the greedy policy $\mu(s_t) = \arg \max_{\mathbf{a}} Q(s_t, \mathbf{a}_t)$. Let θ^Q parametrize the Q function and the loss function is :

$$L(\theta^Q) = E_{s_t \sim \rho^\beta, \mathbf{a}_t \sim \beta, r_t \sim E} \left[(Q(s_t, \mathbf{a}_t | \theta^Q) - y_t)^2 \right] \quad (14)$$

where

$$y_t = r_{QoE}^t(s_t, \mathbf{a}_t) + \gamma Q(s_{t+1}, \mu(s_{t+1})) \quad (15)$$

and ρ is a stochastic distribution, β is a different stochastic policy. We optimize the Q function by using gradient descent with $\frac{\partial L(\theta^Q)}{\partial \theta^Q}$:

$$\begin{aligned} & \frac{\partial L(\theta^Q)}{\partial \theta^Q} \\ &= E_{s_t \sim \rho^\beta, \mathbf{a}_t \sim \beta, r_t \sim E} \left[\nabla_{\theta^Q} (Q(s_t, \mathbf{a}_t | \theta^Q) - y_t)^2 \right] \\ &= E_{s_t \sim \rho^\beta, \mathbf{a}_t \sim \beta, r_t \sim E} \left[\nabla_{\theta^Q} Q(s_t, \mathbf{a}_t | \theta^Q) (Q(s_t, \mathbf{a}_t | \theta^Q) - y_t) \right] \end{aligned} \quad (16)$$

To use Q-learning for continuous problem, we should also define the value function $V^\pi(s_t)$ and advantage function $A^\pi(s_t, \mathbf{a}_t)$:

$$\begin{aligned} V^\pi(s_t) &= \int \pi(s_t, \mathbf{a}_t) Q^\pi(s_t, \mathbf{a}_t) d\mathbf{a} \\ A^\pi(s_t, \mathbf{a}_t) &= Q^\pi(s_t, \mathbf{a}_t) - V^\pi(s_t) \end{aligned} \quad (17)$$

Then a neural network that separately outputs a value function term V and an advantage term A is used to represent the Q function, and the network is parameterized as a quadratic function [20]:

$$\begin{aligned} Q(s, \mathbf{a} | \theta^Q) &= A(s, \mathbf{a} | \theta^A) + V(s | \theta^V) \\ A(s, \mathbf{a} | \theta^A) &= -\frac{1}{2} (\mathbf{a} - \mu(s | \theta^\mu))^T \mathbf{P} (s | \theta^P) (\mathbf{a} - \mu(s | \theta^\mu)) \end{aligned} \quad (18)$$

where $\mathbf{P}(s | \theta^P)$ is a state dependent positive definite square matrix, and it is parameterized by $\mathbf{P}(s | \theta^P) = \mathbf{L}(s | \theta^L)^T \mathbf{L}(s | \theta^L)$, and $\mathbf{L}(s | \theta^L)$ is a lower triangular matrix. We have three variables that are parameterized differently $\mu(s | \theta^\mu)$, $V(s | \theta^V)$ and $\mathbf{P}(s | \theta^P)$, they are three neural network and we use target networks to combine them [28].

Imitation learning: As large amounts of on-policy experience are required in addition to good off-policy samples, we should improve the sampling efficiency of the algorithm. For the video rate allocation problem in continuous action space some actions, for example, discrete values obtained by sampling in the action space at equal intervals [10], are useful but not easily sampled. For this reason we propose an imitation learning we called it imitated sampling learning (ISL).

Algorithm 1 ISL

- 1: **Initialize:** Initialize the set $I = \{b_{\min}, b_{\min} + \Delta b, b_{\min} + 2\Delta b, \dots, b_{\max}\}$ where $\Delta b = \frac{b_{\max} - b_{\min}}{d}$ and d is a fixed constant.
 - 2: **Step 1:** Sample a random minbatch of K samples $\{(S_i, A_i, r_i, S_{i+1})\}_{i=1}^K$ from replay buffer Σ .
 - 3: **Step 2:** Using these samples get a policy $\pi_\theta(A_i|S_i)$.
 - 4: **Step 3:** Using the policy $\pi_\theta(A_i|S_i)$ get new set (\tilde{S}_i, A'_i) and discrete A'_i i.e., $\tilde{A}_i = (b, \dots, b), b \in I$. Then get a new dataset $\tilde{\Sigma} = (\tilde{S}_i, A'_i, \tilde{r}_i, S_{i+1})$.
 - 5: **Step 4:** Return aggregate dataset $\Sigma \leftarrow \Sigma \cup \tilde{\Sigma}$.
-

In Algorithm 1, the **Initialize** gets a candidate bit rate set like [12]. **Step 2** provides a learned model π_θ and this model will be used to get new samples. **Step 3** and **Step 4** can be seen as a sample imagination rollout [21]. Q-learning inherently requires noisy on-policy actions to succeed. It implies that the policy must be allowed to make its own mistakes during training, which might involve taking undesirable actions. ISL can avoid this problem while still allowing for a large amount of on-policy exploration by generating synthetic on-policy trajectories under the learned model π_θ .

ISL synthesizes a series of operations by discretizing A_j^t and use them to get a lot of state action trajectories and add them to the replay buffer. This will effectively augment the amount of experience available for Q-learning, and improve its data efficiency substantially.

Self-Transfer learning: Using pseudo-rewards in reinforcement learning can make other auxiliary predictions that serve to focus the agent on important aspects of the task. It also accelerates the acquisition of a useful representation [29]. Our QoE model will focus on different aspects when we choose different λ_i . This inspired us to choose different λ_i to improve the generalization ability of our agent. And we give this ability to the original agent by using self-transfer learning (STL).

In Algorithm 2, we construct different reward set by using different e_i from a noise process \mathcal{N} . For each reward r_i , $e_i r_i$ can be seen as the main part of the r_i , and $(1 - e_i)r_i$ is considered to be a noise added to r_i , and vice versa. This inspired us to propose a method which called reward lifting scheme similar to lifting-schemed wavelet transform [30] to obtain the main components of the rewards. The reward lifting scheme has three main steps: split, predict, and update, which correspond to **Step 3**, **Step 4**, and **Step 5** in Algorithm 2, respectively.

STL synthesizes a series of pseudo-rewards and use them to train a virtual agent. Then copies its knowledge to the original agent. This can improve training efficiency and generalization performance. The training method to our deep self-transfer reinforcement learning is described in Algorithm 3.

In Algorithm 3, Γ is the total number of training times, r is the learning rate. We use Adam [31] and the rectified non-linearity [32] to learn the parameters for our neural network with a learning rate of 10^{-3} . We set the mini-batch sizes is 64

Algorithm 2 STL

- 1: **Initialize:** Randomly initialize normalized networks $\tilde{P}(s, a|\theta^{\tilde{P}})$, $\tilde{V}(s|\theta^{\tilde{V}})$, $\tilde{\mu}(s|\theta^{\tilde{\mu}})$ and a prediction network $G(r|\theta^G)$.
 - 2: **Step 1:** Copy $\tilde{P}(s, a|\theta^{\tilde{P}})$ and $\tilde{V}(s|\theta^{\tilde{V}})$ to $P(s, a|\theta^P)$ and $V(s|\theta^V)$.
 - 3: **Step 2:** Using these $\tilde{\mu}(s|\theta^{\tilde{\mu}})$ get dataset $\Sigma' = \{(S_i, A_i, r_i = (r_1, r_2, \dots, r_d), S_{i+1})\}$.
 - 4: **Step 3:** Split each r_i : $Split(r_i) = (e_i r_i, (1 - e_i)r_i)$ where e_i obeys a probability distribution β and $\sum_{i=1}^d e_i = 1$.
 - 5: **Step 4:** Use $G(r|\theta^G)$ for prediction error: $error_i = (1 - e_i)r_i - G(e_i r_i|\theta^G)$. And train $G(r|\theta^G)$ using $Split(r_i)$.
 - 6: **Step 5:** Update $\hat{r}_i = e_i r_i + ((1 - e_i)r_i - error_i)/2$. And get $\Sigma = \{(S_i, A_i, \sum_{i=1}^d \hat{r}_i, S_{i+1})\}$.
 - 7: **Step 6:** Use Σ train $\tilde{P}(s, a|\theta^{\tilde{P}})$, $\tilde{V}(s|\theta^{\tilde{V}})$ and $\tilde{\mu}(s|\theta^{\tilde{\mu}})$. And update $\theta^P \leftarrow \theta^{\tilde{P}}$ and $\theta^V \leftarrow \theta^{\tilde{V}}$.
 - 8: Return $P(s, a|\theta^P)$, $V(s|\theta^V)$ and $\tilde{\mu}(s|\theta^{\tilde{\mu}})$.
-

Algorithm 3 Deep Self-Transfer Reinforcement Learning for Cache-Enabled Video Rate Allocation

- 1: **Initialize:** Use STL and (18) initialize normalized Q network $Q(s, a|\theta^Q)$. And freeze $P(s, a|\theta^P)$ and $V(s|\theta^V)$. Initialize target network Q' with weight $\theta^{Q'} = \theta^Q$. Initialize replay buffer $\Sigma = \emptyset$.
 - 2: **For episode = 1, ..., Γ do:**
 - 3: Initialize a random process \mathcal{N} for action exploration.
 - 4: Get a initial observation state s^{t_1} .
 - 5: **For x = 1, ..., \aleph do:**
 - 6: Select action A^{t_x} according to the current policy and exploration method $A^{t_x} = \mu(S^{t_x}|\theta^\mu) + \mathcal{N}$.
 - 7: Execute A^{t_x} and observe r_{t_x} and $S^{t_{x+1}}$.
 - 8: Store $(S^{t_x}, A^{t_x}, r_{t_x}, S^{t_{x+1}})$ in Σ .
 - 9: **If mod(episode · \aleph + x) = 0**
 - 10: Run Algorithm ISL.
 - 11: **End If**
 - 12: Sample a random minbatch of K samples $\{(S_i, A_i, r_i, S_{i+1})\}_{i=1}^K$ from Σ .
 - 13: Set $y_i = r_i + \gamma V(S_{i+1}|\theta^{Q'})$.
 - 14: Update θ^Q by minimizing the loss function $L = \frac{1}{K} \sum_{i=1}^K (y_i - Q(S_i, A_i|\theta^Q))^2$.
 - 15: Update the target network $\theta^{Q'} = r\theta^Q + (1 - r)\theta^{Q'}$.
 - 16: **End For**
 - 17: **End For**
-

and the replay buffer size is 10^6 . Fig. 2 shows the workflow of the deep self-transfer reinforcement learning in detail. STL is first executed to get $P(s, \mathbf{a}|\theta^P)$, $V(s|\theta^V)$ and $\tilde{\mu}(s|\theta^\mu)$ with good generalization performance. This corresponds to the left part of Fig. 2. When the STL is sufficiently trained, the parameters of the STL will be passed to the deep neural network which located in the right of Fig. 2 and the ISL will run at the appropriate time. Fig. 3 is the structure of the deep self-transfer reinforcement learning neural network. $\mu(s|\theta^\mu)$, $V(s|\theta^V)$ and $L(s|\theta^L)$, each of the three neural networks has three hidden layers with 128, 64 and 32 units, respectively. The output layer of $\mu(s|\theta^\mu)$ uses Sigmoid activation function. The training clock controls the time of knowledge transfer.

Analysis of Algorithm 3: The training of the parameters of the deep neural network in algorithm 3 uses Adam based on the first-order gradient information which makes the neural network update sufficiently stable. Similar to [25], despite lacking any theoretical convergence guarantees, our method is able to train large neural networks using reinforcement learning signal and gradient descent in stable manner. There are two neural networks to calculate Q , Q'^θ and Q^θ , only the target Q -network calculates Q values according to the next state with the target network's output. The using of target Q -network can cut off the dependencies of training samples and improve stability and convergence.

A major challenge of transfer learning in different tasks is catastrophic forgetting [33]. It means that the tendency of a neural network to completely and abruptly forget previously learned information upon learning new information. We teach a learned model parameter to a new task via lateral connection [24]. Since the trained model will be freezed in network design (line 1 in Algorithm 3), the way parameters are optimized for back-propagation does not affect the network that has been learned. This kind of neural network design naturally avoids the appearance of catastrophic forgetting.

The challenge of exploration bring us to learn a effective policy in continuous action spaces can be treated by adding noise sampled from a noise process \mathcal{N} (line 6 in Algorithm 3). The main reason is the off-policies algorithms such as Algorithm 3 can treat the problem of exploration independently from the learning algorithm.

Notice that in order to solve (10) by using Bellman equation the knowledge of the transition probability is needed, but the formulated Markovian domain herein lacks the state transition mapping as state in Remark 1. Therefore, the traditional approaches cannot be used to solve our problem. Although we do not know the transition probability, our algorithm is still valid, we prove this by Theorem 1.

Theorem 1 (Effectiveness): The changes in the distribution of n^{t_x} and C^{t_x} do not cause Algorithm 3 to fail to update its parameters, no matter what the distribution function of n^{t_x} and C^{t_x} is Algorithm 3 is valid.

Proof: Please refer to Appendix B.

We can find that the optimal policy π^* is existed from lemma 1. A popular measure of the performance of a reinforcement learning algorithm is its regret relative to executing

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Number of base station F	20
The length of each video chunk ΔT_m (s)	10
Available rates (Mb/s)	$\{0\} \cup [2, 10]$
Capacity space (Mb/s)	$[2, 80]$
Buffer size U (Mb)	180
Random process \mathcal{N}	Gauss process
Average sojourn time $\mathbb{E}[\Omega_j]$ (s)	60

the optimal policy in current MDP. The regret interests in the difference (in rewards during learning) between an optimal policy and a reinforcement learner:

$$\Delta(\pi, \pi^*) = \lim_{T \rightarrow \infty} \frac{1}{T} \left(\sum_{x=0}^T \gamma_j^{t_x} r_{Q^o E}^x(\mathbf{S}^{t_x}, \mathbf{A}^{t_x} | \pi) - \sum_{x=0}^T \gamma_j^{t_x} r_{Q^o E}^x(\mathbf{S}^{t_x}, \mathbf{A}^{t_x} | \pi^*) \right) \quad (19)$$

From a value function point of view (21) we can be define the regret as:

$$\text{Regret}(\pi, \pi^*) = E \left[V^\pi(\mathbf{S}) - V^{\pi^*}(\mathbf{S}) \right] \quad (20)$$

Where the expectation $E[\cdot]$ is with respect to the state \mathbf{S} . We give a regret bound of our learning algorithm following the assumption of the estimation errors $Q^\pi(\mathbf{S}, \mathbf{A}) - V^*(\mathbf{S})$ are uniformly random in $[-1, 1]$ [34].

Theorem 2 (Regret bound): Consider a state \mathbf{S} in which all the true optimal action values are equal at $Q^*(\mathbf{S}, \mathbf{A}) = V^*(\mathbf{S})$. Suppose that the estimation errors $Q^\pi(\mathbf{S}, \mathbf{A}) - V^*(\mathbf{S})$ are independently uniformly randomly in $[-1, 1]$. Then,

$$\text{Regret}(\pi, \pi^*) \leq 3 \quad (21)$$

Proof: Please refer to Appendix C.

V. PERFORMANCE EVALUATION

In this section, simulation results are provided to illustrate the effectiveness of the proposed algorithm. There are another four solutions used for comparison. Specifically, actor-critic solution [35], normalized advantage function (NAF) without deep neural network [36], normalized advantage function with deep neural network [21] and random rate allocation solution. We show the performances in terms of the buffer size and the maximum network capacity. The Zipf distribution is used in our scenario, which is used to describe the popularity of the contents in user requirements. This distribute predicts that out of a population of N elements, the frequency of elements of rank k is:

$$f(k, z, N) = \frac{1/k^z}{\sum_{i=1}^N (1/i^z)} \quad (22)$$

where z is the zipf factor, in our scenario $z = 0.8$ [37]. The default parameter values are shown in Table I.

Fig. 4 shows the training process of our deep self-transfer reinforcement learning. The training clock is 1500 means that the self-transfer neural network run 1500 times. Then the deep

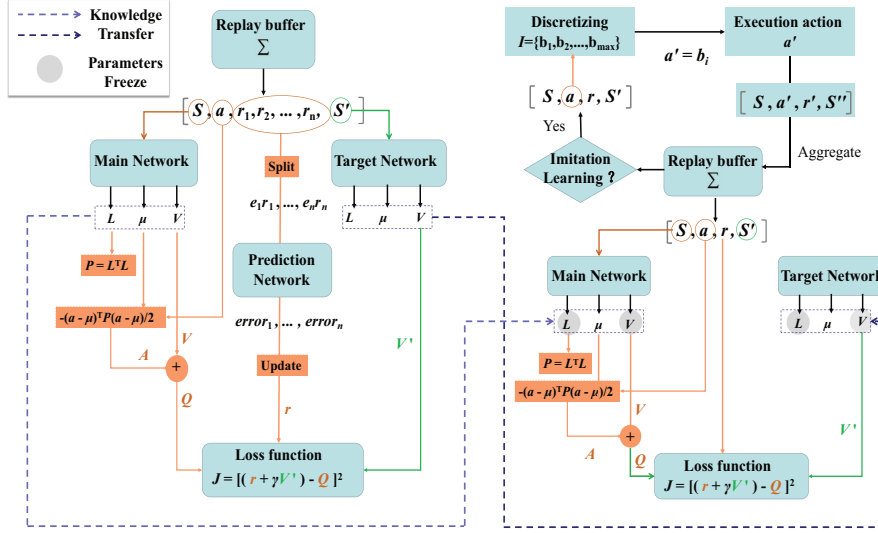


Fig. 2: Deep self-transfer reinforcement learning.

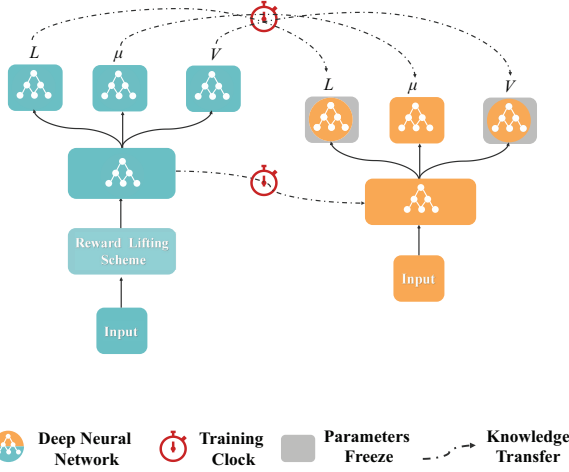


Fig. 3: The structure of deep self-transfer reinforcement learning neural network.

reinforcement neural network start work. We can find that compared with the deep neural network without STL, using knowledge transfer make our agent get higher rewards.

We gain insight into the convergence behaviors of our approach. We compare it to other methods in terms of performance. Fig. 2 shows the convergence behavior of our algorithm. After 2000 training, the agent has mastered the knowledge of the whole network system. We can find that our proposed algorithm is obviously better than the another four methods.

We let the buffer size change from 80Mb to 230Mb to compare the effect of the buffer size of the five methods. Fig. 3 demonstrates the impact of the buffer size on our approach. It can be found that as the buffer size increases, the average reward gained by the user is increasing because

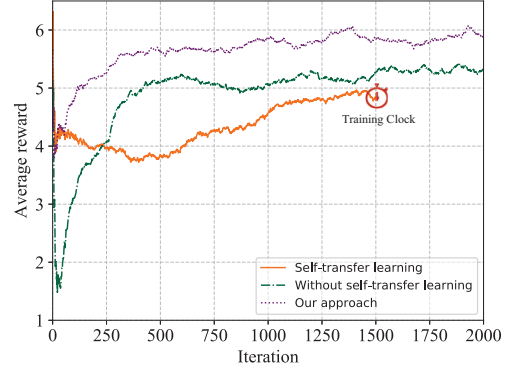


Fig. 4: Training of deep self-transfer reinforcement learning.

buffer size becomes larger means that more video chunks can be downloaded at the appropriate time. However, it is noticed that this performance improvement is becoming smaller as the buffer size becomes larger, which means that the average reward tends to converge as the buffer size increases. We can see from Fig. 4 that in the process of increasing buffer size, our algorithm is always better than the other four algorithms.

Fig. 5 demonstrates the impact of the maximum network capacity on the five approaches. In order to compare the performance of different algorithms, we set the user's storage space as 180Mb. It is shown that as the maximum network capacity increase, our algorithm has always been able to maintain a higher QoE than the another three methods it can also maintain a higher QoE when the maximum network capacity is low.

VI. CONCLUSION

Existing researches on cache-enabled video rate adaptive allocation do not take into account the continuous system capacity and candidate rate. In this paper, we leverage MDP

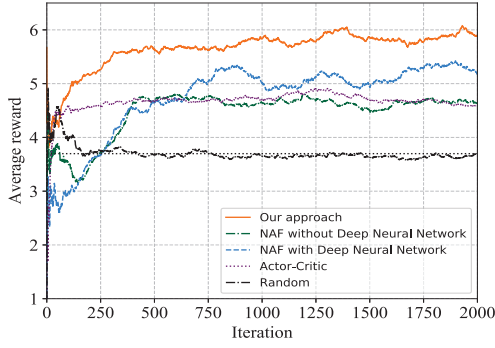


Fig. 5: Convergence behavior.

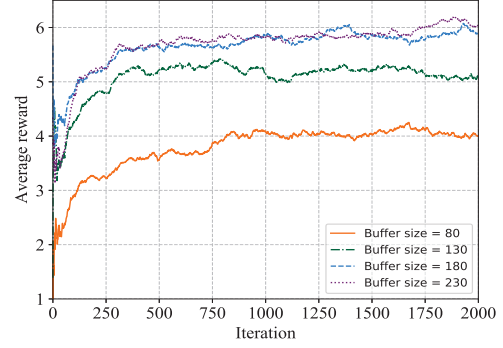


Fig. 6: The impact of buffer size on our algorithm

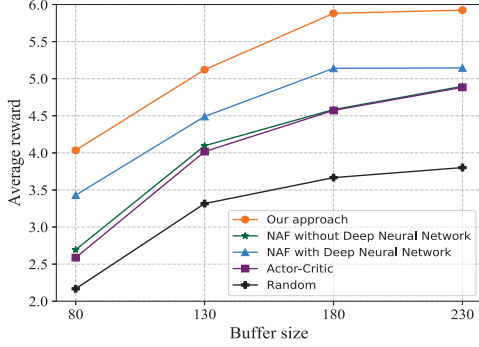


Fig. 7: Performance comparison with different buffer size

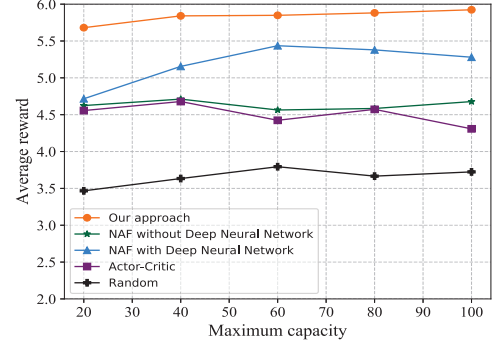


Fig. 8: Performance comparison with different maximum network capacity

to model the cache-enabled video rate allocation problem and propose a QoE-driven dynamic video rate adaptation method via deep continuous Q-learning scheme. We study the impacts of the buffer size and the maximum network capacity on our approach, and the performance evaluation results have shown that our approach can greatly improve the QoE. Future work is going to consider a more efficient scheme by adding more prior information to the model.

APPENDIX A PROOF OF LEMMA 1

By Propositions 4.2.1, 4.2.3, and 4.2.5 in [27], the optimal system reward of problem (10) is the same for all initial states and the solution $(\delta, V(S^{t_x}))$ to the following Bellman equation exists.

$$\delta + V(S) = \max_{A \in \mathcal{A}} \left\{ \pi(S|A) (r_{\text{QoE}}^{t_x} + \gamma^{t_x} \sum_{S' \in \mathcal{S}} p(S'|S, A) V(S')) \right\} \quad (23)$$

The transition probability is given by

$$\begin{aligned} p(S'|S, A) &\triangleq p(S^{t_{x+1}} = S' | S^{t_x} = S, A^{t_x} = A) \\ &= E_{n,C} [p(S^{t_{x+1}} = S' | S^{t_x} = S, A^{t_x} = A), n^{t_x} = n, C^{t_x} = C] \end{aligned} \quad (24)$$

Substituting (31) into (30) leads to (11).

APPENDIX B PROOF OF THEOREM 1

Since n^{t_x} and C^{t_x} affect the state transition probabilities, we let $d^\mu(S) = \sum_{x=0}^{\infty} \gamma^{t_x} p(S^{t_x} = S | S^{t_0}, A)$. Then use policy gradient [38] and cost function L (line 14 in Algorithm 3) we get

$$\frac{\partial L}{\partial \theta} = \sum_S d^\mu(S) \sum_A \frac{\partial \mu(S, A)}{\partial \theta} [A^\mu(S, A) + V^\mu(S)] \quad (25)$$

The gradient is no terms of the form $\frac{\partial d^\mu(S)}{\partial \theta}$, so the effect of policy changes on the distribution of states does not appear.

APPENDIX C PROOF OF THEOREM 2

Define $\sigma_A = Q^\pi(S, A) - Q^{\pi^*}(S, A)$. σ_A is a uniform random variable in $[-1, 1]$. As we use the Sigmoid function as the activation function, the output of $\mu(S)$ belong to $[0, 1]$. We quantize the interval $[0, 1]$ into m values at the interval $1/m$. Thus we get m discrete actions. When m tends to infinity we can get continuous actions. Because the estimation errors are independent, we can derive

$$\begin{aligned} P(\max_A \sigma_A \leq \sigma) &= P(\sigma_1 \leq \sigma, \sigma_2 \leq \sigma, \dots, \sigma_m \leq \sigma) \\ &= \prod_{i=1}^m P(\sigma_i \leq \sigma) \end{aligned} \quad (26)$$

The function $P(\sigma_i \leq \sigma)$ is the cumulative distribution function of σ_i , which here is defined as

$$P(\sigma_i \leq \sigma) = \begin{cases} 0, & \text{if } \sigma \leq 1 \\ (1 + \sigma)/2, & \text{if } 0 < \sigma < 1 \\ 1, & \text{if } \sigma \geq 1 \end{cases} \quad (27)$$

Then we get that

$$\begin{aligned} P(\max_A \sigma_A \leq \sigma) &= \prod_{i=1}^m P(\sigma_i \leq \sigma) \\ &= \begin{cases} 0, & \text{if } \sigma \leq -1 \\ [(1 + \sigma)/2]^m, & \text{if } -1 < \sigma < 1 \\ 1, & \text{if } \sigma \geq 1 \end{cases} \end{aligned} \quad (28)$$

The expectation of $\max_A \sigma_A$ can be given as

$$E[\max_A \sigma_A] = \int_{-1}^1 \sigma f_\sigma(\sigma) d\sigma \quad (29)$$

where f_σ is the probability density function of σ , it can be written as $f_\sigma(\sigma) = \frac{m}{2} [(1 + \sigma)/2]^{m-1}$. Note that we use Sigmoid function as the activation function of the $\mu(S)$ and $L(S)$. Thus

$$P(S) = L(S)^T L(S) \leq 1. \quad (30)$$

$$A^\pi(S, A) = -\frac{1}{2} (A - \mu(S))^T P(S) (A - \mu(S)) \geq -2 \quad (31)$$

This implies that

$$\begin{aligned} &E[V^\pi(S) - V^{\pi^*}(S)] \\ &= E[Q^\pi(S, A) - A^\pi(S) - V^{\pi^*}(S)] \\ &\leq E[Q^\pi(S, A) - V^{\pi^*}(S)] + 2 \\ &= E[\max_A \sigma_A] + 2 \\ &= \left[\left[(1 + \sigma)/2 \right]^m \frac{m\sigma - 1}{m+1} \right] \Big|_{-1}^1 + 2 \\ &= \frac{m-1}{m+1} + 2 \end{aligned} \quad (32)$$

In our cache-enabled bit rate allocation problem, the actions lie in the continuous space thus when $m \rightarrow \infty$

$$E[V^\pi(S) - V^{\pi^*}(S)] \leq 3 \quad (33)$$

We complete the proof.

REFERENCES

- [1] K. Zheng, L. Zhao, J. Mei, and M. Dohler, "10 gb/s hetsnets with millimeter-wave communications: access and networking - challenges and protocols," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 222–231, 2015.
- [2] M. Jaber, M. A. Imran, R. Tafazolli, and A. Tukmanov, "5g backhaul challenges and emerging research directions: A survey," *IEEE Access*, vol. 4, pp. 1743–1766, 2016.
- [3] C. V. N. Index, "Global mobile data traffic forecast update, 2012–2017," 2013.
- [4] N. Carlsson, D. Eager, V. Krishnamoorthi, and T. Polishchuk, "Optimized adaptive streaming of multi-video stream bundles," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1637–1653, 2017.
- [5] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *ACM Conference on Multimedia Systems*, 2011, pp. 157–168.
- [6] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.
- [7] J. Qiao, Y. He, and X. S. Shen, "Proactive caching for mobile video streaming in millimeter wave 5g networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 7187–7198, Oct 2016.
- [8] B. Zhou, Y. Cui, and M. Tao, "Optimal dynamic multicast scheduling for cache-enabled content-centric wireless networks," *IEEE Transactions on Communications*, vol. 65, no. 7, pp. 2956–2970, July 2017.
- [9] A. Liu and V. K. N. Lau, "Cache-enabled opportunistic cooperative mimo for video streaming in wireless systems," *IEEE Transactions on Signal Processing*, vol. 62, no. 2, pp. 390–402, Jan 2014.
- [10] Y. Chen, F. Zhang, K. Wu, and Q. Zhang, "Qoe-aware dynamic video rate adaptation," in *IEEE Global Communications Conference*, 2016, pp. 1–6.
- [11] J. Xie, R. Xie, T. Huang, J. Liu, and Y. Liu, "Energy-efficient cache resource allocation and qoe optimization for http adaptive bit rate streaming over cellular networks," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.
- [12] Y. Wang, X. Zhou, M. Sun, L. Zhang, and X. Wu, "A new qoe-driven video cache management scheme with wireless cloud computing in cellular networks," *Mobile Networks & Applications*, vol. 22, no. 1, pp. 72–82, 2017.
- [13] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "Qoe-driven cache management for http adaptive bit rate streaming over wireless networks," *IEEE Transactions on Multimedia*, vol. 15, no. 6, pp. 1431–1445, Oct 2013.
- [14] W. Bao and S. Valentin, "Bitrate adaptation for mobile video streaming based on buffer and channel state," in *IEEE International Conference on Communications*, 2015, pp. 3076–3081.
- [15] A. Bokani, S. A. Hoseini, M. Hassan, and S. S. Kanhere, "Implementation and evaluation of adaptive video streaming based on markov decision process," in *ICC 2016 - 2016 IEEE International Conference on Communications*, 2016, pp. 1–6.
- [16] M. Pesce, D. Munaretto, and M. Zorzi, "A markov decision model for source video rate allocation and scheduling policies in mobile networks," in *Ad Hoc NETWORKING Workshop*, 2014, pp. 119–125.
- [17] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [18] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995, vol. 1, no. 2.
- [19] 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," *Computer Science*, vol. 8, no. 6, p. A187, 2015.
- [20] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *Computer Science*, pp. 1889–1897, 2015.
- [21] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, 2016, pp. 2829–2838.
- [22] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, p. 233, 1999.
- [23] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," 2017.
- [24] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016.
- [25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, and M. Lanctot, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.
- [26] D. Liu and C. Yang, "Energy efficiency of downlink networks with caching at base stations," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 907–922, 2015.
- [27] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.
- [28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
- [29] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," 2016.

- [30] C. Chrysafis and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 378–389, Mar 2000.
- [31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Computer Science*, 2014.
- [32] X. Glorot, B. Antoine, and Y. Bengio, "Deep sparse rectifier networks," *Learning/statistics & Optimisation*, 2010.
- [33] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning & Motivation*, vol. 24, pp. 109–165, 1989.
- [34] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," *Proceedings of the Fourth Connectionist Models Summer School*, 1993.
- [35] D. P. Bertsekas, J. N. Tsitsiklis, and A. Volgenant, "Neuro-dynamic programming," *Encyclopedia of Optimization*, vol. 27, no. 6, pp. 1687–1692, 2011.
- [36] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3389–3396.
- [37] M. Cha, H. Kwak, P. Rodriguez, Y. Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Transactions on Networking*, vol. 17, no. 5, pp. 1357–1370, Oct 2009.
- [38] R. S. Sutton, "Policy gradient methods for reinforcement learning with function approximation," *Submitted to Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, 1999.