Joint Offloading and Streaming in Mobile Edges: A Deep Reinforcement Learning Approach

Soohyun Park, Junhui Kim, Dohyun Kwon, MyungJae Shin, and Joongheon Kim
School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea

***Abstract*—This paper proposes a joint dynamic video streaming and deep reinforcement learning (DRL) based offloading method in mobile edge computing systems. In order to utilize video services in mobile edge networks, efficient streaming and offloading algorithms are essentially required. In this paper, therefore, a novel dynamic offloading algorithm is proposed and the algorithm is fundamentally based on deep Q-network (DQN) which is one of widely used deep reinforcement learning algorithms.**

I. Introduction

Nowadays, there are a lof of research results in mobile edge computing (MEC) architectures, algorithms, and testbed implementation results. This is an obvious evidence that the MEC system is one of key technologies which can enable 5G and beyond 5G network evolution and realization.

One of the main reasons why MEC-based systems are getting a lot of attentions is that it is the best solution which can support seamless video streaming and contents delivery. Therefore, efficient and application-specific MEC-based algorithms those are for video contents sharing are essentially desired.

In this paper, a novel video contents offloading algorithm which aims at the joint optimization in terms of delay, energy consumption, and video quality considerations. In order to achieve this goal, deep Q-network based improved methods is utilized which is a special type of neural network aims at the computation of reinforcement learning (e.g., Q-learning).

The rest of this paper is organized as follows. Sec. II shows the related work that presents the applications of deep reinforcement learning algorithms. Sec. III presents our proposed dynamic offloading algorithm that is fundamentally based on deep Q-network; and the novelty of the proposed algorithm is evaluated via data-intensive simulations in Sec. IV. Lastly, Sec. II concludes this paper.

II. Related Work: Sequential Decision Making via Deep Reinforcement Learning

A. Intelligent system

Liangbing Feng, *et. al.* proposed a hybrid intelligent control model. The model combines high-level time PetriNet (HLTPN) and reinforcement learning [1]. The control system of the model is modelled by HLTPN and the final system state is presented as transitions delay time. In order to minimize transitions delay time, two algorithms are used based on Q-learning, a type of deep reinforcement learning. Sarhak Bhagat, *et. al.* proposed the possibility of developing self-sufficient agent that can be learned based on information collected from the agent's environment by merging deep reinforcement algorithms and soft bio-inspired structure [2]. They also present the examples and algorithms that are deployed in actual scenarios. Furthermore, the paper describes various studies that apply deep reinforcement learning in soft robotics.

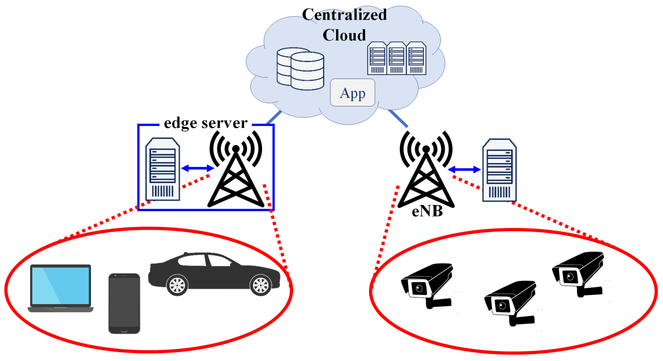

Fig. 1: Mobile edge computing (MEC) architecture.

B. Autonomous driving

Y. Zhang, *et. al.* improved the instability seen by the Q-learning algorithm using double Q-learning as an automatic decision-making approach for vehicle speed control [3]. When video data consisting of high-dimensional data is entered into the model, double DQN is improved in both value accuracy and policy quality compared to conventional DQNs. X. Xu, *et. al.* developed an intelligent overtaking decision making method for highway autonomous driving [4]. This overtaking decision making policy is learned using a reinforcement learning method called the Q-learning via a series of simulated driving scenarios. In addition, there are a number of control-related studies for safe autonomous driving in highway environments [5], [6]. M. Shin, *et. al.* presents a novel imitation learning algorithm for autonomous driving control under the benefits of augmented random search [7]–[10].

C. Others

D. Silver, *et. al.* presented AlphaGo Zero, which evaluates the location using a tree search and determines the movement using a trained deep neural network (DNN) based on reinforcement learning [11]. AlphaGo Zero carries out the learning through its own self without a guide or knowledge of

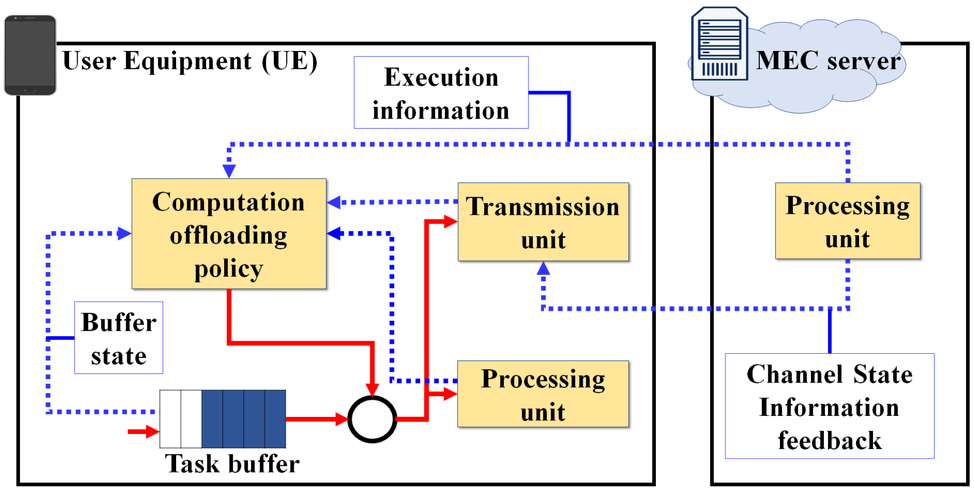

Fig. 2: General task offloading sequential decision-making process.

human data and game rules through its reinforcement learning based algorithm, and improves the strength of the tree search. AlphaGo Zero has achieved 100 wins against AlphaGo, who won the title against a human Go player.

III. Joint Streaming and Offloading

A. *Overall Architecture*

When real-time video services are provided to individual users such as video streaming and AR/VR, it is important to (i) maintain the high quality of video streaming under the constraint of tolerance latency as well as (ii) minimize the total execution time and total energy consumption of devices. There is always a trade-off between energy consumption minimization and latency while keeping the quality of the video data at a high level. To solve these problems there are many solutions using mobile edge computing (MEC). The MEC technique lets the processing performance and energy efficiency of the device improve by offloading the task to the edge server around the user device which has better performance than the user devices (terminal devices).

The execution latency of the terminal device can be categorized into two parts, i.e., local computing latency and edge computing latency. In edge computing architecture, the latency includes transmission delay to edge server, processing delay on the edge server, and downloading delay from the edge server. Because of the transmission and downloading delays, even if the edge server has better computing power than the terminal devices, it is important to decide (i) whether to offload or not and (ii) when which task will be offloaded. The decision may affect to the reduction of the battery consumption and the improvement of the processing performance of the terminal devices. In general, each device decides local computing or offloading to edge server for each task according to the 'internal offloading policy' as illustrated in Fig. 2. In this paper, we propose a DQN based offloading decision algorithm Each UE (terminal device) is trained by the algorithm and finally gets the best offloading policy. The idea and algorithm we proposed in this paper will be explained below.

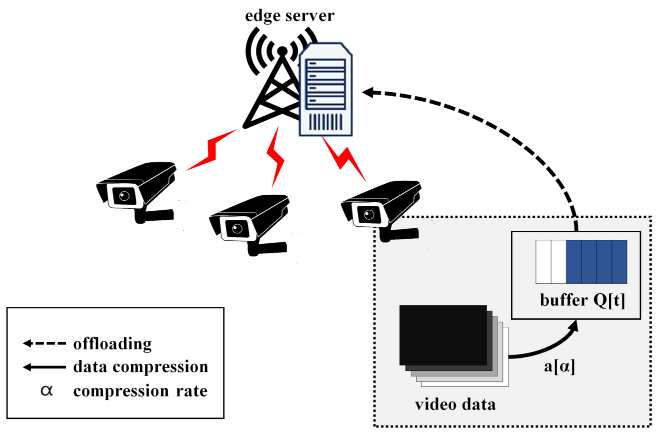

Fig. 3: Video quality decision.

After the decision of offloading is made, the data to be transmitted to the edge server will be accumulated in each UE buffer. When more data is accumulated in the buffer, more transmission time (i.e., time to wait for wireless transmission) of newly added data increases. Then, eventually, the total execution delay also increases. For this reason, after the offloading is decided in each UE, additional step of a decision whether the transferred data should be compressed or not by considering the buffer size and the degree of data accumulation is needed.

Finally, in this paper, we pursue the minimization of the energy consumption and execution delay of UE while satisfying the optimal quality of the video data which appropriates for the UE situation via the proposed DQN based offloading algorithm.

Algorithm 1 Deep Q-learning with Energy and Execution Delay

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\bar{\theta}$ = $\theta$
for episode = 1, $M$ do
    Initialize sequence $s_1$ = $\{x_1\}$ and preprocessed sequence $\phi_1$ = $\phi(s_1)$
    for $t$ = 1, $T$ do
        With probability $\epsilon$ select a random action $a_t$
        otherwise select $a_t$ = $argmax_a Q(\phi(s_t), a; \theta)$
        Execute action $a_T$ in emulator and observe reward $r_t^e$, $r_t^d$ and image $x_{t+1}$
        Set $r_t$ = $\alpha$ * $r_t^e$ + $(1-\alpha)$ * $r_t^d$ with weight parameter $\alpha$
        Set $s_{t+1}$ = $s_t$,$a_t$,$x_{t+1}$ and process $\phi_{t+1}$ = $\phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
        Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+'})$ from $D$
        Set $y_j$ = $\begin{cases} \mathrm{r}_j, & \text{if episode terminates at step j+1} \\ \mathrm{r}_j + \gamma max_{a\prime}\hat{Q}(\phi_{j+1}, a\prime; \bar{\theta}), & \text{otherwise} \end{cases}$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
        Every $C$ steps reset $\hat{Q}$=$Q$
    end for
end for
```

B. Offloading Decision with Deep Q-Networks (DQN)

We use deep Q-network (DQN) for the decision of offloading where DQN is the advanced form of Q-learning. In conventional Q-learning, it has some problems, i.e., correlation between samples and a target that is non-stationary. The DQN overcomes these problems through more deep layers, replaying with random samples of experience and distribution of the target network.

For the DQN-based offloading decision algorithm, we have designed the following:

State. The system has two components, UE capacity and edge server capacity.

Action. In our system, the action means whether to offload and is expressed as 0 or 1.

Reward. Each agent will get a reward in each step, it is expressed as the sum of two aspects of the return. In addition, the reward follows the formulas, i.e.,

$$R = \gamma \times R_e + (1-\gamma) \times R_d, \tag{1}$$

$$R_e = E_{local} - E_{total}, \tag{2}$$

$$R_d = D_{local} - D_{total}, \tag{3}$$

where R represents the reward that agent will get through action. R_e and R_d mean the reward considered on the energy consumption side and the execution delay side. When offloading computing is determined for tasks in the UE baseds on DQN, the energy consumed in the UE is called E_{total} and E_{local} means the energy consumed in the UE when all the tasks are processed in the local (UE). Similarly, D_{total} is the total execution delay when offloading computing is determined based on DQN, D_{local} is the execution delay in the UE when all the tasks are processed in the local (UE). The two values by DQN-based offloading determination, E_{total} and D_{total}, are as follows:

$$E_{total} = \sum_{n=1}^{N} \alpha_n * E_n^o + (1-\alpha_n) * E_n^l \tag{4}$$

$$D_{total} = \sum_{n=1}^{N} \alpha_n * D_n^o + (1-\alpha_n) * D_n^l \tag{5}$$

IV. Performance Evaluation

In this section, we present the simulation results using Deep Q-Network based offloading decision.

We assume a scenario as follows. User Equipment (UE) could have the task capacity between 30 and 50. Mobile Edge Computing (MEC) server could also have the task capacity between 800 and 1000. The agent should perform 10000 tasks per episode. Each task is selected between 0 and remained tasks uniform randomly. If the agent performs all tasks, that episode is terminated. At the start of the episode, the UE task capacity and MEC server task capacity are randomly set from 30 to 50, and from 800 to 1000, respectively.

In our DQN learning, the input size of the model is 3 which is the number of dimension of observation (UE capacity, MEC server capacity and remained tasks). The output size is 2 which is the number of dimension of action (offload or not). We consider the learning model using discount rate = 0.99, replay memory = 50000, batch size = 64, target update frequency = 5, total episodes = 500.

Fig. 4 shows the total learning reward with respect to the increasing episodes. More the episodes learned, the greater the reward value is shown. The total reward in 0$\sim$150 episode shows the best learning performance. In Fig. 4, reward of the first episode is $1.34 * 10^5$, but reward of episode 150 is $1.44 * 10^5$. Since then, the increase of the reward has been steadily decreasing. Especially, episode 300$\sim$500 section had

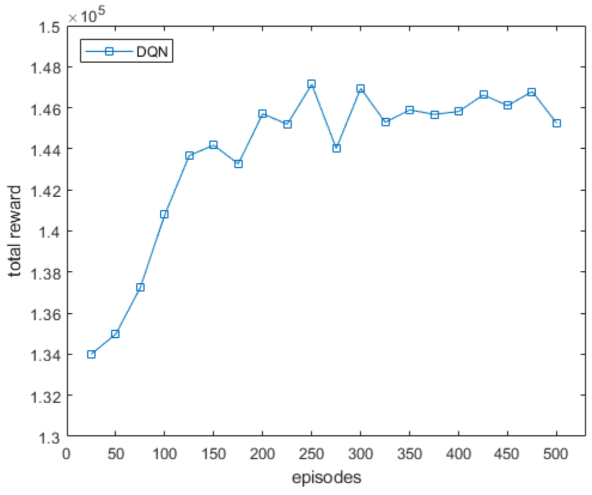

Fig. 4: Total reward versus the episode of DQN

reward between $1.45 * 10^5$ and $1.46 * 10^5$ without significant change. The reason why reward doesn't increase continuously but repeats increase and decrease is the task to be processed at each step is randomly selected.

V. Concluding Remarks and Future Work

This paper proposes a joint dynamic video streaming and deep Q-network based intelligent offloading method in mobile edge computing systems. For utilizing video services in mobile edge networks, efficient streaming and offloading algorithms are essentially required. In this paper, therefore, a new dynamic offloading algorithm is proposed and the algorithm is fundamentally based on deep Q-network (DQN).

For our future work, following topics are worthy to consider.

- The various performance evaluation with various settings are desired.
- Specific streaming control algorithms can be further discussed. In this research domain, Lyapunov optimization based algorithms are definitely considerable as the algorithm is widely used in various topics [12]–[18]

Acknowledgment

This research was supported by the National Research Foundation of Korea (2017R1A4A1015675, 2019R1A2C4070663); IITP grant funded by the Korea government (MSIT) (No. 2018-0-00170, Virtual Presence in Moving Objects through 5G). J. Kim is the corresponding author of this paper (e-mail: joongheon@gmail.com).

References

[1] L. Feng, *et. al.*, "An intelligent control system construction using high-level time Petri net and Reinforcement Learning," in *Proceedings of the IEEE International Conference on Control, Automation and Systems (ICCAS)*, Gyeonggi-do, Korea, October 2010.

[2] S. Bhagat, H. Banerjee, Z. T. H. Tse, and H. Ren, "Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges," in *Robotics*, January 2019.

[3] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, June 2018.

[4] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sunm, "A Reinforcement Learning Approach to Autonomous Decision Making of Intelligent Vehicles on Highways," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, December 2018.

[5] R. Zheng, C. Liu, and Q. Guo, "A decision-making method for autonomous vehicles based on simulation and reinforcement learning," in *Proceedings of International Conference on Machine Learning and Cybernetics (ICMLC)*, July 2013.

[6] M. Spryn, A. Sharma, D. Parkar, and M. Shrimal, "Distributed Deep Reinforcement Learning on the Cloud for Autonomous Driving," in *Proceedings of the IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, May 2018.

[7] M. Shin and J. Kim, "Randomized Adversarial Imitation Learning for Autonomous Driving," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Macau, China, August 2019.

[8] M. Shin and J. Kim, "Randomized Adversarial Imitation Learning for Autonomous Driving," *arXiv preprint arXiv:1905.05637*, May 2019.

[9] M. Shin and J. Kim, "Adversarial Imitation Learning via Random Search," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, July 2019.

[10] M. Shin and J. Kim, "Adversarial Imitation Learning via Random Search in Lane Change Decision-Making," in *Proceedings of International Conference on Machine Learning (ICML) Workshop on AI for Autonomous Driving*, Long Beach, California, USA, June 2019.

[11] D. Silver, *et. al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, pp 354–359, October 2017.

[12] J. Kim, G. Caire, and A.F. Molisch, "Quality-Aware Streaming and Scheduling for Device-to-Device Video Delivery," *IEEE/ACM Transactions on Networking*, 24(4):2319–2331, August 2016.

[13] M. Saad, L. Njilla, C.A. Kamhoua, D. Nyang, and A. Mohaisen, "Mempool Optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems," in *Proceedings of IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, Seoul, Korea, May 2019.

[14] J. Kim, F. Meng, P. Chen, H. E. Egilmez, D. Bethanabhotla, A. F. Molisch, M. J. Neely, G. Caire, and A. Ortega, "Demo: Adaptive Video Streaming for Device-to-Device Mobile Platforms," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, Miami, FL, USA, September 2013.

[15] J. Koo, J. Yi, J. Kim, M. A. Hoque, and S. Choi, "REQUEST: Seamless Dynamic Adaptive Streaming over HTTP for Multi-Homed Smartphone under Resource Constraints," in *Proceedings of the ACM International Conference on Multimedia (MM)*, Mountain View, CA, USA, October 2017.

[16] J. Koo, J. Yi, J. Kim, M. A. Hoque, and S. Choi, "Seamless Dynamic Adaptive Streaming in LTE/Wi-Fi Integrated Network under Smartphone Resource Constraints," *IEEE Transactions on Mobile Computing*, (To Appear), DOI: `10.1109/TMC.2018.2863234`

[17] M. J. Neely, "Energy Optimal Control for Time Varying Wireless Networks," *IEEE Transactions on Information Theory*, 52(7):2915–2934, July 2006.

[18] Y. J. Mo, W. Lee, D. Nyang, "Dynamic Security-Level Maximization for Stabilized Parallel Deep Learning Architectures in Surveillance Applications," in *Proceedings of the IEEE Symposium on Privacy-Aware Computing (PAC)*, Washington DC, USA, August 2017.