Opportunistic Task Offloading in UAV-assisted Mobile Edge Computing: A Deep Reinforcement Learning Approach

Taewon Song
Department of Internet of Things
Soonchunhyang University
Asan, Korea
twsong@sch.ac.kr

***Abstract*—Mobile edge computing (MEC) aims to extend cloud services to the network edge to reduce network traffic and latency for 5G mobile networks. Unmanned aerial vehicles (UAVs) are being used as assisted edge clouds for large-scale sparsely-distributed user equipment, due to their flexible deployment, wide coverage, and reliable wireless communication. In this paper, we propose a deep Q learning-based opportunistic task offloading algorithm for UAV-assisted mobile edge computing. To this end, we formulate a Markov decision process (MDP) model in which the UAV can choose whether to offload tasks to the cloud server or process them on the local MEC server. Extensive simulations show that our task offloading algorithm outperforms both offload-only and local-only algorithms, ensuring satisfactory service quality for 5G services.**

***Index Terms*—5G mobile networks, task offloading, deep reinforcement learning, DQN, mobile edge computing**

I. Introduction

With the advent of the Internet, wireless devices such as smartphones, tablets, laptops, smartwatches and other smart devices have become more common. These devices are convenient and used for communication, entertainment and work. This trend will continue with new technologies that have increased the demand for wireless connectivity, including fifth generation (5G) technology standards and its beyond. As more people use wireless technology, the need for efficient and reliable wireless networks becomes critical.

Mobile edge computing (MEC) provides cloud computing services at the network edge, reducing latency in 5G mobile networks. With traditional cloud computing, data received by a macro base station (BS) is sent to the cloud for processing, which can result in long processing times depending on the communication and processing environment to and from the core network. However, MEC allows for local processing through the MEC server near the BS, increasing throughput and reducing delay time.

Many user equipments (UEs) struggle to obtain reliable computation services, especially in remote or mountainous areas where communication infrastructures are sparse and MEC environments are uncertain. Fortunately, unmanned aerial vehicles (UAVs) have been used to assist MEC systems in executing computation-intensive tasks due to their flexible deployment and large coverage [1]–[4]. By establishing LoS links with ground UEs, UAVs can act as “flying gateways to MEC servers” and offer significant offloading services with low network overhead and execution latency. Although prior research on UAV-assisted networks has focused mainly on communication aspects [5], [6], there is still some work on UAV-assisted MEC systems, such as trajectory design [5], [7], resource management [8], [9], and computation offloading [9]. However, most existing research has only considered a single UAV for computation offloading.

This paper proposes an opportunistic task offloading algorithm in UAV-assisted MEC (OTO-MEC) based on deep Q-network (DQN). First, we establish a Markov decision process (MDP) model in which the UAV can choose whether to perform local processing on the MEC server or offload the task. We then simulate OTO-MEC with a reinforcement environment tailored for UAV-assisted MEC and verify the proposed technique through extensive simulation. Results show that OTO-MEC processes 548% more computing units than the offloading-only method and 44% more than the local edge computing-only method.

II. System Description

Fig. 1 illustrates an edge computing-enabled UAV cooperative MEC system. Multiple user equipments (UEs) are associated with a UAV, a decision-making agent that manages a specific area. Each UE has tasks to process, and each task has specific compute unit and processing time requirements.

Fig. 2 shows an episode composed of media access phase and task process phase. The task gathering occurs during the medium access phase. The UE transmits its task with the communication protocol according to medium access control (MAC) of 5G standards. After the medium access phase, the queued tasks are processed on a first-in-first-out (FIFO) policy in task process phase. The UAV decides whether to 1) offload the task to the cloud, or 2) process it on the local MEC server per decision epoch.

This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1F1A1076069).

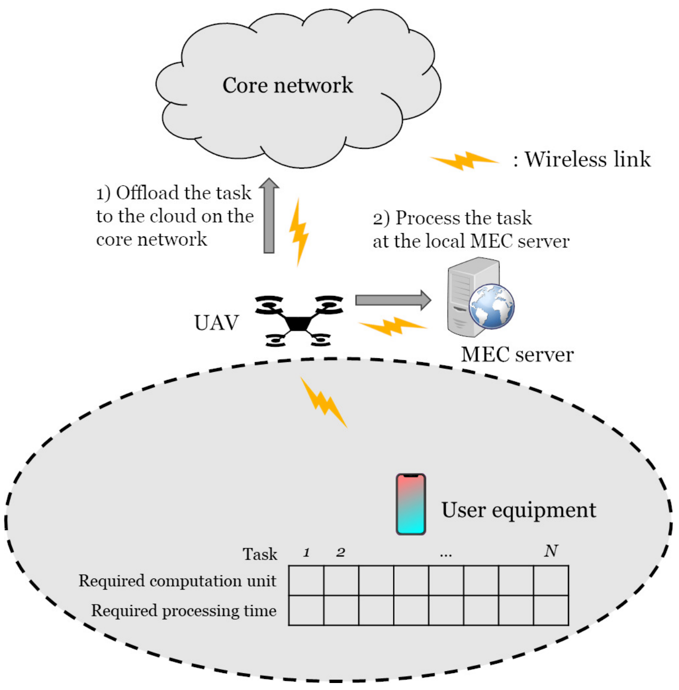
Core network
: Wireless link
1) Offload the task to the cloud on the core network
2) Process the task at the local MEC server
UAV
MEC server
User equipment
Task
1
2
...
N
Required computation unit
Required processing time

Fig. 1. System illustration for OTO-MEC. UE has a number of tasks that consist of required computation units and required processing times.

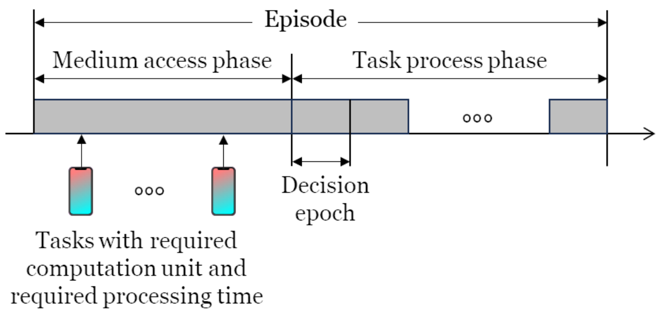
Episode
Medium access phase
Task process phase
Decision epoch
Tasks with required computation unit and required processing time

Fig. 2. The timing diagram of OTO-MEC consisting of medium access phase and task process phase.

III. Markov Decision Process Formulation

The MDP model is an appropriate mathematical decision-making framework. In this section, we present an MDP model for offloading decision.

A. State Space

We define the state space of a finite set $\mathbf{S}$ as follows:

$$\mathbf{S} = \mathbf{F} \times \mathbf{C} \times \mathbf{T} \times \mathbf{Q}, \tag{1}$$

where $\mathbf{F}$ is the set of the available computing unit, $\mathbf{C}$ is the set of the discretized channel conditions from UAV to the portal of the core network, $\mathbf{T}$ is the set of the remain decision epoch, and $\mathbf{Q}$ is the set of information of gathered tasks during medium access phase.

Assuming the MEC server has the maximum f_{max} units, $\mathbf{F}$ is defined as

$$\mathbf{F} = \{0, 1, \cdots, f_{max}\}. \tag{2}$$

Regarding the channel model between UAV and core network, we use a two-state Gilbert-Elliot model [10], [11] to represent the wireless channel from the UAV to the core network. The channel has a good state and a bad state, based on the signal-to-noise ratio (SNR) being above or below a threshold value. Using this model, channel conditions are represented as $\mathbf{C}$ as follows:

$$\mathbf{C} = \{0, 1\}, \tag{3}$$

where $c(\in \mathbf{C}) = 0$ represents a good state and $c = 1$ be a bad state. Additionally, we assume that the wireless channel conditions between UAV and UE, UAV and MEC server are always good.

Given there are t_{max} epochs per an episode, then $\mathbf{T}$ can be defined as

$$\mathbf{T} = \{0, 1, \cdots, t_{max}\}. \tag{4}$$

Next, assuming there are L rooms to be stored in the queue, $\mathbf{Q}$ can be represented as

$$\mathbf{Q} = \prod_{i=1}^{L} \mathbf{Q}_i, \tag{5}$$

that is, $\mathbf{Q}_i$ represents the task information array in the i-th order. As defined in Fig. 1, tasks are gathered on queue in UAV and the queue consists of required computation unit and required processing time. Hence $\mathbf{Q}_i$ is defined in the form of a tuple with two elements as

$$\mathbf{Q}_i = (\mathbf{Q}_i^f, \mathbf{Q}_i^t), \tag{6}$$

where the computational unit required for the task at the ith order in the queue to be executed, $\mathbf{Q}_i^f = \{0, 1, \cdots, f_{max}\}$ and the required time for the task at the same order, $\mathbf{Q}_i^t = \{0, 1, \cdots, t_{max}\}$, respectively.

B. Action

Based on the state information in Section III-A, the agent chooses the action a configured to offload or process locally. To do end, we define the action state as follows:

$$\mathbf{A} = \{0, 1\}, \tag{7}$$

where $a(\in \mathbf{A}) = 0$ and $a = 1$ stand for offload and process locally, respectively. When an offload action is selected, the task from the queue head is forwarded to the core network by the UAV via a wireless channel. Since the channels can be good or bad as defined in (3), it will fail to deliver the task to the core network and send the task later if the channel is bad. In the case when the agent decides to process the task locally, the UAV will deliver the task to the local MEC server.

C. Reward Function

To define the reward function, we consider the weighted processed computation amount since the primary mission is to fully utilize computational capability for both cloud and MEC servers. In particular, reward is defined as the computing unit of the task when the processing time of the task has expired. Additionally, since the core network is a place where tasks

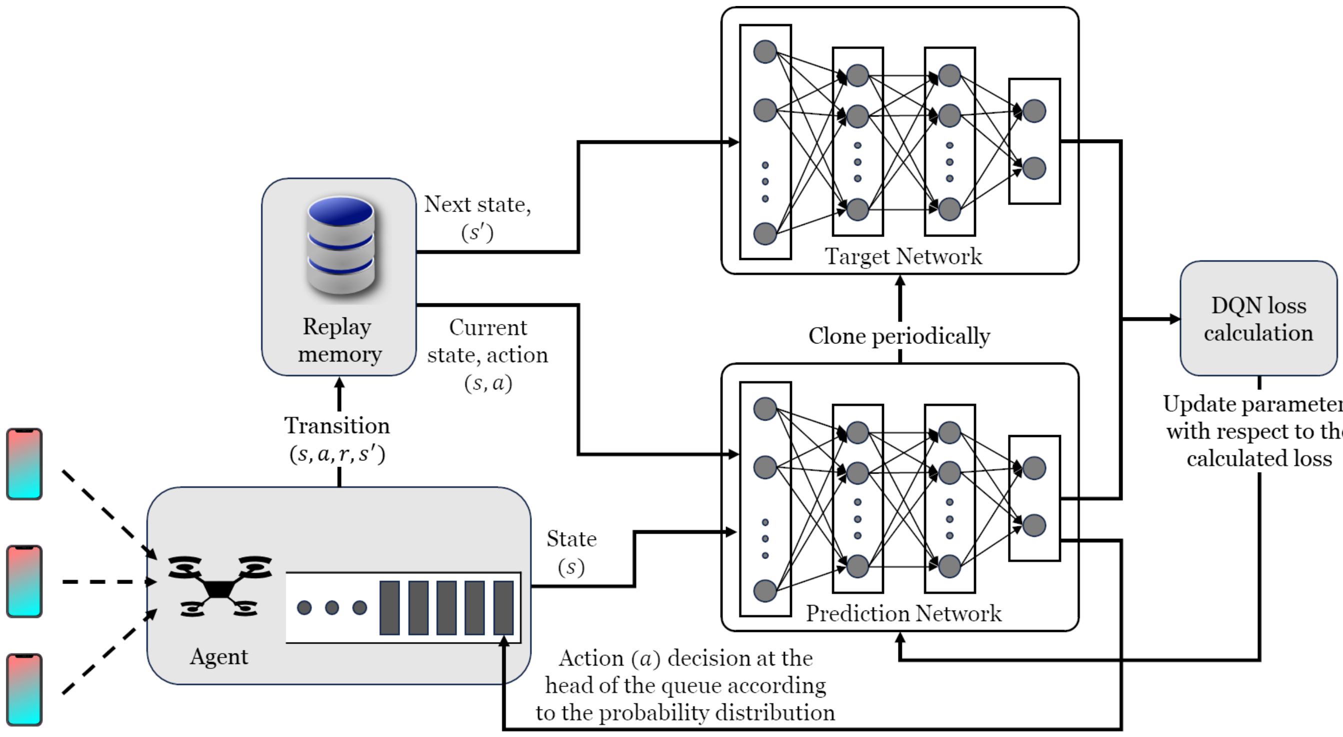

Fig. 3. Neural network architecture with a learning agent, a replay memory, and neural networks.

from multiple UEs are concentrated, congestion and delay that may occur are reflected through a weight. Then the reward is defined as follows:

$$R = weight * q_i^f, \quad \text{if} \quad q_i^t = 0, \tag{8}$$

where $weight$ is 1 when the task is processed at the MEC server and is $w_c(< 1)$ when the task is processed at the cloud server.

IV. Neural Network Architecture

Deep reinforcement learning (DRL) can learn and make optimal decisions in complex and uncertain network environments through trial and error, even with large state spaces. As shown in (1), OTO-MEC has a large number of states. In order to solve the MDP model, we propose using a deep reinforcement learning algorithm. Specifically, we suggest using a DRL model based on the deep Q-network [12], which is a model-free reinforcement learning algorithm that learns the value of actions in specific states. By adopting this approach, we can improve the efficiency of learning.

To tackle the task offloading decision problem, we utilized a DQN architecture as depicted in Fig. 3 that consisted of two neural networks: the target and prediction networks. The target network is responsible for computing the target Q-values, while the prediction network calculates the estimated Q-values. By doing so, we were able to determine the optimal course of action for task offloading with greater precision.

Furthermore, we utilize experience replay to train the DQN. Specifically, by using experience replay technique in the learning process. We store the learned experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in a replay memory at each time epoch. Then, we randomly choose a batch of stored experiences as samples to train the DQN. This can deal with unstable training problem that occurs due to autocorrelation by transforming the problem into one that is similar to a supervised learning problem.

TABLE I
Simulation parameters

Description	Value
Hyperparameters	
Number of input nodes	83
Number of neurons for the hidden layers	42
Number of the hidden layers	2
Optimizer	AdamW [13]
Activation function	ReLU
Learning rate	0.0001
Discount rate	0.99
Batch size of experience replay	128
Environment parameters	
Maximum computation unit for MEC server	100
Epoch size per episode	20
Maximum buffer size	20
Reward weight, w_c	0.1

V. Simulation Results

For performance analysis, we conduct extensive simulations with Python 3.10 along with Gymnasium 0.29.0 [14] which is a standard API for reinforcement learning and PyTorch 2.0.1 [15] which is an open source machine learning framework. We inherit the given gymnasium environment and create

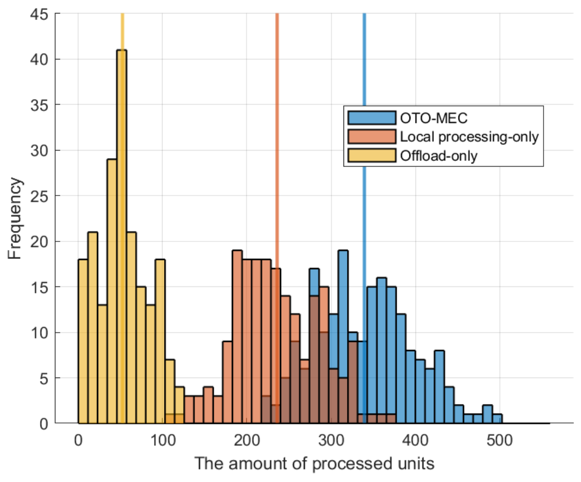

Fig. 4. Histograms of OTO-MEC and comparison algorithms. The test is performed after 2000 training episodes. Each algorithm is tested for 200 different episodes. Vertical bars represent the average value of the amount of processed units for each algorithm.

a custom environment to fit our scenario. Regarding the neural network structure and the environments, we summarize the hyperparameter and environment parameter in Table I.

Fig. 4 displays histograms of the number of processed units for each algorithm. The model was trained for 2000 episodes. OTO-MEC outperforms the other comparative algorithms, including the local processing-only algorithm and the offload-only algorithm. Specifically, in the first algorithm, the UAV always processes the tasks on local MEC servers, while in the second algorithm, the UAV always offloads the tasks to the cloud server. It has been shown that OTO-MEC processes more than 548% of the offload-only algorithm and more than 44% of the local processing-only algorithm.

VI. Conclusion and Future Works

In this paper, we proposed OTO-MEC, an opportunistic task offloading algorithm for UAV-assisted MEC system in which the UAV selects the action among offloading the task to the cloud and processing it on the local MEC server. To find out the optimal policy to maximize the amount of processed tasks, we formulated an MDP that considers available computing unit for MEC, current channel condition, and the task information. Performance analysis demonstrated that OTO-MEC outperforms other comparative algorithms. We will use the custom environment and proposed algorithm as a basis to expand the algorithm's capabilities to cover scenarios involving multiple UAVs and MEC servers. We will also take the power consumption of the UAVs into account for the future work. Additionally, we will investigate the application of federated learning techniques to this expanded environment.

References

[1] Y. Wang, Z. Su, J. Ni, N. Zhang, and X. Shen, "Blockchain-empowered space-air-ground integrated networks: Opportunities, challenges, and solutions," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 160–209, 2021.

[2] N. Zhao, Z. Ye, Y. Pei, Y. C. Liang, and D. Niyato, "Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[3] J. Yao, S. Zhang, Y. Yao, F. Wang, J. Ma, J. Zhang, Y. Chu, L. Ji, K. Jia, T. Shen, A. Wu, F. Zhang, Z. Tan, K. Kuang, C. Wu, F. Wu, J. Zhou, and H. Yang, "Edge-Cloud Polarization and Collaboration: A Comprehensive Survey for AI," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 1–1, 2022.

[4] Z. Ning, H. Hu, X. Wang, L. Guo, S. Guo, G. Wang, and X. Gao, "Mobile Edge Computing and Machine Learning in The Internet of Unmanned Aerial Vehicles: A Survey," *ACM Computing Surveys*, 2023.

[5] N. Zhao, Z. Liu, and Y. Cheng, "Multi-Agent Deep Reinforcement Learning for Trajectory Design and Power Allocation in Multi-UAV Networks," *IEEE Access*, vol. 8, pp. 139 670–139 679, 2020.

[6] G. Yang, R. Dai, and Y.-C. Liang, "Energy-efficient uav backscatter communication with joint trajectory design and resource optimization," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 926–941, 2020.

[7] Q. Luo, T. H. Luan, W. Shi, and P. Fan, "Deep Reinforcement Learning Based Computation Offloading and Trajectory Planning for Multi-UAV Cooperative Target Search," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 504–520, 2023.

[8] W. Liu, B. Li, W. Xie, Y. Dai, and Z. Fei, "Energy Efficient Computation Offloading in Aerial Edge Networks With Multi-Agent Cooperation," *IEEE Transactions on Wireless Communications*, vol. PP, p. 1, 2023.

[9] H. Guo, Y. Wang, J. Liu, and C. Liu, "Multi-UAV Cooperative Task Offloading and Resource Allocation in 5G Advanced and Beyond," *IEEE Transactions on Wireless Communications*, vol. PP, p. 1, 2023.

[10] Q. Zhang and S. Kassam, "Finite-state markov model for rayleigh fading channels," *IEEE Transactions on Communications*, vol. 47, pp. 1688–1692, 1999. [Online]. Available: http://ieeexplore.ieee.org/document/803503/

[11] H. Boujemaa, M. B. Said, and M. Siala, "Throughput Performance of ARQ and HARQ I Schemes over a Two-states Markov Channel Model," *2005 12th IEEE International Conference on Electronics, Circuits and Systems*, pp. 1–4, 12 2005. [Online]. Available: http://ieeexplore.ieee.org/document/4633423/

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[13] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[14] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023. [Online]. Available: https://zenodo.org/record/8127025

[15] "PyTorch." [Online]. Available: https://pytorch.org