J Inf Process Syst, Vol.20, No.2, pp.226~238, April 2024
https://doi.org/10.3745/JIPS.03.0194

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Computation Offloading with Resource Allocation Based on DDPG in MEC

Sungwon Moon and Yujin Lim*

Abstract

Recently, multi-access edge computing (MEC) has emerged as a promising technology to alleviate the computing burden of vehicular terminals and efficiently facilitate vehicular applications. The vehicle can improve the quality of experience of applications by offloading their tasks to MEC servers. However, channel conditions are time-varying due to channel interference among vehicles, and path loss is time-varying due to the mobility of vehicles. The task arrival of vehicles is also stochastic. Therefore, it is difficult to determine an optimal offloading with resource allocation decision in the dynamic MEC system because offloading is affected by wireless data transmission. In this paper, we study computation offloading with resource allocation in the dynamic MEC system. The objective is to minimize power consumption and maximize throughput while meeting the delay constraints of tasks. Therefore, it allocates resources for local execution and transmission power for offloading. We define the problem as a Markov decision process, and propose an offloading method using deep reinforcement learning named deep deterministic policy gradient. Simulation shows that, compared with existing methods, the proposed method outperforms in terms of throughput and satisfaction of delay constraints.

Keywords

Computation Offloading, DDPG, MEC, Resource Allocation

1. Introduction

As vehicular networks begin to evolve rapidly in the 5G era, interactive and intelligent vehicular applications, e.g., autonomous driving, navigation, and virtual reality/augmented reality, are emerging [1,2]. Such vehicular applications typically require computationally intensive computing, which need to be processed with low latency. Despite the improved computing power of the vehicle, processing these applications on vehicular terminal is a heavy computation burden [3].

To alleviate the burden and efficiently facilitate application processing, multi-access edge computing (MEC), a technology that deploys MEC servers (MECSs) with high-capability computing at network edges such as roadside units (RSUs), has emerged [4]. By providing sufficient computational resources close to vehicles, the MEC can satisfy the demand for the heavy computation of vehicles [5]. The vehicle can offload their tasks to MECSs, thereby improving the quality of experience (QoE) of vehicular applications by reducing power consumption and latency.

Manuscript received August 8, 2022; first revision September 15, 2022; accepted September 19, 2022.
* **Corresponding Author:** Yujin Lim (yujin91@sookmyung.ac.kr)
Dept. of IT Engineering, Sookmyung Women's University, Seoul, Korea (sungwon268@sookmyung.ac.kr, yujin91@sookmyung.ac.kr)

However, the efficiency of wireless data transmission, e.g., channel conditions due to the channel interference among the vehicles, and path loss due to the mobility of vehicles time-varying, affects offloading. Moreover, the task arrival of vehicles is actually stochastic, so determining which part of the task to offload to MECS is a challenge in the offloading decision. As a result, offloading methods have become one of the challenges of the dynamic MEC system for channel conditions, mobility of vehicles, and task arrival. Because of dynamic channel conditions and stochastic task arrivals, the MEC system is complex and difficult to optimize and use its resources. Reinforcement learning is applied to the MEC system because it can recognize changes and make corresponding decisions through interactions with the environment. In particular, deep reinforcement learning (DRL) based offloading methods have been proposed [6-13]. Compared to deep Q-network (DQN) based methods [6-8], deep deterministic policy gradient (DDPG) based methods [9-13] have good performance to control continuous quantities such as power consumption or latency because they consider continuous action spaces.

In this paper, we study offloading with resource allocation using DDPG in the dynamic MEC system with stochastic task arrivals and channel conditions. To maximize the long-term reward in terms of throughput and power consumption while meeting the delay constraints of tasks, we formulate the problem that is considered a continuous decision-making problem as a Markov decision process (MDP).

The remainder of this paper is organized as follows. We review the related work in Section 2, and introduce the system model and formulate the problem in Section 3. Section 4 presents the computation offloading method in detail. In Section 5, the performance of the proposed method is analyzed with experimental results. In Section 6, we conclude this paper.

2. Related Work

There are several recent studies using RL for computation offloading in the MEC system [6-16]. In [14,15], the authors proposed an offloading method using Q-learning to obtain optimal offloading decisions with resource allocation. The goal is to minimize energy consumption for all mobile devices, taking into account resource requirements and delay constraints of computation tasks. However, because Q-learning stores states and actions in table form, too large states and actions create curses of dimensions and reduce efficiency.

Thus, offloading methods based on DRL called DQN, which combines deep neural network (DNN) with Q-learning, are being studied [6-8]. In [6], the authors proposed a method using DQN to maximize the long-term utility of vehicular network, considering stochastic channel conditions and traffic. In [7], the authors proposed a method using DQN to obtain optimal policy for offloading under dynamic MEC system. The goal is to minimize the long-term cost, consisting of transmission latency and energy consumption for mobile devices. In [8], the authors proposed a method to minimize the long-term cost, consisting of energy, task queue states, and communication conditions between mobile devices and base station. However, because these methods make decisions in discrete action spaces, they have limitations in processing continuous values such as energy or latency. In [7], to address the problem of discrete action spaces in DQN, the authors proposed a proximal policy optimization (PPO) offloading method. Simulation results show that the PPO method has stable performance and much better results.

Some of the recent offloading researches have adopted a DDPG method based on continuous action spaces [9-13]. In [9], a method using DDPG is proposed to obtain task partitioning and power allocation

decisions that maximize long-term system utility consisting of energy consumption and latency. In [10], the authors proposed a DDPG based method to determine offloading and computing of vehicles to minimize service costs, including latency and service failure penalty. In [11], the authors proposed a method to address task offloading and power allocation policies to minimize long-term energy consumption under latency constraints. In [12], a DDPG based method is proposed to address an optimal offloading policy that minimizes energy consumption and total latency cost.

However, existing methods did not consider uncertain channel conditions and stochastic task arrival in the MDP formulation, so they optimize offloading or resource allocation using predefined criteria. Therefore, we reformulate the problem in the dynamic MEC system without predefined criteria.

There are a few offloading methods using DDPG in the dynamic MEC system [13,16]. In [13], the authors proposed an offloading method to allocate power and local execution resources that minimize energy consumption and queueing latency of the vehicle, and compared DQN and DDPG methods. As a result of the comparison, the DDPG method was more efficient for continuous problems than the DQN method in the dynamic MEC system. Therefore, in this paper, we adopt the DDPG method to learn optimal strategy in the dynamic MEC system. In [16], the authors proposed an offloading method with power allocation to optimize transmission power and local execution power. The reward function is modeled on the task buffer size and power consumption. However, [16] mainly focused on power consumption. Compare to existing methods, the MDP formulation and objective of this paper are different. The goal of this paper is to optimize the long-term reward in terms of throughput and power consumption while meeting the delay constraints of tasks by dynamically allocating CPU resources for local execution and transmission power for offloading.

3. System Model and Problem Formulation

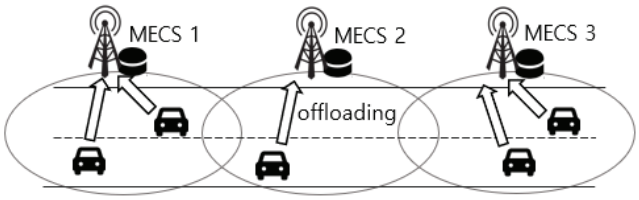


Fig. 1. Computation offloading in the MEC system.

The MEC system proposed in this paper is comprised of a set of MECSs, M, located in RSU and a set of vehicles, N as shown in Fig. 1. The vehicles travel at different speeds on the road within the coverage of an RSU. Time is divided into slots, and which of duration is τ_0. At each time slot t, each vehicle generates tasks based on the mean task arrival rate $\lambda_m = \mathbb{E}[z_n(t)]$, where $z_n(t)$ is a quantification of the number of task arrivals. We assume that tasks can be partitioned. In practice, a task typically consists of multiple threads. For example, the navigation application includes threads of video processing, camera preview, and graphics [17]. The computation tasks of the application can be partitioned because each thread can be regarded as a subtask. It enables the vehicle to offload part of the task to MECS m. The tasks generated from the vehicle n arrive stochastically in the queue of each vehicle n, which is a limited buffer operating in a first-in-first-out manner. Accordingly, each vehicle executes tasks in the queue

locally or offload to MECS by allocating resources for local execution and transmission power for offloading. Because MECS has high computational capability, it can process tasks in parallel and ignore the computation delay [16]. Moreover, the feedback delay can be ignored because the size of the computation result is too small [9]. The transmission delay is the duration of the time slot. Therefore, the amounts of tasks that the vehicle can offload to MECS at time slot t according to the channel state $H_{n,m}(t)$ and the transmission power $p_n^{tr}(t)$ of the vehicle n is as follows:

$$d_n^{tr}(t) = \tau_0 B log\left(1 + \frac{p_n^{tr}(t) H_{n,m}(t)}{\sigma^2}\right), \tag{1}$$

where B is the channel bandwidth and $p_n^{tr}(t) \in [0, P_n^{tr}]$ is the transmission power of the vehicle n. $P_n^{tr}(t)$ is the maximum constraint on the transmission power for vehicle n. σ^2 and $H_{n,m}(t)$ are the background noise power and channel gain between the vehicle n and MECS m, respectively.

$f_n(t) \in [0, F_n^{max}]$ is the CPU cycle frequency allocated to perform the computation task in the vehicle n at time slot t. F_n^{max} is the maximum constraint of the CPU cycle frequency for vehicle n. Therefore, the amount of task that the vehicle can process locally given the allocated local execution frequency $f_n(t)$ is as follows:

$$d_n^{l0} = \frac{\tau_0 f_n(t)}{C_m}, \tag{2}$$

where C_m is the CPU cycle frequency that requires to process one bit of data.

To execute this amount of data, the power consumption of CPU is as follows [18]:

$$p_n^{lo}(t) = \eta (f_n(t))^3, \tag{3}$$

where η is the coefficient according to the hardware chip architecture of vehicle n. Note that $p_n^{lo}(t) \in [0, P_n^{max}]$ indicates local execution power of vehicle n, and $P_n^{max} = \eta (F_n^{max})^3$ is the maximum local execution power of vehicle n.

For time slot t, the amount of task $z_n(t)$ is generated, some bits of tasks marked d_n^{l0} are processed by the vehicle, and some other bits marked d_n^{tr} are offloaded to MECS and then executed by MECS. Therefore, $q_n(t)$ represents the length of the vehicle n's queue at time slot t:

$$q_n(t+1) = [q_n(t) - (d_n^{l0}(t) + d_n^{tr}(t))]^+ + z_n(t). \tag{4}$$

In this paper, the goal of the proposed method is to maximize throughput while minimizing power consumption by dynamically allocating CPU resources $f_n(t)$ for local execution and transmission power $p_n^{tr}(t)$ for offloading. We modeled the problem as a logarithmic function considering power consumption and throughput. As a result, given the maximum frequency constraint F_n^{max} and power constraint P_n^{max}, the following problem is formulated as follows:

$$\max \log_{p_n^{lo}(t) + p_n^{tr}(t)} (d_n^{l0}(t) + d_n^{tr}(t) + 1), \forall t \in T, \tag{5}$$

s.t.

$$f_n(t) \in [0, F_n^{max}], \forall t \in T, \tag{6}$$

$$p_n^{tr}(t) \in [0, P_n^{max}], \forall t \in T. \tag{7}$$

4. Proposed Method

In this section, we propose a DRL-based offloading method in the dynamic MEC system. The objective of the proposed method is to maximize throughput and minimize power consumption while meeting the delay constraints of tasks by allocating optimal resources for local execution and optimal transmission power for offloading. Each vehicle is considered as an agent to learn offload policy independently and interact with the environment. The offloading problem can be defined as a MDP, which comprise of a set of states S, a set of actions A and an immediate reward R, considering the stochastic channel conditions and task arrival. At each time slot t, each vehicle as an agent observes a state s_t, and performs action a_t based on its own observations of environment state. Thereafter, the agent receives the reward r_t and the environment transfer to the next state s_{t+1}. These three MDP components are defined as follows.

State: The state observed in the environment at time slot t by each vehicle agent n to determine the optimal offloading policy is the queue length of vehicle $q_n(t)$ and the channel state information $H_{n,m}(t)$ between the vehicle agent n and MECS m. Thus, the state can be described as:

$$s_n^t = [q_n(t), H_{n,m}(t)]. \tag{8}$$

Action: Based on the current state s_n^t observed by the vehicle agent n, the agent performs an action. The vehicle agent has to find an optimal offloading policy that allocates optimal resources for local execution and optimal transmission power for offloading to MECS. Accordingly, the action can be described as:

$$a_n^t = [f_n(t), p_n^{tr}(t)]. \tag{9}$$

Since the local execution resource $f_n(t)$ and transmission power for offloading $p_n^{tr}(t)$ are allocated within continuous spaces of $[0, F_n]$ and $[0, P_n^{tr}]$, respectively, the action space is defined as a continuous domain.

Reward: The agent receives a reward by taking an action a_n^t, which is expected to provide the highest reward. We aim to optimize the offloading strategy in terms of throughput and power consumption. Therefore, to maximize the throughput and minimize power consumption, the agent allocates the local execution resources and transmission power. The reward function should be associated with the objective. Accordingly, the reward function is defined as follows:

$$r_n^t = \log_{p_n^{lo}(t)+p_n^{tr}(t)}(d_n^{l0}(t) + d_n^{tr}(t) + 1). \tag{10}$$

The DDPG is a method based on an actor-critic architecture applied to a continuous action space. The DDPG based computation offloading method is shown in Algorithm 1. The DDPG method obtains optimal policy through iterative policy improvement and evaluation, which is used by the actor for policy improvement, and the critic for policy evaluation. The DDPG method adopts DNNs on actor and critic, respectively, and is called actor network and critic network. The state is the input of the actor network, and the determined action value is the output. The state and action are the input of the critic network, and Q-value is the output. The actor network comprises of evaluation network μ with parameter θ^{μ}, and a target network μ' with parameter $\theta^{\mu'}$. The critic network comprises of evaluation network Q parameter θ^{Q}, and the target network Q' with parameter $\theta^{Q'}$.

Algorithm 1. DDPG based computation offloading method

Initialize actor network $\mu(s|\theta^{\mu})$ and critic network $Q(s,a|\theta^{Q})$
Initialize target network with weights $\theta^{Q'} = \theta^{Q}$ and $\theta^{\mu'} = \theta^{\mu}$
Initialize experience replay memory
for each episode **do**
  Initialize parameters for simulation in MEC environment
  Generate randomly an initial state s_n^t for each vehicle agent $n \in N$
  for each time slot $t \in T$ **do**
    for each agent $n \in N$ **do**
      Determine the CPU resources for local execution and transmission power for offloading by selecting an action $a_n^t = \mu(s^t|\theta^{\mu}) + \Delta\mu$
      Execute the action a_n^t and receive reward r_n^t
      Store the tuple $(s_n^t, a_n^t, r_n^t, s_n^{t+1})$ into experience replay memory
      Sample randomly a mini-batch of samples from experience replay memory
      Update the critic network and actor network via (11) and (13)
      Update target networks via (15) and (16)

To break up the correlation among data, we adopt experience replay memory. A batch of experience tuples is extracted from experience replay memory in each training step. By minimizing the loss function, the critic network updates its weights as follows:

$$L(\theta^{Q}) = \mathbb{E}\left[\left(y^t - Q(s^t, a^t|\theta^{Q})\right)^2\right], \tag{11}$$

where $Q(s^t, a^t|\theta^{Q})$ means a Q-value approximated by the evaluation network. y^t means a Q-value approximated by the target network.

$$y^t = r^t + \gamma Q\left(s^{t+1}, \mu'\left(s^{t+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right), \tag{12}$$

where $\mu'(s^{t+1}|\theta^{\mu'})$ refers to the action taken by the target actor network at s^{t+1}. By minimizing the loss function (11), the parameter θ^{Q} of the critic network is updated.

Similarly, the actor network updates its parameters using policy gradient as follows

$$\nabla_{\theta^{\mu}}\mathcal{J} \approx \frac{1}{N}\sum_{t} \nabla_a Q(s^t, a^t|\theta^{Q})|_{\,a=\mu(s^t)} \cdot \nabla_{\theta^{\mu}}\mu(s^t|\theta^{\mu}). \tag{13}$$

From (13), according to the direction proposed by the critic, actor parameters are updated at each training step.

DDPG method can handle the exploration and learning process independently. By adding the sampled noise $\Delta\mu$ in the random noise process to the actor, the exploration policy μ' may be configured as follows:

$$\mu'(s^t) = \mu(s^t|\theta^{\mu}) + \Delta\mu. \tag{14}$$

Each agent updates the parameters of the target networks of the actor and critic. It improves the stability of learning by suppressing the rapid change of target values in the learning process by updating as follows:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \tag{15}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}. \tag{16}$$

5. Simulation Results and Discussion

We compared the proposed method with other methods to verify the performance of it. We randomly distributed three MECSs, which are deployed in RSUs, with a communication radius of 250 m within the MEC system. The vehicles were distributed on the road following a Poisson distribution [19], and the mobility speed of the vehicles followed a random distribution of 15–20 m/s. The simulation was conducted with the SUMO simulator [20] and TensorFlow 1.14.0 with Python 3.6. The task arrival rate follows a Poisson distribution with a mean task arrival rate of λ_n Mbps.

The actor and critical networks of DDPG adopted in the simulation consist of four fully connected neural networks with two hidden layers. There are 400 and 300 neurons in two hidden layers, respectively. The ReLU activation function is used for all hidden layers. The learning rate of the actor is 10^{-4} and critic network is 10^{-3}, respectively. The experience replay memory size is 2.5×10^5, and batch size is 128. The remaining parameters used in the simulation are shown in Table 1.

We compared the proposed method to demonstrate the performance with four other methods:

- *Compare1* [13]: *Compare1* modeled the reward function as a weighted sum of power consumption and queueing latency, and set the weight to 0.5. The goal is to minimize power consumption and queueing latency.
- *Compare2* [16]: *Compare2* allocates transmission and local execution power to minimize power consumption and task buffer size. The reward function is modeled as a weighted sum, and set the weight to 0.9.
- *Greedy throughput (GD_T)*: The vehicle agent executes the computation tasks to maximize the throughput by processing as many data as possible locally and offloading without considering power consumption.
- *Greedy power (GD_P)*: The vehicle agent performs the computation tasks to minimize power consumption of the vehicle with minimal data processing in local and offloading.

Table 1. Parameters used in the simulation

Parameter	Value
Number of CPU cycles to process one bit data (C_m)	500 cycles/bit
Background noise power (σ^2)	10^{-9} W
Channel bandwidth (B)	1 MHz
Length of time slot (τ_0)	1 ms
Task arrival rate (λ_n)	1–3 Mbps
Local execution power of vehicle n (P_n^{lo})	2 W
Transmission power of vehicle n (P_n^{tr})	2 W
Maximum CPU cycle frequency of vehicle n (F_n^{max})	2.51 GHz
Discount factor of long-term reward (γ)	0.99
Update parameter for target network (τ)	0.001

Fig. 2 shows the throughput rate between offloading and local computing under different distance rates. The distance rate is defined as the ratio of the distance between the vehicle and the MECS to the communication radius of the MECS. For example, the distance rate of 0.5 equals to 50% of the RSU radius. This is a comparison of the amount of data executed on MECS by offloading and the amount of data executed locally by the vehicle according to the distance. As the distance between the vehicle and the MECS increases, the throughput by offloading decreases, while the throughput by local computing increases. This is because the channel condition is affected by path loss. The closer the vehicle is to the MECS, the less path loss and the better the channel condition. This increase offloading throughput by consuming more transmission power to offload the MECS as the approaches the MECS. Conversely, when the vehicle moves away from the MECS, the vehicle consumes less transmission power for offloading, resulting in less offloading throughput than local execution.

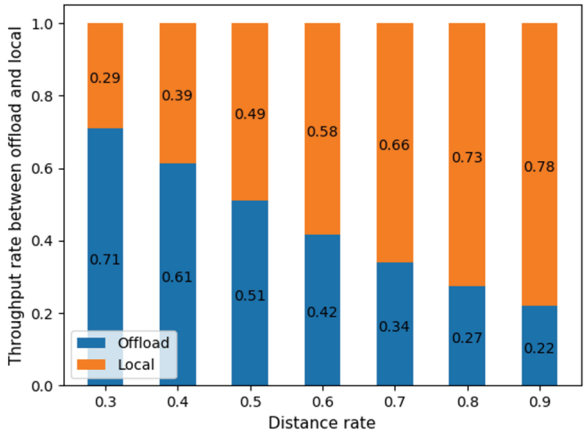


Fig. 2. Throughput rate between offload and local.

In Fig. 3, power consumption and throughput under different distance rates are shown when the task arrival rate is 3 and the number of vehicles is 30. The throughput is defined as the total amount of data executed by local and offload. The larger the distance rate, the greater path loss and the worse the channel conditions. The vehicle consumes more transmission power to offload to their neighboring MECSs. As the distance rate increases, the offloading throughput decreases due to deteriorated channel condition even if it consumes more power than when the distance rate is low. *Compare1* makes an offloading decision considering power consumption and queuing delay, so it consumes slightly less power than *Proposed*. *Compare2* focuses on minimum power consumption, so it consumes less power. However, *Proposed* processes about 18-33% more data than *Compare1* and *Compare2* compared to the amount of power consumption. *GD_T* consumes the most amount of power because it aims to maximize throughput regardless of power consumption. On the contrary, *GD_P* processes the least amount of data because *GD_P* aims to achieve lowest power consumption.

Fig. 4 shows power consumption and throughput under different task arrival rates, respectively. We conducted experiments by setting the distance rate to 0.4 and the number of vehicles to 30. *Proposed* consumes about 23% more power than *Compare1*, and processes about 34% more data than it. *Proposed* is better than *GD_T* because *GD_T* has a throughput growth rate of about 12% compared to the power consumption growth rate, while *Proposed* has a throughput growth rate of about 34% compared to the

power consumption growth rate. *Proposed* processes more data than *Compare2* and *GD_P*, which is focused on minimal power consumption, while it consumes more power than these. As the task arrival rate increases, the load on the system increases, and the increase in power consumption of *Proposed* compared to the load increase rate is lower than that of other methods. This is because *Proposed* considers throughput compared to power consumption. *Compare1* and *Compare2* show that as the task arrival rate increases, power consumption increases and throughput increases rapidly increases. These mainly consider power consumption, so power consumption results in low throughput when the load is low.

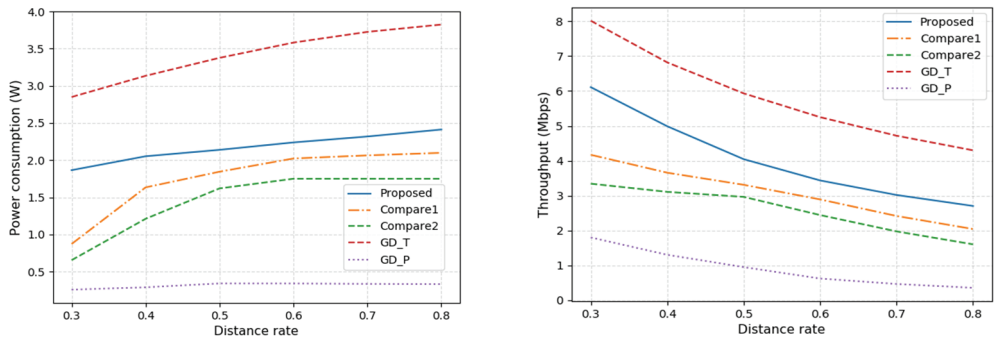


Fig. 3. Performance comparison for the distance rate: (a) power consumption and (b) throughput.

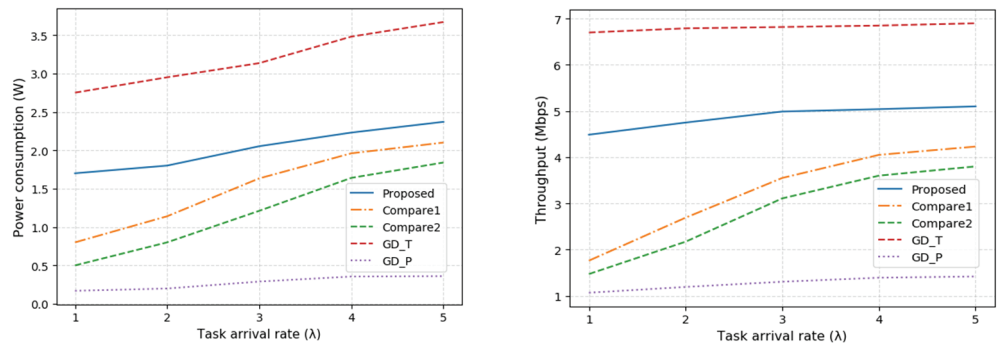
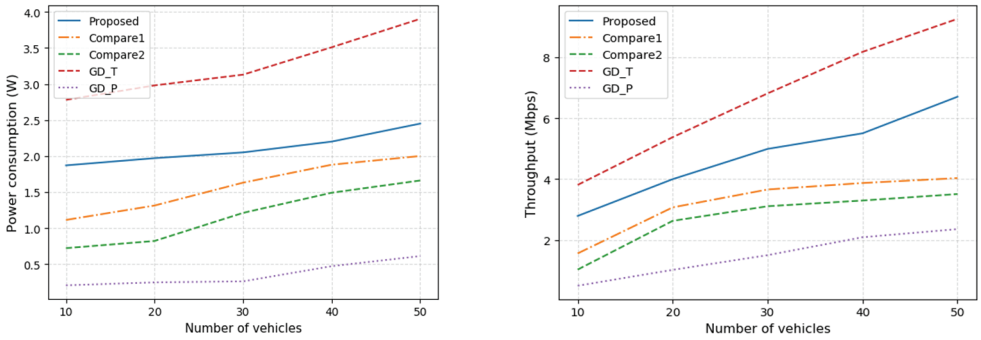


Fig. 4. Performance comparison for the task arrival rate: (a) power consumption and (b) throughput.

Fig. 5 shows the power consumption and throughput under different the number of vehicles. We conducted experiments by setting the distance rate to 0.4 and the task arrival rate to 3. *Proposed* consumes about 25-43% more power than *Compare1* and *Compare2*, while it processes about 33-49% more data than these. *Compare1* and *Compare2* model reward as a weighted sum, and *Proposed* models reward as a logarithmic function. For the reward by using a weighted sum, the weighting coefficient is used to weight on defined parameters. Because the performance of experiments is varying according to the weighting coefficient, the coefficient needs to be selected carefully. However, reward as a logarithmic function uses a ratio in the relationship between defined parameters. Therefore, *Proposed* balances between power consumption and throughput, while reducing power consumption and increasing throughput. *GD_P* and *GD_T* show the best performance on power consumption and throughput, which aims to minimize power consumption and maximize throughput, respectively.



(a) (b)

Fig. 5. Performance comparison for the number of vehicles: (a) power consumption and (b) throughput.

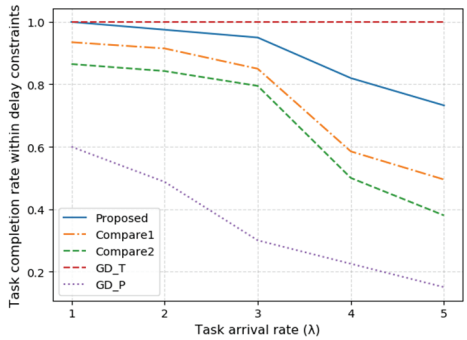


Fig. 6. Comparison of the task completion rate within delay constraints for the task arrival rate.

Fig. 6 shows the task completion rate within delay constraints under different task arrival rates. We conducted experiments by setting the distance rate to 0.4 and the number of vehicles to 30. The rate of task completion within the delay constraint is defined as the ratio of the number of completed tasks while satisfying the delay constraint to the total number of tasks. The task completion rate is affected by the throughput. Low throughput means that there is a lot of tasks in the queue to be processed. This can increase the waiting time in the queue, resulting in a lower task completion rate. *GD_T* has the highest task completion rate because it processes a lot of tasks due to high throughput. *Proposed* is about 10% close to *GD_T*. *GD_P* processes the least amount of data due to low throughput, so there is a lot of tasks in the queue and has the low task completion rate. *Proposed* has a task completion rate of about 16%–24% higher than *Compare1* and *Compare2*.

It is difficult to update the optimal offloading policy in real time by observing the dynamic state of the MEC system. Moreover, in the real world, the environment is very dynamic, and large amounts of tasks occur continuously. Existing methods usually model reward functions as a weighted sum, mainly focusing on power consumption. Because a weighted sum has a fixed weight coefficient of parameters, the performance varies greatly according to the weight coefficient, and it is difficult to select the optimal weight coefficient in the dynamically changing environment. However, the logarithmic function we used does not need to set an optimal weight like a weighted sum. Reward increases in proportion to throughput, and reward decreases rapidly as energy consumption increases using the base of the logarithmic function. It reflects a sharp decrease in QoE for power consumption. Our method has a higher task completion rate within the delay constraint due to higher throughput and consumes a little more power than existing

methods. Our method is more efficient because it processes a larger amount of data than it consumes a little more power, which can lead to improved QoE. However, there is a limitation in that the status of MECS is not considered. Considering this, future research will be conducted.

5. Conclusion

In this paper, we considered the dynamic MEC environment with stochastic channel conditions and task arrival rate. We proposed a DRL based offloading method with resource allocation to maximize the long-term reward in terms of throughput and power consumption by dynamically allocating CPU resources for local execution and transmission power for offloading. We described the system model and formulated the offloading problem as MDP. The DDPG is adopted to optimal offloading method, and each vehicle as an agent learns independently its offloading policy. We compared the method proposed in this paper with conventional methods through simulations, and simulation results indicated the proposed method minimizes power consumption of vehicle and maximizes throughput by local execution and by offloading according to the distance between the vehicle and MECS, and the task arrival rate.

In future research, we will study an offloading method how to select the MECS for processing the tasks in consideration of environment such as MECS status. In addition, we will consider each agent learning in cooperation with each other rather than learning independently in a dynamic MEC system. We will extend the offloading method based on a counterfactual multi-agent policy gradient learning, which determines how much each agent contributes to the overall reward in cooperation with each other.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1047113). This paper is the extended version of the Annual Spring Conference of KIPS (ASK 2022) held in Seoul, Republic of Korea dated May 19-21, 2022 [13].

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: a survey on enabling technologies, protocols, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015. https://doi.org/10.1109/COMST.2015.2444095

[2] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, “Edge computing for autonomous driving: opportunities and challenges,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697-1716, 2019. https://doi.org/10.1109/JPROC.2019.2915983

[3] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, “Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems,” *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773-785, 2020. https://doi.org/10.1109/JIOT.2019.2953047

[4] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, “Deep Learning empowered task offloading for mobile edge computing in urban informatics,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7635-7647, 2019. https://doi.org/10.1109/JIOT.2019.2903191

[5] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377-4387, 2019. https://doi.org/10.1109/JIOT.2018.2876298

[6] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168, 2019. https://doi.org/10.1109/TVT.2019.2935450

[7] A. Sadiki, J. Bentahar; R. Dssouli and A. En-Nouaary, "Deep reinforcement learning for the computation offloading in MIMO-based edge computing," *Ad Hoc Networks*, vol. 141, article no. 103080, 2023. https://doi.org/10.1016/j.adhoc.2022.103080

[8] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," i*n Proceedings of 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, USA, 2018, pp. 1-6. https://doi.org/10.1109/VTCFall.2018.8690980

[9] Z. Cheng, M. Min, M. Liwang, L. Huang, and G. Zhibin, "Multi-agent DDPG-based joint task partitioning and power control in fog computing networks," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 104-116, 2022. https://doi.org/10.1109/JIOT.2021.3091508

[10] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1122-1135, 2020. https://doi.org/10.1109/TCCN.2020.3003036

[11] J. Ren and S. Xu, "DDPG based computation offloading and resource allocation for MEC Systems with energy harvesting," in *Proceedings of 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, 2021, pp. 1-5. https://doi.org/10.1109/VTC2021-Spring51267.2021.9448922

[12] X. Chen, H. Ge, L. Liu, S. Li, J. Han, and H. Gong, "Computing offloading decision based on DDPG algorithm in mobile edge computing," in *Proceedings of 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, Chengdu, China, 2021, pp. 391-399. https://doi.org/10.1109/ICCCBDA51879.2021.9442599

[13] S. Moon and Y. Lim, "Performance Comparison of Deep Reinforcement Learning based Computation Offloading in MEC," *Proceedings of Annual Conference of KIPS*, vol. 29, no. 1, pp. 52-55, 2022.

[14] K. Jiang, H. Zhou, D. Li, X. Liu, and S. Xu, "A Q-learning based method for energy-efficient computation offloading in mobile edge computing," in *Proceedings of 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, 2020, pp. 1-7. https://doi.org/10.1109/ICCCN49398.2020.9209738

[15] B. Dab, N. Aitsaadi, and R. Langar, "Q-learning algorithm for joint computation offloading and resource allocation in edge cloud," in *Proceedings of 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Arlington, VA, USA, Apr. 2019, pp. 45-52.

[16] H. Zhu, Q. Wu, X. J. Wu, Q. Fan, P. Fan, and J. Wang, "Decentralized power allocation for MIMO-NOMA vehicular edge computing based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12770-12782, 2022. https://doi.org/10.1109/JIOT.2021.3138434

[17] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 301-313, 2019. https://doi.org/10.1109/TCC.2016.2560808

[18] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268-4282, 2016. https://doi.org/10.1109/TCOMM.2016.2599530

[19] S. Raza, M. A. Mirza, S. Ahmad, M. Asif, M. B. Rasheed, and Y. Ghadi, "A Vehicle to vehicle relay-based task offloading scheme in vehicular communication networks," *PeerJ Computer Science*, vol. 7, article no. e486, 2021. https://doi.org/10.7717/peerj-cs.486

[20] P. A. Lopez, M. Behrisch, L. B. Walz, J. Erdmann, Y. P. Flotterod, R. Hilbrich, L. Lucken, J. Rummel, P. Wagner, and E. Wiessner, “Microscopic traffic simulation using SUMO,” in *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, USA, 2018, pp. 2575-2582. https://doi.org/10.1109/ITSC.2018.8569938

Sungwon Moon **https://orcid.org/0000-0003-1255-0387**

She received B.S. degree in IT Engineering from Sookmyung Women's University, Korea, in 2019. Her research interests include edge computing, intelligent agent system, and artificial intelligence.

Yujin Lim **https://orcid.org/0000-0002-3076-8040**

She received B.S., M.S., and Ph.D. degrees in Computer Science from Sookmyung Women's University, Korea, in 1995, 1997 and 2000, respectively, and Ph.D. degree in Information Sciences from Tohoku University, Japan, in 2013. From 2004 to 2015, she was an associate professor in Department of Information Media, Suwon University, Korea. She joined the faculty of IT Engineering at Sookmyung Women's University, Seoul, in 2016, where currently she is a professor. Her research interests include edge computing, intelligent agent system, and artificial intelligence.