J Inf Process Syst, Vol.21, No.1, pp.63~70, February 2025
https://doi.org/10.3745/JIPS.03.0203
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Edge Server Activation Control Method with Early Congestion Avoidance by Using Deep Q Network

Minseok Koo[1] and Jaesung Park[2,*]

Abstract

Edge networks have emerged as a solution to provide high-speed and localized data processing by utilizing distributed computing models with numerous edge servers (ESs). However, the increasing deployment of ESs has significantly escalated energy consumption, raising critical concerns regarding operational costs, environmental sustainability, and economic efficiency. A dynamic network topology management approach has been proposed to address this issue by adaptively switching the operation mode of each ES according to load condition. Even though various methods have been proposed to address this issue, there is still room for performance improvement. To fill the performance gap, in this paper, we take a deep Q network-based approach and devise a novel ES activation control method that dynamically manages ES states, balancing energy efficiency and service quality through a specially designed reward function. Simulation results demonstrate the effectiveness of the proposed approach in reducing the amount of consumed energy while increasing the service quality across diverse scenarios, compared to existing methods.

Keywords

DQN, Edge Network, Edge Server Activation, Energy Efficiency, Service Delay

1. Introduction

The fourth industrial revolution has catalyzed the rapid expansion of data-driven technologies such as the Internet of Things, autonomous vehicles, and smart cities [1]. These advancements demand real-time data processing and ultra-low latency, which makes edge networks increasingly vital [2]. Unlike traditional centralized architectures, edge networks employ distributed computing models, wherein numerous edge servers (ESs) process data at the edge of the network. This paradigm shift addresses the growing need for high-speed, localized data processing. However, the proliferation of edge servers has led to a substantial increase in energy consumption, raising critical concerns not only on the operation cost of the network operators but also on the environmental sustainability and economic efficiency.

Energy-efficient operation of edge networks has emerged as a pressing research challenge, particularly given the exponential growth in energy demands associated with distributed server infrastructures. A promising solution involves dynamically managing edge servers based on their workload status, activating or deactivating servers as needed [3]. As the number of active ESs decreases, the amount of energy consumed by an edge network decrease. However, the service delay of the tasks offloaded to edge

Manuscript received December 17, 2024; accepted December 24, 2024.
* **Corresponding Author:** Jaesung Park (jaesungpark@kw.ac.kr)
[1] Department of Artificial Intelligence Applied, Graduate School, Kwangwoon University, Seoul, Korea (tiger1472999@kw.ac.kr)
[2] School of Information Convergence, Kwangwoon University, Seoul, Korea (jaesungpark@kw.ac.kr)

servers increases because more tasks are concentrated on active ESs. Excessive service delay provided by an edge network can degrade user experience, potentially leading to service disruptions or a decline in trust. Therefore, achieving a balance between the benefits and drawbacks of dynamically managing the states of edge servers is crucial.

This paper addresses these challenges by proposing a novel ES activation control method for edge networks that minimizes energy consumption while ensuring maximum service delay remains within acceptable thresholds. We cast the issue as a combinatorial optimization problem with constraints and devise ES state control method by resolving the problem with the deep Q network (DQN). Especially, we design a novel reward function that can balance the competing objectives of energy savings and constraining the maximum service delay. Through simulation studies across diverse network scenarios, we validate the efficacy of the proposed methodology, demonstrating the energy savings alongside the maintenance of high service quality.

This paper is organized as follows. In Section 2, we formally state the problem. In Section 3, we propose the ES activation scheme for energy efficient edge network. In Section 4, we compare the existing and proposed schemes in terms of the amount of energy consumed and the service delay violation rate through simulation studies. Section 5 concludes this paper.

2. Problem Formulation

We consider an edge network composed of N edge servers. We assume that time is divided into equal-sized slots and users offload tasks to the nearest ES. Each ES i has a service queue X_i that accommodates the workloads imposed by tasks offloaded to it. We denote the capacity of ES i as c_i and its state during a time slot t as $a_i(t) \in \{0,1\}$. If we denote the workload newly imposed on ES i during a time slot t as $x_i(t)$, the dynamics of $X_i(t)$ is expressed $X_i(t+1) = \max(X_i(t) + x_i(t) - c_i(t), 0)$ when $a_i(t) = 1$. To avoid task migration complexity, we assume that an ES determined at the end of a slot t to sleep during the next slot processes all tasks in $X_i(t)$ before transitioning to sleep Since the slot length is much larger than task processing time, $X_i(t+1)$ becomes zero when $a_i(t) = 0$. Then, if we denote the maximum service delay during a time slot t becomes $d_i(t) = \max(X_i(t-1), X_i(t))/c_i(t)$. Then, the maximum service delay in the edge network during a time slot is given as

$$d(t) = \max\big(d_i(t)\big). \tag{1}$$

According to [4], the energy consumed by ES i is given as $P_i(t) = \alpha E_m + (1-\alpha)E_m l_i(t)$, where E_m and $l_i(t)$ are the load and the maximum power of the ES respectively, and $0 \leq \alpha < 1$. Then, the amount of energy that an edge network consumes during a time slot is given as

$$P(t) = \sum_{i=1}^{N} a_i(t) P_i(t). \tag{2}$$

Therefore, we can formalize ES activation control problem as the following combinatorial optimization problem:

$$\begin{aligned} a_*(\mathrm{t}) = & \underset{a(t)=\{a_1(t),\ldots a_N(t)\}}{\text{minimize}} P(t) \\ & \text{satisfying } d(t) \leq d_{th} \text{ and } \textstyle\sum_{i=1}^{N} a_i(t) \geq 1, \text{ for all } t. \end{aligned} \tag{3}$$

To resolve the formulated problem, we take a DQN-based approach.

3. DQN Agent for Edge Server Activation Control

DQN is a representative deep reinforcement learning method that combines traditional Q-learning with neural networks. Specifically, DQN utilizes deep neural networks to approximate the state-action value function $Q(s, a)$ which represents the expected cumulative future rewards a DQN agent can obtain by taking action a in state s. This enables DQN to learn even in problems with very large states or behavioral spaces and show effective results in problems dealing with high-dimensional state spaces.

To resolve the problem formulated in Eq. (3) by using the DQN, we define the state space S, and the action space A as follows. As we can see in Eq. (1) and Eq. (2), $d(t)$ and $P(t)$ are influenced by $\{(X_i(t), x_i(t), c_i): i = 1, \dots, N\}$. Therefore, we define the state of an edge network at the start of a time slot t as $s(t) = [X_i(t), x_i(t), c_i]$. Since our goal is to determine an optimal $\{a_i(t), \dots, a_N(t)\}$, we define the action of our DQN agent as $a(t) = \{a_i(t): i = 1, \dots, N\}$.

The reward function provides guidance for a DQN agent to learn which actions are beneficial and which are not. Since the agent improves its policy in the direction of reward maximization, the reward function serves as a crucial criterion for determining the optimal actions in a given environment. Our objective is to minimize the energy consumption of the edge network while ensuring that the maximum service delay provided by the network remains below a certain threshold. Therefore, both energy consumption and maximum service delay must be incorporated into the reward function. The impact of each performance factor on the reward varies depending on $s(t)$. However, existing methods often overlook this aspect and define the reward function as a linear combination of these two performance factors using heuristic approaches [5]. Consequently, these methods exhibit suboptimal performance in terms of energy consumption and maximum service delay violation rates. To address this issue, we differentiate the impact of the two performance factors on the reward function based on the load state of the edge network. Specifically, if we denote the minimum and maximum energy that an edge network can consume as P_m and P_M, respectively, we propose the following reward function:

$$r(t+1) = \begin{cases} -(P(t)/N - P_m)/(P_M - P_m) & if\ d(t) < d_z \\ -d(t)/d_{th} & if\ d_z \le d(t) < d_{th}. \\ -3 & if\ d_{th} \le d(t) \end{cases} \quad (4)$$

In Eq. (4), we introduce an early congestion detection parameter $d_z = zd_{th}$ $(0 \le z \le 1)$. The rationale behind using d_z is as follows. Unlike supervised learning using indicative feedback, the DQN agent is learned using evaluative feedback based on rewards. Therefore, an agent learned according to rewards set based on d_{th} will act in a direction that makes $d(t)$ smaller than d_{th}. In this case, $d(t)$ will oscillate around d_{th}. Additionally, in an actual system, the agent cannot know the necessary $s(t)$ in advance when determining $a(t)$, so it uses predicted $s(t)$. As a result, $a(t)$ may be accompanied by errors, and even when determining the optimal $a(t)$ using the predicted $s(t)$, it cannot guarantee that $d(t)$ will be smaller than d_{th}. To mitigate such influences, we introduce d_z. In other words, we guide the agent to determine actions that make $d(t)$ oscillate around d_z, which reduces the probability that $d(t)$ becomes larger than d_{th}.

The reward function in Eq. (4) guides our agent to behave as follows. In low load conditions, the agent uses only energy-based rewards to minimize network energy consumption, thereby decreasing the number of active ESs. When the load increases and $d(t)$ approaches d_{th}, the agent uses only rewards based on the maximum service delay to determine active ESs such that $d(t)$ becomes smaller than d_{th}.

For actions that would make $d(t)$ larger than d_{th}, the agent is given extremely low rewards to discourage such behaviors.

Fig. 1 shows the agent training process. Before training begins, the weights of the Q-network parameterized by Φ and target network parameterized by Ψ are randomly initialized, and the experience replay memory is cleared. The agent selects action $a(t)$ through an ε-greedy policy in the current $s(t)$, thereby determining the state of each ES in the next slot. At the end of the $(t+1)$ slot, the agent observes the reward $r(t+1)$ and the next state $s(t+1)$ according to the determined $a(t)$, and stores the experience $(s(t), a(t), r(t+1), s(t+1))$ in the replay memory. Then, we randomly sample a batch from the replay memory and calculate the loss function based on the Bellman equation, $L(t) = r(t+1) + \gamma \max_{a \in A} Q(s(t), a(t); \Psi) - Q(s(t), a(t); \Phi)$. $L(t)$ is backpropagated on the Q-network to update its parameters Φ. The target network is periodically updated by synchronizing its weights with the weights of the Q-network. After completing the training, the DQN agent uses the target network to infer an action according to the greedy method when $s(t)$ is given.

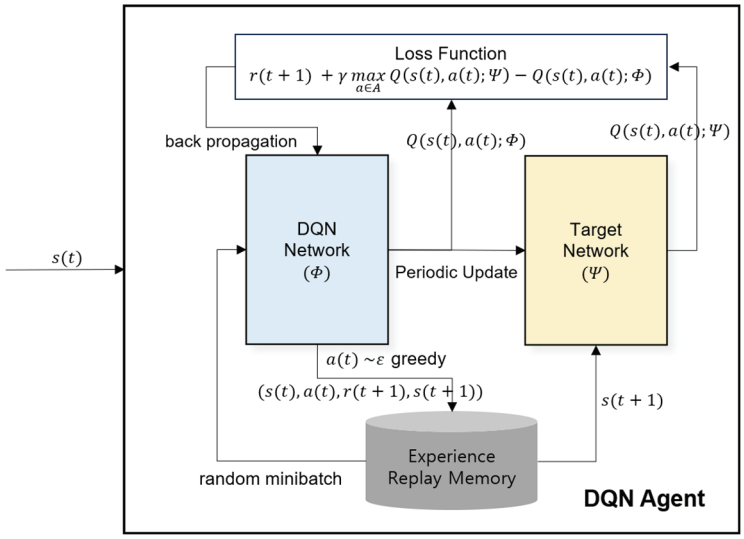

Fig. 1. DQN agent training process.

4. Performance Evaluation

In this section, we validate our method by comparing its performance with that of the existing method using a reward function formed by linear combination of $P(t)$ and $d(t)$. To ensure a fair comparison in the same environment, the DQN models for both methods were configured identically except for the reward functions. Specifically, the reward function that the comparison target model uses is defined as

$$r(t+1) = -\left(w \frac{\left(\frac{P(t)}{N} - P_m\right)}{P_M - P_m} + (1-w)\frac{d(t)}{d_{th}} \right). \quad (5)$$

The Q-network consists of a one-dimensional convolutional neural network (1D-CNN) followed by a

fully connected neural network. The 1D-CNN is designed to process the input state $s(t)$, effectively reducing its dimensionality from $3N$ to $m(\ll 3N)$ by extracting relevant features. Following the feature extraction, the fully connected network comprises H hidden layers, where each hidden layer $h \in [1, H]$ contains l_h nodes employing the rectified linear unit (ReLU) activation function. The output layer subsequently generates the estimated state-action value $Q(s, a)$. When $s(t)$ is given, the agent determines $a(t) = \max_{a \in A} Q(s(t), a)$.

To construct a simulation topology, we use two real-world datasets. One dataset is the Geographical Cellular Traffic (GCT) dataset [6] containing cellular traffic information measured at 21 locations. The other dataset is Alibaba dataset [7] that contains information about machine learning jobs executed on 1,800 machines. By combining the information from these two datasets, we determine the positions of the seven edge servers within the topology to ensure that the traffic load received by each ES is similar. In addition, from these datasets, we derive the user distribution, their mobility information, and the workload they impose during each time slot. To configure simulation parameters, we calculate the load imposed on each ES during each time slot when all ESs are in an active state. Based on these measurements, we set c_i to 70% of the highest load observed across all time slots. We also set the slot length to 30 minutes, α=0.5, E_m=20 W, and d_{th}=0.5 seconds, and d_z=0.6d_{th}.

Since $s(t)$ contains $x_i(t)$, the agent does not know $s(t)$ when it must determine $a(t)$. Generally, prediction methods are often used to estimate $x_i(t)$ at the end of each time slot $(t-1)$ [8]. However, prediction process inevitably entails errors. Thus, the agent uses $y_i(t) = x_i(t) + e_i(t)$ instead of $x_i(t)$. From [8], $e_i(t)$ follows the Gaussian distribution with mean zero and standard deviation σ_e.

To inspect the robustness of each method against the prediction error, we vary σ_e in our simulation study. To configure $e_i(t)$, we follow the approach in [8] and use STGCN (spatio-temporal graph convolutional network) to predict $x_i(t)$. In our simulation configurations, the standard deviation of the prediction error is σ_{STGCN}=1.0. Based on σ_{STGCN}, we adjust σ_e by either increasing or decreasing it (i.e. $\sigma_e = e\sigma_{STGCN}$). Henceforth, we use the following notations to represent each method. $R_{z=0.6}$ and $R_{z=1.0}$ are the proposed method when z=0.6 and z=1.0, respectively. $R_{w=0.6}$, and $R_{w=0.5}$, and $R_{w=0.4}$ are the comparison target methods when w=0.6, 0.5, and 0.4, respectively.

In Fig. 2, we show the amount of energy consumed by an edge network when each method is used with various σ_e. In Fig. 3, we also compare the maximum service delay violation rate (i.e., the proportion of the event $d(t) > d_{th}$ which is denoted by d_v) of the edge network when each method is applied. In these figures, we observe the trade-off between the amount of consumed energy and the maximum service delay violation rate. $R_{w=0.6}$ is trained with a greater emphasis on energy consumption reduction than on service delay, resulting in lower energy consumption but a significantly higher violation rate. In contrast, $R_{w=0.4}$ is trained with a stronger focus on reducing service delay rather than energy consumption, leading to a lower violation rate but significantly higher energy consumption. This is attributed to the way the DQN agent is trained. Existing methods use a reward function formed by linearly combining $P(t)$ and $d(t)$. Therefore, each term acts as a regularizer for the other, depending on the load of an edge network. For example, When the network load is low, the probability of $d(t)$ exceeding the threshold is very small. Thus, efforts should focus on further reducing energy consumption. However, in conventional methods, even if the service delay of active ESs is small, it still impacts the reward, which can lead to keeping ESs active that could otherwise be put to sleep. In contrast, the proposed method determines the reward solely based on energy consumption when $d(t)$ is low, which allows the number of active ESs to be reduced and achieves greater energy savings for the network.

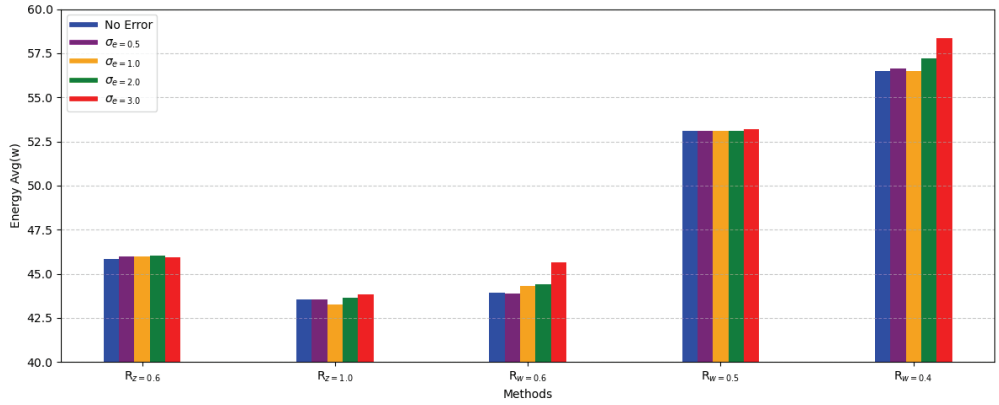

Fig. 2. Comparison of the amount of energy consumed by an edge network. $R_{z=0.6}$ and $R_{z=1.0}$ are the proposed method when z=0.6 and z=1.0, respectively. The other three methods are the existing method with w in Eq. (5) is 0.6, 0.5, and 0.4, respectively. No error represents the case when σ_e=0.

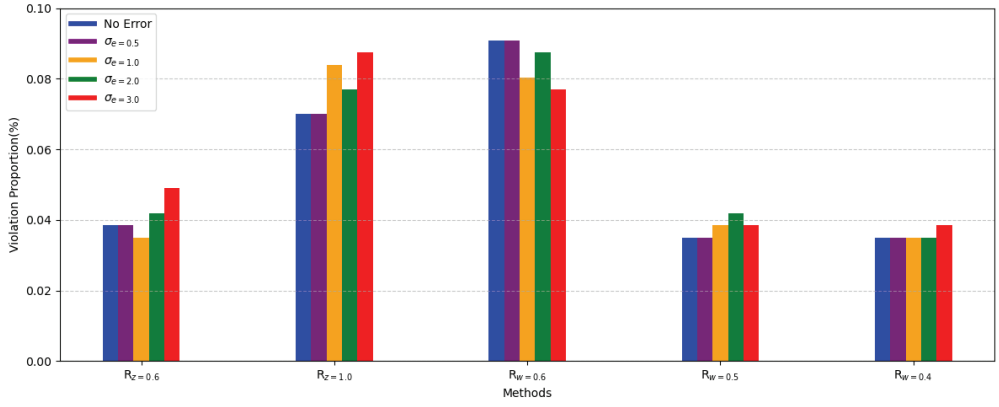

Fig. 3. Comparison of the maximum service delay violation rate. $R_{z=0.6}$ and $R_{z=1.0}$ are the proposed method when z=0.6 and z=1.0, respectively. The other three methods are the existing method with w in Eq. (5) is 0.6, 0.5, and 0.4, respectively. No error represents the case when σ_e=0.

Conversely, when $d(t)$ is large, the risk of being $d(t) > d_{th}$ increases, necessitating an increase in the number of active ESs without considering energy consumption. However, in conventional methods, the reward is always determined by simultaneously considering $P(t)$ and $d(t)$. As a result, even when $d(t)$ is large, these methods may still aim to reduce energy consumption, potentially failing to activate a sufficient number of ESs to prevent the maximum service delay condition from being violated. Therefore, in such cases, conventional methods tend to have a higher violation rate compared to the proposed method.

When we compare $R_{z=1.0}$ and $R_{z=0.6}$, we observe that $R_{z=1.0}$ achieves a smaller amount of consumed energy than $R_{z=0.6}$ for all σ_e. However, $R_{z=0.6}$ achieves a smaller d_z than $R_{z=1.0}$ for all σ_e. This is attributed to the fact that unlike exhaustive search, which explores all possible scenarios to find the optimal solution, DQN determines the optimal action for the current state based on the learned policy during the training process. Therefore, the actions determined by the agent may differ from the optimal results found through exhaustive search. When $d(t)$ approaches d_{th}, such differences can increase the

likelihood of $d(t) > d_{th}$ when the actions determined by the agent is taken. To mitigate this issue, we introduced d_z. When $d(t)$ approaches d_z, if the action determined by the agent results in $d(t) > d_z$, the agent receives a reward based on $d(t)$. Consequently, in such cases, the agent is incentivized to activate more ESs to reduce $d(t)$, thereby decreasing the likelihood of $d(t) > d_{th}$. In Figs. 2 and 3, we can see that the benefits obtained by introducing d_z in terms of the reduction in the maximum service delay violation rate are much higher than the loss in terms of the increase in $P(t)$. For example, even when $\sigma_{e=0.0}$, compared to $R_{z=1.0}$, $R_{z=0.6}$ increases the energy consumption by 4.98% but decreases the maximum service delay violation rate by 81.82%. We can observe the same effects in Fig. 4 that shows a box plot for the service delay provided by each server when $\sigma_{e=0.0}$.

From the simulation results, we conclude that compared to other approaches, our method achieves a better balance between the competing objectives of energy savings and constraining the maximum service delay.

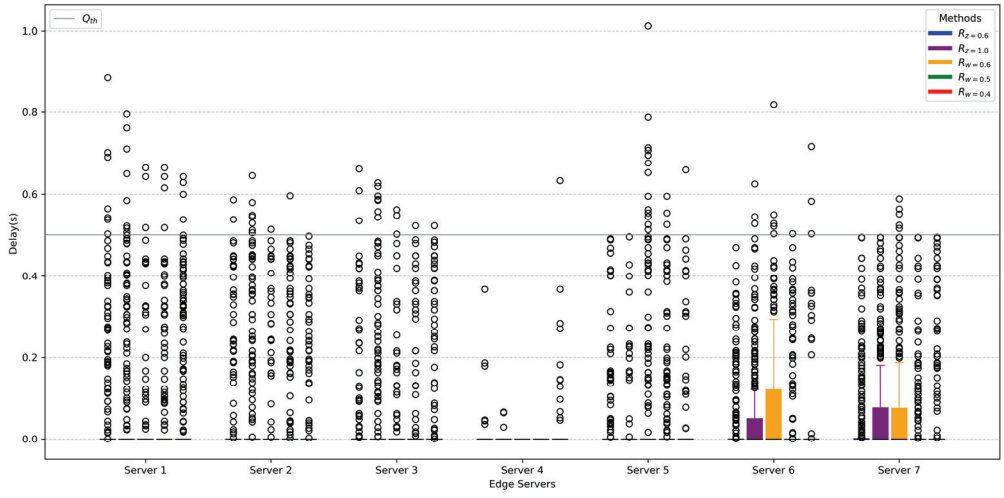

Fig. 4. Box plot illustrating the distribution of the service delay provided by each server ($\sigma_{e=0.0}$).

5. Conclusion

In this paper, we have proposed an edge server activation scheme by using a DQN. In the proposed scheme, we separate the influence of the energy consumption and the maximum service delay of an edge network on the reward function according to the load level of the network. In addition, we introduce an early congestion detection parameter to reduce the maximum service delay violation rate. From the simulation studies with real-world datasets, we show that our method achieves a better balance between the amount of energy consumed by an edge network and its maximum service delay violation rate.

Conflict of Interest

The authors declare that they have no competing interests.

Funding

Following are results of a study on the “Convergence and Open Sharing System” Project, supported by the Ministry of Education and National Research Foundation of Korea.

References

[1] J. Yu, A. Alhilal, P. Hui, and D. H. Tsang, “Bi-directional digital twin and edge computing in the metaverse,” *IEEE Internet of Things Magazine*, vol. 7, no. 3, pp. 106-112, 2024. https://doi.org/10.1109/IOTM.001.2300173

[2] H. Bae and J. Park, “Proactive service caching in a MEC system by using spatio-temporal correlation among MEC servers,” *Applied Sciences*, vol. 13, no. 22, article no. 12509, 2023. https://doi.org/10.3390/app132212509

[3] P. Hou, Y. Huang, H. Zhu, Z. Lu, S. C. Huang, and H. Chai, “Intelligent decision-based edge server sleep for green computing in MEC-enabled IoV networks,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 2, pp. 3687-3703, 2024. https://doi.org/10.1109/TIV.2023.3347833

[4] S. Wang, X. Zhang, Z. Yan, and W. Wenbo, “Cooperative edge computing with sleep control under nonuniform traffic in mobile edge networks,” *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4295-4306, 2019. https://doi.org/10.1109/JIOT.2018.2875939

[5] R. Ma, X. Zhou, H. Zhang, and D. Yuan, “Joint optimization of energy consumption and latency based on DRL: an edge server activation and task scheduling scheme in IIoT,” in *Proceedings of 2022 14th International Conference on Wireless Communications and Signal Processing (WCSP)*, Nanjing, China, 2022, pp. 203-208. https://doi.org/10.1109/WCSP55476.2022.10039283

[6] Github, “Geographical Cellular Traffic (GCT) dataset,” 2023 [Online]. Available: https://github.com/cylin-cmlab/GCT-Prediction.

[7] Github, “Alibaba cluster dataset,” 2020 [Online]. Available: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2020.

[8] M. Koo and J. Park, “MEC server sleep strategy for energy efficient operation of an MEC system,” *Applied Sciences*, vol. 14, no. 2, article no. 605, 2024. https://doi.org/10.3390/app14020605

Minseok Koo **https://orcid.org/0009-0002-7883-6505**

He received B.S. from Kwangwoon University in 2024. Since March 2024, he is with the Department of Artificial Intelligence Applied, Graduate School, Kwangwoon University as a M.S. candidate. His current research interests include reinforcement learning, sensor data analysis, and Wi-Fi sensing.

Jaesung Park **https://orcid.org/0000-0001-8976-6480**

He received B.S., M.S., and Ph.D. degrees in electronic engineering from Yonsei University, Korea in 1995, 1997, and 2002, respectively. From 2005 to 2019, he was an associate professor in the Department of Information Security, Suwon University, Korea. He joined the School of Information Convergence at Kwangwoon University, Korea, in 2019, where currently he is working as a professor. His research interests include artificial intelligence, wireless sensing, and intelligent agent systems.