J Inf Process Syst, Vol.18, No.6, pp.803~812, December 2022
https://doi.org/10.3745/JIPS.03.0180

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

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Deep Reinforcement Learning-Based Edge Caching in Heterogeneous Networks

Yoonjeong Choi and Yujin Lim*

Abstract

With the increasing number of mobile device users worldwide, utilizing mobile edge computing (MEC) devices close to users for content caching can reduce transmission latency than receiving content from a server or cloud. However, because MEC has limited storage capacity, it is necessary to determine the content types and sizes to be cached. In this study, we investigate a caching strategy that increases the hit ratio from small base stations (SBSs) for mobile users in a heterogeneous network consisting of one macro base station (MBS) and multiple SBSs. If there are several SBSs that users can access, the hit ratio can be improved by reducing duplicate content and increasing the diversity of content in SBSs. We propose a Deep Q-Network (DQN)-based caching strategy that considers time-varying content popularity and content redundancy in multiple SBSs. Content is stored in the SBS in a divided form using maximum distance separable (MDS) codes to enhance the diversity of the content. Experiments in various environments show that the proposed caching strategy outperforms the other methods in terms of hit ratio.

Keywords

Edge Caching, Heterogeneous Networks, Reinforcement Learning

1. Introduction

Mobile data traffic is increasing worldwide; by 2023, approximately 70% of the world's population is expected to have mobile connectivity through devices such as smartphones and machine-to-machine (M2M) applications [1]. Devices connected to the Internet provide a variety of services to users, which includes enabling video or VR streaming. However, to enjoy such services, content must be transmitted from a server or cloud and large content data take long time for users to download. In addition, bottlenecks may occur when there are many users. One way to overcome these problems is to use mobile edge computing (MEC). MEC was developed to address problems arising from mobile cloud computing (MCC), and can provide benefits such as fast computing, energy efficiency, and mobility support [2]. In particular, MEC provides the ability to store audio and video content; it is located physically close to end users, and therefore it reduces the transmission data traffic from a server and transmission latency when used as a caching node [2]. Representative facilities that can be used as MEC for caching nodes include small base stations (SBSs), macro base stations (MBSs), and roadside units (RSUs). However, because they have a smaller storage capacity than the server, it is important to decide which content to store and

Manuscript received August 8, 2022; first revision October 4, 2022; accepted October 13, 2022.
* **Corresponding Author:** Yujin Lim (yujin91@sookmyung.ac.kr)
Dept. of IT Engineering, Sookmyung Women's University, Seoul, Korea (potatochips@sookmyung.ac.kr, yujin91@sookmyung.ac.kr)

how much content to cache.

In an environment where the service areas of the SBSs overlap, reducing the duplication of content that is cached in each SBS can more efficiently utilize the limited capacity of SBS while increasing the hit ratio for the content requested by users. In addition, caching a variety of content results in a high hit ratio if the service areas of the caching nodes overlap [3]. Therefore, a method is proposed in this study to increase the hit ratio by reducing the redundantly cached content in SBSs for mobile devices when the service areas of the SBSs overlap.

2. Related Work

The technique of caching to a mobile edge server, such as an MEC, is called mobile edge caching; it reduces the mobile traffic and latency during data transmission [4]. Many studies have been conducted on edge caching with various goals. In [5], a technique of caching in RSUs for vehicles with the aim of reducing the transmission latency and service cost was studied. In [6] and [7], caching methods in RSUs were used to improve the quality of experience (QoE) and quality of service (QoS), respectively. To increase QoE in a vehicular network environment, [6] proposed a deep deterministic policy gradient (DDPG)-based algorithm to cache files optimally in RSUs. To improve the QoS by caching files on vehicles and RSUs in vehicular network environments, [7] studied a long short-term memory (LSTM) model to predict file requests and proposed a Q-learning-based algorithm to increase the hit ratio. In some cases, it is intended to reduce system costs using mobile edge caching. A DDPG-based method was proposed in [8] to decrease the system cost, which comprises evaluating the communication cost and bandwidth cost. In an environment where vehicles request data, two timeslots reflect the mobility of the vehicles. In [9], the system cost was reduced, which included not only the caching cost but also the computation cost with vehicular mobility and service deadline constraints.

In mobile edge caching, the factors to be considered vary depending on the user type. In [10], an artificial intelligence-based method was proposed to use a multitime scale. The multitime scale method helps deal with the high mobility of vehicles by reducing the system cost. In [11], a cellular caching technique was proposed to decrease the delay by grouping cells in heterogeneous cellular networks. It studies a caching strategy for mobile users aimed at minimizing energy consumption for content transmission in an ultra-dense network (UDN) [12]. In this strategy, the cells of the four-tier UDN are clustered and solved by dividing a problem into sub-problems of inter-/intra-clusters. In [13], the Q-learning algorithm was adopted in the UDN environment to cache files with the goal of increasing traffic served directly from SBSs without accessing the MBS.

3. System Model

An MBS and multiple SBSs form a heterogeneous network. The MBS and SBSs are connected with a backhaul link, and the SBS and end users communicate wirelessly using radio access. It is assumed that the MBS can periodically receive status messages from the SBSs to recognize all SBS situations. The MBS is assumed to be a content provider with all K content. If the content requested by a mobile user is not cached in an accessible SBS, the MBS sends the content to the SBS. There are U mobile users who

request content for each time slot t. N = {1, 2, …, N}, K = {1, 2, …, K}, and U = {1, 2, ..., U} denote a set of SBSs, content, and users, respectively. The storage capacity of SBS n is limited to C_n; on exceeding this limit, the content cannot be cached in the SBS. Mobile users are located in the overlapping area of jSBSs, therefore j is the number of SBSs that the users can access, as shown in Fig. 1.

A mobile user in a heterogeneous network receives the requested content from a nearby SBS. The popularity of mobile users requiring content is assumed to follow a Zipf-like distribution [14].

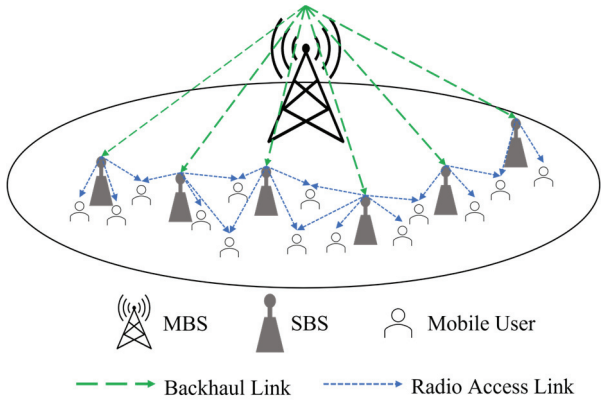

Fig. 1. System model.

$$P_{K(i)} = \left(\sum_{i=1}^{K} \frac{1}{i^{\alpha}}\right)^{-1} \times \frac{1}{i^{\alpha}}. \tag{1}$$

Here α is a parameter affecting the distribution shape and $P_{k(i)}$ denotes the popularity of content k, where $k \in K$. Using a Zipf-like distribution, each mobile user requests content in every time slot.

To compute the popularity of the content requested by mobile users, the MBS collects all requests that enter each SBS. The popularity of content is calculated by counting each requested content at every time slot t. q_k^t denotes the number of requests for content k at time t and MBS calculates the content popularity using q_k^t.

It is assumed that ideal MDS rateless codes are used when caching content to the SBSs. This means that regardless of order, only the number of parity symbols obtained by the mobile user affects the retrieval of full content [15]. The full size of the content is 1, and the content is divided into pieces and distributed to various SBSs through MDS-coded caching. If the content is divided into m pieces, it is set to the default caching size of $1/m$. For example, if $m = 4$, the 0.25-sized content can be stored; therefore, caching can be performed at 0, 0.25, 0.5, 0.75, or 1. $x_{n,k}^t$ is the size of content k cached in SBS n at time t.

4. Problem Formulation

Deep Q-Network (DQN), which emerged in [16], is a type of reinforcement learning that uses deep neural networks to compute a value function. The DQN agent observes the current states, selects an appropriate action, and learns policies through rewards that can be obtained (Fig. 2). In this study, we

aimed to increase the hit ratio of accessible SBSs for the mobile users. Each SBS observes its service area as an environment, assuming the agent is situated in the MBS, and periodically notifies the MBS about these observations. The MBS figures out the current state based on the observations received from the SBSs and determines the content type and size to be cached in the SBSs.

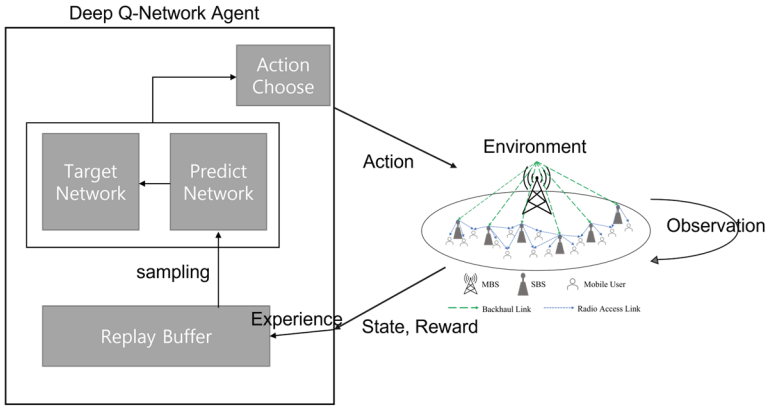

Fig. 2. Interaction between Deep Q-Network (DQN) agent and the environment.

At time t, the state s^t is defined as follows:

$$s^t = \{g_k^t, q_k^t\}. \tag{2}$$

When the set of SBSs that a user can access is J, g_k^t represents the sum of the content k cached in j SBSs at time t.

$$g_k^t = \sum_{n \in J} x_{n,k}^t, x_{n,k}^t \in \left\{0, \frac{1}{m}, \frac{2}{m}, \ldots, 1\right\}, \forall m > 0. \tag{3}$$

Here, $x_{n,k}^t$ is the content k stored in the SBS n at time t, and m determines the cached content size. By storing a variety of small-sized content rather than a small amount of content in a large size, the hit ratio can be increased. This is because the more densely the SBSs are installed, the higher the hit ratio that can be obtained when diverse contents are present [3].

q_k^t denotes the popularity of the content k requested at time t. The number of content k requests at time t is expressed as a vector. It is also important to store diverse content on each SBS in a crowded environment; however, mobile users are more likely to request popular content. If SBS has popular content, it helps to improve the hit ratio. In addition, popular content can change over time, and q_k^t is helpful to figure out the popularity of content even if the popularity of the content changes. All states were used after normalization.

The action a^t is calculated as follows:

$$a^t = \left\{x_{n,k}^t\right\}, \forall n \; and \; \forall k, \tag{4}$$

$$\text{subject to } \sum_{k=1}^{K} x_{n,k}^t \leq C_n. \tag{5}$$

The action determines the cached size of the content k in SBS n at time t, $x_{n,k}^t$. Assuming MDS-coded caching, the full size of all content is the same as 1, and the size that can be cached to the SBS is $\{0, 1/m, 2/m, \dots, 1\}$. There are various types and sizes of content, and the agent observes the states and chooses the action that can provide the highest reward. By using the ε-greedy method, which randomly selects an action with epsilon as the probability, the likelihood of staying in the local optimum solution is reduced. The epsilon declines over time until it reaches a minimum value and creates a well-trained model that chooses less random action.

The goal of the agent is to maximize the hit ratio of the SBSs. h^t is the number of hits, which means that the requested content is downloaded from the SBSs without accessing the MBS at time t. Thus, the reward is defined as the hit ratio, which is calculated by dividing h^t by the total number of requests from users. The reward r^t at time t is given by

$$r^t = \frac{h^t}{\sum_{k=0}^{K} q_k^t}. \tag{6}$$

5. Simulation Results

For the experiments, we assumed a heterogeneous network that included one MBS and multiple SBSs. It is assumed that all mobile users are located in the area where jSBSs overlap. Mobile users request content at each time t and generate the content request using the Zipf-like distribution in (1). Every mobile user can request only one piece of content at a time. If the requested content is not cached in the SBSs close to the users, the SBSs receive the content from the MBS.

We compared the proposed algorithm to three other caching strategies. *Identical random caching* randomly stores content in the SBSs, and all SBSs have the same content. *Random caching* is the same as *Identical random caching*, but it differs in that the cached content is selected independently for each SBS. MDS-coded cooperative caching (*MCC*) is a modified version of the algorithm proposed in [13]. In [13], a Q-learning-based caching method was proposed, and its states were the content to be requested and a previously taken action. In *MCC*, the action determines the size of the content cached in the SBS, and the reward is the amount of traffic directly imported from the SBS without accessing the MBS. Based on this action, all SBSs store the same type and size of content, and all contents are cached using MDS codes. For comparison, *MCC* is modified from Q-learning to DQN, whereas the states, actions, and rewards are not changed. The parameters used in the experiments are listed in Table 1.

Table 1. Experiment parameters

Parameter	Setting value
Number of content (K)	50
Number of SBSs overlapped (J)	[1, 4]
Content segmentation parameter (m)	[1, 4]
Zipf-like distribution parameter (α)	[0.2, 0.8]

The performance criteria are defined as the hit ratio from the SBS and the average update cost. The hit ratio can be calculated using Eq. (6). The content cached in the SBSs is updated over time, and the update

cost is defined as the total size of the content that is changed in the SBS from time t to time $t+1$ [8].

$$update\ cost_n = \sum_{k=1}^{K} |x_{n,k}^{t} - x_{n,k}^{t-1}| \tag{7}$$

$$average\ update\ cost = \frac{1}{N}\sum_{n=1}^{N} update\ cost_n. \tag{8}$$

When the update cost is high, MBS sends more content to SBS. Updating the SBS content depending on the dynamics of the environment can increase the hit ratio. However, if the caching policy primarily focuses on the current situation, the cost of updating the content increases sharply.

The hit ratio was measured while changing the number of SBSs that users could access, as shown in Fig. 3. The difference between the proposed algorithm and *Random* method decreases as the number of SBSs increase, because when there are many SBSs, the content diversity grows accordingly. The *MCC* method learns the amount of content directly served by the SBS as a reward; therefore, it has a smaller hit ratio than the proposed algorithm. Both *Identical random* and *MCC* methods store the same types and sizes of content in every SBS; thus, they have relatively little diversity in content.

Fig. 4 shows the results of measuring the average update cost by changing the number of SBSs that users can request. The cost was calculated using (8) and normalized. *Random* and *Identical random* methods show the best results because they determine the content to be cached randomly every time. The *MCC* method has an average update cost close to zero, because the cached content hardly changes, even in different environments. The proposed algorithm has a lower result than *Random* method but a higher one than *MCC*. This is because the proposed algorithm caches more content, depending on the situation.

Fig. 5 illustrates the hit ratio while changing m from 1 to 4. The proposed algorithm has a higher update cost than *Random* method, but the difference decreases when $m = 4$ because the small size of the cached content encourages rich diversity in all SBSs. Contrastingly, the difference between *MCC* and the proposed algorithm increases because the proposed algorithm selects which content to cache in each SBS; however, in the case of *MCC*, the same content is cached in all SBS. Therefore, the proposed algorithm has more diverse content. In the case of *Identical random* method, the hit ratio increases as m increases, but it shows the lowest result because the cached content is not diverse compared with other algorithms.

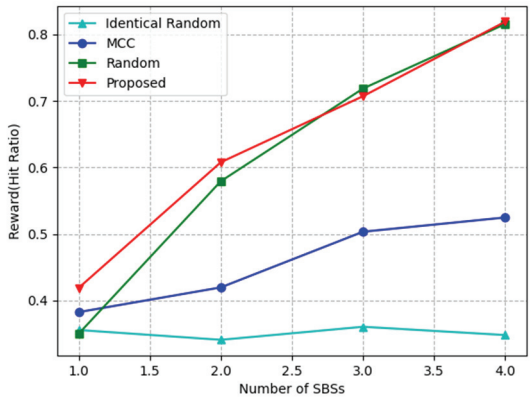

Fig. 3. Hit ratio as the number of SBSs grows.

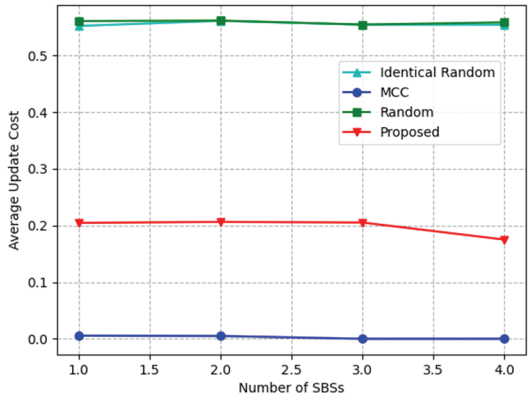

Fig. 4. Average update cost as the number of SBSs grows.

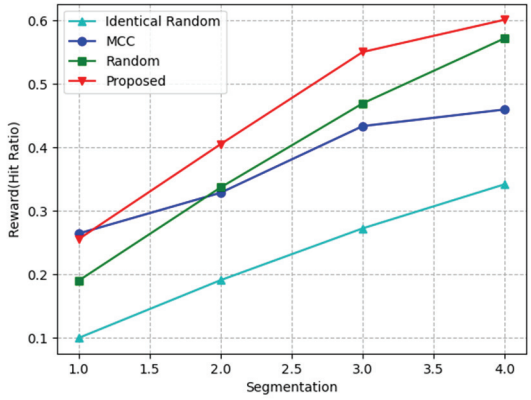

Fig. 5. Hit ratio as content segmentation changes.

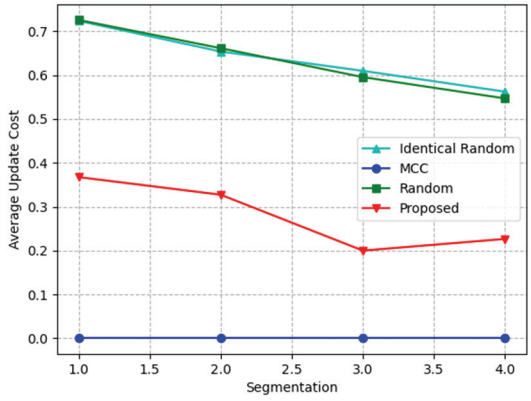

Fig. 6. Average update cost as the content segmentation changes.

In Fig. 6, the average update cost is measured by increasing m from 1 to 4. *Random* and *Identical random* methods both cache diverse content in a smaller size as m increases, and relatively less content is adjusted when updating over time. The proposed algorithm also has a low cost as m increases, but it is lower than that of *Random* because the size of the updated content in the proposed algorithm is smaller than that in the *Random* method. In the case of *MCC*, it usually stores similar content and has a cost close to zero.

The hit ratio was measured by changing parameter α used for the Zipf-like distribution, as shown in Fig. 7. The smaller the α (that is lower than 1), the more similar the content popularity to a random distribution. As α increases, a small amount of content becomes more popular, affecting mobile users requesting for their favored content. If SBS has popular content, there are many advantages, such as increasing the hit ratio. The proposed algorithm finds popular content and obtains a better result than *Random* method. *Random* and *Identical random* methods always show a uniform appearance because they randomly cache, and in the case of *MCC*, the larger the α, the more popular content is cached, and it shows a higher result.

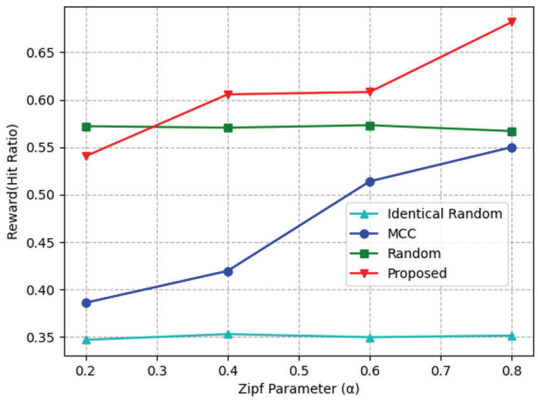

Fig. 7. Hit ratio as α increases.

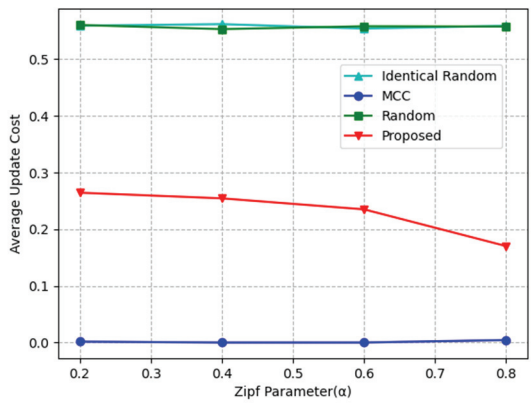

Fig. 8. Average update cost as α increases.

Fig. 8 depicts the effect of α, which is used for the Zipf-like distribution, on the average update cost. *Random* and *Identical random* methods always exhibit the highest and most constant appearances. In the case of the proposed algorithm, the update cost decreased when $\alpha = 0.8$. The reason is that popular content is clarified, which helps the algorithm to cache popular content. *MCC* shows the lowest result compared to the other algorithms because *MCC* knows the content that was cached in SBS at a previous step.

6. Conclusion

In this study, we investigate a DQN-based caching strategy that increases the hit ratio for mobile users

in overlapping SBS coverage. Diverse content is provided to mobile users by reducing the duplication of cached content in the SBSs. Using MDS coding, it was assumed that the content was partially coded and cached. The state of the DQN is determined by considering the time-varying popularities of content and duplicated content in several SBSs. The reward is the hit ratio that stores the appropriate content for each SBS. According to the experimental results, depending on the number of SBSs where the mobile users downloaded content, the proposed algorithm shows an improvement of approximately 26% on an average over *MCC* in the hit ratio, and about 64% on an average over *the Random* method in the update cost. As a result of measuring while varying the size of the content to be cached, the hit ratio of our algorithm increased by approximately 15% on average compared to both the *MCC* and *Random* methods, and the update cost decreased by approximately 56% on an average compared to the *Random* method. By changing the popularity of the content required by mobile users, the proposed method shows an improvement of approximately 16% compared to the *Random* method and approximately 30% enhancement compared to *MCC*, and the update cost declined by approximately 58% on average compared to the *Random* method.

In future studies, it will be possible to expand the UDN environment. UDN is an environment in which the access point density is much higher than the user density [17]. In UDN, services requiring large amounts of data traffic can be used. However, when the number of SBSs increases, there are many duplicated contents; thus, an efficient caching technique is needed.

Acknowledgement

This paper is the extended version of paper published in the Annual Spring Conference in KIPS2022 on May 19–21, 2022; titled “Edge Caching Based on Reinforcement Learning Considering Edge Coverage Overlap in Vehicle Environment” by Y. Choi and Y. Lim.

This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the ICT Challenge and Advanced Network of HRD (ICAN) program (No. IITP-2022-RS-2022-00156299) supervised by the Institute of Information & Communications Technology Planning & Evaluation (IITP), and also supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1F1A1047113).

References

[1] Cisco, “Cisco Annual Internet Report (2018–2023),” 2022 [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: a survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, 2018.

[3] P. Lin, Q. Song, and A. Jamalipour, “Multidimensional cooperative caching in CoMP-integrated ultra-dense cellular networks,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1977-1989, 2020.

[4] J. Yao, T. Han, and N. Ansari, “On mobile edge caching,” *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2525-2553, 2019.

[5] J. Chen, H. Wu, P. Yang, F. Lyu, and X. Shen, “Cooperative edge caching with location-based and popular contents for vehicular networks,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10291-10305, 2020.

[6] C. Song, W. Xu, T. Wu, S. Yu, P. Zeng, and N. Zhang, "QoE-driven edge caching in vehicle networks based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5286-5295, 2021.
[7] R. Wang, Z. Kan, Y. Cui, D. Wu, and Y. Zhen, "Cooperative caching strategy with content request prediction in Internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 8964-8975, 2021.
[8] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 247-257, 2020.
[9] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: a deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10190-10203, 2018.
[10] L. T. Tan, R. Q. Hu, and L. Hanzo, "Twin-timescale artificial intelligence aided mobility-aware edge caching and computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3086-3099, 2019.
[11] F. Rezaei, B. H. Khalaj, M. Xiao, and M. Skoglund, "Performance analysis of heterogeneous cellular caching networks with overlapping small cells," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1941-1951, 2022.
[12] I. Keshavarzian, Z. Zeinalpour-Yazdi, and A. Tadaion, "Energy-efficient mobility-aware caching algorithms for clustered small cells in ultra-dense networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6833-6846, 2019.
[13] S. Gao, P. Dong, Z. Pan, and G. Y. Li, "Reinforcement learning based cooperative coded caching under dynamic popularities in ultra-dense networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5442-5456, 2020.
[14] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proceedings of IEEE Conference on Computer Communications (INFOCOM'99)*, New York, NY, 1999, pp. 126-134.
[15] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402-8413, 2013.
[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with deep reinforcement learning," 2013 [Online]. Available: https://arxiv.org/abs/1312.5602.
[17] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-dense networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2522-2545, 2016.

Yoonjeong Choi **https://orcid.org/0000-0003-1169-1102**

She received B.S. in software convergence from Sookmyung Women's University, Korea, in 2021. Since March 2021, she is with the Department of IT Engineering from Sookmyung Women's University, Korea, as an M.S. candidate. Her current research interests include caching, vehicular networks, and reinforcement learning.

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 at the Department of Information Media, Suwon University, Korea. She joined the Faculty of IT Engineering at Sookmyung Women's University, Seoul in 2016, where she is currently a professor. Her research interests include edge computing, intelligent agent systems, and artificial intelligence.