Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9929318, 10 pages
https://doi.org/10.1155/2021/9929318

WILEY | Hindawi

Research Article

Task Migration Based on Reinforcement Learning in Vehicular Edge Computing

Sungwon Moon,[1] **Jaesung Park**,[2] **and Yujin Lim**[1]

[1]*Department of IT Engineering, Sookmyung Women's University, Seoul 04310, Republic of Korea*
[2]*School of Information Convergence, Kwangwoon University, Seoul 01897, Republic of Korea*

Correspondence should be addressed to Yujin Lim; yujin91@sookmyung.ac.kr

Received 16 March 2021; Accepted 30 April 2021; Published 12 May 2021

Academic Editor: Dongkyun Kim

Multiaccess edge computing (MEC) has emerged as a promising technology for time-sensitive and computation-intensive tasks. With the high mobility of users, especially in a vehicular environment, computational task migration between vehicular edge computing servers (VECSs) has become one of the most critical challenges in guaranteeing quality of service (QoS) requirements. If the vehicle's tasks unequally migrate to specific VECSs, the performance can degrade in terms of latency and quality of service. Therefore, in this study, we define a computational task migration problem for balancing the loads of VECSs and minimizing migration costs. To solve this problem, we adopt a reinforcement learning algorithm in a cooperative VECS group environment that can collaborate with VECSs in the group. The objective of this study is to optimize load balancing and migration cost while satisfying the delay constraints of the computation task of vehicles. Simulations are performed to evaluate the performance of the proposed algorithm. The results show that compared to other algorithms, the proposed algorithm achieves approximately 20–40% better load balancing and approximately 13–28% higher task completion rate within the delay constraints.

1. Introduction

With the rapid development of Internet of Things technologies, numerous complex applications such as automatic driving, augmented/virtual reality, and image recognition have emerged recently. These applications are usually time-sensitive and computation-intensive; they require a large amount of computation resources and high quality of service (QoS) [1]. These high-complexity services are difficult to guarantee; therefore, users offload their computation-intensive tasks to a remote cloud server with sufficient computational resources. However, processing the exponentially increasing requests in the cloud and long-distance communication between users and remote cloud servers leads to considerable latency, which results in failure to meet the QoS requirements. To overcome this problem, multiaccess edge computing (MEC) has emerged as a promising technology [2]. Compared to centralized cloud servers, MEC servers with computing and storage resources are deployed at the edge of networks. Offloading computation-intensive tasks to MEC servers that are geographically closer to users can better fulfil the QoS requirements and improve reliability of computation tasks.

One of the challenges of an MEC system is mobility; this is because users move across the coverages areas of MEC servers [3]. When a user moves out of the coverage area of the current serving MEC server, the system needs to perform task migration from the current MEC server to the target MEC server. Task migration in a vehicular environment with high mobility should be carefully considered. Because the number of task migrations in high-mobility environments is greater than that in low-mobility environments, the choice of the target MEC server highly influences the performance of an MEC system.

Load balancing among vehicular edge computing servers (VECSs) is necessary to meet the QoS requirements of computational tasks and also improve the throughput of the system. A VECS acts as the MEC server in a vehicular environment. Task migration considering load balancing is beneficial because it can reduce the computational congestion of VECSs, which shortens the computation time

required to process a task and improve task throughput in a VECS system. However, task migration requires additional costs of migration time and computation resources. Accordingly, we propose an efficient computational task migration strategy to improve the balance of loads in VECSs and minimize the migration cost in a vehicular environment. We adopt a cooperative VECS group environment that can collaborate with the VECSs in the group [4]. Migrating the tasks to the nearest VECS can result in unequal distribution of loads at a certain VECS; thus, the task may not be completed within the delay constraint due to overload. In addition, the throughput of a VEC system may reduce. Thus, we adopt a task migration strategy in the cooperative VECS group environment to increase the number of candidate VECSs for migration. We can select the best VECS among the candidate VECSs and balance the loads of the VECSs. Load balancing can improve not only the fulfilment of the QoS requirements of a task but also the throughput of computational tasks in a VEC system.

The remainder of this paper is organized as follows. Section 2 introduces related works on computational task migration and load balancing. Section 3 describes the system model and proposed algorithm using reinforcement learning (RL). Section 4 presents and discusses the results of the simulation. Finally, Section 5 concludes the study.

2. Related Work

In this section, we discuss two main categories of related work: computational task migration and load balancing. The high mobility of vehicles across multiple VECSs as well as an increase in the number of vehicles can lead to unbalanced loads among VECSs, which can reduce the QoS and lead to computation delay. Therefore, some studies on load balancing with offloading optimize server selection [5–7].

In [5], authors proposed a joint load balancing and offloading method to maximize the system utility in VEC networks. They designed an algorithm to adopt task computation delay to formulate the system utility. By jointly optimizing server selection, offloading ratio, and computation resources among VECSs, the requirements could be satisfied using a logarithmic utility function. The authors in [6] proposed a task offloading method based on a heuristic algorithm to achieve load balancing of the computation resources among VECSs. The proposed method was aimed at minimizing the processing delay by utilizing the computation resources of the VECS more efficiently. In [7], the load-balancing problem was handled in a distributed *software-defined networking* framework using an M/M/1 queuing system that optimized the migration operations that select the sever to which the task should be migrated. To achieve better load balancing, the trade-off between load balancing and migration operation costs was optimized. The load consisted of each server's response time; the migration cost was the sum of the initial and target servers' response time. The loads were balanced by lowering the average response time. In this study, we perform load balancing with the number of computational task requests and reduce migration cost with the size of the task; therefore, loads are balanced by lowering the difference between the number of tasks at each VECSs. In addition, we adopt RL. Because of the high mobility and time-sensitive services, the environment becomes more complex and dynamic; thus, there are limitations in obtaining optimal solutions with heuristic algorithms.

Task migration has been seen as an efficient solution to deal with the QoS and latency issues faced due to the high mobility of users and high-complexity applications. There are few studies, addressing the task migration problem, that utilize RL, such as Sarsa [8], Q-learning [9, 10], and deep Q-network (DQN) [10–13].

The authors in [8] proposed an on-policy RL- (i.e., "Sarsa-") based computation migration method to determine the optimal VECS. The aim was to minimize the overall response time of the offloaded computation tasks. The decision was based on the VECS hosting the previous task, task characteristics, and vehicle locations. In [9], the authors addressed joint task offloading and migration schemes based on Q-learning to obtain the maximum system revenue. The decision was made based on the property of computation tasks, mobility of users, and cell sojourn time of users.

In [10], the authors considered a single user scenario by exploiting the predefined mobility of the user; the user passes through many VECSs and the corresponding services decide the migration strategy and communication strategy. The authors proposed a Q-learning and DQN-based service migration algorithm to minimize the migration and communication costs, respectively, for detailed evaluation. In [11], a service migration algorithm based on DQN was proposed, which considered the dynamics of the network bandwidth, battery level of mobile devices, and remaining computation. The proposed algorithm was aimed at minimizing the total cost, which consisted of delay and energy consumption. The authors in [12] proposed DQN-based computation migration and resource allocation; the goal was to minimize the total cost, which consisted of the delay cost, energy computation cost, and bandwidth cost. In [13], a DQN-based algorithm was designed for task migration and resource management at the network edge. With the model mobility, the goal was to make the right decision to balance the migration cost and data transfer cost to reduce the overall cost.

As discussed above, some studies were aimed at maximizing the system utility or minimizing the overall delay of tasks. In contrast, in this study, we aim to balance the loads of the VECSs and minimize the migration cost while satisfying the vehicle requirements. In addition, we adopt RL to better manage the dynamic and complex environments in VEC.

3. Methodology

3.1. System Model. As shown in Figure 1, we consider a VEC system consisting of M VECSs, each of which is deployed at a roadside unit (RSU). The VECSs communicate among each other through backhaul links, which allows load balancing among them. The RSUs are connected to a base station equipped with a VECS controller. The VECS controller can collect information on vehicles and VECSs in a system for making migration decisions for load balancing of VECSs to meet the task requirements of the vehicles.

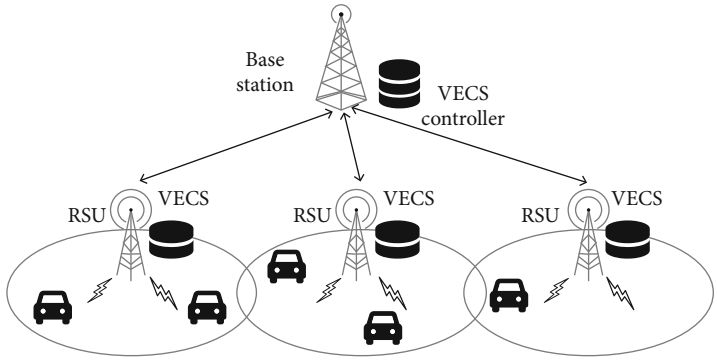

Figure 1: Vehicular edge computing system (VECS) model.

In a system, N vehicles are deployed following a Poisson distribution on the road. The vehicles run at speed v. Vehicle n has a computation-intensive task, which can be described as $s_n = \{\lambda_n, \eta_n, T_n^{\max}\}$, where λ_n denotes the size of the computation task, η_n denotes the computation resources required to process the task, and $T_n^{\max}$ denotes the delay constraint of the task.

Considering vehicle mobility, we focus on a dynamic scenario in which vehicles move across multiple VECSs. Once a vehicle moves out of the coverage of its serving VECS, the VECS controller should decide whether to migrate the computation task of the vehicle to another VECS. The VECS controller makes migration decisions to satisfy the QoS constraints of the task and improves the throughput of the system. The following models were used to describe the system:

3.1.1. Communication Model. The communication model depicts the communication time for a computational task to be transmitted from vehicle n to VECS m via a wireless channel. Each vehicle $n \in N$ offloads its computation-intensive task to one of the VECSs in its communication coverage. The data transmission rate between vehicle n and VECS m is given as

$$R_{n,m} = B_n \log_2\left(1 + \frac{P_n H_{n,m}}{\sigma^2}\right), \tag{1}$$

where B_n and $H_{n,m}$ denote the channel bandwidth and channel gain between vehicle n and VECS m, respectively; P_n is the transmission power of the vehicle; and σ^2 is the power level of the white noise.

The communication time for offloading the computation task of vehicle n to VECS m, T_n^{trans}, is given as follows:

$$T_n^{\text{trans}} = \frac{\lambda_n}{R_{n,m}}. \tag{2}$$

The size of the computation result is smaller than that of the offloading task; therefore, the download time from the VECS to the vehicle is not considered.

3.1.2. Computation Model. The computation model describes the computation time required to execute a computational task. The computing capacity of the VECS m is denoted by C_m, which is measured by the CPU cycles per second. It is assumed that the computation requests that are hosted on VECS m share computing capacity C_m evenly. The computing delay is expressed as follows:

$$T_n^{\text{comp}} = \frac{\psi \cdot \eta_n}{C_m / \Sigma_{n \in N} \phi_n^m}, \tag{3}$$

whereψdenotes how many remaining CPU cycles are required. Further, ϕ_n^m denotes whether task s_n is hosted on VECS m. In the equation, the computation time of VECS increases linearly with the number of computational requests in the VECS, which can lead to unreliable services and increase in the latency of computational tasks. By balancing the computation loads among VECSs, the QoS requirements and throughput of the system can be improved as congestion and bottlenecks of the VECS are reduced.

3.1.3. Migration Model. The migration time is the time required to transmit a task from VECS m to VECS m' via backhaul links when a VECS controller decides to migrate computation task s_n. The migration time is expressed as follows:

$$T_n^{\text{mig}} = \begin{cases} 0, & \text{if } m = m', \\ \dfrac{\lambda_n}{R_b}, & \text{if } m \neq m', \end{cases} \tag{4}$$

where R_b is the bandwidth between VECSs via backhaul links.

Computational migration caused by the mobility of vehicles incurs additional costs, such as the cost for computational replication from VECS m to VECS m' and the resource release of current hosting VECS m [14]. The migration cost is determined by the size of task n to be migrated from VECS m to VECS m'; the migration cost is denoted as follows, where μ is a positive coefficient:

$$C_n^{\text{mig}} = \begin{cases} 0, & \text{if } m = m', \\ \mu|\lambda_n|, & \text{if } m \neq m'. \end{cases} \tag{5}$$

3.1.4. Load Balance Model. One of the goals of this study is to balance the loads among VECSs. The computing load of VECS m can be calculated as follows:

$$L_m = \sum_{m \in M} \sum_{n \in N} \varnothing_n^m, \tag{6}$$

where $\varnothing_n^m$ is a binary variable that determines whether task s_n is hosted on VECS vecs_m, which is denoted by

$$\varnothing_n^m = \begin{cases} 1, & \text{if } s_n \text{ is hosted on vecs}_m, \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

The average computing load of all VECSs is calculated as

$$L = \frac{1}{M} \sum_{m=1}^{M} L_m. \tag{8}$$

To determine whether the computing load is distributed fairly among the VECSs in the system, load balance is measured by the deviation of the computing load at each VECS [15]. In this study, we define the load-balancing factor LB as follows:

$$\text{LB} = \frac{1}{M} \sum_{m=1}^{M} |L_m - L|. \tag{9}$$

3.1.5. Problem Definition. Considering the mobility of vehicles, a VECS controller decides whether and where to migrate the computational task among VECSs. In this study, we aim to achieve the goal of balancing the loads of VECSs and minimizing the migration cost while satisfying the delay constraints of the computation tasks. The two objectives are defined as a weighted sum with weight factor $\omega \in [0, 1]$. Consequently, the problem is formulated as

$$\begin{aligned} \min \quad & \omega \cdot \text{LB} + (1 - \omega) \cdot C^{\text{mig}}, \\ \text{s.t.} \quad & T_n^{\text{total}} < T_n^{\max}. \end{aligned} \tag{10}$$

C^{mig} denotes the total migration costs of the tasks to be migrated into a system. The total computation time for vehicle n is given as

$$T_n^{\text{total}} = T_n^{\text{trans}} + \sum_{i=0}^{\tau} T_n^{\text{mig}} + T_n^{\text{comp}}, \tag{11}$$

where τ is the number of migrations that occur during the task computation time. The size of the problem can increase rapidly as the number of vehicles and VECSs increase. Instead of using the existing optimization strategy, in this study, we propose an RL-based strategy to address this problem.

3.2. Proposed Algorithm. In this section, we propose a computation migration strategy based on RL. The VECS controller acts as an agent. The controller can observe the load states for all VECSs and vehicle information and make a migration decision to balance the loads and minimize the migration cost while satisfying the delay constraint of the computation task.

The Q-learning algorithm is a model-free RL algorithm that learns to determine the optimal policy that maximizes the expected reward. Q-learning consists of a set of states S, set of actions A, and set of rewards R. The agent chooses an action according to the optimal policy based on its current state. The state transitions from one state to another by taking action a. The Q-function is updated as follows:

$$Q(s, \text{a}) = Q(s, a) + \alpha \left[r + \gamma \cdot \max_a Q\left(s', a'\right) - Q(s, a) \right], \tag{12}$$

where s and a indicate the state and action at time t, respectively. When an agent selects action a at time t, state s enters a new state s'. Here, α represents the learning rate. γ and r represent the discount factor and reward at time $t + 1$, respectively. The optimal Q-function can be expressed as

$$Q^*(s, a) = \max_a Q\left(s', a'\right). \tag{13}$$

However, the high dimensionality of the state and action spaces reduces the efficiency of the Q-learning algorithm. To address this problem, we propose a deep Q-learning-based migration strategy that predicts the reward of each action using the DQN, a trained neural network. Deep Q-learning is more applicable to high-dimensional environments with large and complex state spaces.

Deep Q-learning uses a neural network to learn $Q^*(s, a)$. The Q-function in the DQN is defined as $Q^*(s, a; \theta)$, where θ stands for the weights of the main DQNs. Then, the network is trained by minimizing the loss function, denoted as $\text{Loss}(\theta)$, which is given by

$$\text{Loss}(\theta) = \mathbb{E}[y - Q(s, a; \theta)]^2, \tag{14}$$

where $y = r + \gamma \cdot \max_{a'} Q(s', a'; \theta')$ is the target Q-value, and the weights θ' are updated to θ periodically. Algorithm-based deep Q-learning is shown in Algorithm 1. The controller observes state s and takes action a at every interval t. The elements in the RL model are defined as follows:

(1) *State*: to optimize the load balance and reduce the migration cost, the state reflects the task information of the vehicles, list of VECSs serving N tasks, total loads of a VEC system, and migration decision variable. Therefore, state s is defined as

$$s = \{V_1, \text{V}_2, \cdots, V_n, l\}, \tag{15}$$

where V_n denotes $\{\lambda_n, e_n, d_n\}$. λ_n represents the size of the computation task n, e_n represents the VECS that hosts computation task n, and d_n denotes a binary variable that indicates whether the task needs to be migrated. l denotes the sum of the computing loads of the serving VECSs.

Initialize main DQN $Q(s, a; \theta)$ with random weights θ
Initialize target DQN $Q(s, a; \theta')$ with weights $\theta' = \theta$
Initialize replay memory D to capacity N
For each episode **do**
  Initialize initial state s_0, reward r_0
  For time slot $t = \tau,\ 2\tau \cdots \Gamma$ **do**
    The controller acquires information about vehicles, tasks, and VECS by interacting with the environment
    If the random number $< \epsilon$:
      Select action $a_t = \text{argmax}_a Q(s_t, a_t; \theta)$
    Else:
      Randomly select action a_t
    Execute action a_t at controller, observe reward r_t and next state s_{t+1}
    Store the tuple $<s_t, a_t, r_t, s_{t+1}>$ in D
    Randomly sample a minibatch of tuple $\{<s_j, a_j, r_j, s_{j+1}>\}_{j=1}^{J}$ from D
    If episode terminates at $j + 1$, then
      $y_j = r_j$
    Else:
      $y_j = r_j + \gamma \bullet \max_{a'} Q(s_{j+1}, a'; \theta')$
    Perform a gradient descent step on Loss(θ) with respect to the network parameters θ, where loss function is $\text{Loss}(\theta) = (1/J)\sum[y_j - Q(s_j, a_j; \theta)]^2$ and update $\theta' = \theta$
    Terminate when all the vehicles are out of simulation region
  End for
End for

Algorithm 1: Deep Q-learning method.

Table 1: Parameters used in simulation.

Parameter	Value
B	10 MHz
λ	[100,150] MB
η	[0.4,0.5] Gcycles/MB
P_n	1 W
C_m	10 GHz
μ	0.002/MB
σ^2	10^{-11} mW

(2) *Action*: for each time step, the controller selects and executes an action based on its current state. In the proposed VECS network, the controller determines where to migrate tasks for optimizing load balancing and migration cost in a system. Accordingly, the action is denoted as follows:

$$a = \{a_1, a_2, \cdots, a_n\}, a_n \in M, \tag{16}$$

where a_n refers to the serving VECS of task n. If $a_n = e_n$, the computation task does not migrate. If $a_n \neq e_n$, then the task is migrated to a_n.

(3) *Reward*: after each time step, the controller receives feedback from the environment, which is reward r. In general, the RL reward is related to optimization. The goal of our optimization problem is to minimize the average variation in the VECS loads and migration cost. Therefore, we formulate the function of negative reward as follows:

$$r = \omega \cdot \text{LB} + (1 - \omega) \cdot C^{\text{mig}}. \tag{17}$$

4. Experimental Comparison

To illustrate the performance of the proposed algorithm, we considered seven VECSs connected to the RSU environment. The radius of the VECS was 250 m, and the capability of each VECS was 10 GHz. The mobility dataset of San Francisco city was used in the simulation [16]. The cooperative VECS group could have a maximum of 100 vehicles with computation-intensive tasks. Both the size of each task and the required number of CPU cycles per bit followed a random distribution with [100,150] MB and [0.4, 0.5] Gcycles/MB, respectively. The path loss between the RSU and a vehicle was modeled as $140.7 + 36.7 \log_{10}(\text{distance (km)})$ [17]. The deep Q-learning parameters were $\gamma = 0.9$ and $\epsilon = 0.9$. The capacity of the replay memory was 500, and the size of the minibatch was 32. The parameters used in the simulations are listed in Table 1.

In this study, weight factor ω and time interval τ are strategic parameters. Weight ω trades off between the load balance and the migration cost in the reward function (Equation (17)). Interval τ represents the frequency with which the controller makes migration decisions. Figures 2, 3, and 4

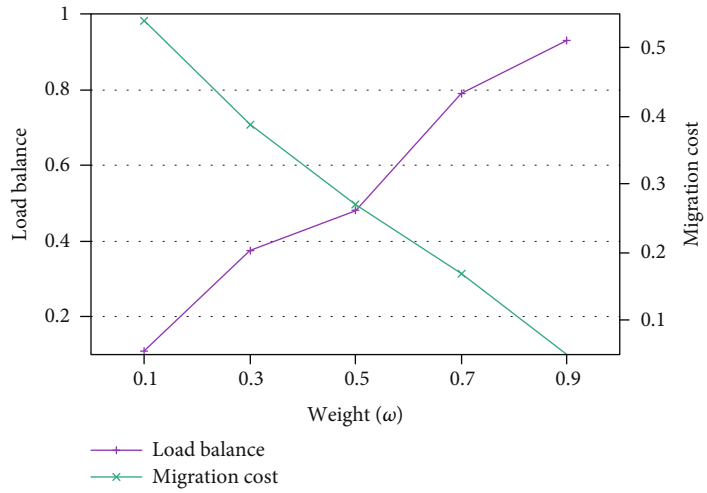

Figure 2: Effect of weight on load balancing and migration cost.

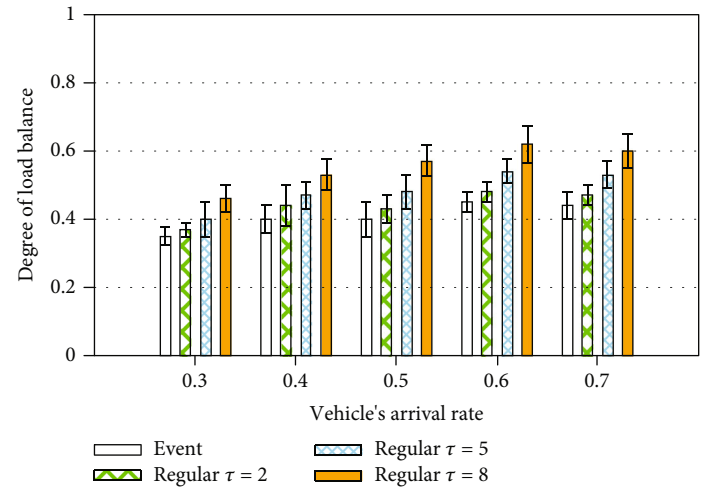

Figure 3: Degree of load balance for different intervals τ.

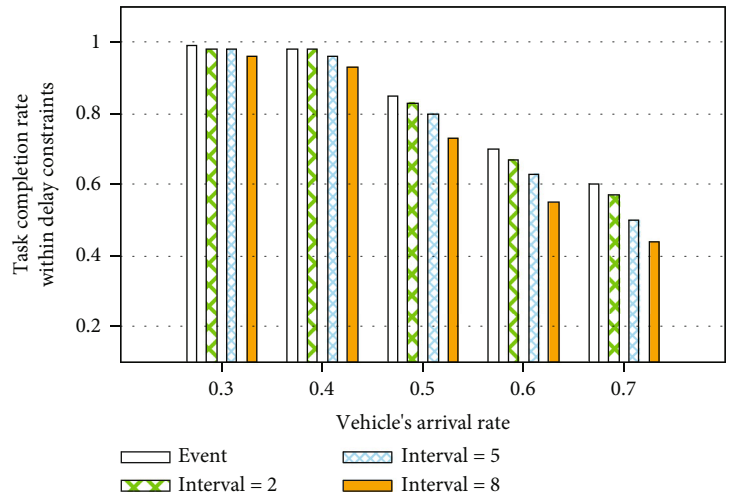

Figure 4: Task completion rate for different intervals τ.

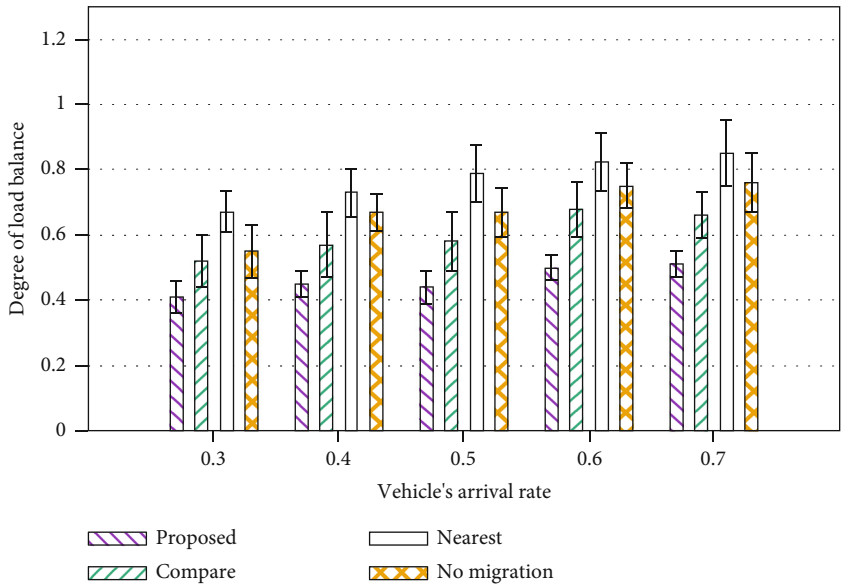

Figure 5: Comparison of the degree of load balance for different arrival rates.

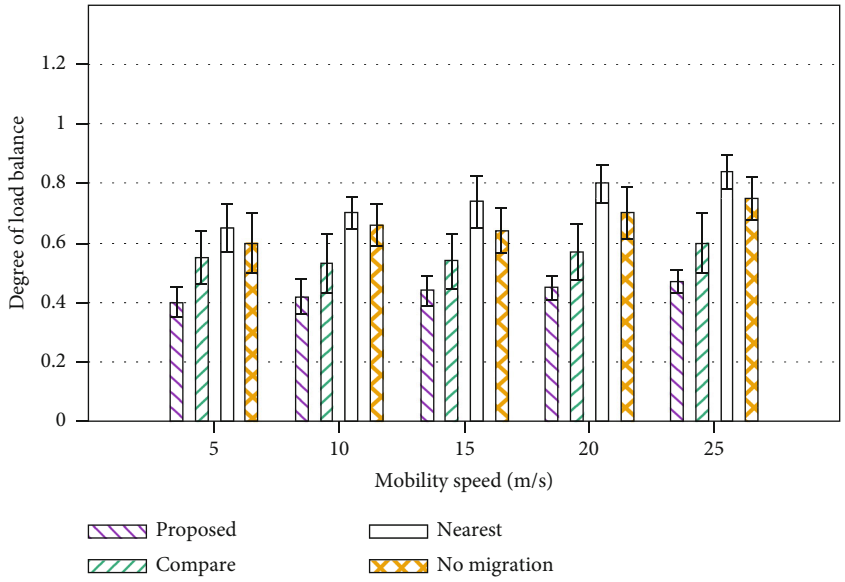

Figure 6: Comparison of the degree of load balance for different mobility speeds.

show the manner in which weights ω and intervals τ affect the system performance.

To find optimal weight ω for load balancing and migration cost, we evaluate the performance by varying weight ω when the arrival rate is 0.5 and the vehicle's mobility is 15 m/s (Figure 2). In the following simulations, we set weight ω to 0.5, which affected the load balancing and migration cost equally.

To analyze the effect of interval factor τ on the performance, we measured the degree of load balance and task completion rate within the delay constraints, as shown in Figures 3 and 4. In the figures, "Event" indicates that the controller makes migration decisions whenever a vehicle moves between adjacent VECSs. "Regular τ = 2, 5, or 8" indicate that the controller regularly makes migration decisions every τ = 2, 5, or 8. Compared to regular τ = 2, 5, and 8, Event shows approximately 6%, 14%, and 26% better load balancing performances, respectively. Event has higher completion rates within the delay constraints of approximately 3%, 8%, and 16%, respectively. However, as the number of vehicles increases, Event puts too much pressure on the controller. Therefore, we set the VECS controller to make migration decisions at regular intervals of τ = 5 considering the controller's burden, load balancing, and latency.

The performance of our proposed DQN-based migration algorithm is compared with three algorithms: *Compare* [7], *Nearest*, and *No Migration*. In the *Compare* and proposed algorithms, the optimization problem is formulated for reducing migration cost and load balancing. The *Compare* algorithm makes the migration decision with the response

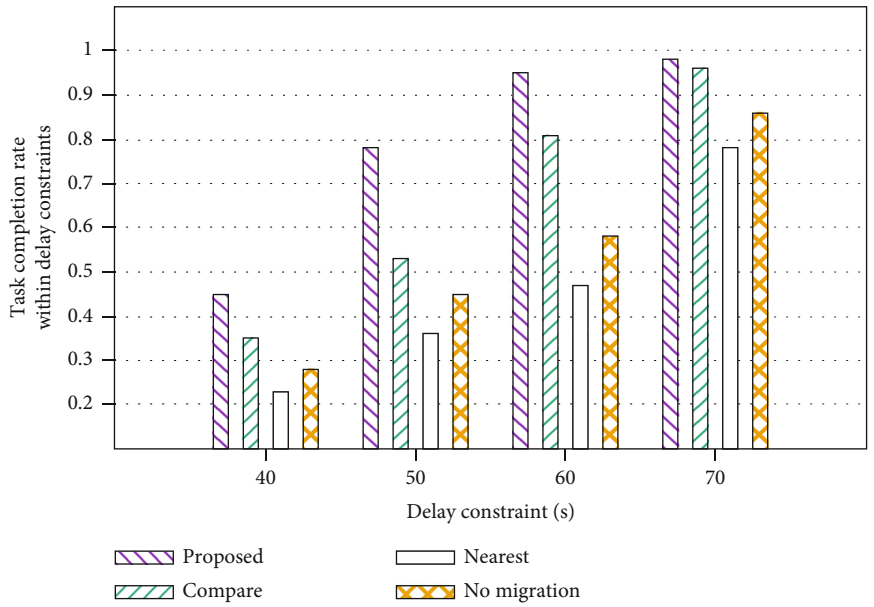

Figure 7: Comparison of the task completion rate for different delay constraints.

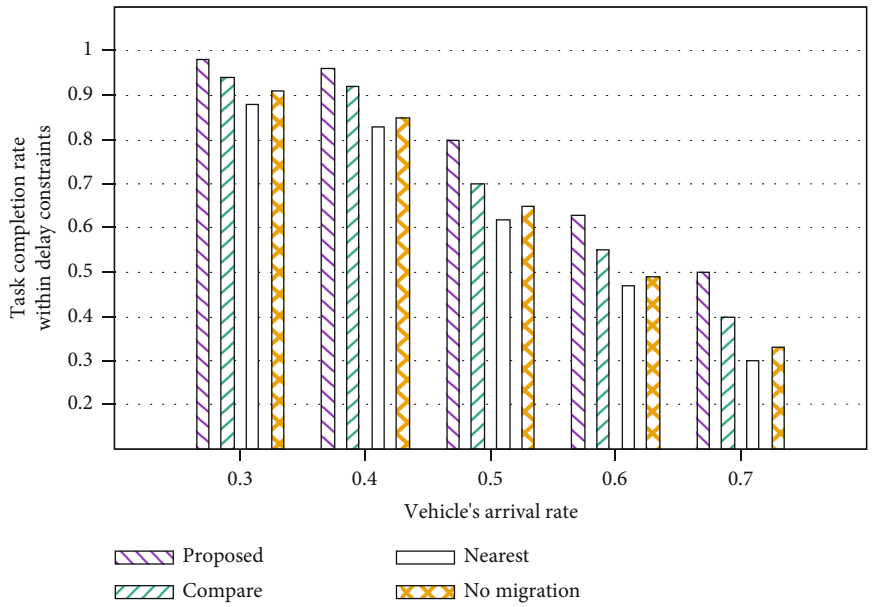

Figure 8: Comparison of the task completion rate for different arrival rates.

time modeled as an M/M/1 queuing system. The migration cost is also calculated by summing the response times of the current and target VECSs. In the *Nearest* algorithm, the computation task always migrates to the nearest VECS of the vehicle. In the *No Migration* algorithm, the computation task does not migrate to another VECS until the computation task is complete. Figures 5–9 show the degree of load balance and task completion rate within the delay constraints with varying arrival rates and mobility speeds.

Figure 5 depicts a comparison of the degree of load balance with respect to the arrival rate at a vehicle mobility speed of 20 m/s. The degree of load balance represents the deviation of the computing loads of the VECSs in Equation (9). The proposed algorithm performed approximately 23%, 39%, and 32% better than *Compare*, *Nearest*, and *No Migration*, respectively. The *Compare* algorithm focuses on the response time of VECSs, whereas the proposed algorithm focuses on the deviation of the VECS loads. The *Compare* algorithm tries to balance loads by lowering the average response time; the proposed algorithm attempts to balance loads by distributing computation tasks evenly among VECSs. Meanwhile, the *Nearest* algorithm selects the nearest VECS to a vehicle when it decides the target VECS. In this case, a small number of servers may experience workload

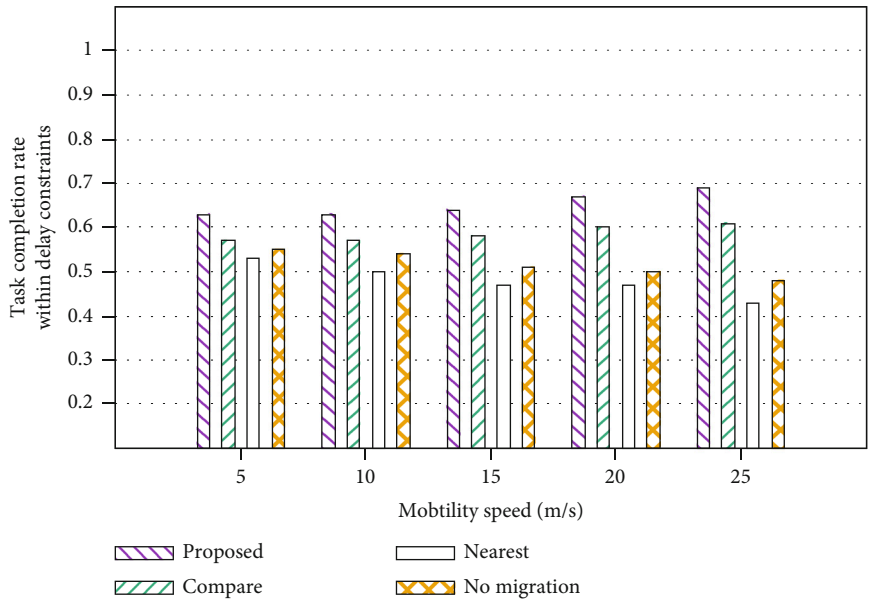

Figure 9: Comparison of the task completion rate for different mobility speeds.

concentration. Because the proposed algorithm considers the deviation of server loads when selecting the target VECS, congestion of certain VECSs can be avoided.

Figure 6 shows the degree of load balance with respect to the mobility speed of the vehicles at an arrival rate of 0.6. A higher mobility speed allows a vehicle to pass through the coverages of several VECSs within a certain period of time. From Figure 6, we can see that the degree of load balance for the proposed algorithm is almost constant regardless of the increase in speed. The proposed algorithm performs approximately 21%, 42%, and 35% better than *Compare*, *Nearest*, and *No Migration* algorithms, respectively. The *Compare* algorithm also shows a steady performance regardless of the speed. The *Nearest* and *No Migration* algorithms are sensitive to the increasing speeds.

In Figure 7, the rate of task completion within the delay constraint is shown for different values of delay constraint. The arrival rate and mobility speed are 0.6 and 20 m/s, respectively. 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 offloaded tasks. Figure 7 shows that the proposed algorithm has the highest task completion rate. Because overall load balancing is optimized, the computation time on the VECS is less than that when using other algorithms, and thus, the proposed algorithm achieves a relatively higher task completion rate.

Figures 8 and 9 show the performance in terms of the rate of task completion within the delay constraint with varying arrival rates and mobility speeds. In these simulations, we set the delay constraint to 50 s. Figure 8 shows that the proposed algorithm achieves approximately 14%, 25%, and 21% higher completion rates compared to the *Compare*, *Nearest*, and *No Migration* algorithms, respectively. As the arrival rate increases, regardless of how well the load balancing is performed, the latency increases because the number of computation requests handled by a VECS increases. In other words, as the workload of the VECS increases, the computing time of a task on the VECS increases, which degrades the performance. From the figure, we can see that despite the increase in the number of vehicles, the proposed algorithm is most optimized for load balancing, resulting in high task completion throughput. Because the workload is not biased to one VECS, the computation time and waiting time are lower.

Figure 9 shows that the proposed algorithm has better performance; the rate of task completion within delay constraints is approximately 13%, 28%, and 23% higher compared to those of *Compare*, *Nearest*, and *No Migration*, respectively. As the mobility speed increases, the proposed and *Compare* algorithms achieve a higher rate of completion. This is because the higher the mobility speed, the higher will be the migration decisions, which will result in a higher possibility of balancing loads among VECSs. However, the proposed algorithm balances the loads better than *Compare*; thus, the proposed algorithm achieves a higher rate of completion within the delay constraint.

5. Conclusions

In this study, we proposed a computation migration strategy using RL in VEC. To handle the high-dimensional state and action spaces, we proposed a deep RL-based migration strategy to optimize load balancing and migration cost while satisfying the delay constraints of computation tasks in a cooperative VECS group. We compared the proposed strategy with other strategies through simulations; the results showed that the proposed strategy achieves good load balancing among VECSs in terms of the number of computation requests.

This study considered a single agent in a multiuser environment. In future work, this can be expanded to multiagent

RL, each of which can be learned independently by the user [18]. In addition, various other factors can be considered, such as VECS group clustering [19] and load balancing with load prediction [20].

Data Availability

The data used to support the findings of this study have not been made available, because our funding agency has not agreed to this.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Stage 4 BK21 Project in Sookmyung Women's University of the National Research Foundation of Korea Grant. The present research has been conducted by the Excellent Researcher Support Project of Kwangwoon University in 2021.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[3] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23511–23528, 2018.

[4] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26652–26664, 2019.

[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.

[6] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[7] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Preemptive SDN load balancing with machine learning for delay sensitive applications," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 15947–15963, 2020.

[8] F. Sun, N. Cheng, S. Zhang, H. Zhou, L. Gui, and X. Shen, "Reinforcement learning based computation migration for vehicular cloud computing," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Abu Dhabi, 2018.

[9] D. Wang, X. Tian, H. Cui, and Z. Liu, "Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network," *China Communications*, vol. 17, no. 8, pp. 31–44, 2020.

[10] Z. Gao, Q. Jiao, K. Xiao, Q. Wang, Z. Mo, and Y. Yang, "Deep reinforcement learning based service migration strategy for edge computing," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 116–1165, San Francisco, 2019.

[11] Y. Cheng and X. Li, "A compute-intensive service migration strategy based on deep reinforcement learning algorithm," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pp. 1385–1388, Chongqing, China, 2020.

[12] H. Wang, H. Ke, G. Liu, and W. Sun, "Computation migration and resource allocation in heterogeneous vehicular networks: a deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 171140–171153, 2020.

[13] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: a deep reinforcement learning approach," *IEEE Network*, vol. 33, no. 3, pp. 26–33, 2019.

[14] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041–9052, 2020.

[15] X. Xu, Q. Wu, C. He, S. Wan, L. Qi, and H. Wang, "Edge computing-enabled resource provisioning for video surveillance in internet of vehicles," in *Smart City and Informatization: Communications in Computer and Information Science*, pp. 128–140, Springer, 2019.

[16] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD data set Epfl/mobility," Feb. 2009, http://crawdad.org/epfl/mobility.

[17] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12313–12325, 2018.

[18] C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 7, pp. 1603–1614, 2021.

[19] J. Lee, H. Ko, and S. Pack, "Trajectory-aware edge node clustering in vehicular edge clouds," in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–4, Las Vegas, 2019.

[20] J. Li, G. Luo, N. Cheng et al., "An end-to-end load balancer based on deep learning for vehicular network traffic control," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 953–966, 2019.