Graph-powered Reinforcement Learning for Intelligent Task Offloading in Vehicular Networks

Ihsan Ullah
Advanced Technology Research Center
KOREATECH
Cheonan, South Korea
ihsan@koreatech.ac.kr

Hyun-Kyo Lim
Quantum Network Research Center
KISTI, Daejeon,
South Korea
hk.lim@kisti.re.kr

Yeong-Jun Seok
Computer Science and Engineering
KOREATECH
Cheonan, South Korea
dsb04163@koreatech.ac.kr

Naeem Ul Islam
Computer Science and Engineering
Yuan Ze University
Taoyuan, Taiwan
naeem@saturn.yzu.edu.tw

Chang-Hun Ji
Future Convergence Engineering
KOREATECH
Cheonan, South Korea
koir5660@koreatech.ac.kr

Youn-Hee Han
Future Convergence Engineering
KOREATECH
Cheonan, South Korea
yhhan@koreatech.ac.kr

***Abstract*—In vehicular edge computing, task offloading optimization is crucial for balancing computational demands with minimizing delays and costs. However, the dynamic nature of the vehicular environment, including vehicle mobility, network topology, and available computing resources, poses significant challenges. This paper presents a task offloading scheme that enables vehicles to dynamically decide between local task execution and offloading to nearby vehicles, edge servers, or the cloud. Our objective is to optimize task offloading by minimizing cost and delay. We integrate graph convolutional networks with deep reinforcement learning to optimize task offloading decisions and achieve our goal. The Graph Convolutional Network is integrated with Deep Reinforcement Learning to enhance network representation and decision efficiency of agent. The optimization problem is formally formulated within the framework of a Markov Decision Process. Simulation results demonstrate the superiority of proposed scheme, which achieves cost-efficiency by maximizing resource utilization, minimizing costs, and optimizing task offloading while reducing task rejection.**

***Index Terms*—Deep Q-Network, Edge cloud computing, Resource allocation, Markov decision process (MDP), IoT, 5G network.**

I. INTRODUCTION

As the Internet of Thongs (IoT) and artificial intelligence (AI) advance, autonomous driving is a prominent topic in academic and engineering fields, especially in intelligent transportation. The Internet of Vehicles (IoV), an integral part of the Internet of Things, plays a crucial role in Intelligent Transport Systems by integrating vehicles, communication networks, and cloud computing [1]. IoV aims to enhance traffic safety, reduce congestion, and improve the overall user experience through Vehicle Ad Hoc Networks and various communication modes like Vehicle-to-Everything [2]. Nevertheless, the surge of data and complex computational tasks due to intelligent vehicles poses challenges such as limited network bandwidth, high latency, and security concerns. Vehicular edge computing, combining mobile edge computing and IoV, facilitates computing offloading and interaction between vehicle and roadside units via wireless access networks [3].

Task offloading is a key technique in vehicular edge-cloud computing (VECC), essential for optimizing resource utilization and minimizing latency [4]. Efficient offloading is crucial due to strict latency requirements in many vehicular applications. Research efforts have focused on developing task offloading algorithms for VECC to make optimal offloading decisions [5]–[8]. Edge cloud networks for vehicles play a critical role in VECC task offloading, offering reduced latency, improved resource utilization, and increased scalability. Challenges in deploying edge cloud networks for vehicles include deployment cost, resource management, security, and privacy.

This paper introduces a novel task offloading scheme for VECC networks. The proposed scheme, named GRLVTO (Graph-powered Reinforcement Learning for Vehicular Task Offloading). GRLVTO leveraging GCN layers in the DQN framework enhances feature extraction from the network graph, improving the deep Q-network (DQN) ability to identify optimal task offloading solutions. The optimization problem undergoes a formal formulation within the framework of a Markov decision process (MDP). Simulations show the proposed scheme outperforms the heuristic approach, achieving minimal cost, processing delay, and reduced task rejection.

The remainder of the paper is organized as follows: Related work is discussed in Section II. Section III provides the system model in detail. Section IV presents the proposed scheme. The performance evaluation is discussed in Section V. Section VI presents the conclusions.

II. RELATED WORK

Recently, there has been a surge in research within the realm of vehicular networks, focusing on leveraging the underutilized resources inherent in vehicular systems. In the work of Luo et al. [9], the objective was to minimize offloading delay. To achieve this, they proposed a multi-objective particle swarm optimization method that incorporates game theory analysis. This method takes into account various aspects, including communication protocols, offloading decision-making processes, and the efficient allocation of computing resources. In a related study, researchers in [10] advocated for a novel vehicle-end-edge cloud architecture designed to offload task computations. To determine the optimal decision in this offloading process, they employed an A3C-based offloading method, aiming to enhance the overall efficiency of resource allocation. Addressing the stringent low-latency requirements of vehicle-to-vehicle communication links and striving to improve the throughput of vehicle-to-infrastructure connections, Fu et al. [11] proposed a DRL-based algorithm. This algorithm stands out in its ability to intelligently allocate resources, contributing to more efficient and effective utilization. In the pursuit of maximizing the use of idle resources within vehicles, Wu et al. [12] introduced a hybrid task offloading strategy. This strategy involves a combination of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) modes to achieve the highest possible efficiency in resource utilization. Turning attention to resource allocation schemes within wireless networks, [13] proposed an intelligent DRL-based approach. This scheme dynamically allocates computational and network resources to minimize service time and achieve a balanced distribution of resources, contributing to enhanced overall network performance. Lai et al. [13] put forward a comprehensive three-tier vehicular network model, encompassing the cloud layer, the fog layer, and the network layer. Meanwhile, Feng et al. [14] introduced a hybrid cloud computing infrastructure within vehicular networks. In this infrastructure, tasks are intelligently offloaded either to neighboring vehicles or to Roadside Units (RSUs), depending on the specific requirements and conditions. Adding to the spectrum of offloading schemes, Li et al. [15] presented predictive combination-mode and load-aware Mobile Edge Computing (MEC) strategies. These strategies involve task offloading through either V2V relay transmission to the MEC server or V2I uploading, considering the dynamic nature of the vehicular environment and load conditions.

III. SYSTEM MODEL

In this section, we introduce the system model designed for the efficient task offloading in vehicular edge computing, incorporating a three-tier architecture (vehicle, edge (RSU), and cloud layers). Following sections provide a detailed exploration of task, communication, and computation models.

A. Task Modeling

Assume the system encompasses a set of tasks generated by vehicles v_n. A task $t_i = (z_i, \tau_i)$ is a tuple of two variables, where z_i and τ_i represent the input data size (measured in bytes) and the tolerated delay latency requirement, respectively. A variable $x_i \in [0, 1]$ serves as the computation offloading decision of the vehicle, a binary variable that indicates whether the task t_i is offloaded or not. Specifically, $x_i = 0$ if vehicle v_i decides to compute its task locally and $x_i = 1$ if the vehicle v_i decides to offload the task to a neighbor vehicle, an RSU, or the cloud. The offloading decision is denoted as $o_i \in [\mathrm{nv}_i, \mathrm{rsu}_i, \mathrm{cloud}_i]$, such as $o_i \in [1, 0, 0]$ offloading to a neighbor vehicle, $o_i \in [0, 1, 0]$ offloading to an RSU, and $o_i \in [0, 0, 1]$ offloading to the cloud. We assume that ζ represents the number of CPU cycle units for processing one byte of data; then, c_i indicates the total CPU cycles required to process the task t_i.

$$c_i = z_i \zeta. \tag{1}$$

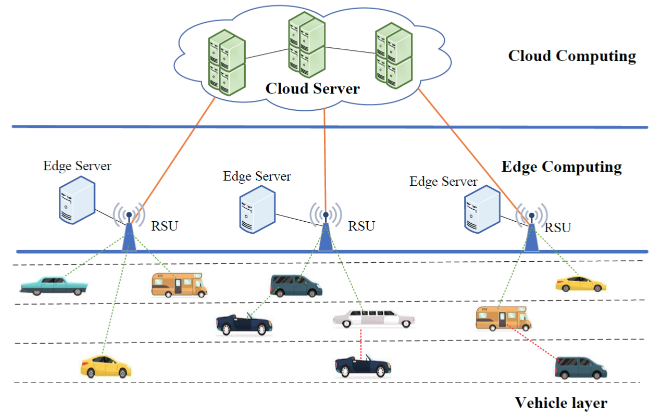

Fig. 1. Three-tier computing architecture comprising end devices, and edge and cloud layers

B. Communication Modeling

We consider a system that employs OFDMA as the multiple access scheme. OFDMA is utilized in each base station (BS) to mitigate intra-cell interference. The operational frequency band B_m (MHz) owned by a BS m is divided into k orthogonal sub-channels. The vehicle associated with the closest RSU can use an available sub-channel to offload a task. The binary variable $\rho_{(i,j)} \in \{0, 1\}$, where $\rho_{(i,j)} = 1$ indicates that the sub-channel between the vehicle v_i and the target for task offloading can be either nv_i, rsu_i, or cloud_i. The efficiency of the sub-channel, denoted as $\eta^k_{(i,j)}$, can be approximated using Shannon's formula as follows:

$$\eta^k_{(i,j)} = \log_2(1 + \varphi^k_{(i,j)}), \tag{2}$$

where $\varphi^k_{(i,j)}$ represents the signal-to-noise ratio of the k-th sub-channel used by vehicle v_i, and it can be expressed as follows:

$$\varphi^k_{(i,j)} = \frac{e_{(i,j)} h^k_{(i,j)}}{\sigma^2}. \tag{3}$$

Here, $e_{(i,j)}$ denotes the transmission power between vehicle v_i, $h^k_{(i,j)}$ denotes the channel gain (dB), and σ^2 represents the power of the additive white Gaussian noise of a sub-channel. The achievable transmission rate $\mu^k_{(i,j)}$ is given by:

$$\mu^k_{(i,j)} = b^k_{(i,j)} \eta^k_{(i,j)}, \tag{4}$$

where $b^k_{(i,j)}$ represents the bandwidth assigned to the sub-channel.

C. Computational Framework

1) Local Processing: When the vehicle v_i decides to process the local task t_i by itself, the processing delay can be calculated as:

$$T_l^{exec} = \frac{c_i}{r_{v_i}}, \tag{5}$$

where c_i is the computational demand of task t_i and r_{v_i} is the computational capacity of vehicle v_i.

2) Neighbor Vehicle Processing: When the agent decides to offload the task t_i to the neighbor vehicles, the processing delay can be calculated as:

$$T_{nv}^{exec} = \frac{c_i}{r_{nv_i}}, \tag{6}$$

where c_i is the computational demand of task t_i and r_{nv_i} is the computational capacity of the neighbor vehicle.

3) Edge Processing: Consider a set of the edge servers represented by $\mathcal{S} = \{s_1, s_2, ..., s_P\}$. The time required for task t_i at server s_p is:

$$T_{edge}^{exec} = \frac{c_i}{r_p}, \tag{7}$$

where c_i is the computational demand of task t_i and r_p is the computational capacity of the edge server where the task is offloaded. The resource utilization $U_{\mathcal{S}}(t)$ of edge servers can be calculated as:

$$U_{\mathcal{S}}(t) = \frac{\sum_{p=1}^{P} r_p(t)}{R_{edge}}, \tag{8}$$

where R_{edge} denotes the computation capacity of all ege servers.

4) Cloud Processing: Consider the cloud server set $\mathcal{M} = \{m_1, m_2, ..., m_Q\}$. The computational capability of server m_q is denoted by r_q. The time taken for task t_i at server m_q is:

$$T_{cloud}^{exec} = \frac{c_i}{r_q}, \tag{9}$$

where c_i is the computational demand of task t_i and r_q is the computational capacity of the cloud server where the task is offloaded. The resource utilization $U_{\mathcal{C}}(t)$ for cloud servers is:

$$U_{\mathcal{C}}(t) = \frac{\sum_{q=1}^{Q} r_q(t)}{R_{cloud}}. \tag{10}$$

D. Delay Model in Computation Offloading

The offloading process encompasses three major delay types: transmission, propagation, and processing.

1) Transmission Delay: Data transmission occurs bidirectionally: from the vehicle to the edge/cloud server (size z_i) and back to the vehicle (size y_i). Transmission delays to the neighbor vehicle T_{nv}^{trans}, edge server, T_{edge}^{trans}, and cloud T_{cloud}^{trans}, are given by:

$$T_{nv}^{trans} = \frac{z_i}{\mu_{(v_i,nv_i)}^k} + \frac{y_i}{\mu_{(v_i,nv_i)}^k}, \tag{11}$$

$$T_{edge}^{trans} = \frac{z_i}{\mu_{(v_i,s_p)}^k} + \frac{y_i}{\mu_{(v_i,s_p)}^k}, \tag{12}$$

$$T_{cloud}^{trans} = T_{edge}^{trans} + \frac{z_i}{\mu_{(v_i,m_q)}^k} + \frac{y_i}{\mu_{(v_i,m_q)}^k}. \tag{13}$$

2) Propagation Delay: Propagation delays are assumed constant. Specifically, $T_{nv}^{prop} = T_{edge}^{prop} = 5$ ms for the neighbor vehicle and edge server, and $T_{cloud}^{prop} = 50$ ms for the cloud server. This assumption facilitates simpler analysis, although real-world propagation may vary based on resource positioning.

3) Processing Delay: Processing delays for task t_i at neighbor vehicle, edge server, and cloud server, can be denotes T_{nv}^{exec}, T_{edge}^{exec}, and T_{cloud}^{exec}, respectively.

4) Overall Task Completion Time: The total time for task completion on the edge, rtt_n^e, and cloud, rtt_n^c, sums up the aforementioned delays:

$$rtt_{v_i}^{nv} = T_{nv}^{trans} + T_{nv}^{prop} + T_{nv}^{exec}, \tag{14}$$

$$rtt_{v_i}^{edge} = T_{edge}^{trans} + T_{edge}^{prop} + T_{edge}^{exec}, \tag{15}$$

$$rtt_{v_i}^{cloud} = T_{cloud}^{trans} + T_{cloud}^{prop} + T_{cloud}^{exec}. \tag{16}$$

The offloading decision is denoted as $o_i \in \{\text{nv}_i, \text{rsu}_i, \text{cloud}_i\}$ where if we put $o_i = [1,0,0], [0,1,0]$ and $[0,0,1]$ in the following equation, the decision is neighbor-vehicle, edge (rsu), and cloud, respectively. The round-trip time equation is given by:

$$\begin{aligned} rtt_{v_i} = & o_i rtt_{v_i}^{nv} + o_i(1-o_i) rtt_{v_i}^{edge} \\ & + (1-o_i)(1-o_i) rtt_{v_i}^{cloud}. \end{aligned} \tag{17}$$

IV. Graph-powered Reinforcement Learning Scheme

Our integrated model incorporates a GCN layer within the DQN framework, facilitating a more profound comprehension of network dynamics and enabling dynamic allocation of tasks. The network is represented as a directed graph, where nodes correspond to vehicles, edge servers, and cloud servers. GCN processes graph data by considering node features and connections, creating a concise representation of the state called embedding. The embedding layer is responsible for creating low-dimensional representations of the nodes in the graph by using GCN. In our integrated model, the DQN relies on neural networks to estimate Q-values.

$$Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a) \tag{18}$$

where s_t denotes the current state, a_t represents the current action, r_t is the reward received after taking action a_t in state s_t, and γ is the discount factor.

The Q-network benefits from the insights provided by the GCN layer and the Q-learning update is employed to refine Q-values based on the integration of GCN insights: It first use the GCN to learn a representation of the state of the environment and then uses the DQN to learn a policy for taking actions in the environment. it can be mathematically formulated as follows:

$$h_s = GCN(s), \tag{19}$$

$$Q(s,a) = \sigma(W * h_s + h_a), \tag{20}$$

where h_s is the representation of the state of the environment also called embedding, h_a is the representation of the action,

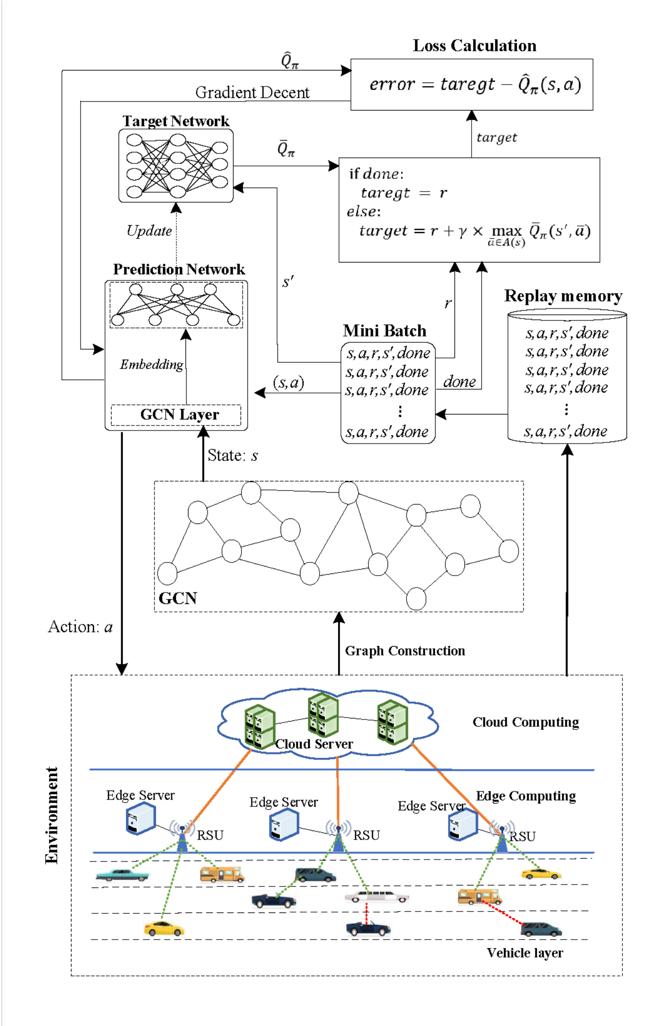

Fig. 2. Three-tier computing architecture comprising vehicles, and edge (RSU), and cloud layers

GCN is the graph convolution operation, and W is the weight matrix of the DQN.

A. Markov Decision Process

A MDP models sequential decision-making problems, where an agent seeks to maximize reward through decisions. It comprises elements like agent, state, action, policy, and reward. We formulate task offloading and resource optimization as an MDP to determine the optimal policy π^*. In GRLVTO, the agent observes state s_t, selects an action a_t via a deterministic policy for computing server selection, and receives immediate reward r_t. The agent uses action-value function $Q(s_t, a_t)$ to update its policy, aiming to maximize long-term rewards through optimal resource allocation.

1) State Space: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent the graph, where $\mathcal{V}$ is the set of nodes (entities) and $\mathcal{E}$ is the set of edges (relationships) in the graph. We define the state space as s_t, where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is the node feature matrix, with N nodes and D features. $\mathbf{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix, representing connections between nodes. The combined state space can be represented as follows:

$$s_t = (\mathbf{X}_{\text{Tasks}}, \mathbf{X}_{\text{Vehicles}}, \mathbf{X}_{\text{RSU}}, \mathbf{X}_{\text{Cloud}}, \mathbf{A}). \qquad (21)$$

- $\mathbf{X}_{\text{Tasks}} \in \mathbb{R}^{N_{\text{Tasks}} \times D_{\text{Tasks}}}$ represent the feature matrix for tasks nodes, where N_{Tasks} is the number of tasks and D_{Tasks} is the number of features.
- $\mathbf{X}_{\text{Vehicles}} \in \mathbb{R}^{N_{\text{Vehicles}} \times D_{\text{Vehicles}}}$ represent the feature matrix for vehicle nodes, where N_{Vehicles} is the number of vehicles and D_{Vehicles} is the number of features.
- $\mathbf{X}_{\text{RSU}} \in \mathbb{R}^{N_{\text{RSU}} \times D_{\text{RSU}}}$ represent the feature matrix for RSU nodes, where N_{RSU} is the number of RSUs and D_{RSU} is the number of features.
- $\mathbf{X}_{\text{Cloud}} \in \mathbb{R}^{N_{\text{Cloud}} \times D_{\text{Cloud}}}$ represent the features matrix for cloud nodes, where N_{Cloud} is the number of cloud nodes and D_{Cloud} is the number of features.
- $\mathbf{A}$ is the adjacency matrix representing connections between nodes in the graph.

2) Action: We can define an action space $\mathcal{A}$ that consists of actions representing choices among vehicle nodes, RSU nodes, or cloud nodes based on the state embedding obtained from the GCN. In each time step t, the DQN agent selects an action to offload the task t_i and allocates resources to the task for execution within the task deadline. In each time step, the DQN agent makes a decision according to the task offloading policy derived from the current state embedding obtained through the GCN, and then it receives the reward from the environment at time step $t+1$.

3) Reward: In our optimization framework, the reward function R_t captures the primary objective of maximizing resource utilization and minimizing cost while adhering to a specified delay. The cost ξ values, set at 0 for local, 1 for a neighboring vehicle, 2 for the edge, and 3 for the cloud, can be adjusted according to environmental and system configurations The reward is computed differently based on the success or failure of the task:

$$R_t = \begin{cases} 10 - \max[0, (rtt_{v_i} - \tau_i)] + \xi & \text{if successful} \\ -1.0 & \text{if unsuccessful} \end{cases} \qquad (22)$$

The term $\max[0, (rtt_{v_i} - \tau_i)]$ represents the extent to which the round-trip time (rtt_{v_i}) exceeds the given delay threshold (τ_i). This value is subtracted from 10, ensuring that a positive contribution is added to the reward when the delay constraint is satisfied. The cost term (cost) is also factored in, contributing to the overall optimization goal. The task unsuccessful case, a fixed penalty of -1.0 is applied.

V. Performance Evaluation

In this section, we use a computer simulation to evaluate our scheme's performance in terms of resource utilization, task acceptance, rejection, and cost. Our simulation runs on hardware with an i9-10900K CPU, 64GB RAM, an RTX 3090 GPU, and uses Linux Ubuntu 20.04.02 LTS with Python 3.8 and PyTorch 1.9.0. We compare our scheme with two heuristics: Heuristic1 (prioritizing high-resource-demand tasks) and Heuristic2 (pbest match server).

Algorithm 1 Training Stage of the DQN algorithm with GCN

Input: Complete information on the vehicular network, including tasks and the edge-cloud network
Output: Selection of a server at the edge or cloud for offloading tasks, considering resource constraints and task requirements.
Initialize two neural networks as Q-networks Q_π and $\overline{Q}_\pi$ with random weights (and biases) or parameters θ and $\overline{\theta}$.
Initialize GCN with parameters ϕ.
Initialize replay memory M.
for episode $= 1$ to e **do**
  Initialize the first state s_0.
  for $t = 1$ to T **do**
    Gather the state s_t from the environment contains, current task nodes and link (graph).
    Apply GCN to obtain the graph embedding h_t.
    if random value $< \epsilon$ **then**
      Select a random action a_t.
    else
      Select $a_t = \arg\max_a Q(h_t, a|\theta)$.
    end if
    Execute action a_t, receive reward r_t, and obtain the next state s_{t+1}.
    Store (s_t, a_t, r_t, s_{t+1}) in M.
  end for
  Get a mini-batch of size b: (s_i, a_i, r_i, s_{i+1}) from M.
  for $i = 1$ to b **do**
    if s_{i+1} is terminal **then**
      $y_i = r_i$.
    else
      Apply GCN to obtain the graph embedding $\overline{h}_{i+1}$ for the next state.
      $y_i = r_i + \gamma \cdot \max_a \overline{Q}_\pi(\overline{h}_{i+1}, a|\overline{\theta})$.
    end if
    $q_i = Q(h_i, a_i|\theta)$.
  end for
  $\theta = \theta - \alpha\Delta_\theta \sum_{i=1}^{b}(q_i - y_i)^2/b$ ▷ Gradient descent
  if episode is a multiple of K **then**
    Copy θ to $\overline{\theta}$.
  end if
end for

In Figure 3, we compare three schemes based on their task rejection ratios. As the number of tasks increases, all three schemes experience higher rejection rates. Nevertheless, the efficiency of the GRLVTO scheme in server selection through agent learning enables it to accept more tasks, conserving resources for future use. This heightened acceptance rate enhances resource utilization and reduces system idle time. Consequently, GRLVTO outperforms other algorithms, underscoring the success of integrating GCN with DQN in improving the vehicle network's performance in the edge-cloud system

Figure 4 depicts a cost analysis of three schemes. The GRLVTO scheme exhibits a minimal cost increase compared to the other two schemes, highlighting its cost-effectiveness. Notably, GRLVTO maintains a significantly lower task rejection rate compared to heuristic1 and heuristic2. Despite costs, GRLVTO's intelligent matching of tasks to servers based on resource requirements, latency, and cost contributes to efficient cost management

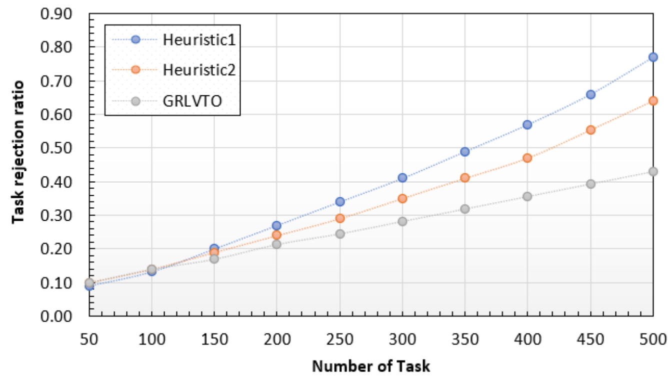

Fig. 3. The comparison of task rejection rates for three different schemes concerning various tasks

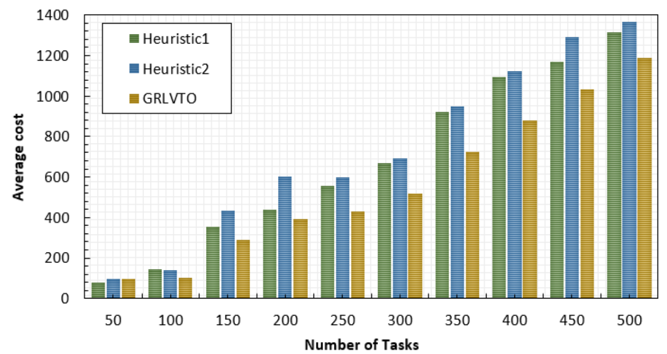

Fig. 4. The comparison of cost for three different schemes concerning various tasks

Figure 5 illustrates the comparison of average resource utilization among the proposed scheme and two other schemes, heuristic1 and heuristic2. GRLVTO consistently exhibits higher resource utilization than the other two algorithms, confirming the findings in Figure 3. Task rejection rates directly impact resource utilization, with lower rejections leading to higher efficiency. Our scheme's ability to select optimal servers based on task requirements enhances system efficiency through intelligent resource allocation. GRLVTO outperforms heuristic algorithms due to the effective feature extraction capabilities of GCN from the entire network and its integration with DQN. Moreover, the integration of GCN effectively manages resource demands across vehicle, edge, and cloud networks, thereby improving the performance of the vehicle edge-cloud system

VI. Conclusion

In the context of vehicle edge cloud network environments, the optimization challenge of task offloading and resource

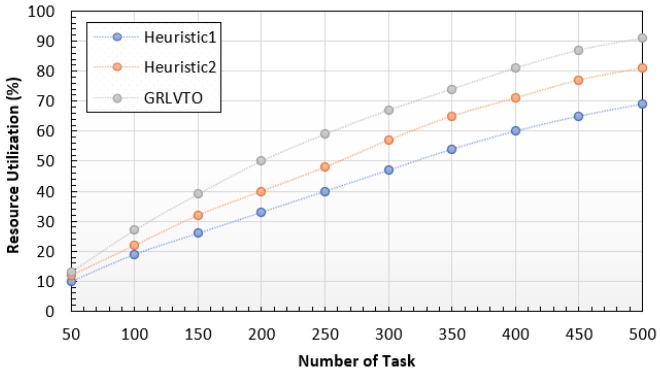

Fig. 5. The comparison of respource utilization for three different schemes concerning various tasks

allocation stands as a fundamental and formidable issue. To address this, we formulated the problem as a MDP and employed a powerful combination of GCN with the DQN algorithm to seek an optimal solution. The GRLVTO model intelligently manages vehicular tasks, optimizing performance by efficiently handling computation in edge or cloud servers. It shows improved cost, resource utilization, and reduced task rejection rates, as validated by simulations.

The GRLVTO scheme will leverage advanced machine learning and AI techniques to optimize its potential. A thorough analysis of the vehicular network will enhance realism for managing diverse IoT devices. This forward-looking approach aims to continuously improve efficiency and adaptability to evolving challenges in vehicular edge cloud networks.

Acknowledgments

This research was supported by two Basic Science Research Programs through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2018R1A6A1A03025526 and NRF-2023R1A2C1003143).

References

[1] C. R. Storck and F. Duarte-Figueiredo, "A survey of 5g technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by internet of vehicles," *IEEE access*, vol. 8, pp. 117 593–117 614, 2020.

[2] X. Kong, Y. Wu, H. Wang, and F. Xia, "Edge computing for internet of everything: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 472–23 485, 2022.

[3] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6g networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4584–4595, 2021.

[4] Y. Sun, Z. Wu, K. Meng, and Y. Zheng, "Vehicular task offloading and job scheduling method based on cloud-edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[5] 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, 2019.

[6] T. H. Binh, H. Vo, B. M. Nguyen, H. T. T. Binh *et al.*, "Reinforcement learning for optimizing delay-sensitive task offloading in vehicular edge-cloud computing," *IEEE Internet of Things Journal*, 2023.

[7] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, A. Aziz, and Z. Han, "Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8315–8338, 2022.

[8] Y. Qi, Y. Zhou, Y.-F. Liu, L. Liu, and Z. Pan, "Traffic-aware task offloading based on convergence of communication and sensing in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17 762–17 777, 2021.

[9] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Transactions on Services Computing*, vol. 15, no. 5, pp. 2897–2909, 2021.

[10] C. Chen, H. Li, H. Li, R. Fu, Y. Liu, and S. Wan, "Efficiency and fairness oriented dynamic task offloading in internet of vehicles," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1481–1493, 2022.

[11] J. Fu, X. Qin, Y. Huang, L. Tang, and Y. Liu, "Deep reinforcement learning-based resource allocation for cellular vehicular network mode 3 with underlay approach," *Sensors*, vol. 22, no. 5, p. 1874, 2022.

[12] C. Wu, Z. Huang, and Y. Zou, "Delay constrained hybrid task offloading of internet of vehicle: A deep reinforcement learning method," *IEEE Access*, vol. 10, pp. 102 778–102 788, 2022.

[13] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "Drl-based resource allocation for computation offloading in iov networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 8027–8038, 2022.

[14] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling," *IEEE vehicular technology magazine*, vol. 14, no. 1, pp. 28–36, 2018.

[15] L. Li, H. Zhou, S. X. Xiong, J. Yang, and Y. Mao, "Compound model of task arrivals and load-aware offloading for vehicular mobile edge computing networks," *IEEE Access*, vol. 7, pp. 26 631–26 640, 2019.