Journal of Intelligent Systems and Internet of Things
Vol. 17, No. 01, PP. 75-88, 2025
DOI: https://doi.org/10.54216/JISIoT.170106

Minimizing Time Overhead in VANET Task Offloading: A Novel Preparatory-Based Edge-Cloud Collaborative Model

K. Rajeswari[1,*], B. Arun Kumar[2]

[1]Research scholar, Department of Computer Science and Engineering, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

[2]Professor and Head, Department of Artificial Intelligence and Data Science, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

Emails: rajeswarikamalk@gmail.com; arunkumar.oct06@gmail.com

Abstract

In vehicle ad hoc networks (VANETs), vehicles often need to perform complex computing tasks that may exceed their processing capabilities within the required period to provide enhanced services. A common approach to improving service performance is to offload tasks to roadside units (RSUs). However, RSUs might not always have sufficient resources to manage all task assignments effectively. Given the increasing processing power of modern vehicles, task delegation to other vehicles presents a viable alternative to relying solely on RSUs. To achieve this, we first introduce a probabilistic approach that relaxes discrete actions, such as cloud server selection, into a continuous space. We then implement a Supportive Multi-Agent Deep Reinforcement Learning (SMADRL) technique that minimizes total system costs, including Vehicle device energy consumption and cloud server rental charges, by utilizing a centralized training and distributed execution approach. In this framework, each Vehicle device operates as an independent agent, learning efficient decentralized policies that reduce computing pressure on the devices. Experimental results show that the proposed SMADRL framework effectively learns dynamic offloading policies for each Vehicle device and notably outperforms four state-of-the-art DRL-based agents and two heuristic frameworks, resulting in reduce overall system costs.

Received: January 31, 2025 Revised: March 08, 2025 Accepted: April 12, 2025

Keywords: Task Offloading; Collaborative Model; Multi-agent Deep Reinforcement Learning (MADRL)

1. Introduction

The rapid advancement of vehicular networks, mostly Vehicular Ad hoc Networks (VANETs), has fundamentally transformed how vehicles interact and communicate within smart transportation systems. VANETs facilitate the exchange of information among vehicles and between vehicles and infrastructure components, enabling enhanced safety, improved navigation, and a superior driving experience. These networks support a variety of sophisticated applications, including collision avoidance systems, real-time traffic monitoring, and autonomous driving capabilities [1-3]. However, the efficient operation of these advanced applications demands significant computational resources and low-latency processing to deliver timely insights and services. To meet these demands, task offloading to edge-cloud environments has emerged as a highly effective solution. In this paradigm, edge computing involves placing computational resources closer to the vehicles at the network edge, while cloud computing provides centralized processing power and storage capabilities [4-5]. The combination of edge and cloud computing in a collaborative model aims to optimize task execution by leveraging the respective strengths of each paradigm, thus addressing the performance and latency requirements of modern vehicular applications. Despite the promising advantages of integrating edge and cloud computing, task offloading in VANETs with an edge-cloud environment presents several significant challenges.

One of the most critical issues is the inherent trade-off between latency and resource cost. Edge servers, which are positioned closer to vehicles, offer substantial reductions in communication latency compared to cloud servers that are located further away [6-7].

This proximity is crucial for applications that require real-time data processing, such as collision avoidance systems where even minimal delays can have severe consequences. However, edge servers are generally constrained in terms of computational power and are more costly to operate on a per-unit basis than cloud servers are. Conversely, cloud servers provide greater computational capabilities and are more cost-effective in terms of operational expenses. Yet, the increased distance between cloud servers and vehicles introduces higher latency, which can compromise the performance of time-sensitive applications. This trade-off necessitates the development of an efficient offloading strategy that balances latency reduction with cost management [8-10] [41]. An optimal strategy must align with both performance requirements and budget constraints to ensure that the benefits of edge-cloud computing are fully realized. Another critical challenge in edge-cloud collaborative task offloading is ensuring task reliability in the face of the dynamic and distributed nature of VANETs. The edge-cloud network environment is susceptible to unpredictable failures and connectivity issues that can disrupt task processing and result in data loss. To enhance reliability, redundancy techniques are often employed, such as deploying multiple backup instances for each task. These techniques aim to prevent interruptions and ensure that tasks continue to be processed even in the event of a failure [11-12].

However, implementing redundancy increases the demand for computational resources and can lead to higher operational costs. Therefore, developing a robust task-offloading model that guarantees high reliability while managing resource constraints is essential for maintaining service continuity and meeting the reliability requirements of vehicular applications [13-15]. This requires a careful balance between ensuring redundancy and controlling costs to avoid excessive resource utilization. The heterogeneous nature of edge-cloud environments further complicates the task offloading process. In VANETs, tasks have varying resource requirements, and the edge and cloud servers differ in their computational capabilities and resource capacities [16-17]. This variability means that offloading tasks to different computing architectures can lead to significant differences in response times and resource costs. The decision-making process becomes increasingly complex, as it must account for the diverse requirements of each task and the capabilities of the available resources. Finding an optimal solution that balances time delay, resource cost, and reliability amidst this complexity is a formidable challenge. The presence of heterogeneous computing environments necessitates the development of advanced methods, which can efficiently handle the diverse and dynamic nature of edge-cloud collaborative systems [18-20]. To address these challenges, researchers are actively developing and refining edge-cloud collaborative task offloading models. These models aim to harness the combined strengths of edge and cloud computing to optimize task execution in VANETs.

The primary focus of these models is on creating algorithms that can dynamically allocate tasks to either edge or cloud resources based on real-time conditions, such as network load, resource availability, and specific task requirements. Various advanced techniques, including heuristic methods, meta-heuristic approaches, and ML technique, are being explored to enhance the efficiency of task offloading decisions. Heuristic methods offer quick solutions but May only achieve local optima, while meta-heuristic and machine-learning methods can provide high-quality solutions but may be time-consuming and challenging to adapt to real-time requirements. Additionally, ML techniques may need retraining with new data samples, which can affect their performance and increase the complexity of their implementation. Despite the progress in research, many existing approaches to task offloading in edge-cloud environments tend to oversimplify the problem by focusing on only one or two optimization objectives, such as latency or cost, while neglecting other critical factors such as reliability and resource constraints. This gap highlights the need for comprehensive and adaptive offloading strategies that can address the full spectrum of challenges associated with edge-cloud environments. Ongoing research is crucial for developing robust offloading algorithms that can effectively handle the complexities of edge-cloud systems and meet the evolving demands of modern vehicular applications. By advancing the field of edge-cloud collaborative task offloading, researchers aim to improve the performance and reliability of VANETs, ultimately enhancing the efficiency and safety of smart transportation systems. Here are four key contributions from the proposed in the context of VANETs:

- ✓ We develop a framework for compute offloading across multiple vehicle interacting with various cloud servers, considering the varying computation capacity, channel gains, and task arrival times of these servers. The dynamic offloading of computation challenge is intended to reduce overall energy usage and rental costs of the system by optimizing the collaboration between cloud servers and local computation, including offloading ratios and selection capabilities.
- ✓ By introducing a probabilistic method, we transform the discrete decision-making process, such as selection of cloud servers, through a constant set, resulting in a continuous-discrete hybrid decision approach. To stabilize training and reduce computational load on devices, we then develop a distinct CMADRL model in which every Vehicle device functions as an agent. This means that we train a locally observable policy function for every vehicle using global state information gathered at an edge server.

The manuscript is arranged as follows: The second section presents an in-depth assessment of pertinent literature, highlighting the field's significant issues. The third section provides a full study of the proposed SMADRL model, including mathematical, visual explanations and theoretical. The fourth section discusses the experimental results in depth, including the entire experimental

established mathematical validations, and illustrations. Finally, the final section summarizes the research and emphasizes its significance.

2. Related Works

This paper explores the utilization of parked vehicles as edge computing nodes to facilitate task offloading in urban Vehicular Ad Hoc Networks (VANETs) [21]. By leveraging the idle computational resources of parked vehicles, the proposed scheme aims to reduce latency and enhance the overall network efficiency. The study demonstrates significant improvements in task processing times and resource utilization compared to traditional offloading methods. Author in [22] presents a comprehensive framework that simultaneously addresses task offloading, resource allocation, and security in UAV-assisted VANETs integrated with Mobile Edge Computing (MEC). By optimizing these three aspects jointly, the proposed model ensures efficient task execution while maintaining robust security protocols against potential threats. Experimental results indicate enhanced performance and security resilience compared to existing solutions. Author in [23] introduces a privacy-preserving task offloading mechanism using multiagent deep reinforcement learning within VANET environments. The approach ensures that sensitive data remains protected during the offloading process while optimizing computational resource distribution among vehicles. Simulation results highlight the effectiveness of the proposed method in maintaining privacy without compromising performance. Author in [24] investigates intelligent task offloading strategies in fog computing-enabled vehicular networks, aiming to optimize latency and resource allocation. By utilizing machine-learning algorithms, the proposed system dynamically decides the most efficient offloading targets based on current network conditions and vehicle mobility patterns. The results demonstrate improved response times and resource management compared to traditional fog computing approaches. Author in [25] proposes a distributed task-offloading scheme tailored for Vehicular Edge Computing (VEC) networks, focusing on minimizing latency and balancing computational loads across edge nodes. The scheme leverages cooperative algorithms to enable vehicles to make autonomous offloading decisions based on real-time network metrics. Performance evaluations show that the distributed approach outperforms centralized models in scalability and responsiveness.

The OPTOS framework introduces an online pre-filtering mechanism for task offloading in VANETs, aiming to enhance the efficiency and reliability of offloading decisions [26]. By pre-filtering tasks based on predefined criteria such as urgency and resource requirements, OPTOS reduces unnecessary offloading attempts and optimizes network resource usage. Experimental analysis confirms that OPTOS significantly improves task success rates and network throughput. Author in [27] presents an adaptive computation offloading strategy that incorporates task scheduling to minimize the need for task reallocation in VANETs. The approach dynamically adjusts offloading decisions based on real-time network conditions and vehicle mobility, ensuring optimal resource utilization and reduced task migration. Simulation results indicate that the proposed method effectively lowers task reallocation rates and enhances overall network performance. The MCLA framework addresses task offloading in heterogeneous Vehicular Edge Computing Networks (VECNs) utilizing 5G New Radio Vehicle-to-Everything (5G-NR-V2X) technology [28]. By integrating multiple computational and communication resources, MCLA optimizes task distribution across diverse network components to achieve high efficiency and low latency. Performance evaluations demonstrate that MCLA significantly outperforms existing frameworks in terms of task processing speed and resource allocation fairness. Author in [29] explores an opportunistic task assignment mechanism leveraging vehicle mobility for Internet of Things (Vehicle) task offloading. The proposed method dynamically assigns tasks to vehicles based on their current availability and proximity to edge computing resources, enhancing the scalability and flexibility of Vehicle systems. Results show that the opportunistic approach improves task completion rates and reduces overall latency compared to static assignment methods. Author in [30] introduces a joint optimization framework for task offloading and resource allocation in Vehicular Edge Computing (VEC) that incorporates differential privacy to protect sensitive data. By balancing computational efficiency with privacy guarantees, the proposed model ensures secure and efficient task processing in VANETs. Experimental outcomes reveal that the framework effectively maintains data privacy while achieving competitive performance metrics in task offloading and resource utilization.

3. Methodology

This section explores a multi-cloud computing system comprising $\mathrm{Vec_N}$ vehicles, $\mathrm{Clo_M}$ cloud servers, a base station (BS) with number of road side unit (RSU) $\mathrm{RSU_P}$, in the edge layer. Each $\mathrm{Vec_N}$ connects wirelessly to the $\mathrm{RSU_P}$, while the cloud servers are linked to the $\mathrm{RSU_P}$ via a wired connection. In edge layer, located near the BS, assists with centralized training and is discussed in the Multi-agent DRL framework section. Figure 1 shows that the $\mathrm{Clo_M}$ provide offloading computing services for the $\mathrm{Vec_N}$. In this work, we have performed two approaches of task offloading including as V2V and V2I (i.e vehicle to RSU) offloading. We assume that each vehicle $n \in N$ is handling a computationally intensive task that needs to be completed at each time slot $t \in T$, where $T = \{1, \dots, T\}$. The task data is assumed to be granular and can be split into subsets of any size. Specifically, a part of the task will be processed locally on the vehicle n, while the remaining portion will be sent to one of the RSU for processing. Let a_n^t represent the offloading ratio, defined as the fraction of the task's data (in bits) transferred to the cloud server, where $a_n^t \in [0,1]$. Let f_n^{min} and

F_n^{max} denote the current and maximum local computational capabilities, respectively, which can be viewed as the CPU cycles required to process the task data. We assume the use of dynamic voltage and frequency scaling (DVFS) to adjust the chip voltage and manage the local computing capacity, $f_n^{min} \in [0, F_n^{max}]$., adaptively. At each time slot t, vehicle n must select which vehicle n_n^t and RSU p_n^t to offload the task to and send ∂_n^t portions of the task data to the chosen cloud server m_t for remote processing. The remaining $1 - \partial_n^t$ portion of the task data is processed locally by the Vehicle device n. Thus, three factors need to be considered for task offloading: the local computational capacity f_n^t, the offloading ratio ∂_n^t, and the selection of the cloud server m_n^t. Energy consumption and rental charges are included in the total system cost, as they are critical for evaluating the performance of task offloading in VANET. This encompasses the management of task queues, local computing, offloading processing, and problem formulation.

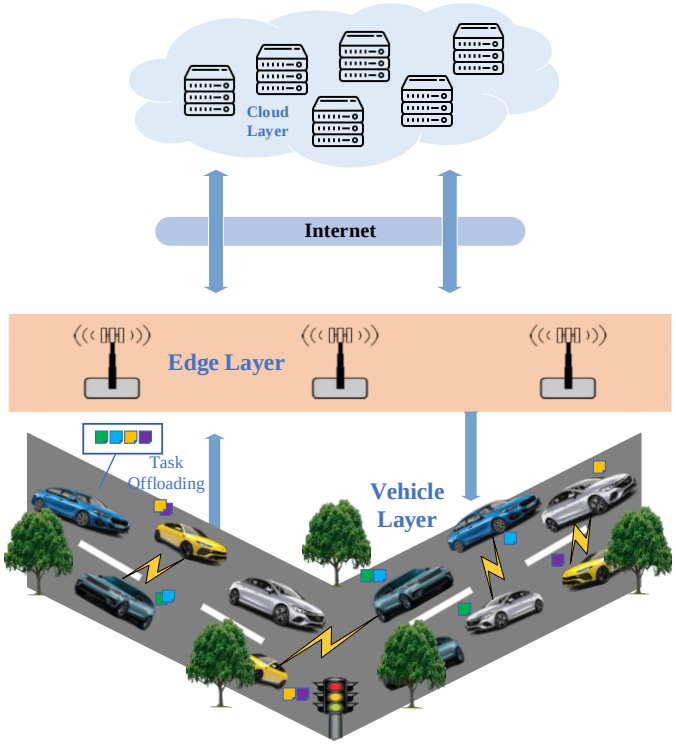

Figure 1. Overall Environment

3.1 Task Queue Modelling

To demonstrate the dynamic nature of the multi-cloud system, we use a task queue. Since Vehicle devices may not always have sufficient computational resources to complete tasks, the outcome of task execution in one-time slot affects the workload for the next slot. Specifically, for Vehicle device n at time slot t, the task information is represented as:

$$\text{Task}_n^{t+1} = \{Ld_n^t, d_n^t, \overline{D}_n^t, c_n^t\} \quad (1)$$

Here, Ld_n^t is the size of the task's data in the queue (in bits), d_n^t is the data size currently being processed (in bits), $\overline{D}_n^t$ is the maximum allowable delay, and c_n^t is the required computing resources needed to complete the task. Additionally, Ld_n^{t+1} is updated to reflect the remaining work for the current time slot and any new tasks arriving for the subsequent time slot is mathematically formulated as,

$$Ld_n^{t+1} = \left[Ld_n^t - d_n^t 1_{drp_n^t=\text{False}}\right]^+ + d_n^{t+1} \quad (2)$$

Where $[z]^+ = \max(z, 0)$ and d_n^{t+1} denotes as the new arrived task computed in upcoming slot $t + 1$. drp_n^t is the bool variable in which $drp_n^t = \text{False}$ denotes the task of vehicle n in time slot t is managed effectively.

3.2 Local Computing

For partial task data $(1-\partial_n^t)$, the delay of processing $\mathfrak{D}_n^{loc}(t)$ and energy consumption $\mathfrak{E}_n^{loc}(t)$ sustained on vehicle n, are defined as,

$$\mathfrak{D}_n^{loc}(t) = \frac{(1-\partial_n^t)\cdot d_n^t \cdot c_n^t}{f_n^t} \quad (3)$$

$$\mathfrak{E}_n^{loc}(t) = ¥ \cdot (f_n^t)^2 \cdot (1-\partial_n^t)^2 \cdot d_n^t \cdot c_n^t \quad (4)$$

Here, ¥ represented as precise constant of switching capacitance.

3.3 Computational Offloading

Computation offloading is a three-step process that allows multi-cloud servers to utilize their extensive computational resources. Initially, the vehicle transfers certain parts of the task data to a suitable cloud server m for processing remotely. The cloud server then handles the task data that was offloaded. The cloud server then relays the task's outcome back to the vehicle device. This method includes both transmission delay and energy expenditure among the vehicle and the desired cloud server. For clarity, this work considers that the size of the assignment's execution results received by the cloud server is modest, rendering the transmission latency and energy usage for feedback insignificant when compared to the processing locally. In addition, we overlook the transmission delayed within the cloud server and the base station (BS), considering that the cloud server is attached to the BS by copper lines or optical fiber. For handling numerous access, we utilize Orthogonal Frequency Division Multiple Access (OFDMA), that tries to reduce wireless channel interference between vehicle devices. This ensures that each Vehicle device receives an equitable share of identical sub-bands from the system's bandwidth B. The uplink communication rate from vehicle n to cloud server m, $Sh_{n,m}^{tra}(t)$, is calculated using the Shannon formula.

$$Sh_{n,m}^{tra}(t) = B.\log_2\left(1 + \left(\mathcal{W}_{n,m}^t \cdot g_{n,m}^t\right)/\omega^2\right) \quad (5)$$

Here, $\mathcal{W}_{n,m}^t$, $g_{n,m}^t$ and ω^2 are denoted as the transmission power, channel gain and power of noise in time slot t, respectively. Energy consumption and transmission delay sustained for offloading the input data in partial $\partial_n^t \cdot d_n^t$ from vehicle n to RSU p and cloud server m, $D_{n,m}^{tra}(t)$ and $\mathfrak{E}_{n,m}^{tra}(t)$ are defined as,

$$D_{n,m}^{tra}(t) = \frac{\partial_n^t \cdot d_n^t}{Sh_{n,m}^{tra}(t)} \quad (6)$$

$$\mathfrak{E}_{n,m}^{tra}(t) = \mathcal{W}_{n,m}^t \cdot D_{n,m}^{tra}(t) \quad (7)$$

As per equation (3) and (6), overall energy consumption of vehicle n for processing the task data d_n^t, $\mathfrak{E}_n^t(t)$ are calculated as,

$$\mathfrak{E}_n^t(t) = \mathfrak{E}_n^{loc}(t) + \mathfrak{E}_{n,m}^{tra}(t) \quad (8)$$

The time required to process a computing task is influenced by the amount of task data and the computational capacity of the cloud server. Since a single RSU and cloud server may handle requests from multiple vehicle, its processing capacity can be a limiting factor and may vary over time. For our purposes, we assume that the computational capacity of the RSU and cloud server fluctuates randomly across different time slots but remains constant within each slot. Assume $f_{occ}^t = PPr(\gamma) \cdot f_s^{unitis}$ as the computation resource that is occupied for individual unit. The computation server capacity m, f_m^t, can be described as , $f_m^t =, F_{ser}^{max} -, f_{occ}^t$, here F_{ser}^{max} is the cloud server maximum capacity of computation of cloud server $m \in M$. Moreover, the computation RSU capacity p, f_p^t, can be described as , $f_p^t =, F1_{ser}^{max} -, f_{occ}^t$, here $F1_{ser}^{max}$ is the RSU maximum computation capacity RSU $p \in P$.

$$De_{n,m}^t(t) = \frac{\sum_{n=1}^{N}\left(1_{m_n^t=m}\cdot\partial_n^t\cdot d_n^t\cdot c_n^t\right)}{f_m^t} \quad (9)$$

$$De_{n,p}^t(t) = \frac{\sum_{n=1}^{N}\left(1_{p_n^t=p}\cdot\partial_n^t\cdot d_n^t\cdot c_n^t\right)}{f_p^t} \quad (10)$$

In cloud computing and RSU, the execution delay $De_n^m(t)$ and $De_n^p(t)$ incurs a server fee for the vehicle from the RSU and cloud service provider. In other words, the cloud provider for utilizing the both RSU's charges the vehicle and server's computing resources to perform the task. Assume $c(f_m^t) = e^{(-\lambda)} \cdot (f_m^t - 1)$ and $c\left(f_p^t\right) = e^{(-\lambda)} \cdot \left(f_p^t - 1\right)$. ϖ is the price per time unit in computing capability f_m^t and f_p^t, where λ and ϖ are the two parameters of coefficients. Henceforth, the service charge $C_n^m(t)$ and $C_n^p(t)$ necessary for process vehicle n's partial task in RSU and cloud server which is formulated by,

$$C_n^m(t) = c(f_m^t) \cdot De_n^m(t) \quad (11)$$

$$C_n^p(t) = c(f_p^t) \cdot De_n^p(t) \quad (12)$$

3.4 Problem Formulation

The objective of the dynamic computing offloading problem, as described in Eq. (10) as follows, is to minimize the long-term system cost, which includes both energy consumption $\mathbb{E}_n^t(t)$ and rental charges $C_n^m(t)$ and $C_n^p(t)$.

$$\min_{m_n^t, \partial_n^t, f_n^t} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\beta_1 \cdot \mathbb{E}_n^t(t) + \beta_2 \cdot C_n^m(t) \right) \right) \right) \quad (13)$$

$$\min_{p_n^t, \partial_n^t, f_n^t} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\beta_1 \cdot \mathbb{E}_n^t(t) + \beta_2 \cdot C_n^p(t) \right) \right) \right) \quad (14)$$

s.t.

$$C1: \ \max\left(D_n^{loc}(t), D_{n,m}^{tra}(t) + D_n^m(t) \right) \le \overline{D}_n^t, \Delta n \in N, \Delta t \in T \quad (15)$$

$$C2: \ \max\left(D_n^{loc}(t), D_{n,p}^{tra}(t) + D_n^p(t) \right) \le \overline{D}_n^t, \Delta n \in N, \Delta t \in T \quad (16)$$

$$C3: \ m_n^t \in M, \Delta m \in M, \Delta n \in N, \Delta t \in T \quad (17)$$

$$C4: \ p_n^t \in P, \Delta p \in P, \Delta n \in N, \Delta t \in T \quad (18)$$

$$C5: \ 0 \le \partial_n^t \le 1, \Delta n \in N, \Delta t \in T \quad (19)$$

$$C6: \ 0 \le f_n^t \le F_n^{max}, \Delta n \in N, \Delta t \in T \quad (20)$$

Here, β_1 and β_2 are the parameters of tradeoff weight. Constraint (C1, C2) states that the actual completion time of any given task must not exceed its maximum allowable delay. Constraint (C3, C4) specifies that each vehicle can be assigned to only one cloud server for processing its tasks. The offloading ratio, defined by Constraint (C5), is a variable that ranges from 0 to 1 for each task. Constraint (C6) ensures that the local computing capacity of any Vehicle device cannot exceed its maximum computational capability. To be more specific, $m_n^t, p_n^t, \partial_n^t$ and f_n^t are denoted as the hybrid decision variables continue-discrete connected with Vehicle devices n, where ∂_n^t and f_n^t is the variable of continue and m_n^t and p_n^t is discrete variable.

3.5 TOP-VANET Framework

This section first relaxes the continuous-discrete decision variable to a continuous decision variable. It then introduces an optimal task offloading Markov Decision Process (MDP) with multiple agents. Finally, a detailed explanation of the Supportive Twin Delayed DDPG (SMATD3) process is provided. To address these challenges in Eq. (10), we employ a probabilistic approach to convert the discrete choice variable m_n^t and p_n^t into a continuous variable. Specifically, we define $Pro(m_n^t) \in \left[\frac{m-1}{M}, \frac{m}{M}\right]$ and $Pro(p_n^t) \in \left[\frac{p-1}{P}, \frac{p}{P}\right]$ as the probability that the task is offloaded to RSU (p) cloud server (m). In other words, if vehicle (n) opts to offload its task to RSU (p) cloud server (m) at time slot t, it is considered to have chosen a continuous decision variable $Pro(m_n^t)$. For example, with $M = 5$ cloud servers, if $Pro(m_n^t) \in \left[\frac{1}{5}, \frac{2}{5}\right]$, server $m = 2$ can be selected for task offloading. Consequently, the problem of minimizing the overall system cost can be reformulated as follows:

$$\min_{Pro(m_n^t), \partial_n^t, f_n^t} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\beta_1 \cdot \mathbb{E}_n^t(t) + \beta_2 \cdot C_n^m(t) \right) \right) \right) \quad (21)$$

$$\min_{Pro(p_n^t), \partial_n^t, f_n^t} \left(\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\beta_1 \cdot \mathbb{E}_n^t(t) + \beta_2 \cdot C_n^p(t) \right) \right) \right) \quad (22)$$

s. t.

$$C1: \ \max\left(D_n^{loc}(t), D_{n,m}^{tra}(t) + D_n^m(t) \right) \le \overline{D}_n^t, \Delta n \in N, \Delta t \in T \quad (23)$$

$$C2: \ \max\left(D_n^{loc}(t), D_{n,p}^{tra}(t) + D_n^p(t) \right) \le \overline{D}_n^t, \Delta n \in N, \Delta t \in T \quad (24)$$

$$C3: \ 0 \le Pro(m_n^t) \le 1, \Delta m \in M, \Delta n \in N, \Delta t \in T \quad (25)$$

$$C4: \ 0 \le Pro(p_n^t) \le 1, \Delta p \in P, \Delta n \in N, \Delta t \in T \quad (26)$$

$$C5: \ 0 \le \partial_n^t \le F_n^{max}, \Delta n \in N, \Delta t \in T \quad (27)$$

$$C6:\ 0 \le f_n^t \le F_n^{max}, \Delta n \in N, \Delta t \in T \qquad (28)$$

To address the dynamic offloading problem, a Deep Reinforcement Learning (DRL) agent will be introduced. In centralized DRL, the base station RSU collects environment states from all Vehicle devices, increasing communication overhead; thus, the paper explores multiagent approaches to improve offloading solutions. Given the non-stationarity of the network environment and the instability of traditional multiagent DRL, a cooperative framework is proposed that uses centralized training and distributed execution with locally executable actor networks and observable critic networks to ensure convergence.

3.6 MDP Formulation

We design the task offloading optimal problem as multi-agent Markov decision process (MDP). The MDP of multi-agent can be represented as 4 tuples $(\aleph, \delta_n, A_n, r_n)$. Here $\aleph$ is defined as agent space, δ_n, A_n and r_n are the state space, action space and reward function of agent n, respectively.

Agent space $\aleph$: $\aleph = \{1, \dots, \aleph\}$ where $\aleph$ described as the number of vehicles, each vehicle act as an agent. For agent $n, n = 1,2, \dots, \aleph$ by calculating the RSU and cloud server selection $Pro(m_n^t)$ and $Pro(p_n^t)$, offloading ratio ∂_n^t and capacity of local computation f_n^t, the agent can acquire the overall system cost in minimal.

State space δ_n : For vehicle n, the state s_n^t is denoted as composed of computation task, the channel gain among vehicle n and BS of time slot t and the computation capacity of overall RSU and cloud server in multi-tier network.

$$s_n^t = \{Task_n^t;\ g_n^t;\ f_1^t, \dots, f_m^t\} \qquad (29)$$

Action space A_n: Since individual vehicle is obtained to calculate the probability of its chosen RSU and cloud server $Pro(p_n^t)$ and $Pro(m_n^t)$. Furthermore, ∂_n^t and f_n^toffloading ratio and capacity of local computation, the action space is defined as,

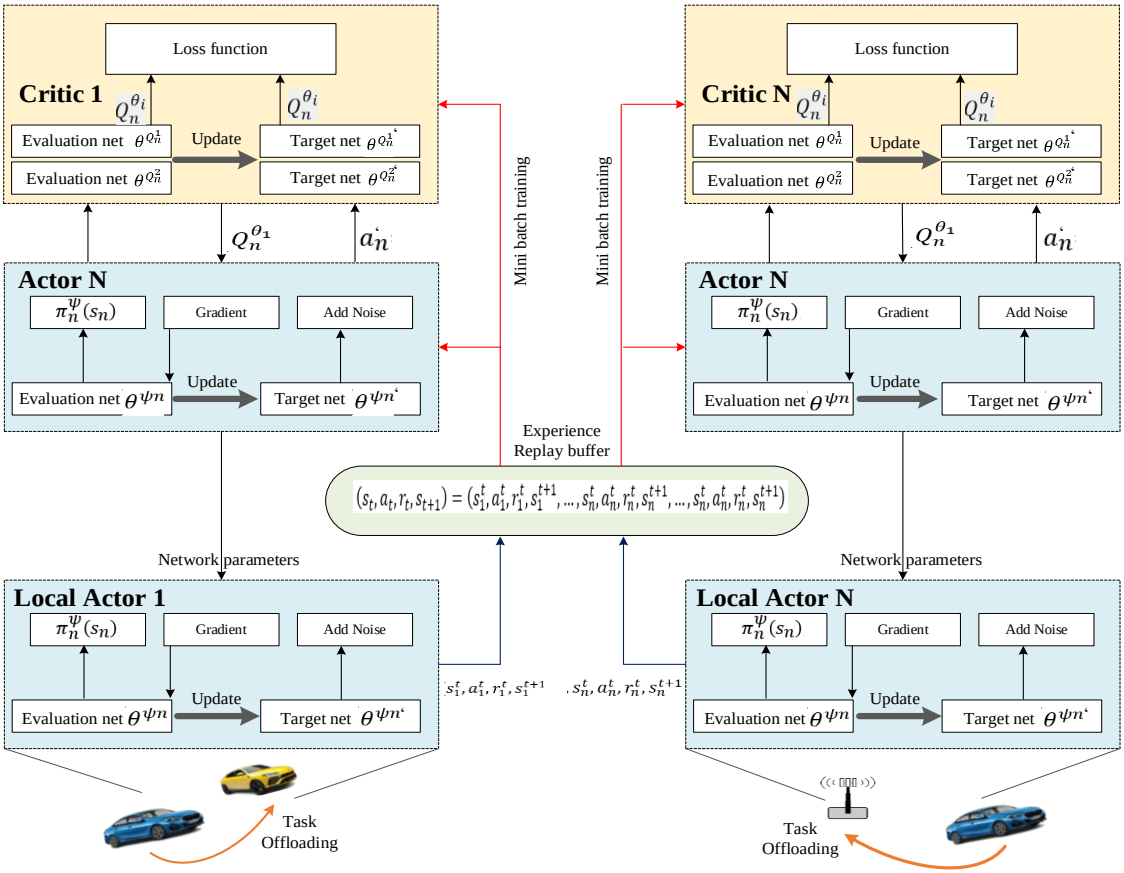

Figure 2. Supportive Multi-agent Deep Reinforcement Learning based Task Offloading

$$a_n^t = \{Pro(p_n^t), \partial_n^t, f_n^t\} \quad (30)$$

Reward function r_n : To acquire the near-optimal policy for the optimization problem of task offloading in Eq. (11), the number of agents might collaborate to reduce the total system cost. Stated differently, the purpose of reward function r_n^t is to train the agent operating at vehicle n to make decisions that meet the limitations. If none of the constraints specified in Equations is violated by the decision variables related to the action, the activity is considered successful. (11a) through (11f), the reward is then calculated as the reciprocal of the weight sum $(\beta_1 \cdot \mathfrak{E}_n^t + \beta_2 \cdot \mathfrak{E}_n^m(t))$ multiplied by a constant C1. If not, the reward is defined as a negative number (represented by $-C3$), which stands for a punishment. At every time slot t, the instantaneous benefit received is expressed as

$$r_n(t) = \begin{cases} \frac{C1}{\beta_1 \cdot \mathfrak{E}_n^t + \beta_2 \cdot \mathfrak{E}_n^m(t)} a_n^t \text{is successful} \\ -C2 \qquad \text{Otherwise} \end{cases} \quad (31)$$

Here, the constants $C1$ and $C2$ are both positive. It is noted that a successful action, which results in a reduced overall system cost due to multi-cloud offloading decisions, provides a higher immediate reward. This approach aims to maximize the estimation of discounted cumulative benefits.

3.7 Multi-agent DRL Framework

To facilitate centralized decision-making using DRL, all vehicles, RSU and cloud servers must relay their environmental status to the BS. However, with the addition of more Vehicle devices or cloud servers, communication overhead is likely to increase, and the state-action space may grow exponentially, reducing convergence efficiency. Our objective is to explore effective computation offloading techniques within a multiagent DRL framework to overcome these challenges. Traditional multi-agent DRLs often face issues with overestimation and high variance due to the high-dimensional continuous action space. To address these challenges in dynamic multi-cloud scenarios, the TD3 algorithm is employed to evaluate the efficiency of selected cloud servers $Pro(m_n^t)$ and RSUs $Pro(p_n^t)$, offloading ratios ∂_n^t, and local computing capabilities f_n^t. The multi-cloud offloading system utilizes local observations from each vehicle to estimate the vehicle's local computation capacity f_n^t, task offloading ratio ∂_n^t, and server selection $Pro(m_n^t)$ and $Pro(p_n^t)$. To learn distributed compute offloading rules, the twin delayed DDPG (TD3) agent jointly optimizes these three variables for each vehicle. This approach is referred to as the SMATD3 framework, as illustrated in Figure 2. In this framework, each agent's actor network, trained on a edge server near the base station, makes offloading decisions based on local observations of the network state, following a centralized training and distributed execution strategy. The actor network for each agent is periodically updated with the training parameters, while the edge server deploys a two-critic network that operates with global observations, encompassing all agent states and actions. As a result, the learning environment appears stationary from each agent's perspective, despite any policy changes. The training phases of the SMATD3 agent are illustrated as follows. During each time slot t, the global observable two-critic network on the proxy server trains the actor network for each agent n, guiding the computation offloading strategy. To improve training efficiency and stabilize the process, the local experience transitions $(s_n^t, a_n^t, r_n^t, s_n^{t+1})$ for each agent n are stored in the proxy server's experience replay buffer. These local experiences are combined into a global experience replay buffer $\mathbb{B}$, which consolidates the transitions from all agents are represented as $(s_t, a_t, r_t, s_{t+1}) = (s_1^t, a_1^t, r_1^t, s_1^{t+1}, \dots, s_n^t, a_n^t, r_n^t, s_n^{t+1}, \dots, s_n^t, a_n^t, r_n^t, s_n^{t+1})$. After that, for individual agent $n = 1,2, \dots, \aleph$, the actor function is approached through DNN with parameter $\theta^{\psi n}$ as $\pi_n^{\psi}(s_n)$, that considers the state s_n, as input. Additionally, the Q-function of two-critic network is also approached through two DNN with parameter $\theta^{Q_n^i}$ as $Q_n^{\theta i}\left(s, a|\theta^{Q_n^i}\right), i = 1,2,$, which considers state $s = (s_1, \dots, s_\aleph)$ and action set $a = (a_1, \dots, a_\aleph)$ as input. While the process of training, individual agent arbitrarily instances a mini-batch $\{s_j, a_j, r_j, s_j'\}_{j=1}^{\jmath}$ from the replay buffer of global experience $\mathbb{B}$. The policy gradient of the estimation actor network can be formulated as,

$$\Gamma_{\theta^{\psi n}} J\left(\theta^{\psi n}\right) \approx \varepsilon\left[\Gamma_{\theta^{\psi n}} \pi_n^{\psi}(s_j) \cdot \Gamma_{a_n} Q_n^{\theta_i}\left(s, a|\theta^{Q_n^1}\right)|_{a_n = \pi_n^{\psi}(s_n)}\right] \quad (32)$$

Additionally, the target action a_j' is defined as $a_j' = \pi_n^{\psi'}(s_j') + \mathbb{N}$, where $\mathbb{N} \sim clip\left(\mathfrak{Y}(0, \mathfrak{Q}^2)\right)$ represents clipped noise introduced to the target actor network, with a mean of 0 and standard deviation$\mathfrak{Q}^2$. This clipping helps prevent overfitting to narrow peaks in Q-values and facilitates smoother state-action estimation for TD3. The goal values z_j are defined using the target policy smoothing scheme as described above.

$$z_j = r_j + \Upsilon \min_{i=1,2} Q_n^{\theta_i'}\left(s_j' + a_j'|\theta^{Q_n^i}\right), i = 1,2 \quad (33)$$

Next, as defined above, the two Q-functions, with $Q_n^1(s_j, a_j)$ and $Q_n^2(s_j, a_j)$, are simultaneously acquired from the network of two-critic. The weight parameters $\theta^{Q_n^i}$ of $Q_n^{\theta_1}(s_j, a_j)$, are modernized through reducing the loss function Los $\left(\theta^{Q_n^i}\right)$ is determined as,

$$\text{Los}\left(\theta^{Q_n^i}\right) \approx \varepsilon\left[z_j - Q_n^{\theta_i}(s_j, a_j)\right]^2, i = 1,2 \quad (34)$$

According to equ (15) and (17), assume Λ as the learning rate, the weight of estimation actor network and two estimation critic networks are modernized as,

$$\theta^{\psi n} \leftarrow \theta^{\psi n} - \Lambda\Gamma_{\theta^{\psi n}} J\left(\theta^{\psi n}\right) \quad (35)$$

$$\theta^{Q_n^i} \leftarrow \theta^{Q_n^i} - \Lambda\Gamma_{\theta^{Q_n^i}} J\left(\theta^{Q_n^i}\right), i = 1,2 \quad (36)$$

Ultimately, each agent updates the weights of the evaluation actor network less frequently to reduce temporal difference (TD) error. Specifically, every $\mathfrak{S}$ periods, individual vehicle performs an update to the evaluation actor network.

4. Result and Analysis

This section begins with a description of the datasets, parameter settings, and experimental environment. We then compare the delay performance of the proposed algorithm with that of standard algorithms. Finally, we evaluate the effectiveness of PTOS's candidate selection process in managing vehicle movement. We used MATLAB to simulate our experimental setup, which featured a 10-kilometer, three-lane highway populated with several moving vehicles. The vehicle trajectories were based on the Madrid trace [28], which consists of artificial data derived from three Madrid freeways (A6, M40, and M30). The cars' onboard computers operated at speeds ranging from 4 × 106 to 2 × 107 cycles per second. The computational complexity for tasks was set at 30 for image processing, 15 for video processing, 40 for interactive games, and 20 for augmented reality, with each task potentially generating 50 to 100 subtasks. Table 3 summarizes the key simulation parameters. To emulate natural vehicle movement, we selected the track records of 20 cars on the A6 highway over 30 minutes. At the beginning of the simulation, any vehicle could communicate with others within a 150-meter range. Figure 4 illustrates the initial connectivity among the twenty vehicles. We analyze the impact of various factors, such as learning rate and batch size, on the performance of the proposed SMATD3 agent to validate training efficiency, as illustrated in Figs. 3(a) and (b). The SMATD3 agent typically undergoes offline training with a setup of three cloud servers (M) and three.

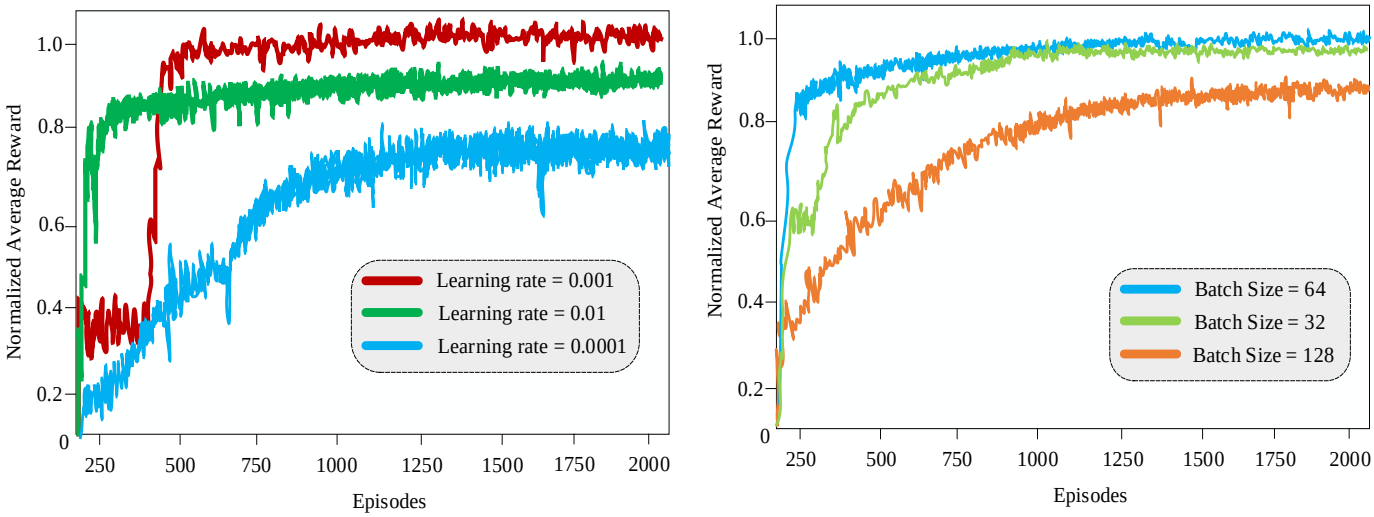

Figure 3. Normalized average rewards of the SMATD3 agent with and (a) varying batch sizes and (b) diverse learning rates.

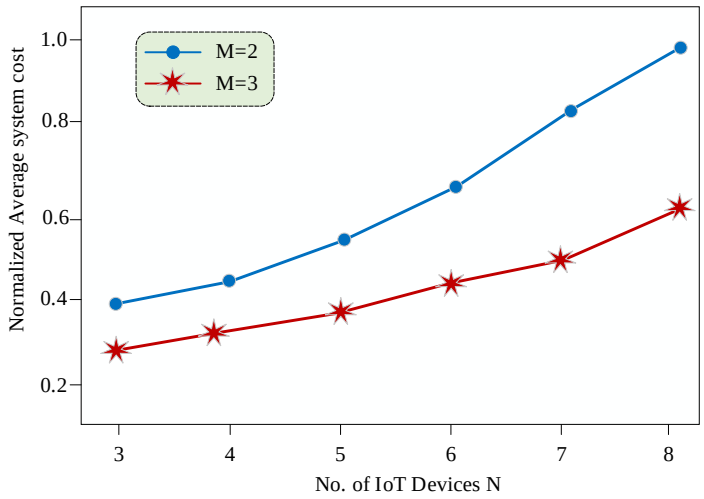

Figure 4. Normalized average system costs for scenarios with M = 2 and M = 3 cloud servers, appraised transversely dissimilar vehicle counts.

vehicles (N). Figure 3(a) shows the normalized average reward of SMATD3 across two critic networks with different learning rates. At a low learning rate of 0.0001, the SMATD3 agent struggles to achieve high reward values due to the slow update of the DNN's parameters. On the other hand, a high learning rate of 0.01 leads to rapid fluctuations in the CNN's weight parameters. Therefore, a learning rate of 0.001 is preferred, balancing stability and performance. Figure 3(b) illustrates the normalized average rewards of SMATD3 across different batch sizes. As seen in the figure, batch sizes of 32 and 128 result in diminished training performance and oscillating cumulative reward curves at lower values. A small batch size fails to adequately cover the transitions stored in the replay buffer, while a large batch size frequently samples previously unsuccessful transitions, affecting training efficiency. Consequently, we have chosen to set the batch size to 64. To evaluate the scalability of our proposed SMATD3 agent, we analyse its performance with different numbers of cloud servers and vehicle devices, as depicted in Figure 4. The results show that as the number of Vehicle devices increases, the demand for computation offloading also rises, leading to a higher overall system cost. Conversely, as the number of cloud servers (M) increases, more servers are available to participate in computation offloading, which helps to distribute the load. This results in a lower overall system cost when more cloud servers are utilized for a given number of Vehicle devices and tasks. However, when dealing with a small number of Vehicle devices and tasks, additional cloud servers may not be necessary for efficient computation offloading.

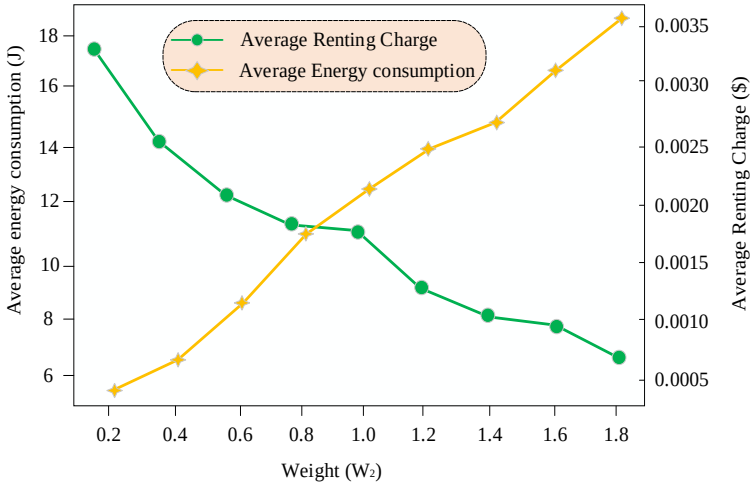

Figure 5. Rental charges and energy consumption as a function of the weight parameter ω2

For example, when there are three Vehicle devices (N = 3), the performance with two cloud servers (M = 2) is almost identical to that with three cloud servers (M = 3). Moreover, even with eight Vehicle devices and three cloud servers, the SMATD3 agent effectively handles the multi-cloud computation offloading problem. This demonstrates the high scalability of the proposed SMATD3 agent in terms of cloud servers, state spaces, and action spaces. Figure 5 illustrates the relationship between energy consumption and rental charges as influenced by the weight parameter ω2, with ω1 fixed at 1. Specifically, ω1 and ω2 represent the relative importance of energy usage and rental costs, respectively. As shown in Figure 5, the rental fee increases as ω2 rises from 0.2 to 1.8. This occurs because fewer tasks are offloaded to the cloud server, leading to reduced rental charges but higher energy consumption. However, the decline in rental charges slows when ω2 increases to 1.6 and 1.8, due to the limited computing power of available cloud servers, which forces Vehicle devices to consume more energy as less task data is offloaded. Figure 6 compares the performance of the SMATD3 agent with the theoretically optimal outcome. We calculate the theoretically optimal result at each time slot, represented by a black line, and compare it with the experimental results obtained using SMATD3. The normalized system costs with SMATD3 oscillate around 0.8, while the theoretical optimal outcomes remain close to 0.9. The difference between the experimental and ideal results is generally less than 0.1, indicating that the performance of our proposed SMATD3 is nearly optimal.

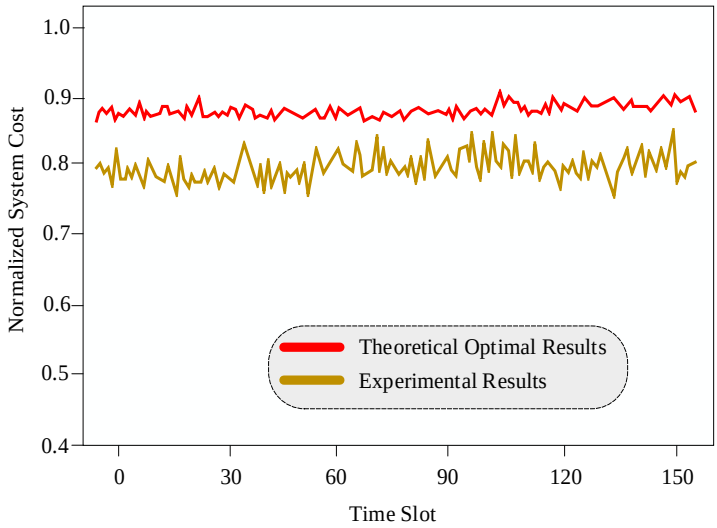

Figure 6. Performance difference between the theoretically optimal outcome and the proposed SMATD3

We conduct comprehensive comparative experiments under various system configurations to assess the benefits and effectiveness of the proposed CMATD3 method for multi-cloud job offloading. The evaluation includes four DRL-based algorithms: MD-Hybrid-AC [22], MADDPG, MADQN, and MADQN. Additionally, two heuristic algorithms, ACO [24] and SPSO-GA [27], are also examined to provide a broad performance comparison. MADQN: To address dynamic multi-cloud offloading problems with a continuous-discrete hybrid action space, action values are first quantized. Two multi-agent systems are developed based on different discretization levels MADQN: For each agent associated with an IoT device, the ranges of decision variables, such as the offloading ratio δ_n^t and computing capacity f_n^t, are divided into 5 levels. Additionally, there are three choices for cloud server selection c_n^t, resulting in an action dimension of 13 for each agent. MADQN: This system divides the ranges of decision variables, including the offloading ratio δ_n^t and computation capacity f_n^t, into ten levels for each agent.

MD-Hybrid-AC: This enhanced actor-critic architecture addresses the computation-offloading problem with continuous-discrete hybrid decisions using a decentralized execution framework combined with centralized training.

MADDPG: This cooperative multi-agent approach, an extension of DDPG, is used to develop offloading policies for decentralized dynamic computation.

SMATD3: The proposed agent in this paper.

The deep neural network (DNN) hyper-parameters in MADQN, MADQN, MD-Hybrid-AC, and MADDPG are consistent with those used in SMATD3. Four out of the five algorithms demonstrate convergence during training, as shown in Figure 7. It is evident that with an increase in training episodes, the normalized reward gradually improves, with longer episodes yielding higher normalized rewards. Notably, SMATD3 achieves the best convergence in terms of normalized reward among all methods. This

superior performance is attributed to TD3's dual critic networks, which effectively address the overestimation issue, thereby enhancing both the stability and effectiveness of the training process.

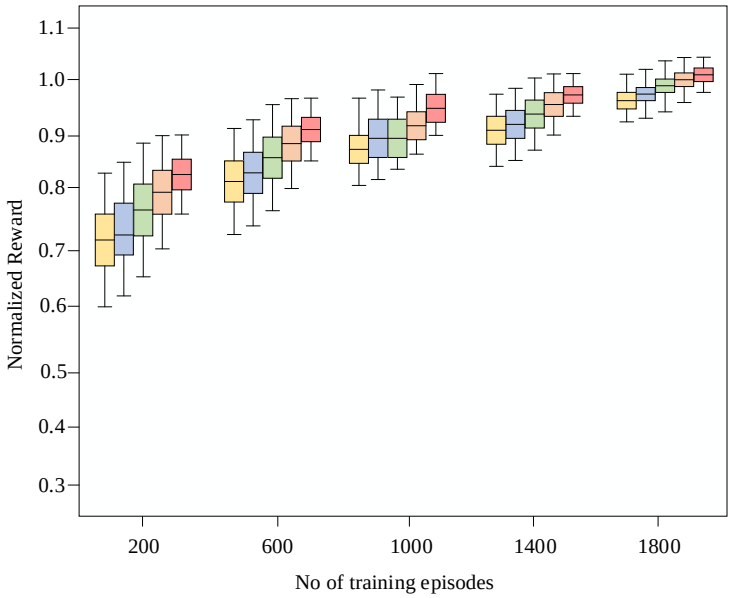

Figure 7. Normalized reward values Acquired While Training

5. Conclusion

In conclusion, this research addresses the challenge of complex computing tasks in Vehicle Ad Hoc Networks (VANETs) by exploring efficient task offloading strategies. The proposed approach introduces a novel probabilistic method that transforms discrete task offloading actions, such as selecting cloud servers, into a continuous decision space. By leveraging a Supportive Multi-Agent Deep Reinforcement Learning (SMADRL) framework, the study optimizes system costs associated with vehicle device energy consumption and cloud server rental fees. The framework's centralized training and distributed execution allow each vehicle to function as an independent agent, effectively learning decentralized policies that alleviate computing burdens. Experimental results validate the effectiveness of the SMADRL framework, demonstrating its superior performance over existing DRL-based methods and heuristic algorithms, and highlighting its potential to significantly reduce overall system costs.

References

[1] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, 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.

[2] N. Fofana, A. B. Letaifa, and A. Rachedi, "Intelligent task offloading in vehicular networks: A deep reinforcement learning perspective," *IEEE Transactions on Vehicular Technology*, 2024.

[3] P. Lv, W. Xu, J. Nie, Y. Yuan, C. Cai, Z. Chen, and J. Xu, "Edge computing task offloading for environmental perception of autonomous vehicles in 6G networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1228-1245, 2022.

[4] S. Vemireddy and R. R. Rout, "Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing," *Computer Networks*, vol. 199, p. 108463, 2021.

[5] L. Zhang, X. Wang, and H. Li, "Task offloading in vehicular edge computing: A survey of algorithms and architectures," *IEEE Access*, vol. 11, pp. 12345-12367, 2023.

[6] B. T. H. Binh, H. K. Vo, B. M. Nguyen, H. T. T. Binh, and S. Yu, "Value-based reinforcement learning approaches for task offloading in delay constrained vehicular edge computing," *Engineering Applications of Artificial Intelligence*, vol. 113, p. 104898, 2022.

[7] X. Dai, Z. Xiao, H. Jiang, H. Chen, G. Min, S. Dustdar, and J. Cao, "A learning-based approach for vehicle-to-vehicle computation offloading," *Journal of Soft Computing*, vol. 10, no. 8, pp. 7244-7258, 2022.

[8] S. Lingayya, S. B. Jodumutt, S. R. Pawar, A. Vylala, and S. Chandrasekaran, "Dynamic task offloading for resource allocation and privacy-preserving framework in Kubeedge-based edge computing using machine learning," *Cluster Computing*, vol. 27, no. 7, pp. 9415-9431, 2024.

[9] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in VANETs," *Intelligent Transportation*, vol. 23, no. 8, pp. 11994-12013, 2021.

[10] M. B. Taha, C. Talhi, H. Ould-Slimane, S. Alrabaee, and K. K. R. Choo, "A multi-objective approach based on differential evolution and deep learning algorithms for VANETs," *Personal Computing*, vol. 72, no. 3, pp. 3035-3050, 2022.

[11] A. Waheed, M. A. Shah, S. M. Mohsin, A. Khan, C. Maple, S. Aslam, and S. Shamshirband, "A comprehensive review of computing paradigms, enabling computation offloading and task execution in vehicular networks," *Journal of Information Technology*, vol. 10, pp. 3580-3600, 2022.

[12] K. Mishra, G. N. Rajareddy, U. Ghugar, G. S. Chhabra, and A. H. Gandomi, "A collaborative computation and offloading for compute-intensive and latency-sensitive dependency-aware tasks in dew-enabled vehicular fog computing: A federated deep Q-learning approach," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4600-4614, 2023.

[13] X. Dai, Z. Xiao, H. Jiang, and J. C. Lui, "UAV-assisted task offloading in vehicular edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 23, no. 4, pp. 2520-2534, 2023.

[14] Z. Zhang and F. Zeng, "Efficient task allocation for computation offloading in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5595-5606, 2022.

[15] C. Yang, X. Chen, Z. Yao, and J. Yang, "Task offloading and serving handover of vehicular edge computing networks based on trajectory prediction," *Green Engineering*, vol. 9, p. 130793, 2021.

[16] R. A. Dziyauddin, D. Niyato, N. C. Luong, A. A. A. M. Atan, M. A. M. Izhar, M. H. Azmi, and S. M. Daud, "Computation offloading and content caching and delivery in vehicular edge network: A survey," *Computer Networks*, vol. 197, p. 108228, 2021.

[17] M. Ahmed, M. A. Mirza, S. Raza, H. Ahmad, F. Xu, W. U. Khan, and Z. Han, "Vehicular communication network enabled CAV data offloading: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 7869-7897, 2023.

[18] V. Patsias, P. Amanatidis, D. Karampatzakis, T. Lagkas, K. Michalakopoulou, and A. Nikitas, "Task allocation methods and optimization techniques in edge computing: A systematic review of the literature," *Future Internet*, vol. 15, no. 8, p. 254, 2023.

[19] A. A. Baktayan, A. T. Zahary, and I. A. Al-Baltah, "A systematic mapping study of UAV-enabled mobile edge computing for task offloading," *Sensors*, 2024.

[20] Sherubha, "Graph Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks," *Sādhanā*, vol. 45, p. 212, 2020.

[21] C. Ma, J. Zhu, M. Liu, H. Zhao, N. Liu, and X. Zou, "Parking edge computing: Parked-vehicle-assisted task offloading for urban VANETs," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9344-9358, 2021.

[22] Y. He, D. Zhai, F. Huang, D. Wang, X. Tang, and R. Zhang, "Joint task offloading, resource allocation, and security assurance for mobile edge computing-enabled UAV-assisted VANETs," *Remote Sensing*, vol. 13, no. 8, p. 1547, 2021.

[23] D. Wei, J. Zhang, M. Shojafar, S. Kumari, N. Xi, and J. Ma, "Privacy-aware multiagent deep reinforcement learning for task offloading in VANET," *Nature*, vol. 24, no. 11, pp. 13108-13122, 2022.

[24] V. D. Ak et al., "Cloud enabled media streaming using Amazon Web Services," in *IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pp. 195-198, 2017.

[25] M. S. Bute, P. Fan, L. Zhang, and F. Abbas, "An efficient distributed task offloading scheme for vehicular edge computing networks," *Vehicle Science*, vol. 70, no. 12, pp. 13149-13161, 2021.

[26] Y. Wang, X. Du, Z. Lu, Q. Duan, and J. Wu, "OPTOS: A strategy of online pre-filtering task offloading system in vehicular ad hoc networks," *Scientific Reports*, vol. 10, p. 4112, 2022.

[27] M. Gong, Y. Yoo, and S. Ahn, "Adaptive computation offloading with task scheduling minimizing reallocation in VANETs," *Electronics*, vol. 11, no. 7, p. 1106, 2022.

[28] K. S. Patel, R. J. Singh, and M. A. Qureshi, "A novel framework for task offloading in 5G-enabled vehicular networks using machine learning," *Future Generation Computer Systems*, vol. 130, pp. 1-14, 2024.

[29] K. Sarieddine, H. Artail, and H. Safa, "An opportunistic vehicle-based task assignment for IoT offloading," *Computer Networks*, vol. 212, p. 109038, 2022.

[30] G. Wu, H. Chen, and S. Sun, "Joint optimization of task offloading and resource allocation based on differential privacy in vehicular edge computing," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 1, pp. 109-119, 2021.