

Traffic Classification-Assisted Multitasking Offloading Strategy in IoV

Chen-Wei Feng, Zhen-Zhen Lin*, and Ya-Xi Yang

School of Opto-electronic and Communication Engineering, Xiamen University of Technology,
Xiamen 361000, China

cwfeng@xmut.edu.cn, zhenzlh@163.com, 915225418@qq.com

Received 14 December 2024; Revised 10 April 2025; Accepted 12 May 2025

Abstract. Mobile Edge Computing (MEC) is one of the key technologies of 5G network, which can better meet the demand of low latency and high reliability of Telematics. However, in a multi-vehicle roadway in a Internet of Vehicles (IoV) scenario, the edge server cannot compute a large number of offloading tasks in a short period of time. In this paper, a joint offloading allocation method with Naive Bayesian combined with reinforcement learning is proposed, which introduces an idle vehicle parked on the roadside as a computational device so that it can perform task offloading together with the edge server and the vehicle terminal device. The offloading problem is formulated as an optimization problem with the objective of minimizing the offloading delay and the energy consumption of the user's vehicle, and it is further transformed into a Markov Decision Process (MDP). In order to solve the problem in discrete and continuous action space, Parametrized Deep Q-Networks Learning (P-DQN) algorithm is used to solve the problem. Simulation results show that the proposed algorithm exhibits superior performance in terms of task vehicle utility and resource utilization relative to other algorithms.

Keywords: Internet of Vehicles, mobile edge computing, task offloading, Naive Bayesian, deep reinforcement learning

1 Introduction

With the growing applications of IoV, the demand for compute-intensive and latency-sensitive in-vehicle services has dramatically risen, already exceeding the vehicle's own computational and storage limitations. In this context, the computation offloading technology comes into being, and its emergence brings a new solution to the computation of vehicle applications, which effectively enhances the computation capability of vehicles and the execution efficiency of applications by assigning part of the computation tasks to the cloud or other computation nodes [1-4]. Though the distribution of computing tasks to the cloud plays an important role in expanding the computing power of vehicles, for latency-sensitive applications, transmission latency can be a limiting factor due to the distance of the cloud from the end device. Therefore, when selecting a computing offloading scheme, it is necessary to comprehensively consider the nature of the computing task, the transmission delay, and the needs of the actual application in order to achieve the best performance and user experience. The European Telecommunication Standardization Society has introduced the concept of mobile edge computing [5], the core concept is to place computing and storage resources in the network close to the mobile device side, enabling mobile device tasks to be offloaded to nearby MEC servers for execution, dramatically reducing task processing time, providing low-latency, high-throughput services, and improving the quality of user experience [6]. The emergence of Vehicular Edge Computing (VEC) marks a high degree of convergence between MEC technologies and IoV. But the computational resources of the edge nodes are very limited, and when in a multi-vehicle roadway in a IoV scenario, the edge servers cannot compute a large number of offloading tasks in a short period of time. Therefore, it is of great research significance to formulate a reasonable offloading strategy and effectively utilize the computational resources of the edge nodes, so as to reduce the task processing time and improve the quality of user experience.

When facing the problems of task offloading in VEC environments, research usually focuses on a few core optimization objectives: reducing the latency of the task offloading process, decreasing the energy consumption, and guaranteeing high quality of the application results [7]. Literature [8] proposed an innovative transportation architecture by integrating IoV, Software Defined Network (SDN) and MEC technologies, which can effectively

* Corresponding Author

alleviate traffic congestion and improve the efficiency of urban management. Although these studies expand the utilization of MEC in IoV application scenarios, the majority of them only touch the surface of MEC technology and lack in-depth discussions on key technologies such as computational offloading and resource allocation. Owing to the inability of vehicle terminals to meet the requirements of computing tasks in terms of latency and energy consumption, literature [9] proposed a cloud-based MEC offloading framework that explored the effectiveness of offloading strategies combining Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communication modes as well as discussed a model architecture of how a vehicle can exchange and share information with other vehicles or roadside units utilizing V2V and V2I communication modes. These studies not only optimized the task offloading process in vehicular edge computing networks, but also greatly improved the computational efficiency and responsiveness of vehicular networks by introducing innovative communication patterns and offloading strategies. Literature [10] proposed an SDN-driven cloud-edge cluster collaborative vehicular networking architecture to address the inability of traditional edge computing architectures to meet the growing demand of applications. By jointly optimizing task offloading and computational resource allocation, it aims to minimize system latency, maximize profit for edge service providers, and balance system energy consumption. Literature [11] studied the optimal management of task offloading for connected vehicles in edge computing based vehicular networking. Analyzed the goals of computational task offloading, constructed a delay model and an energy consumption model for computational task offloading, described the process of offloading computational tasks in vehicular networking, including sensing offloading nodes, task preprocessing and classification, offloading decision, task transmission and processing and result return, aiming at optimizing task allocation, node load balancing and maximizing economic efficiency. Regarding the task offloading methods, literature [12] proposed a scheduling scheme based on genetic algorithm considering the complexity of the task offloading and scheduling optimization problem in the vehicular cloud computing environment. The scheme reduces the total latency of task processing and improves the utilization efficiency of computing resources by offloading computational tasks to edge servers. Literature [13] established an offloading framework and proposes a multi-objective optimization problem to study computational offloading in VEC networks by jointly considering the offloading decision, communication and computational resource allocation with the objective of minimizing the delay and cost of computational offloading. In order to solve the above multi-objective optimization problems, particle swarm optimization algorithm is introduced, which effectively reduces the delay and cost. Literature [14] reduced system energy consumption by proposing a joint computation and caching framework based on a deep deterministic policy gradient algorithm to optimize the computation and caching resource allocation at edge nodes. Meanwhile, it satisfied the latency requirements of applications in IoV and improved the energy efficiency of the system. Literature [15] proposed an in-vehicle computing network based on cloud-side-end collaborative computing, constructs a model of cloud-side-end collaborative computing system for IoV, and proposes a computational offloading strategy based on a multi-strategy collaborative cyclic swarming algorithm and combining task prioritization and computational offloading node prediction. Literature [16] considered the optimization of VEC task offloading performance based on Actor-Critic deep reinforcement learning algorithm. A multi-vehicle assisted MEC system model is proposed that utilizes the Actor-Critic deep reinforcement learning technique to decide whether the task is to be executed in the local vehicle or offloaded to the roadside unit for execution in order to increase the algorithm convergence speed and to achieve the lowest latency system improvement.

Considering the limited resources of edge servers, the researchers further explored the possibility of utilizing the computing resources of idle vehicles. Literature [17] used vehicles' idle resources during congregation periods due to traffic lights or congestion in cities and proposed a task offloading scheme that relies only on V2V communication. By defining task performance as a min-max problem between a task and multiple computational vehicles, max-min fairness theory is utilized to optimize task execution time. Similarly, the literature [18] utilized peripheral vehicles that are temporarily idle in road computing resources and discusses on how to put the resources of these vehicles to use. For the task execution efficiency, a distributed multi-hop task offloading decision model is established, and neighboring vehicles within the wireless communication range are selected as candidate vehicles, lastly, the unloading problem is modeled as a generalized allocation model with constraints and solved by greedy and discrete bat algorithms, respectively. Furthermore, the literature [19] Propose joint task offloading and resource allocation scheme for underutilization of vehicle resources. Considering the scenarios of equipment, parked and moving vehicles, the system model with task attributes and vehicle service time is constructed to form an optimization problem. The GRIA and TSHA algorithms are used to solve the problem. After multi-scenario simulation, the proposed scheme outperforms the other three schemes, with GRIA obtaining the

optimal solution, and TSHA obtaining the near-optimal solution with low complexity and better convergence of the algorithms. Vehicle mobility was not considered in the above studies. For the challenges of frequent changes in network topology and unpredictable computational resources caused by vehicle mobility, a task offloading framework based on multi-armed gambling machines was introduced in literature [20]. The framework demonstrated a high degree of adaptability and predictability by predicting and learning the offloading performance of neighboring vehicles and minimizing task offloading latency when computational resources are sufficient. Due to the mobility of the vehicles, tasks will be migrated between RSUs. Literature [21] designed a mobile-aware multitasking migration and offloading scheme for IoV. To consider the coupling of migration and unloading, a joint migration and unloading optimization problem is proposed. The problem is decoupled into two sub-problems: the computational resource allocation problem and the vehicle node selection problem, and solved based on the linear relaxation of Lagrangian function and branch-and-bound algorithms, respectively, and finally the system performance is effectively improved. The above algorithms lack the flexibility for scenario expansion. For the resource allocation problems, Literature [22] established a stochastic traffic and dynamic network environment scenario, and researched the joint optimization of computational offloading and resource allocation to minimize the system cost of the processing tasks under the constraints of processing latency and transmission rate. In order to address the challenges posed by dynamic environments, Deep Reinforcement Learning (DRL) techniques are applied to deal with high-dimensional continuous state and action spaces. Then, a Computation Offloading and Resource Allocation (CORA) algorithm is designed, which can effectively learn the optimal solution by adapting to the network dynamics. In the field of research on vehicular edge computing, there have been many papers discussing the design of vehicular networked systems. However, the above methods fail to fully consider the multi-tasking scenarios in real problems and lack diversity in the types of optimized tasks, thus failing to prioritize high security level tasks in computing resource allocation. In addition, some of the studies do not synchronize consider of the radius of the communication between the computing vehicle and the edge server, thus limiting the effectiveness of the scheme in practical applications.

In the research field of edge computing for Internet of Vehicles, how to design a reasonable offloading strategy and effectively utilize the computing resources of edge nodes to reduce task processing time and system energy consumption is of great research significance. However, many existing researches are still insufficient in facing the problems of how to efficiently allocate resources and balance latency and energy consumption in dynamic environments. Meanwhile, most of the studies fail to fully consider the multi-tasking scenarios in real problems and lack diversity in the types of optimized tasks, hence failing to prioritize the tasks with high security levels in the allocation of computational resources.

Based on the above analysis formulation, this paper proposes an innovative offloading strategy combining Naive Bayesian and reinforcement learning techniques to study the task offloading problem under the edge base station-idle vehicle joint offloading model. Joint optimization of offloading decisions, bandwidth allocation, and computational resource allocation is aimed at reducing the overall execution latency of tasks in connected vehicle systems and decreasing the energy consumption of in-vehicle devices to maximize system utility. In addition, in order to reduce the iterative process of the algorithm, a Naive Bayesian algorithm was utilized for traffic identification and classification for the offloading task of the vehicle terminals, and the computational devices were preclassified on the basis of the radius of the communication between the vehicle and the edge devices.

The main contributions of this paper are summarized as follows:

- 1) Introducing idle vehicles as a computing resource, enabling them to collaborate with roadside units and vehicle end devices to solve the problem of insufficient computing resources of edge servers.
- 2) Classify the task traffic using a Naive Bayesian algorithm to classify the tasks into high-priority (security tasks) and low-priority (non-security tasks). And combining with edge caching technology, it prioritizes the allocation of computing resources for high-priority tasks. A pre-classification strategy is proposed to reduce the number of iterations of the algorithm by dividing the range of task offloadable devices.
- 3) The task offloading problem is modeled as a MDP, and the P-DQN algorithm is used to handle the problem in discrete and continuous action spaces, and the task offloading strategy is optimized through continuous learning and iteration to achieve the goal of reducing latency and energy consumption.

The rest of the paper is organized as follows: section II presents the associated system model. Section III formulates the problem. Section IV introduces the offload scheduling algorithm based on pre-classification and P-DQN. Section V shows the simulation results and analysis. Finally the conclusion of the paper is given in section VI.

Where $\varepsilon_{m,l}$ is the channel gain between the task vehicle m and the idle vehicle l , P_{mn} is the transmission power when the task vehicle m performs the unloading task Y_{mn} , B_u denotes the spectrum bandwidth available between the task vehicle and the idle vehicle, $B_{mn,u}$ is the bandwidth allocated to task Y_{mn} , α_{mn} denotes the percentage of bandwidth allocated to task Y_{mn} under this offloading scheme, N is the noise power spectral density, l is the relative distance, $l_{m,l}$ represents the actual distance between the two points, and ϖ denotes the path loss index.

2) Tasks are offloaded to the roadside unit:

The vehicle can establish communication with the roadside unit under this offloading scheme. The transmission rate of task Y_{mn} offloaded to the g th roadside unit RSU g can be expressed as

$$R_{mn,g}^r = B_{mn,r} \log_2 \left(1 + \frac{P_{mn} \varepsilon_{m,g} (l/l_{m,g})^\varpi}{NB_{mn,r}} \right). \quad (3)$$

$$B_{mn,r} = \frac{B_r}{S} \beta_{mn}. \quad (4)$$

Where $\varepsilon_{m,g}$ is the channel gain between the task vehicle m and the roadside unit g , $B_{mn,r}$ is the bandwidth allocated to the task Y_{mn} under this offloading scheme, $l_{m,g}$ is the actual distance between the task vehicle m and the roadside unit g , and β_{mn} denotes the percentage of bandwidth allocated to the task Y_{mn} under this offloading scheme.

2.2 Computational Model

Vehicle computation tasks can be processed in the local vehicle or offloaded to be executed on an idle vehicle or on a roadside unit. Next, these three computational models are described.

1) Local Vehicle Calculation: the delay incurred by the local vehicle in processing the computation task is the computation delay, denoted as

$$t_{mn,loc} = \frac{W_{mn}}{\eta_{mn,loc} f_{m,loc}}. \quad (5)$$

Where $f_{m,loc}$ is the computing power of the local vehicle m and $\eta_{mn,loc}$ is the percentage of computing resources allocated to the task offloaded to the local vehicle.

To define the energy consumption of a task to be processed in a local vehicle as $E_{mn,loc}$, it can be expressed as

$$E_{mn,loc} = zW_{mn} (\eta_{mn,loc} f_{m,loc})^2. \quad (6)$$

Where z is the conversion factor of the hardware device's capability into energy consumption [24].

2) Idle vehicle calculation: When a task chooses to offload to an idle vehicle, it incurs transmission delay and computation delay. Where the time it takes for the results to be transmitted back to the task vehicle after the task has been executed is neglected. The transmission delay for task offloading to an idle vehicle on the roadside can be defined as

$$t_{mn,tran}^u = \frac{L_{mn}}{R_{mn,l}^u}. \quad (7)$$

The computational delay for task offloading to idle vehicles on the roadside can be defined as

$$t_{mn,com}^u = \frac{W_{mn}}{\theta_{mn,l} J_l^u}. \quad (8)$$

Where f_I^u is the computational power (CPU cycles/sec) of the idle vehicle I, and $\theta_{mn,I}$ is the percentage of computational resources allocated to task Y_{mn} in the idle vehicle.

Therefore, the total delay in offloading the task from the local vehicle to the idle vehicle is

$$t_{mn}^u = \frac{L_{mn}}{R_{mn,I}^u} + \frac{W_{mn}}{\theta_{mn,I} f_I^u}. \quad (9)$$

The energy consumption generated by task offloading to the idle vehicle is a combination of three parts: the energy consumption of the task vehicle which is in an idle state while waiting for the task to be computed, the energy consumption of the task vehicle for upstream transmission of data, and the energy consumption of the idle vehicle for computation of the task. So, E_{mn}^u can be represented as

$$E_{mn}^u = P_{mn} \frac{L_{mn}}{R_{mn,I}^u} + Q_m \frac{W_{mn}}{\theta_{mn,I} f_I^u} + z W_{mn} (\theta_{mn,I} f_I^u)^2. \quad (10)$$

Where Q_m is the power of the idle state of the task vehicle.

3) Roadside unit calculations: The same transmission delay and computation delay are incurred when the task is selected to be offloaded to the roadside unit, and the transmission delay can be defined as

$$t_{mn,tran}^r = \frac{L_{mn}}{R_{mn,g}^r}. \quad (11)$$

The computational delay of task offloading to the roadside unit can be defined as

$$t_{mn,com}^r = \frac{W_{mn}}{\varphi_{mn,g} f_g^r}. \quad (12)$$

Where f_g^r is the total computational resource of the roadside unit g, and $\varphi_{mn,g}$ is the percentage of computational resources allocated to the task Y_{mn} in the roadside unit.

Therefore, the total delay in offloading the task from the local vehicle to the roadside unit is

$$t_{mn}^r = \frac{W_{mn}}{\varphi_{mn,g} f_g^r} + \frac{L_{mn}}{R_{mn,g}^r}. \quad (13)$$

In the system model considered in this paper, it is assumed that the energy of the roadside unit is not limited, and based on this setting, when tasks are offloaded to the roadside unit, the energy consumption incurred by the roadside unit in performing computational tasks is not considered. Therefore, the energy consumption for task offloading to the roadside unit is expressed as

$$E_{mn}^r = P_{mn} \frac{L_{mn}}{R_{mn,g}^r} + Q_m \frac{W_{mn}}{\varphi_{mn,g} f_g^r}. \quad (14)$$

2.3 System Utility

The utility of this paper is dominated by the task vehicle, and the main consideration is the benefit of the task vehicle. The system utility consists of two parts: delay and energy consumption, and the introduction of delay benefit coefficients and energy cost coefficients is aimed at achieving the purpose of balancing the delay and energy consumption magnitudes, so that the utility function describes the user's total benefit in a more comprehensive

way.

1) Time-delayed utility: using the coefficient of revenue μ_1, μ_2 , to describe the time-delayed utility coefficients H_t

$$H_t = \sum_{m=1}^S \sum_{n=1}^Q q \mu_1 (T_{mn}^{\max} - t_{mn}) + o \mu_2 (T_{mn}^{\max} - t_{mn}^s). \quad (15)$$

Where, t_{mn}^s denotes the security class task offloading delay, $o=1$ when the offloading task type is security class task, and vice versa, $o=0$; When the offloading task type is a non-security task, $q=1$ and vice versa $q=0$. $T_{mn}^{\max} - t_{mn}$ with $T_{mn}^{\max} - t_{mn}^s$ denotes the difference between the maximum delay and the actual delay of the unloading task, defined as the time consumed by the vehicle unloading task Y_{mn} less relative to the maximum tolerated delay of the unloading task. The larger the difference, the larger H_t is, meaning the better the delay utility. The types of tasks are categorized as non-security tasks and security tasks. Both their delay t_{mn} and t_{mn}^s can be expressed in terms of the delay t , which can be defined as

$$t = b_{mn} t_{mn,loc} + e_{mn} t_{mn}^u + d_{mn} t_{mn}^r. \quad (16)$$

2) Energy Efficiency: Using the cost coefficients $\gamma_1, \gamma_2, \gamma_3$, the energy utility function is expressed as follows

$$H_e = \sum_{m=1}^S \sum_{n=1}^Q b_{mn} \gamma_1 E_{mn,loc} + e_{mn} \gamma_2 E_{mn}^u + d_{mn} \gamma_3 E_{mn}^r. \quad (17)$$

The system utility function can be described by the difference between the system delay benefit and the energy cost. The difference between the benefit and the cost is the net benefit, therefore, the larger the system utility function, the better the system performance. The system utility function U can be expressed as

$$U = H_t - H_e. \quad (18)$$

3 Formulation of the Problem

$$(Z1) \quad \max_{e_{mn}, b_{mn}, d_{mn}, \varphi_{mn,g}, \theta_{mn,l}, \eta_{mn,loc}, \alpha_{mn}, \beta_{mn}, \forall l, g} U. \quad (19)$$

s.t.

$$\begin{aligned} C1: & e_{mn} + b_{mn} + d_{mn} = 1, \forall m \in S, \forall n \in Q, \\ C2: & t_{mn,loc}, t_{mn}^u, t_{mn}^r \leq T_{mn}^{\max}, \forall m \in S, \forall n \in Q, \\ C3: & t_{mn}^s \leq \min\{t_{mn,loc}, t_{mn}^u, t_{mn}^r\}, \forall m \in S, \forall n \in Q, \\ C4: & \sum_{m=1}^S \sum_{n=1}^Q b_{mn} \eta_{mn,local} \leq 1, \forall m \in S, \forall n \in Q, \\ C5: & \sum_{m=1}^S \sum_{n=1}^Q e_{mn} \theta_{mn,l} \leq 1, \forall m \in S, \forall n \in Q, \\ C6: & \sum_{m=1}^S \sum_{n=1}^Q d_{mn} \varphi_{mn,g} \leq 1, \forall m \in S, \forall n \in Q, \\ C7: & \sum_{m=1}^S \sum_{n=1}^Q \alpha_{mn} \leq 1, \forall m \in S, \forall n \in Q, \\ C8: & \sum_{m=1}^S \sum_{n=1}^Q \beta_{mn} \leq 1, \forall m \in S, \forall n \in Q. \end{aligned} \quad (20)$$

In the given constraint (20), constraint (C1) ensures that the task can choose only one of the three offloading schemes of local vehicles, idle vehicles or roadside units to perform the offloading operation. Constraint (C2) ensures that the task offloading delay under all three offloading scenarios does not exceed the maximum tolerable delay allowed for the task. Constraint (C3) ensures that when offloading a security task, the delay does not exceed the lowest delay of the three offloading policies. Constraints (C4-C6) ensure that the computational resources allocated to task Y_{mn} are within the capacity of the device's own total computational resources. Constraints (C7-C8) ensure that the bandwidth allocated to task Y_{mn} cannot exceed the bandwidth of the available spectrum between the task vehicle and the device.

The decision variable in problem (Z1) is binary and is a nonconvex problem. The optimization objective function has multiple local optimal solutions rather than one global optimal solution, so the problem can be transformed into a MDP and solved using DRL. DRL algorithms find locally optimal solutions in non-convex problems by using policy search, value iteration, and gradually approaching the global optimal solution through continuous iterative search.

4 Task Offload Scheduling Algorithm Based on Pre-classification and P-DQN

In this section, a combination of Naive Bayesian traffic classification as well as edge caching techniques are first used for pre-classification, and then a parameterized deep reinforcement learning algorithm is proposed for problem solving.

1) Pre-Classification Strategy

The pre-classification method mainly combines the Naive Bayesian traffic classification as well as the edge caching technique to classify the vehicle offloading traffic. Considering the vehicle communication radius as well as the driving speed, categorizing the task off-loadable servers in advance for the task off-loadable servers before the resource allocation step using reinforcement learning for the purpose of reducing the latency of the safety class tasks and compressing the action space of the neural network to reduce the algorithm iterations, the steps are as follows:

a. Perform traffic categorization. Naive Bayesian Traffic Classification is a network traffic classification technique based on the Naive Bayesian algorithm, which is mainly used to classify and identify the network traffic. This technique transforms the packet features in network traffic into vector form and classifies them using a Naive Bayesian algorithm. The basic idea is to determine the category to which each packet belongs by calculating the probability of each feature under different classifications according to Bayes' theorem, using a priori and conditional probabilities [25].

In a given sample set, the probability distribution of each category is known and the effects of the attributes on the categories are independent of each other. Consider a sample set containing n attributes: A_1, A_2, \dots, A_n . These attribute values form the feature vector of the sample. There exist m possible categories of $\{C_1, C_2, \dots, C_m\}$. For a specific sample X to be classified, its feature vector is $\{x_1, x_2, \dots, x_m\}$. In this paper, by calculating the probability $P(C_i|X)$ that X belongs to each category and using the category with the highest probability as the predicted classification of X [26].

From Bayes' formula (21):

$$P(C_x | X) = \frac{P(X | C_x)P(C_x)}{P(X)}. \quad (21)$$

In the simulation, the studied traffic classifications are all offline traffic and the problem of how to collect and classify the traffic from the vehicle side in real time is not considered. The methodology used in this paper is as follows: first simulate the vehicle unloading tasks and randomly assign different traffic feature attributes to each task, use the publicly available dataset Android-zoo to obtain different types of traffic, and classify the traffic using the Naive Bayesian algorithm. Generally traffic classification can be based on the following characteristics: source and destination MAC addresses, IP protocol version, destination port, TCP message length, application layer protocol used, and key fields in the protocol. However, with some new applications moving away from fixed and pre-known port numbers, many applications do not have port numbers assigned or registered by IANA, and instead simply use publicly known default ports, making it difficult for port number-based methods to accurately identify application-type traffic. So in this paper the MAC address, IP address, keywords of the protocol,

minimum, maximum, average length, mean square deviation and duration of the flow will be used for categorization.

In this paper, the offloaded task types are mainly assumed to be map navigation class and language communication class, and the recognized secure tasks are put into the set $Sa = \{T_{mn}^s\}$.

b. Divide the range of equipment that can be offloaded from the task. The two types of security tasks are randomly cached in three edge servers and are obtained directly from the edge device when the vehicle requests the content.

When dividing the range of task off-loadable devices, it is first determined what edge devices are in the offload range of the vehicle m based on the communication range of the roadside unit and the idle vehicle. Suppose the communication radius of the roadside unit is $R0$, the communication radius of the idle vehicle is $R1$, the initial coordinates of the task vehicle are (x, y) , the coordinates of the roadside unit are (x_0, y_0) , and the coordinates of the idle vehicle are (x_1, y_1) , which should satisfy the relationship of Eqs. (22) and Eqs. (23)

$$d_0 \leq R0. \quad (22)$$

$$d_1 \leq R1. \quad (23)$$

Where, $d_0 = \sqrt{(x-x_0)^2 + (y-y_0)^2}$, $d_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2}$.

The constraint is that the vehicle does not drive out of the communication radius of the edge device within the time delay used by the vehicle m to offload the task to the edge device, and suppose that the road is a straight section, and the task vehicle travels with the speed of v , then the conditions of Eq. (24) and Eq. (25) should be satisfied

$$l_0 \leq R0. \quad (24)$$

$$l_1 \leq R1. \quad (25)$$

Where, $l_0 = \sqrt{[(x+v \cdot t_{mn}^s) - x_0]^2 + [y - y_0]^2}$, $l_1 = \sqrt{[(x+v \cdot t_{mn}^{\max}) - x_1]^2 + [y - y_1]^2}$.

Next the vehicle m searches the off-loadable edge server and the local scope to see if any devices have cached security tasks. If it exists then determine if the cached task in the edge server is of the same type as the vehicle offload task. If only one edge server satisfies the condition, the decision variable for it is set directly; If more than one edge server meets the conditions, then those edge servers are taken into the offload range. At this time, the uplink delay of the task is regarded as 0, and the uplink transmission power is also set to 0, which is represented by Eq. (26) and Eq. (27), respectively

$$t_{mn}^s = b_{mn} t_{mn,loc} + e_{mn} t_{mn,com}^u + d_{mn} t_{mn,com}^r. \quad (26)$$

$$E_{mn}^u = Q_m \frac{W_{mn}}{\theta_{mn,l} f_l^u}. \quad (27)$$

$$E_{mn}^r = Q_m \frac{W_{mn}}{\varphi_{mn,g} f_g^r}. \quad (28)$$

In conclusion, the specific algorithm steps for pre-classification are shown in Fig. 2:

Inputs: roadside unit communication radius $R0$, idle vehicle communication radius $R1$, task vehicle initial coordinates (x, y) , roadside unit coordinates (x_0, y_0) , idle vehicle coordinates (x_1, y_1) , task vehicle running speed v .

Output: set of off-loadable ranges for task Y_{mn} , decision variables for a subset of safety class tasks.

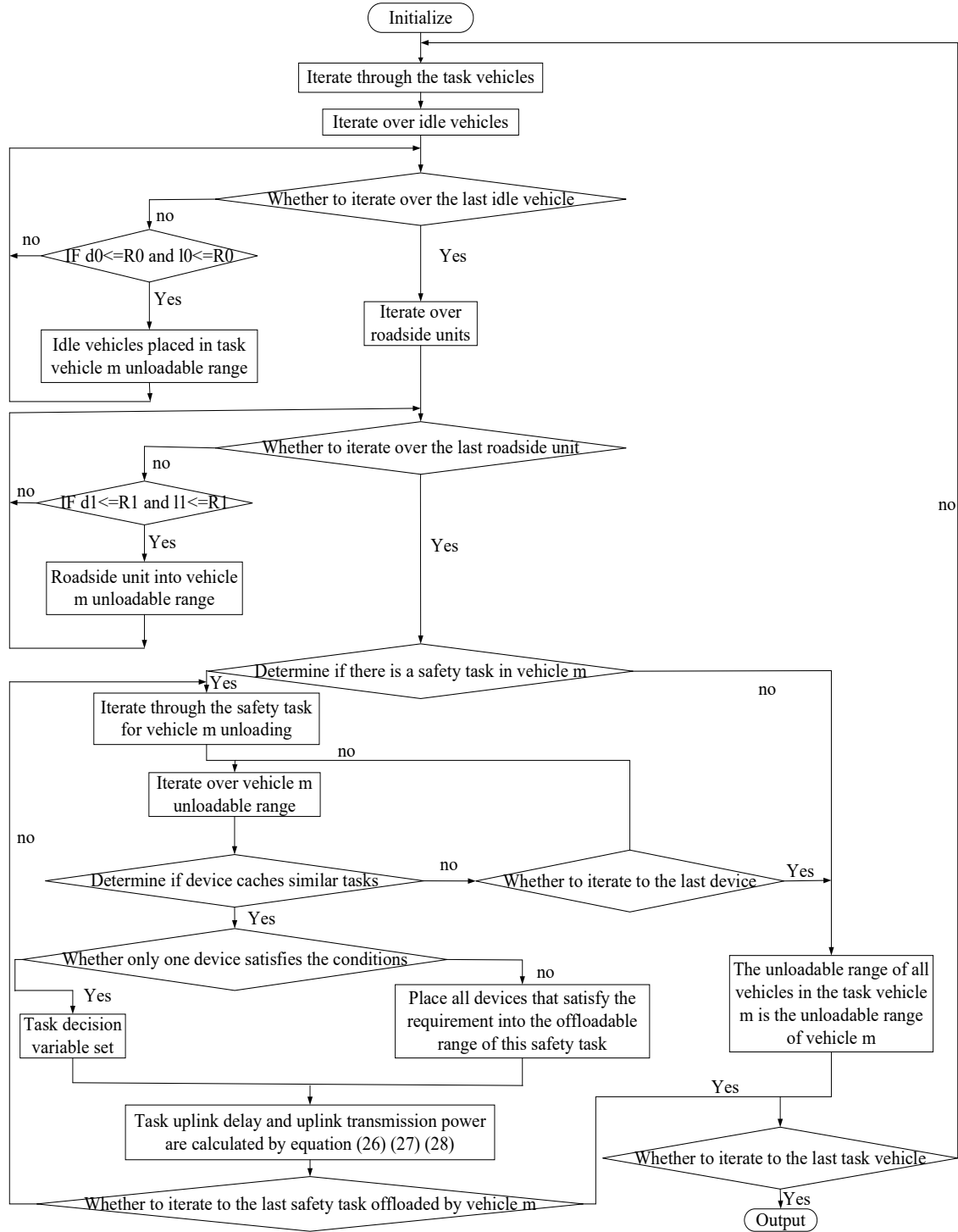


Fig. 2. Pre-categorization strategy

2) Resource allocation strategies based on reinforcement learning

A. States, Action Spaces, and Rewards

1) States: The main states are task Y_{mn} offload scheduling, bandwidth allocation, task Y_{mn} off-loadable range device set $U_j \in \{1, 2, 3, \dots, J(mn)\}$, $r_g \in \{1, 2, 3, \dots, K(mn)\}$ and device compute resource utilization

$$s = \left[\begin{array}{l} R_{mn,1}^u, R_{mn,2}^u, \dots, R_{mn,J(mn)}^u, R_{mn,1}^r, R_{mn,2}^r, \dots, R_{mn,K(mn)}^r, \\ f_1^u, f_2^u, \dots, f_{J(mn)}^u, f_1^r, f_2^r, \dots, f_{K(mn)}^r \end{array} \right], \text{ location information and off-loadable device location}$$

information. Where, $R_{mn,J(mn)}^u, R_{mn,K(mn)}^r$ is the rate at which the task offloads to idle vehicles and roadside units task vehicles within the off-loadable range, and $f_{J(mn)}^u, f_{K(mn)}^r$ is the computational capacity of idle vehicles and roadside units within the off-loadable range of the task.

2) Action: When performing task offloading, it is important to determine the appropriate offloading strategy. Tasks can choose one of three different offloading strategies to perform the task offloading process. This paper also determines both the percentage of bandwidth allocated to each vehicle offloading task Y_{mn} under the three different offloading scenarios as well as the percentage of computing resources allocated to task Y_{mn} by the devices in the three different offloading scenarios.

3) Rewards: With the optimization goal of minimizing latency and energy consumption. In reinforcement learning, states, actions, and rewards are a constantly looping triad, with different rewards for actions produced in each state, and so on until the end state or maximum number of iterations is reached. For each step, the intelligent body receives a reward r after performing an action a in a certain state s . In this paper, the reward is defined as the product of the system's utility function and the coefficient of return τ .

$$r = \tau \cdot U. \quad (29)$$

B. P-DQN based task offloading algorithm

1) Markov Decision Process

The vehicle task offloading problem can be formulated as a MDP. A basic MDP can be represented by (S, A, P) , with S denoting the state, A the action, and P the state transfer probability, that is, the probability of transferring to s_{t+1} based on the current state s_t and a_t . Introducing the Bellman equation to illustrate the relationship between the value function of the current state and the value function of the next state

$$\Omega(s(t)) = E[re_{t+1} + \zeta\Omega(S(t+1) | S(t) = s)]. \quad (30)$$

Where $S(t)$ is defined as the state at moment t and ζ is defined as the discount factor.

Thus one can define an action value function, defined as the cumulative reward for using the strategy π after executing the action a , starting from the state s .

$$Q^\pi(s, a) = E_x[re + \zeta Q^\pi(s', a') | s, a]. \quad (31)$$

Next, find the optimal value function for state s . Finding the optimal policy is equivalent to solving the optimal value function.

$$\Omega^x(s) = \max_a E_x[re + \zeta\Omega^x(s') | s, a] \quad (32)$$

2) P-DQN based allocation algorithm

P-DQN can be viewed as extending DQN to hybrid spaces by first defining a deterministic function for mapping the state and each discrete action to continuous parameters. Next, an action-value function is defined for mapping states and finite hybrid actions to real values, with the lower continuous parameter being optional and different discrete actions can have the same continuous parameter between them [27].

First define the set of actions $v = [b_{mn}, d_{mn}, e_{mn}]$, the set $[V]$ is discrete, from which one is chosen as the upper discrete action, followed by a continuous lower parameter,

$$x_v = \left[\begin{array}{l} \varphi_{mn,1}, \varphi_{mn,2}, \dots, \varphi_{mn,K(mn)}, \\ \theta_{mn,1}, \theta_{mn,2}, \dots, \theta_{mn,J(mn)}, \alpha_{mn,1}, \alpha_{mn,2}, \dots, \alpha_{mn,J(mn)}, \\ \beta_{mn,1}, \beta_{mn,2}, \dots, \beta_{mn,K(mn)}, \eta_{mn,loc} \end{array} \right], x_v \in X_v. \text{ This parameter is related to the previously se-}$$

lected discrete action. X_v is a discrete set containing all $v \in [V]$ correspondences, which can be constituted as an action space as

$$\Delta = \{(v, x_v \mid x_v \in X_v)\}. \quad (33)$$

Consider an MDP with an action space with $Q = (s, a) = Q(s, x_v, v)$ for $a \in A$. The Bellman equation can be rewritten as

$$Q(x_t, x_v, v) = E_x [re + \zeta \max_{v \in [V], x_v \in X_v} \sup Q(x_{t+1}, x_v, v) \mid x_{t+1} = x, a_t = (v_t, x_{v_t}^*)]. \quad (34)$$

Where, v_t is defined as the discrete action chosen at moment t and is the associated continuous parameter.

Calculate $x_v^* = \arg \sup_{x_v \in X_v} Q(x_{t+1}, x_v, v)$ for each $v \in [V]$, followed by choosing the maximal $Q(x_{t+1}, v, x_v^*)$.

As with DQN, a deep neural network $Q(s, x_v, v; w)$ is used to approximate $Q(s, x_v, v)$. When w is fixed, for each $v \in [V]$, it is desirable to find a set of θ satisfying

$$Q(s, x_v(x; \theta), v; w) \approx \sup_{x_v \in X_v} Q(x, x_v, v; w). \quad (35)$$

When $n \geq 1$, define the n -step objective y_t as

$$y_t = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{v \in [V]} Q(s_{t+n}, v, x_v(x_{t+n}, \theta_t); w_t). \quad (36)$$

Where, θ_t, w_t is the value network deterministic strategy network parameter at moment t .

In summary, this paper proposes the FPPDQ (Flow Preprocessing with Parametrized Deep Q-Networks) algorithm after using a pre-classification strategy combined with the P-DQN algorithm, and the specific steps are shown in Table 1.

Table 1. Algorithm steps

Algorithm. FPPDQ (Flow Preprocessing with Parametrized Deep Q-Networks)

Input: Combine the pre-classification phase to obtain the state set S , the discrete action set v , and the continuous action set x_v , Step $\{\alpha t, \beta t\} t \geq 0$, explore step ϵ , small batch size B , probability distribution, initialize network weights w_t, θ_t .

For $t = 1, 2, \dots, T$.

Calculate action parameters $x_v \leftarrow x_v(x_t, \theta_t)$.

Selection of actions based on state and greedy strategy $a_t = (v_t, x_{v_t})$,

$$a_t = \begin{cases} \text{Sampling from distribution } \zeta \text{ according to probability } \epsilon \\ \text{Choose with probability } 1 - \epsilon \text{ such that } v_t = \arg \max_{v \in [V]} Q(x_t, x_v, v; w_t) \end{cases}$$

Execute action a_t , observe reward r_t and next state s_{t+1} .

Store the transformations $[s_t, a_t, r_t, s_{t+1}]$ into the experience playback buffer D . Sample B transitions $[s_b, a_b, r_b, s_{b+1}]$ at random from D .

Define the goal y_b :

$$y_b = \begin{cases} r_b \\ r_b + \max_{v \in [V]} \gamma Q(s_{b+1}, x_v(s_{b+1}, \theta_t, V; w_t)) \end{cases}$$

Use the data $\{y_b, s_b, a_b\}, b \in [B]$ to compute the stochastic ladder $\nabla_w \ell_t^Q(w)$ and $\nabla_{\theta} \ell_t^Q(\theta_t)$.

Update weights $w_{t+1} \leftarrow w_t - \alpha_t \nabla_w \ell_t^Q(w_t)$ and $\theta_{t+1} \leftarrow \theta_t - \beta_t \nabla_{\theta} \ell_t^Q(\theta_t)$.

End for

5 Analysis of Simulation Results

In order to verify the performance of the offloading scheme proposed in this paper, a $3 \times 0.03\text{km}^2$ road is set up with four idle vehicles randomly parked on both sides of the road, and four RSUs are randomly distributed on both sides of the road, in which the number of task vehicles that are traveling varies between 5 and 9, and the number of tasks to be offloaded per vehicle varies between 10 and 50. The specific parameters were taken from the literature [28, 29] and are shown in Table 2.

Table 2. Simulation parameter settings

Symbolic	Value	Definition
T_{mn}^{\max}	1	Maximum delay in task execution (s)
W_{mn}	[900, 110]	Amount of computing resources/number of CPU cycles for computing tasks
L_{mn}	[300, 50]	Task data size(kbit)
f_m	0.1	Local computing capacity (GHz)
f_l^u	20	Idle vehicle computing capability (GHz)
f_g^r	100	Roadside unit computing capability (GHz)
B_u	5	Spectrum bandwidth between task vehicles and idle vehicles (MHz)
B_r	10	Spectral bandwidth between task vehicle and roadside unit (MHz)
P_{mn}	(200, 500)	Task uplink transmission data power (mW)
N	-174	Ambient Gaussian white noise power density (dBm/Hz)
$\varepsilon_{m,l}, \varepsilon_{m,g}$	10	Channel Gain (dB)
Q_m	100	Power of the task vehicle in idle state (mW)
z	10^{-27}	Conversion factor of the capacity of hardware equipment into energy consumption
l	1	Relative distance (m)
ϖ	2	Path loss index

In this paper, the FPPDQ algorithm is used to compare with the all offloading to curb cell method, the randomized algorithm, and the P-DQN algorithm. In offloading using the stochastic algorithm first one of the three types of offloading targets is randomly selected and then the computational and bandwidth resource percentages are randomly generated within a uniform distribution [0,1].

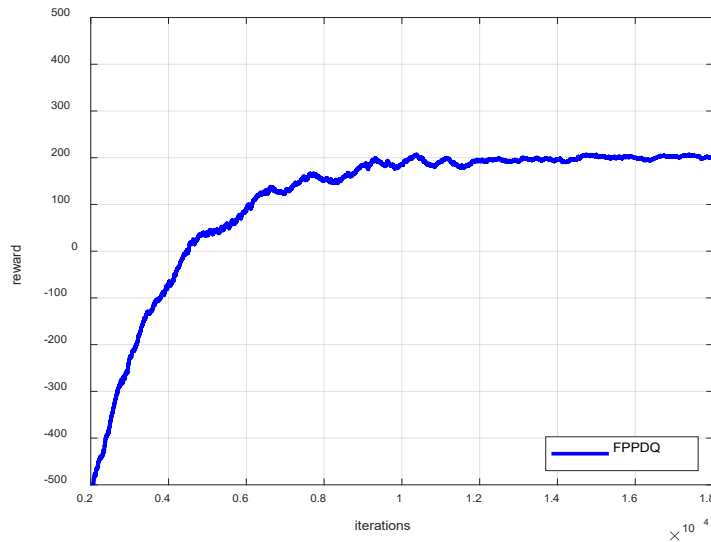


Fig. 3. Convergence of FPPDQ algorithm

The simulation first analyzes the trend of reward change of FPPDQ algorithm during the iteration process. It can be seen from Fig. 3 that the reward value is lower in the initial stage. As the number of iterations increases, the reward value increases rapidly, indicating that the algorithm is learning and adapting to the environment quickly. After the number of iterations reaches about 0.6×10^4 , the growth trend slows down and enters a fluctuating upward phase; After more than 1×10^4 iterations, the reward value stabilizes, indicating that the algorithm has converged and found a relatively stable strategy. Therefore, Fig. 3 shows that the FPPDQ algorithm has good convergence and stability during the training process. Through the introduction of a pre-classification strategy, the algorithm is able to learn rapidly in the early stages of training and maintain a stable performance in the later stages. This fast convergence and stability features make the FPPDQ algorithm highly efficient and reliable in practical applications.

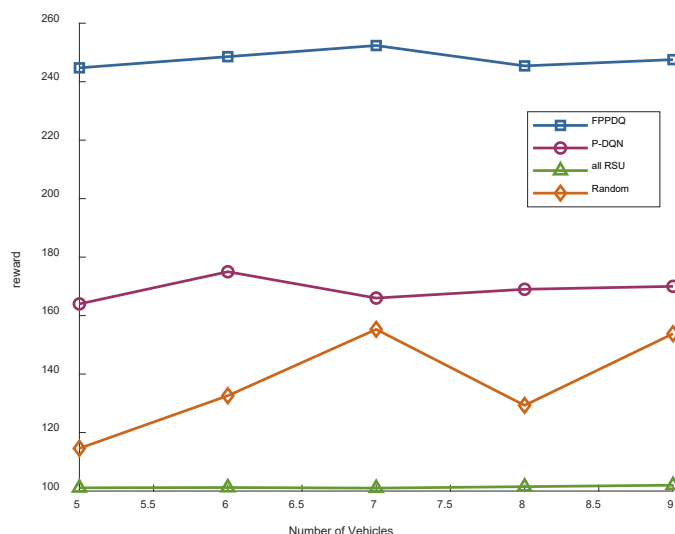


Fig. 4. Incentives versus number of vehicles

Fig. 4 shows the changes of reward values for each algorithm for different number of vehicles in the task. It can be seen that the reward values of all four methods change as the number of vehicles on the task increases, but the FPPDQ algorithm consistently performs better than the other algorithms. The P-DQN algorithm shows an increasing, then decreasing, then steady trend, with the reward value reaching a peak at a vehicle number of 6 and then dropping back, indicating that it is poorly adapted. The utility of the randomized algorithm is unstable and the reward value fluctuates greatly, indicating that it is random in resource allocation and cannot effectively match the task demand. The utility of the all RSU method is almost stable with the increase in the number of vehicles, which is due to the fact that the computational resources of the roadside unit are no longer sufficient to satisfy the minimum delay requirement for each task when the number of vehicles is 5, and the energy consumption for offloading to the roadside unit is negligible, so the system’s utility is little affected by the number of vehicles. There is a decrease in the gain of the FPPDQ algorithm when the number of vehicles is 8, but it is still much greater than the other three methods. This is due to the fact that the system resources are limited with a constant number of edge servers, and the local consumption and system latency also increase with the number of vehicles, and thus the overall gain decreases. The division of the vehicle unloadable range in the pre-classification step reduces the number of iterations of reinforcement learning and improves the efficiency of the algorithm, which enables the FPPDQ algorithm to maintain a high reward value even when the number of vehicles increases.

Fig. 5 shows the latency of the four methods as the task vehicle varies. It can be seen that the latency of the FPPDQ algorithm exhibits slight fluctuations as the number of vehicles increases. When the number of vehicles increases from 5 to 7, the delay decreases slightly, indicating that FPPDQ is able to make better use of the surrounding computational resources to optimize task allocation and reduce delay within a certain range. However, the latency of the FPPDQ rises slightly when the number of vehicles reaches 8. This is due to system resources

beginning to approach saturation, resulting in a slight increase in latency as more tasks are processed. The delay change of P-DQN algorithm is relatively stable, but the overall delay is higher than that of FPPDQ algorithm, indicating that its ability of optimizing resource allocation is slightly lower than that of FPPDQ algorithm. The delay performance of the randomized algorithm shows large fluctuations, especially when the number of vehicles increases from 5 to 7, and the delay decreases rapidly. The latency then reaches a minimum when the number of vehicles is 8, but rises again when the number of vehicles on the task is 9. This instability reflects the stochastic nature of the randomized algorithm in resource allocation, which fails to effectively match the task requirements. The delay of the all RSU method remains almost constant with the increase in the number of vehicles. This is because the all RSU method relies on roadside units for centralized computation, and when the number of vehicles increases to a certain number, the resources of RSUs gradually approach saturation, resulting in latency basically stabilizing at a high level. Overall, the FPPDQ algorithm fluctuates slightly in delay as the number of vehicles increases, but still performs better than the other algorithms in general, showing better dynamic adaptability and resource utilization efficiency.

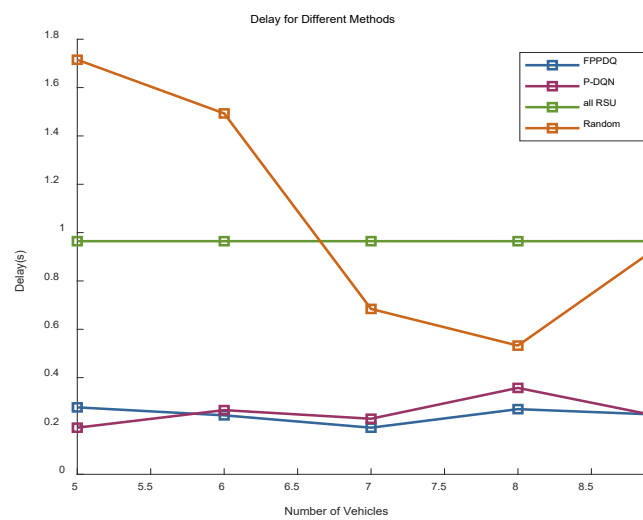


Fig. 5. Delay versus number of vehicles

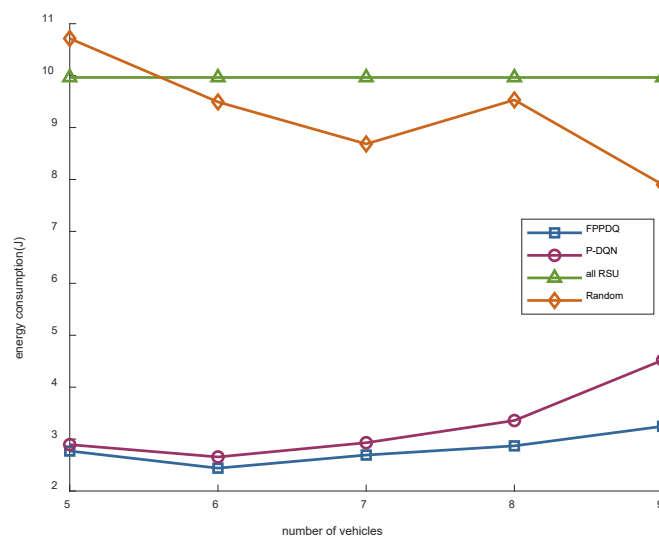


Fig. 6. Energy consumption versus number of vehicles

Fig. 6 shows the energy consumption of the four methods with the change of task vehicles. It can be seen that the energy consumption of all four methods varies as the number of vehicles increases, but the FPPDQ algorithm always consumes lower energy. P-DQN energy consumption varies more steadily, but the overall energy consumption is higher than the FPPDQ algorithm, indicating that it is poorly optimized for resource allocation. The randomized algorithm has higher energy consumption when the number of vehicles is 5 and varies significantly with the number of vehicles on the task, indicating that its randomness in resource allocation leads to unstable energy consumption. The energy consumption of all offloading to the RSUs remains unchanged, due to the fact that all computations are performed by the roadside units and the energy consumption consumed by the roadside units is negligible. Through the pre-classification strategy, the FPPDQ algorithm reduces the computing energy consumption of security tasks offloading to itself or edge servers, so as to reduce the energy consumption of the system as a whole. At the same time, the FPPDQ algorithm continues to learn and optimize through deep reinforcement learning, and dynamically adjusts the unloading strategy, so that the energy consumption can be kept low when the number of vehicles changes.

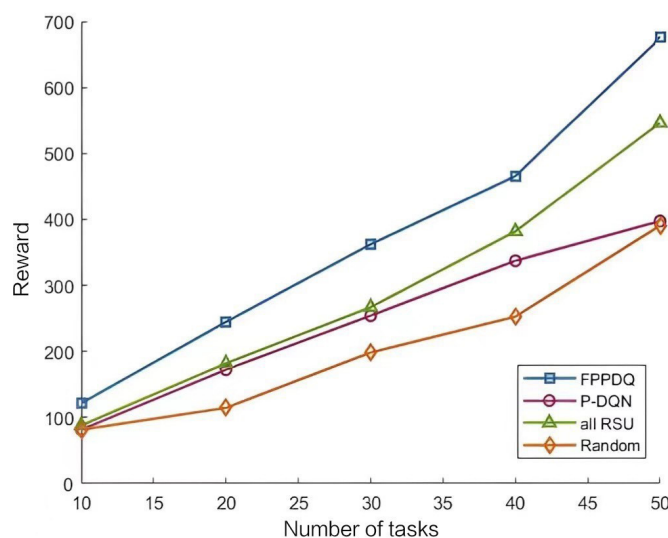


Fig. 7. Rewards versus number of tasks

Fig. 7 shows the reward value variation of the four methods with different number of tasks. With the increasing number of tasks, the reward values of all four methods varied, but the FPPDQ algorithm always had a higher reward value. The reward value of P-DQN increases with the increase of the number of tasks, but the growth rate is slow, indicating that it has poor adaptability and cannot effectively cope with the increase of the number of tasks. The reward fluctuation of Random strategy is large, which indicates that it has randomness in resource allocation and cannot effectively match task requirements. The reward value of all RSU is higher than P-DQN and Random, indicating that it can complete the task well under sufficient resources, but it is lower than FPPDQ in general. FPPDQ algorithm can maintain high reward value under different number of tasks, especially when the number of tasks is large, and its adaptability and global optimization ability make it perform better than other methods. The combination of pre-classification strategy and deep reinforcement learning enables the FPPDQ algorithm to efficiently allocate tasks and optimize resource utilization even when the number of tasks increases, thus performing well in complex vehicle networking scenarios.

6 Conclusion

In this paper, the problem of efficient task allocation and offloading is investigated under a joint edge base station-idle vehicle system model, where the idle vehicle is used as a computational device to work together with

the roadside unit and the vehicle terminal device. Meanwhile, considering the traffic identification and classification of offloading traffic for vehicle terminal devices and the communication radius between task vehicles and edge devices, and proposing an optimization problem aiming at reducing the offloading delay and decreasing the energy consumption of the system. Through experimental validation, the method in this paper optimizes the system utility compared to other algorithms in the same scenario and further reduces the offloading delay for security problems.

In the future, we will further expand the optimization objectives and introduce a multi-objective optimization framework to achieve a finer tradeoff between delay, energy consumption, resource utilization and task priority. At the same time, the complexity of the algorithm is further optimized, and the computational overhead of the algorithm is reduced by introducing sparse representation and dimensionality reduction techniques, so as to improve its applicability in large-scale vehicle networking scenarios.

7 Acknowledgement

This work was supported by Natural Science Foundation of Fujian Province (Grant No. 2022J011276, and No. 2023I0044), Undergraduate Education and Teaching Research Project of Fujian Province (Grant No. FBJY20240120), High-level Talent Project of Xiamen University of Technology (Grant No. YKJ22030R, and No. YKJ23034R), and Postgraduate Science and Technology Innovation Project of Xiamen University of Technology (Grant No. YKJCX2024143).

References

- [1] Y. Lin, P. Wang, M. Ma, Intelligent transportation system (ITS): Concept, challenge and opportunity, in: Proc. 2017 IEEE 3rd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2017. <https://doi.org/10.1109/BigDataSecurity.2017.50>
- [2] F.A. Silva, S. Kosta, M. Rodrigues, D. Oliveira, T. Maciel, A. Mei, Mobile cloud performance evaluation using stochastic models, *IEEE Transactions on Mobile Computing* 17(5)(2017) 1134-1147. <https://doi.org/10.1109/TMC.2017.2749577>
- [3] M.R. Rahimi, N. Venkatasubramanian, A.V. Vasilakos, MuSIC: Mobility-aware optimal service allocation in mobile cloud computing, in: Proc. IEEE sixth international conference on cloud computing, 2013. <https://doi.org/10.1109/CLOUD.2013.100>
- [4] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, F. Giust, Mobile-edge computing architecture: The role of MEC in the Internet of Things, *IEEE Consumer Electronics Magazine* 5(4)(2016) 84-91. <https://doi.org/10.1109/MCE.2016.2590118>
- [5] H. Lin, S. Zeadally, Z. Chen, H. Labiod, L. Wang, A survey on computation offloading modeling for edge computing, *Journal of Network and Computer Applications* 169(2020) 102781. <https://doi.org/10.1016/j.jnca.2020.102781>
- [6] Y. Zha, Y. Yuan, J. Li, H. Liu, Z. Chen, J. Li, A task offloading algorithm in mobile edge cloud computing, *Computer Applications and Software* 37(6)(2020) 135-141. <https://doi.org/10.3969/j.issn.1000-386x.2020.06.025>
- [7] H. Gao, X. Wang, W. Wei, A. Al-Dulaimi, Y. Xu, Com-DDPG: Task Offloading Based on Multiagent Reinforcement Learning for Information-Communication-Enhanced Mobile Edge Computing in the Internet of Vehicles, *IEEE Transactions on Vehicular Technology* 73(1)(2024) 348-361. <https://doi.org/10.1109/TVT.2023.3309321>
- [8] J. Liu, J. Wan, D.Y. Jia, B. Zeng, D. Li, C.-H. Hsu, High-efficiency urban traffic management in context-aware computing and 5G communication, *IEEE Communications Magazine* 55(1)(2017) 34-40. <https://doi.org/10.1109/MCOM.2017.1600371CM>
- [9] L. Liu, C. Chen, Q. Pei, S. Maharjan, Y. Zhang, Vehicular edge computing and networking: A survey, *Mobile networks and applications* 26(2021) 1145-1168. <https://doi.org/10.1007/s11036-020-01624-1>
- [10] X. Shen, L. Wang, P. Zhang, X. Xie, Y. Chen, S. Lu, Computing Resource Allocation Strategy Based on Cloud-Edge Cluster Collaboration in Internet of Vehicles, *IEEE Access* 12(2024) 10790-10803. <https://doi.org/10.1109/ACCESS.2023.3349029>
- [11] D. Xue, Task offload optimization management of networked vehicles in edge computing environment, in: Proc. 2022 2nd International Signal Processing, Communications and Engineering Management Conference (ISPCEM), 2022.

- <https://doi.org/10.1109/ISPCEM57418.2022.00014>
- [12] C. Li, S. Wang, X. Huang, X. Li, R. Yu, F. Zhao, Parked vehicular computing for energy-efficient internet of vehicles: A contract theoretic approach, *IEEE Internet of Things Journal* 6(4)(2019) 6079-6088.
<https://doi.org/10.1109/IJOT.2018.2869892>
- [13] Q. Luo, C. Li, T.H. Luan, W. Shi, Minimizing the delay and cost of computation offloading for vehicular edge computing, *IEEE Transactions on Services Computing* 15(5)(2022) 2897-2909.
<https://doi.org/10.1109/TSC.2021.3064579>
- [14] X. Kong, G. Duan, M. Hou, G. Shen, H. Wang, X. Yan, Deep reinforcement learning-based energy-efficient edge computing for internet of vehicles, *IEEE Transactions on Industrial Informatics* 18(9)(2022) 6308-6316.
<https://doi.org/10.1109/TII.2022.3155162>
- [15] Q. Xu, G. Zhang, J. Wang, Research on cloud-edge-end collaborative computing offloading strategy in the internet of vehicles based on the M-TSA algorithm, *Sensors* 23(10)(2023) 4682.
<https://doi.org/10.3390/s23104682>
- [16] B. Wang, L. Liu, J. Wang, Actor-critic based DRL algorithm for task offloading performance optimization in vehicle edge computing, in: *Proc. 2023 International Wireless Communications and Mobile Computing (IWCMC), 2023*.
<https://doi.org/10.1109/IWCMC58020.2023.10183228>
- [17] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks, *IEEE Access* 8(2020) 18863-18873.
<https://doi.org/10.1109/ACCESS.2020.2968465>
- [18] C. Chen, Y. Zeng, H. Li, Y. Liu, S. Wan, A multihop task offloading decision model in MEC-enabled internet of vehicles, *IEEE Internet of Things Journal* 10(4)(2023) 3215-3230.
<https://doi.org/10.1109/IJOT.2022.3143529>
- [19] W. Fan, J. Liu, M. Hua, F. Wu, Y. Liu, Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles, *IEEE Transactions on Vehicular Technology* 71(5)(2022) 5314-5330.
<https://doi.org/10.1109/TVT.2022.3149937>
- [20] G. Qiao, S. Leng, K. Zhang, Y. He, Collaborative task offloading in vehicular edge multi-access networks, *IEEE Communications Magazine* 56(8)(2018) 48-54.
<https://doi.org/10.1109/MCOM.2018.1701130>
- [21] X. Li, J. Tang, Y. Xu, Y. Sun, Mobility-aware multi-task migration and offloading scheme for Internet of Vehicles, *Chinese Journal of Electronics* 32(6)(2023) 1192-1202.
<https://doi.org/10.23919/cje.2022.00.333>
- [22] J. Huang, J. Wan, B. Lv, Q. Ye, Y. Chen, Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning, *IEEE Systems Journal* 17(2)(2023) 2500-2511.
<https://doi.org/10.1109/JSYST.2023.3249217>
- [23] Y. Yang, C. Feng, Deep Reinforcement Learning-Based Multi-Task Offloading Strategy in Mobile Edge Computing, in: *Proc. 2023 5th International Conference on Frontiers Technology of Information and Computer (ICFTIC), 2023*.
<https://doi.org/10.1109/ICFTIC59930.2023.10456120>
- [24] Y. Wen, W. Zhang, H. Luo, Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones, in: *Proc. 2012 proceedings IEEE Infocom, 2012*.
<https://doi.org/10.1109/INFCOM.2012.6195685>
- [25] A.W. Moore, D. Zuev, Internet traffic classification using bayesian analysis techniques, in: *Proc. of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2005*.
<https://doi.org/10.1145/1064212.1064220>
- [26] H. Linghu, M. Chen, H. Wang, Y. Lou, Bayesian network intrusion detection method based on credibility of mutual information, *Computer Engineering and Design* 30(14)(2009) 3288-3290.
- [27] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, H. Liu, Parametrized Deep Q-Networks Learning: Reinforcement Learning with Discrete-Continuous Hybrid Action Space. <<https://arxiv.org/abs/1810.06394>>, 2018 (accessed 14.09.2024).
- [28] Y. Liu, H. Yu, S. Xie, Y. Zhang, Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks, *IEEE Transactions on Vehicular Technology* 68(11)(2019) 11158-11168.
<https://doi.org/10.1109/TVT.2019.2935450>
- [29] J. Li, H. Gao, T. Lv, Y. Lu, Deep reinforcement learning based computation offloading and resource allocation for MEC, in: *Proc. 2018 IEEE wireless communications and networking conference (WCNC), 2018*.
<https://doi.org/10.1109/WCNC.2018.8377343>