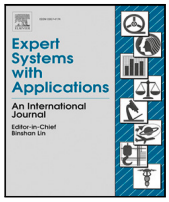
Expert Systems With Applications 249 (2024) 123622

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Deadline-aware task offloading in vehicular networks using deep reinforcement learning

Mina Khoshbazm Farimani [a], Soroush Karimian-Aliabadi [b], Reza Entezari-Maleki [a,c,d,*], Bernhard Egger [e], Leonel Sousa [d]

[a] School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran
[b] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
[c] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran
[d] INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
[e] Department of Computer Science and Engineering, Seoul National University, Seoul, South Korea

ARTICLE INFO

Keywords:
Computation offloading
Vehicular edge computing
Deep reinforcement learning
Deep Q-learning
Internet of vehicles

ABSTRACT

Smart vehicles have a rising demand for computation resources, and recently vehicular edge computing has been recognized as an effective solution. Edge servers deployed in roadside units are capable of accomplishing tasks beyond the capacity which is embedded inside the vehicles. However, the main challenge is to carefully select the tasks to be offloaded considering the deadlines, and in order to reduce energy consumption, while delivering a good performance. In this paper, we consider a vehicular edge computing network in which multiple cars are moving at non-constant speed and produce tasks at each time slot. Then, we propose a task offloading algorithm, aware of the vehicle's direction, based on Rainbow, a deep Q-learning algorithm combining several independent improvements to the deep Q-network algorithm. This is to overcome the conventional limits and to reach an optimal offloading policy, by effectively incorporating the computation resources of edge servers to jointly minimize average delay and energy consumption. Real-world traffic data is used to evaluate the performance of the proposed approach compared to other algorithms, in particular deep Q-network, double deep Q-network, and deep recurrent Q-network. Results of the experiments show an average reduction of 18% and 15% in energy consumption and delay, respectively, when using the proposed Rainbow deep Q-network based algorithm in comparison to the state-of-the-art. Moreover, the stability and convergence of the learning process have significantly improved by adopting the Rainbow algorithm.

1. Introduction

The future is always pictured with modern and fictional cars which are more like robots than classic transportation means. This image is today somehow realized with autonomous vehicles, and we are close to the futuristic transportation style more than ever. The market of the Internet of Vehicles (IoV) was estimated to correspond to $96 billion in 2021, and is approximately predicted to grow to $370 billion in 2028 (FortuneBusinessInsights, 2020). This remarkable investment justifies the ever-growing research attention shifted toward vehicular networks. A significant number of advances are achieved every year in the fields of auto navigation, vehicular augmented reality, autonomous driving, and intelligent object recognition (Chen et al., 2023; Morra et al., 2019; Xu et al., 2022). All these new applications are pushing the available infrastructures to their limits. They require Quality of Service (QoS), while at the same time executing complex tasks on massive data captured by sensors and communicating with the network.

Strict delay constraints are common in the above-mentioned applications, and this results in high energy consumption. However, autonomous vehicles are powered by batteries, and therefore, vehicles experience excessive pressure on the onboard resources. In order to relieve the conflict between application demands and resource-constrained vehicles, Vehicular Edge Computing (VEC) has emerged as a promising computing paradigm (Hu et al., 2019; Sun et al., 2019; Zhang, Guo, Liu, & Zhang, 2019). With the integration of Mobile Edge Computing (MEC) and vehicular networks, Vehicular Edge Computing (VEC) supports IoV by enabling the offloading of resource-intensive tasks onto MEC servers deployed on Road Side Units (RSU) via wireless networks (Liu et al., 2020). MEC servers can alleviate computation

* Corresponding author at: School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran.
E-mail addresses: m_khoshbazmfarimani@comp.iust.ac.ir (M.K. Farimani), skarimian@ce.sharif.edu (S. Karimian-Aliabadi), entezari@iust.ac.ir (R. Entezari-Maleki), bernhard@csap.snu.ac.kr (B. Egger), las@inesc-id.pt (L. Sousa).

https://doi.org/10.1016/j.eswa.2024.123622
Received 29 May 2023; Received in revised form 11 October 2023; Accepted 6 March 2024
Available online 10 March 2024

pressure on the vehicle's onboard resources and reduce the processing delay of these applications by pushing the computation resources in proximity to vehicles. As a result, task offloading has recently received increasing attention in VEC (Chen et al., 2022; Zhang et al., 2020; Zhao et al., 2022). However, due to the unique characteristics of vehicular networks, especially the high mobility of nodes and dynamic channel properties, designing an efficient edge-based task offloading scheme is quite challenging.

There have been many efforts devoted to the task offloading in VEC to seek solutions to the above issues and meeting the various performance requirements. Most of them have employed the Q-learning algorithm, a model-free Reinforcement Learning (RL) method, as a potential solution to solve offloading problems (Alchalabi et al., 2021; Jiang et al., 2020). However, they are slow in a huge state space. The state space for vehicular task offloading environment can be vast and high-dimensional due to the numerous factors involved. To address issues in Q-learning methods, Deep Q-Network (DQN) algorithms, which approximate the Q-function using deep neural networks, have emerged to prove their efficiency in solving offloading decision problems with huge dimensional state and action spaces, where the agent can adapt well to complex environments via continuous interaction (Liu et al., 2019; Xu et al., 2022). However, the major challenge in traditional DQN algorithms is the trend to overestimation which led to unstable training and low-quality policy. As an alternative, Double DQN was proposed (Hasselt et al., 2016), which has two identical neural network models, where one network is used to select the best action and the other is used to estimate the value of that action. It could reduce overestimation by separating the max operation in the target into action selection and action evaluation (Tan et al., 2022). Deep Recurrent Q-Network (DRQN) is also practical with Partially-Observable Markov Decision Process (POMDP) (Hausknecht & Stone, 2015). In DRQN, a recurrent layer is added after the convolutional part of the original DQN to handle long term dependencies. Nevertheless, adding a recurrent layer to a DQN makes DRQN trickier and longer to train compared to classical DQN.

DQN and its derivations were an important milestone; however, the research has identified several drawbacks. Consequently, efforts have been made to propose improvements to DQNs. Yet it is unclear which of these advancements are indeed complementary, and can be integrated into a single agent to go beyond the state-of-the-art performance. In this paper, we focus on making efficient task-offloading decisions for multiple vehicles moving at a non-constant speed along the road. Vehicles generate computation tasks at each time slot and RSUs provide computing services for the vehicles moving within the coverage radius. In order to reach an efficient solution to optimize joint delay and energy consumption, we design a task-offloading algorithm that exploits the state-of-the-art DQN variation designated Rainbow. Rainbow is an integration of six improvements to the DQN algorithm, namely Double DQN, prioritized experience replay, dueling networks, multi-step learning, distributional RL, and noisy nets (Hessel et al., 2018).

The main contributions of this paper are summarized as follows.

- The speed of each vehicle is assumed to be non-constant while related work have mostly simplified the case to constant speed.
- The speed of the learning process is improved by considering penalties for wrong offloading decisions. In other words, when the vehicle is out of the coverage radius of the selected RSU, a penalty is imposed on the agent.
- The optimization problem for computation offloading to simultaneously consider the delay and energy consumption of vehicles is mathematically formulated.
- The target RSU is specified when the offloading action is suggested. In some related work, it is only decided whether to offload or process locally.
- The direction of vehicles is taken into account when selecting the RSU to offload tasks.
- The agent makes offloading decisions for all moving vehicles in the network, while most of the related work assume a single vehicle.
- Real traffic data, instead of synthetic data, is used to evaluate the performance of the proposed algorithm. Extensive simulations are carried out to illustrate the efficiency of the proposed algorithm compared to the state-of-the-art.

The remaining of this paper is organized as follows. Section 2 discusses the related work in the literature. Section 3 provides background information about deep Q-learning and Rainbow algorithm. Section 4 introduces the system model, in terms of communication and computational models, and formulates the optimization problem. In Section 5, a Rainbow DQN-based algorithm is proposed to solve task offloading problem in vehicular networks. Section 6 presents numerical results of the proposed approach, and finally, the paper is concluded in Section 7, with also some guidelines for future research.

2. Related work

Vehicular edge computing is a promising computing paradigm that has attracted considerable attention in recent years. Considering mobile nodes and the time-varying dynamic networks, obtaining an efficient offloading strategy is challenging. In this context, task offloading algorithms play a critical role in optimizing resource utilization and improving system performance. Various approaches have been presented to solve task-offloading problems in VEC models (Fan et al., 2022; Jiang et al., 2021; Liu et al., 2022; Ning et al., 2020; Raza et al., 2020; Song et al., 2022). The high-dimensional state and action space of task offloading in vehicular networks, coupled with the dynamic topology of the network, can pose significant challenges in solving the problem. Due to the complex and dynamic nature of the problem, heuristic algorithms or conventional optimization techniques may fail to capture the full range of possible states and actions, leading to suboptimal performance. As a result, more advanced techniques such as Deep Reinforcement Learning (DRL), which can effectively handle high-dimensional state and action spaces, have become increasingly popular in this field (Alam & Jamalipour, 2022; Jeremiah et al., 2024; Karimi et al., 2022; Li et al., 2020; Peng et al., 2022; Shi et al., 2023, 2020). Although DRL has been successful in handling high-dimensional state and action spaces, conventional DQN algorithms may still face challenges in solving task offloading problems in vehicular networks, due to the dynamic nature of the network. Several improvements to DQN algorithms have been proposed to enhance their performance in these scenarios. These advanced DQN algorithms have shown promising results in reducing the effects of high state and action space dimensionality, and effectively handling the dynamic topology of vehicular networks (Lee et al., 2022; Liao et al., 2023; Tang & Wong, 2022; Tang et al., 2023; Yang et al., 2022; Zhang, Ge, et al., 2019).

Although DQN algorithms have shown desirable results in solving vehicular task offloading, some other approaches have been proposed in recent years as well. Game theory has emerged as a promising solution for task offloading in VEC systems, leveraging its ability to model strategic interactions among multiple entities and optimizing system performance under uncertain conditions. They include the ability to optimize system performance by considering strategic behavior, address load balancing concerns by modeling utility-based task and server choices, and effectively handle uncertainty in dynamic VEC environments. For example, the authors in Jiang et al. (2021) have proposed a two-stage game theory iterative algorithm to solve the problem of load imbalance in VEC systems with RSUs. Game theory has several limitations when applied to vehicular edge computing. It introduces complexity in modeling interactions, relies on assumptions about participant rationality and information, which may not always hold, and faces coordination challenges in achieving equilibrium. Scalability issues arise with a large number of entities, and privacy concerns

can deter cooperation when sensitive information is involved. Adapting to dynamic environments and the computational overhead of solving complex game models are further challenges. Real-world validation is crucial, but adds complexity to deployment and evaluation. These limitations emphasize the need for cautious consideration when using game theory in VEC scenarios.

Apart from game theory, other heuristic methods have succeeded as well in partially addressing this issue. Heuristic algorithms have interesting features for vehicular task offloading problems. They are prized for their simplicity, speed, and scalability, making them well-suited for real-time decision-making in dynamic vehicular environments with numerous vehicles, tasks, and edge servers. These algorithms adapt effectively to changing conditions, require low computational overhead, and can work in a decentralized manner, enhancing their applicability in large-scale vehicular networks. Furthermore, heuristics simplify complex problems, may offer robust solutions, and explore diverse solution possibilities.

The authors in Fan et al. (2022) proposed an iterative algorithm based on reformulation linearization and generalized benders decomposition methods to obtain the optimal solution, and a heuristic algorithm to provide a sub-optimal solution to minimize the total task processing delay through Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I) communication modes. Partial task offloading algorithms, such as the one in Raza et al. (2020), are common, as well. An urban scenario under V2V and V2I communication modes was considered in Raza et al. (2020), and a partial task offloading algorithm was proposed to reduce the task execution delay. Semi-definite relaxation is known to be an effective technique, specially when facing a high number of users and tasks. This technique was investigated in Liu et al. (2022), where multi-hop vehicle computation resources were leveraged to minimize joint response delay and cost. Another attempt to bring heuristics into use is the online Lyapunov optimization-based multi-decision-making algorithm presented in Ning et al. (2020), where the problem of minimizing the total network delay was formulated as a mixed integer nonlinear programming. In order to speed-up the solution, AI-based imitation learning was proposed with limited training samples. In Song et al. (2022), an ant colony optimization algorithm was proposed to find the best solution for task offloading, with a pheromone matrix initialization based on the base station load and user-base station distance.

Heuristic algorithms, while advantageous in vehicular task offloading problems, also come with disadvantages. One notable drawback is their tendency to produce suboptimal solutions due to their reliance on simplifications and rules of thumb rather than rigorous optimization. This limitation can result in inefficient resource utilization, increased energy consumption, and longer task execution times. Additionally, heuristics may struggle to adapt to dynamic and uncertain environments, making them less suitable for scenarios with rapidly changing conditions or unexpected events. Furthermore, their lack of optimality guarantees can hinder performance in critical applications where precise decision-making is essential, such as autonomous driving or emergency response systems. Hence, when employing heuristic algorithms, careful consideration of the trade-off between simplicity and solution quality is paramount in vehicular task offloading contexts.

Due to the high dimensional state–action spaces, and dynamic topology changes, most recent work has focused on DRL algorithms. DRL has emerged as a powerful paradigm for tackling complex decision-making problems in vehicular task offloading and edge computing. DRL encompasses both value-based and policy-based approaches, each offering unique advantages. Value-based DRL, exemplified by DQN, focuses on estimating the value of taking specific actions in given states, allowing for optimal action selection. On the other hand, policy-based DRL, such as Soft Actor Critic (SAC) and Deep Deterministic Policy Gradient (DDPG), directly learns the policy that maps states to actions, providing greater flexibility in handling continuous action spaces and stochastic environments. These DRL techniques excel in capturing the complex dynamics of vehicular networks and enabling adaptive task offloading decisions. Authors of Karimi et al. (2022) have investigated the resource allocation problem by considering cooperation between MEC and a central cloud to guarantee the required response time in a vehicular environment. They utilized a DRL approach to deal with the large state space and real-time network state transitions. In Alam and Jamalipour (2022), a multi-agent DRL-based algorithm was proposed to solve computation offloading problems while minimizing latency, energy consumption, and cost. In another study, a shared offloading strategy was proposed, based on DRL, to solve the offloading problem considering energy consumption and delay (Peng et al., 2022).

The authors in Shi et al. (2020) have explored task offloading in Vehicular Fog Computing (VFC) and emphasized the need to motivate vehicle resource sharing in dynamic environments. Their work focuses on incorporating factors like mobility, task priority, and service availability into the task offloading process. To address this, they formulate the problem as a Markov Decision Process (MDP) and SAC based DRL algorithm to maximize utility while considering latency. Their results, as presented in extensive simulations, demonstrate the effectiveness of their proposed scheme compared to traditional approaches. In Jeremiah et al. (2024), a framework was proposed to enhance VEC performance by addressing challenges related to vehicle mobility and network dynamics. It uses Digital Twin (DT) technology to create virtual network node replicas, enabling real-time condition assessment and edge node collaboration. The framework optimizes RSUs selection using channel state information and employs a Non-Orthogonal Multiple Access (NOMA) protocol for vehicle communication. Its primary objective is to maximize VEC system computation rates and minimize the task completion delays through joint optimization of offloading decisions, subchannel allocation, and RSU association. The MDP model and Advantage Actor–Critic (A2C) algorithm were applied in Jeremiah et al. (2024) to solve the optimization problem.

The paper Li et al. (2020) introduces a collaborative edge computing framework for vehicular networks, aiming to reduce computing service latency and enhance reliability. It comprises two main components: a task partition and scheduling algorithm for workload allocation and task execution scheduling, and an AI-based approach using deep reinforcement learning to optimize task offloading, computing, and result delivery policies. By formulating the problem as a MDP, the authors employed DDPG to find optimal solutions in dynamic urban transportation networks. The paper Shi et al. (2023) introduces a three-tier vehicular edge computing system designed to optimize task offloading decisions, considering factors such as vehicle mobility and discrete variables. To tackle the complexity of the optimization problem, the authors proposed the TODM-DDPG algorithm, which shows promise in enhancing task offloading efficiency in VEC. DRL algorithms offer advantages in optimizing vehicular task offloading, but they face challenges and limitations. Computational complexity can hinder real-time decision-making, and sample inefficiency may pose problems in data-scarce scenarios. Overestimation bias in value-based DRL and training instability in policy-based methods are key concerns. Additionally, traditional DQN requires a large amount of training data, in dynamic vehicular environments, which can be quite challenging.

Recent advances in the context of DQN algorithms have shown good potential in solving task offloading problems in VEC systems. Double DQN is an extension of DQN that addresses overestimation of action values by using the target network to select actions, which leads to more stable training and better performance (Tang et al., 2023). Dueling DQN improves the accuracy of action value estimation by decomposing it into state value and advantage value, by allowing for more effective learning of the Q-function (Tang & Wong, 2022). Prioritized DQN assigns higher priority to more important experiences for replay, which can improve sample efficiency and reduce training time (Liao et al., 2023). Distributed DQN extends the basic algorithm to distributed environments, allowing for faster and more efficient learning (Lee et al., 2022). Noisy DQN introduces random noise into the network weights, encouraging exploration and improving

learning (Yang et al., 2022). Multi-step DQN improves the learning process by taking into account multiple consecutive observations and rewards, leading to more accurate value estimation and better performance (Zhang, Ge, et al., 2019). In summary, the advancements in DQN have indeed shown great potential for enhancing task offloading in VEC scenarios. However, it is important to highlight that these enhancements have typically been developed and evaluated separately which means there is a need for comprehensive investigations that integrate various improvements into a unified framework. This holistic approach could potentially yield even more significant benefits in terms of performance and efficiency in VEC systems. Such integration and evaluation of these advanced DQN techniques in a real-world vehicular environment are essential steps toward achieving the full potential of these advancements.

To the best of our knowledge, no prior studies have explored the full potential of integrating recent advancements in DQN to enhance the optimization and acceleration of task offloading decisions. In this regard, we propose a novel approach that leverages the Rainbow algorithm, which is a combination of six state-of-the-art DQN enhancements. Our approach aims to minimize the delay and energy consumption in task offloading decisions while also taking into account the vehicle's direction. Through this integration, we aim to demonstrate the significant benefits of using the Rainbow algorithm for improving the efficiency and effectiveness of solving task offloading problems in vehicular networks.

3. Background information

This section provides an overview of deep Q-learning to establish a foundational understanding for the proposed approach presented in this paper. Afterward, we focus on the Rainbow algorithm which has demonstrated significant successes in solving problems in complex environments. We also emphasize its effectiveness in solving the problem of task offloading in the vehicular networks.

3.1. Deep Q-learning

Q-learning is a well-known classical RL algorithm that aims to find the optimal policy by estimating the Q-value of each state–action pair (Kumar et al., 2018). The Q-value function $Q_{(s_t,a_t)}$ estimates the expected discounted cumulative reward for each state–action pair. The Q-learning algorithm learns the optimal Q-value function by iteratively updating the Q-values based on the Bellman equation,

$$Q_{(s_t,a_t)} = Q_{(s_t,a_t)} + \alpha \left[r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}\right) - Q\left(s_t, a_t\right) \right], \tag{1}$$

where s_t is the current state in time step t, a_t is the action taken in time step t, r_t is the immediate reward received after taking action a_t in state s_t, α is the learning rate, γ is the discount factor, s_{t+1} is the next state, and finally, a_{t+1} is the action that maximizes the Q-value in the next state s_{t+1}.

The main limitation of Q-learning is that it can achieve the optimal policy only when the state and action spaces are small. With regard to the continuous state space in our proposed approach, we have infinite possibilities, and therefore, a huge state space. Consequently, traditional Q-learning algorithms may not be able to find an optimal solution. To tackle this challenge, we utilize DQN, which integrates neural network techniques in Q-learning to replace ordinary neural networks and extract high-level features from raw input data (Mnih et al., 2013). Deep neural network is used to approximate the action values for a given state s_t, input of the neural network, which can support state and action spaces with larger dimensions. To avoid overestimation and stabilize the learning process, DQN uses two neural networks; one is the main network used to represent the Q function, denoted as $Q\left(s_t, a_t|\theta\right)$, where θ represents the weight parameter of the main network, and the other is the target network used to represent Q' function, denoted as the $Q'\left(s_t, a_t|\theta'\right)$ where θ' represent the weight parameters of the target network.

In the following, we describe the key components of the DQN algorithm, including action selection, experience replay memory, and DQN network training. We discuss how these elements work together to enable an agent to learn effective policies in a wide range of environments.

- **Action selection.** The agent needs to balance between exploiting its current knowledge, i.e., taking the action with the highest Q-value, and exploring new actions that might lead to even higher rewards. To do this, the agent uses an exploration strategy called ϵ-greedy. At each step, the agent chooses an action with the highest Q-value with probability $1-\epsilon$, the exploitation part, and selects a random action with probability ϵ, the exploration part, based on the current state to avoid reaching the local optimal point, as shown in Eq. (2).

$$a_t = \begin{cases} \arg\max_{a_t} Q\left(s_t, a_t\right) & with\ probability\ 1-\epsilon \\ \text{Randomly choose from action space} & with\ probability\ \epsilon \end{cases} \tag{2}$$

- **Experience replay memory.** Experience replay memory refers to a technique used to improve the efficiency of the learning process by storing and reusing past experiences of an agent in its interactions with an environment. Each experience typically consists of the agent's current state, the action it took, the resulting reward, and the next state. During the learning process, the agent can randomly sample experiences from the replay buffer and use them to update its Q-values, which estimate the expected future rewards for taking different actions in different states. By randomly sampling experiences from the replay buffer, the agent can break the correlation between consecutive experiences and reduce the likelihood of getting stuck in local optima. Experience memory is a powerful technique that has been shown to improve the efficiency and stability of DQN learning, enabling agents to learn from past experiences and avoid overfitting to recent experiences.
- **DQN network training.** Finally, the DQN network is trained by minimizing the square temporal difference error between the target value of the target network and the estimated value of the main network. Therefore, the loss function is defined as Eq. (3)

$$L_t\left(\theta_t\right) = \left(y_t - Q\left(s_t, a_t|\theta_t\right)\right)^2, \tag{3}$$

where y_t is the target value of the target network for time step t, and $Q\left(s_t, a_t|\theta_t\right)$ is the estimated value of the main network with parameters θ_t. Also y_t can be defined by Eq. (4).

$$y_t = r_t + \gamma \max Q'\left(s_{t+1}, a_{t+1}|\theta_t'\right). \tag{4}$$

DQN is still facing major decision-making challenges in dynamic environments, hence since the introduction of DQN, several independent improvements have been made by the DRL community. However, it is not determined which of them can be integrated to provide the state-of-the-art DQN learner. In our case, we exploit Rainbow which is an extended DQN that combines several improvements into a single learner. DRL algorithms like Rainbow are particularly well-suited for this kind of task offloading problem, because they can learn to make decisions on complex and dynamic environments.

3.2. Rainbow

In this section, we briefly introduce the improvements of the Rainbow algorithm compared to the DQN. For more information about the Rainbow algorithm refer to Hessel et al. (2018), Jäger et al. (2021) and Obando-Ceron and Castro (2021).

- **Double DQN.** One significant challenge in traditional DQNs is their susceptibility to overestimating action values, a phenomenon that can lead to suboptimal policies, especially in complex environments like vehicular networks. This overestimation bias can hinder convergence, resulting in suboptimal task offloading decisions and increased time and energy consumption. To address this challenge, the Double DQN algorithm was introduced to alleviate overestimation bias arising from estimation errors. The fundamental concept behind Double DQN is to separate the action selection and value estimation processes. This is achieved by employing two distinct networks: the online network, responsible for action selection during the agent's interactions with the environment, and the target network, utilized for estimating target Q-values. While the online network is frequently updated to adapt to the changing environment, the target network's weights are periodically synchronized with those of the online network. The target network plays a crucial role in computing target Q-values for the Q-learning update equation. By employing a separate target network, Double DQN aims to stabilize the training process and mitigate overestimation bias (due to the maximization step in Eq. (4), particularly when dealing with complex tasks like vehicular task offloading, where precise decision-making is essential. Therefore, Eq. (4) is revised as follows (Hasselt et al., 2016; Hessel et al., 2018; Tang et al., 2023):

$$y_t = r_t + \gamma Q'\left(s_{t+1}, \arg\max Q\left(s_{t+1}, a_{t+1}|\theta_t\right)|\theta'_t\right). \quad (5)$$

- **Prioritized experience replay.** Prioritized experience replay is an important technique that enhances the learning process in DQN algorithms, addressing the issue of inefficient learning from uniformly sampled experiences. In the traditional DQN approach, all experiences, consisting of state, action, reward, and next state tuples, are stored in a replay buffer, and training samples are drawn randomly from this buffer. However, not all experiences are equally valuable for learning. Prioritized experience replay assigns priorities to each experience $\left(s_t, a_t, r_t, s_{t+1}\right)$ based on their potential to improve the agent's knowledge. Experiences that lead to unexpected outcomes, large rewards, or experiences that the agent can learn the most from are given higher priorities. When sampling experiences during training, the agent prefers those with higher priorities. This prioritization allows the agent to focus on the most informative experiences, accelerating the learning process and making it more sample-efficient (Liao et al., 2023; Obando-Ceron & Castro, 2021). In vehicular task offloading scenarios, prioritized experience replay can be especially beneficial, as it enables the offloading system to learn from experiences that are most relevant to improving task offloading decisions and optimizing resource utilization. By efficiently selecting and learning from high-priority experiences, the system can adapt to changing conditions and achieve better overall performance in vehicular edge computing tasks. With prioritized replay buffer transitions are sampled with probability p_t relative to the last encountered absolute Temporal Difference (TD) error, which is the difference between the target and the predicted values, as a measure of the expected learning progress:

$$p_t \propto \left|r_t + \gamma \max Q'\left(s_{t+1}, a_{t+1}|\theta'_t\right) - Q\left(s_t, a_t|\theta_t\right)\right|^{\omega}, \quad (6)$$

where ω is a hyperparameter that influences the form of the distribution.

- **Dueling networks.** The Dueling DQN architecture represents a significant advancement in value-based RL, addressing several limitations of the traditional DQN. In the traditional DQN, a single neural network estimates the Q-values for all possible actions given a state. However, the Dueling DQN architecture introduces a novel approach by splitting the network into two streams: one stream estimates the value of the current state, while the other estimates the advantage of each possible action (Jäger et al., 2021; Tang & Wong, 2022). These streams share a common convolutional encoder and are combined using a specialized aggregator layer to generate an estimate of the state–action value function Q. The fusion of these two streams follows a specific factorization of action values, as expressed by Eq. (7).

$$Q\left(s_{t+1}, a_{t+1}|\theta_t\right) = \upsilon_\eta\left(f_\xi(s)\right) + a_\psi\left(f_\xi(s), a\right) - \frac{\Sigma_{a'} a_\psi\left(f_\xi(s), a'\right)}{N_{actions}}, \quad (7)$$

where ξ, η, ψ denote the parameters of the shared encoder f_ξ, while a_ψ represents the value of choosing a specific action at a given state, and υ_η represents the value of the given state regardless of the action taken, and $\theta = \{\xi, \eta, \psi\}$ is their concatenation.

The key advantage of the Dueling DQN architecture lies in its ability to distinguish valuable states from less valuable ones without the need to explicitly calculate the impact of every action in each state. This separation enables Dueling DQN to provide highly accurate Q-value estimates for each action within a given state. This feature becomes particularly critical when numerous actions exhibit similar values, as the advantage stream allows the agent to effectively differentiate between them. Moreover, in scenarios like vehicular networks, there is often no need to compute the value of each action at every time step, especially when actions are only relevant when vehicles have tasks to process. To enhance computational efficiency, it is preferable to focus exploration efforts on actions in states that are currently relevant.

Dueling DQN's versatility extends to real-time optimization in vehicular task offloading scenarios. Training the network with relevant datasets empowers the agent to learn optimal task allocation based on the current system state, leading to faster and more energy-efficient task completion. By employing Dueling DQN in this context, the agent can provide more accurate estimates of the value of each possible action, resulting in improved task offloading decisions.

- **Multi-step learning.** In traditional DQN algorithms, agents update their Q-values based solely on the immediate reward received after taking an action. However, multi-step learning DQN takes a different approach by allowing agents to receive a sequence of rewards and take a sequence of actions before updating their Q-values. This method involves estimating the expected reward for the next n steps, enabling agents to incorporate information about future rewards beyond just the immediate one (Hessel et al., 2018; Zhang, Ge, et al., 2019). To implement this, we first define the truncated n-step return from a given state s_t by applying Eq. (8).

$$r_t^{(n)} = \Sigma_{k=0}^{n-1} \gamma^{(k)} r_{k+1}. \quad (8)$$

Subsequently, we can formulate an alternative y_t for DQN that utilizes these truncated returns as Eq. (9).

$$y_t = r_t^{(n)} + \gamma^{(n)} \max Q'\left(s_{t+1}, a_{t+1}|\theta'_t\right). \quad (9)$$

This mechanism can expedite the learning process when the value of n is optimally tuned. In the context of vehicular networks, multi-step learning empowers agents to proactively consider the future impact of their decisions on system performance. By predicting the likelihood of future task arrivals and offloading decisions, agents can make choices that minimize energy consumption and latency over multiple steps. Taking into account multiple future rewards, agents can learn to make more efficient and effective decisions in the long run. This improvement in the learning process not only reduces variance but also enhances efficiency and robustness to noisy rewards and environmental stochasticity.

- **Distributional reinforcement learning (Distributional RL).** Traditional DQN algorithms predict the expected return reward as a scalar value to determine the optimal action in each state.

However, this approach has limitations in highly dynamic environments, such as vehicular networks, where multiple possible outcomes exist for each action. To address this challenge, Distributional RL has been proposed to learn the entire distribution of the expected return, which can gain more insights and knowledge for the agent, leading to a much faster and more stable learning process (Jäger et al., 2021; Lee et al., 2022). This approach enables the agent to consider multiple possibilities effectively, which results in more informed and robust decision-making in the face of uncertainty.

This technique introduces a novel approach by learning to approximate the distribution of returns, referred to as Z, instead of the traditional Q action value function in Q-Learning. In essence, Z is a mapping that associates state–action pairs with distributions over returns, known as value distributions. The crucial insight here is that these return distributions must adhere to a distributional variant of Bellman's equation. In order to model Z, a discrete distribution is employed, defined by a set of atoms denoted as z. These atoms are derived from a finite range of values between v_{min} and v_{max} and are parameterized as Eq. (10):

$$z_i = v_{min} + (i-1)\frac{v_{max} - v_{min}}{N_{atoms} - 1},\ i \in 1, 2, \ldots, N_{atoms}, \tag{10}$$

where N_atoms represents the number of atoms and v_min and v_max specify the minimum and maximum values of the distribution, respectively. Different combinations of these parameters yield distinct return distributions. The probabilities of the atoms are determined by a model θ, resulting in the sampling probability p_i obtained by Eq. (11).

$$p_i(s,a) = \frac{e^{\theta_i(s,a)}}{\Sigma_j \theta_j(s,a)}. \tag{11}$$

The primary objective is to learn θ for effectively approximating the true distribution of returns, denoted as $d_t = \left(z, p\left(s_t, a_t|\theta_t\right)\right)$, based on the support vector and associated probabilities. This approach enables the development of a distributional variant of Q-Learning. It begins by constructing a new support for the target distribution, and then minimizing the Kullback–Leibler divergence (D_{KL}), a distance metric quantifying the difference between the distribution d_t and the target distribution $d'_t \equiv \left(r_t + \gamma z, p\left(s_{t+1}, \bar{a}^*_{t+1}|\theta'_t\right)\right)$, according to Eq. (12).

$$D_{KL}\left(\Phi_z d'_t \| d_t\right), \tag{12}$$

where, Φ_z represents the L2-projection of the target distribution onto z, and $\bar{a}^*_{t+1}$ is the greedy action concerning the mean action values $Q'\left(s_{t+1}, a|\theta'_t\right) = z^t p\left(s_{t+1}, a|\theta_t\right)$ in state s_{t+1}. Overall, this distributional approach employs a neural network representation with $N_{atoms} \times N_{actions}$ outputs, followed by a *softmax* operation to ensure proper normalization of the distribution.

- **Noisy nets.** One of the primary challenges in traditional DQN is the exploration limit imposed by the ϵ-greedy technique. This technique requires the execution of numerous actions to collect initial samples for training the network. In highly dynamic and complex vehicular networks, the agent must explore various offloading strategies to identify the optimal one. Additionally, the agent needs to learn how to predict and control unknown and often stochastic environments. To address this challenge, a variant of DQN called noisy nets has been proposed, which incorporates exploration by introducing noise to the network's parameters (Obando-Ceron & Castro, 2021; Yang et al., 2022). Noisy Nets introduce a unique layer known as a noisy linear layer which combines both a deterministic and a noisy component. This layer is defined as Eq. (13):

$$y = (b + wx) + \left(b_{noisy} \odot \epsilon^b + \left(w_{noisy} \odot \epsilon^\omega\right) x\right), \tag{13}$$

where ϵ^b and ϵ^ω are random variables, and $\odot$ denotes the element-wise multiplication. This noisy layer is utilized as a replacement for a standard linear layer expressed as $y = b + wx$. Over time, as the network undergoes training, it learns to effectively disregard the noisy component. The key idea is to encourage the agent to explore more by making the Q-values less deterministic, and hence less prone to overfitting. Noisy DQN can facilitate exploration and prevent the agent from overfitting to its past experiences, resulting in better performance and faster convergence to the optimal solution.

Table 1
Key notations used in the problem formulation.

Notation	Description
M	Number of RSUs
r^{rsu}_j	Communication coverage radius of RSU_j
N	Number of Vehicles
$C_{k,i}$	Number of CPU cycles required to complete the $task_i$ generated by $vehicle_k$
$a_{k,i}$	Data size of $task_i$ generated by $vehicle_k$
$d_{k,i}$	Deadline of $task_i$ generated by $vehicle_k$
$x^j_{k,i}$	Binary offloading decision variable
L_k	Effective switched capacitance coefficient of $vehicle_k$

The integration of the Rainbow algorithm into our work significantly improves training stability and convergence. Rainbow successfully counteracts overestimation bias, enhances training data utilization through prioritized experience replay, and refines action value estimation using the Dueling DQN approach. It effectively handles the issue of delayed rewards through multi-step learning and fosters exploration with the inclusion of noisy nets. These collective enhancements lead to greater stability in convergence and the development of superior policies, rendering Rainbow a robust option for complex tasks such as vehicular task offloading.

4. System modeling and problem formulation

A detailed look into the target system and assumptions is essential before describing the proposed approach. In this section, each aspect of the system, namely network, communication, and computation, is modeled with parameters and variables that are directly or indirectly related to the time and energy consumption, to avoid complexity and at the same time increase the suitability of the solution. Finally, the problem is formulated to optimize the time and energy consumption of the task offloading scheme in a vehicle. We provide an explanation of the notation and symbols used in the problem formulation, as summarized in Table 1. Moreover, Fig. 1 provides a visual representation of the vehicular network, highlighting the communication radius and the interactions between vehicles and RSUs.

4.1. Network model

We consider a vehicular network consisting a road where M RSUs, denoted as a set $R = \{1, 2, \ldots, M\}$, with specific communication coverage radius r^{rsu}_j, $1 \le j \le M$, are equidistantly distributed. Each RSU is equipped with an MEC server which provides computation capability for vehicles in adjacency via the wireless channel. It is assumed that all moving vehicles are always within the range of at least one RSU. Vehicles, denoted as a set $V = \{1, 2, \ldots, N\}$, where N is the total number of vehicles, are moving along the road at varying speeds generating indivisible compute-intensive tasks at each time slot. Each task of vehicle k, $1 \le k \le N$, is modeled as $T_{k,i} = \{c_{k,i}, a_{k,i}, d_{k,i}\}$, where $c_{k,i}$ stands for the number of CPU cycles required to complete the task i, $a_{k,i}$ denotes the input data size, and $d_{k,i}$ is the maximum acceptable time to fulfill the task from the moment task i is allocated, as represented in Table 1. Produced tasks are either processed locally or offloaded to an RSU within the communication range via a wireless channel. Let $x^j_{k,i} \in \{0, 1\}$ denote the binary offloading decision variable,

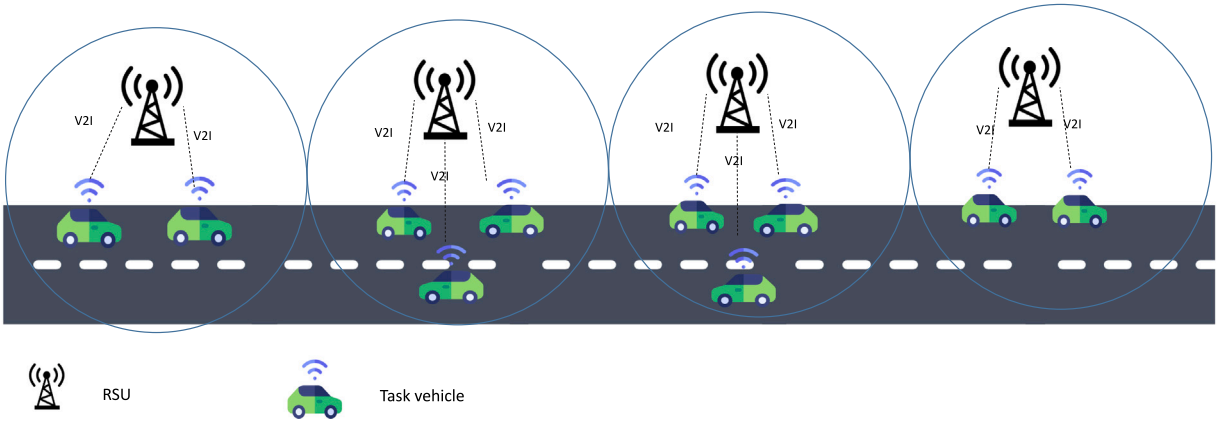


Fig. 1. An abstract representation of a vehicular network.

where $x^0_{k,i} = 1$ means that the computation task i of vehicle k should be processed locally, and $x^j_{k,i} = 1$ means that vehicle k will offload task i to the RSU j.

4.2. Communication model

When $x^j_{k,i} = 1$, the computation task will be offloaded to the RSU j because vehicle k cannot meet the low-latency demand of the task i, due to the lack of necessary computing resources. We assume that there is one mode of communication in the vehicular network, i.e., V2I, and each vehicle communicates with RSUs through a direct wireless link. According to the Shannon formula (Li et al., 2021; Mao et al., 2017), the data transmission rate between vehicle k and RSU j is calculated according to Eq. (14)

$$tr_{k,j} = B\log_2\left(\frac{|h|^2 p_{tr,k}}{\sigma\left(distance^k_j\right)^{\bar{\omega}}}\right), \tag{14}$$

where B denotes channel bandwidth, h is the channel fading coefficient, $p_{tr,k}$ stands for the transmission power of vehicle k, σ represents the power of Gaussian white noise, $\bar{\omega}$ denotes path loss exponent, and $distance^k_j$ is the distance between vehicle k and RSU j. This distance can be expressed by Eq. (15)

$$distance^k_j = \sqrt{(x_k - x_j)^2 + (y_k - y_j)^2}, \tag{15}$$

where x_k and y_k denote the coordinates of vehicle k while x_j and y_j denote the coordinates of RSU j. The time and energy consumed to transmit task i of vehicle k to RSU j are computed using Eqs. (16) and (17), respectively.

$$Time^{comm,j}_{k,i} = \frac{a_{k,i}}{tr_{k,j}}, \tag{16}$$

$$Energy^{comm,j}_{k,i} = p_{tr,k}.Time^{comm,i}_{k,j}. \tag{17}$$

4.3. Computation model

In our proposed computation offloading model, the task can be either accomplished by the vehicle locally or executed by one of the RSUs in proximity, so in the following, we discus both of these models.

4.3.1. Local computing

For $x^0_{k,i} = 1$, computation task will be processed locally on vehicle's own resources. The time consumed to process the task i by the vehicle k can be calculated by Eq. (18)

$$Time^{loc}_{k,i} = \frac{c_{k,i}}{f^{vehicle}_k}, \tag{18}$$

where $f^{vehicle}_k$ denotes the computation capability of the vehicle k (in CPU cycles per second). The corresponding amount of energy consumption for computing is expressed as Eq. (19)

$$Energy^{loc}_{k,i} = p_k Time^{loc}_{k,i}, \tag{19}$$

where p_k represents the local computation power of the vehicle k and is formulated by Eq. (20)

$$p_k = L_k\left(f^{vehicle}_k\right)^3, \tag{20}$$

where L_k represents the effective switched capacitance coefficient depending on the chip architecture in the vehicle k (Mao et al., 2016).

4.3.2. Edge computing

When the computation task is hard to be processed locally under the time constraints, the vehicle transmits the information to one of the adjacent RSUs. After fulfilling the task computation, the result is returned back to the vehicle. In vehicular edge computing scenarios, the decision to omit the energy and time costs associated with transmitting results back from RSUs to vehicles is justified by the small size of the results and the high downlink rates typically available in RSUs (He et al., 2018; Zhao et al., 2017). The total time for executing task i of the vehicle k on the RSU j is computed by Eq. (21)

$$Time^{edge,j}_{k,i} = \frac{c_{k,i}}{f^{rsu}_j}, \tag{21}$$

where f^{rsu}_j denotes the computation capability of the RSU j (in CPU cycles per second).

While the RSU is processing the task, the vehicle should wait until the result gets prepared and returned back. As our primary objective revolves around reducing vehicle energy consumption, we meticulously account for the energy consumption incurred by vehicles while waiting for RSUs to complete task computation. During this time period, we assume that the vehicle is in standby mode and the power consumption of this mode is denoted as $p_{idle,k}$. The corresponding energy consumption of the vehicle k is:

$$Energy^{edge,j}_{k,i} = p_{idle,k} Time^{edge,j}_{k,i}. \tag{22}$$

4.4. Problem formulation

We formulate the optimization problem, with the aim of jointly minimizing the total delay and energy for all vehicles moving within the network, by making offloading decisions for each vehicle. In this paper, we assume that all MEC servers have equal computation resources evenly shared between all vehicles in the network. Total time and energy consumed to process task i of vehicle k can be attained by Eqs. (23) and (24), respectively.

$$Time^{comp}_{k,i} = Time^{loc}_{k,i} x^0_{k,i} + \Sigma^M_{j=1} Time^{edge,j}_{k,i} x^j_{k,i}, \tag{23}$$

$$Energy_{k,i}^{comp} = Energy_{k,i}^{loc} x_{k,i}^{0} + \Sigma_{j=1}^{M} Energy_{k,i}^{edge,j} x_{k,i}^{j}. \tag{24}$$

Total time and energy consumed to transmit task i of vehicle k can be attained using Eqs. (25) and (26), respectively.

$$Time_{k,i}^{comm} = \Sigma_{j=1}^{M} Time_{k,i}^{comm,j} x_{k,i}^{j}, \tag{25}$$

$$Energy_{k,i}^{comm} = \Sigma_{j=1}^{M} Energy_{k,i}^{comm,j} x_{k,i}^{j}. \tag{26}$$

Considering the communication and computation models introduced in Sections 4.2 and 4.3, the total cost imposed on each vehicle is calculated by Eq. (27)

$$\begin{aligned} Cost_k = \Sigma_{i=1}^{I} \Big(& \alpha_k \left(Time_{k,i}^{comp} + Time_{k,i}^{comm} \right) \\ & + \beta_k \left(Energy_{k,i}^{comp} + Energy_{k,i}^{comm} \right) + Penalty_{k,i} \Big), \end{aligned} \tag{27}$$

where I is the total number of tasks generated by vehicle k, and α_k, $\beta_k \in [0, 1]$, under constraint $\alpha_k + \beta_k = 1$, are weighting parameters expressing the importance of delay and energy consumption while making decision for each vehicle, respectively. It is worth noting that all operands in the cost formula of Eq. (27) have been normalized using the min–max method. Normalization is a crucial step in our proposed approach, as it ensures that all operands are on the same scale and have a similar impact on the final result. By using the min–max method, we can ensure that all operands are scaled to a range between 0 and 1, making them easier to compare and combine. We also consider a penalty for situations where a wrong offloading decision is made for a task, which accrued when the distance between the selected RSU and the vehicle is more than RSU's communication coverage radius. The penalty makes the agent learn which nodes to offload tasks to. Consequently, we define the penalty as the maximum cost that can be imposed to the vehicle by offloading decisions as follows in Eqs. (28) and (32).

$$Time_{max}^{comp} = \frac{\max_{k,i} \left(c_{k,i} \right)}{\min_{j} \left(f_{j}^{rsu} \right)}, \tag{28}$$

$$Energy_{max}^{comp} = p_{idle} Time_{max}^{comp}, \tag{29}$$

$$Time_{max}^{comm} = \frac{\max_{k,i} \left(a_{k,i} \right)}{\min_{k,j} \left(tr_{k,j} \right)}, \tag{30}$$

$$Energy_{max}^{comm} = p_{tr} Time_{max}^{comm}, \tag{31}$$

$$Penalty_{k,i} = \begin{cases} 0 & distance_{j}^{k} \le r_{j}^{rsu} \\ \alpha_k \left(Time_{max}^{comp} + Time_{max}^{comm} \right) + & \\ \beta_k \left(Energy_{max}^{comp} + Energy_{max}^{comm} \right) & distance_{j}^{k} > r_{j}^{rsu} \end{cases} \tag{32}$$

For each task i generated by vehicle k, for which a wrong offloading decision is made, $Time_{max}^{comp}$ and $Energy_{max}^{comp}$ denote the maximum possible delay and energy consumption to process the task in remote processing mode, respectively. Furthermore, $Time_{max}^{comm}$ and $Energy_{max}^{comm}$ represent the maximum possible delay and energy consumption to transfer the task data to the RSUs, respectively. With the above cost model, we formulate the computation offloading as an optimization problem with two constraints that aims to minimize the weighted cost of the system in terms of delay and energy as Eq. (33)

$$\begin{aligned} & \min_{x} \Sigma_{k=1}^{N} Cost_k \\ & \Sigma_{j=1}^{M} x_{k,i}^{j} = 1 \qquad c1 \\ & x_{k,i}^{j} \in \{0, 1\} \qquad c2, \end{aligned} \tag{33}$$

where $c1$ denotes constraint on offloading decisions and guarantees that each computation task can be processed only once. This constraint ensures that tasks are processed without duplication or unnecessary redundancy. Moreover, $c2$ denotes that offloading decision variable is binary.

In order to solve this optimization problem, it is necessary to find optimal values for decision variables $x_{k,i}^{j}$. Since offloading decision variables are binary, the optimization problem is not convex. Moreover, we assume a realistic environment where the state of the network dynamically changes. Consequently, the operator needs to collect a large amount of system state information and make the global decision on offloading operation based on the current network's state. Therefore, we apply Rainbow to solve this problem in the following section.

5. Proposed algorithm

A mapping from task offloading decisions in a vehicular network into a DRL formulation includes some key elements. A state space, an action space, and a reward function are necessary for complete mapping. This section starts with the characteristics of these elements and proposes a Rainbow DQN-based algorithm afterward to solve the task offloading problem in a dynamic vehicular network.

5.1. Definition of the state and action spaces, and reward function

At each time slot t, the agent reads important parameter assignments, namely, the state definition. One of the key steps of learning algorithms is to find an appropriate representation of the state space. Given our goal of making decisions for all vehicles, it does introduce complexity to the algorithm. We try to simplify the state space by concentrating on the critical factors essential for decision-making. This strategic simplification ensures that our algorithm remains robust even in complex scenarios.

The most intuitive parameter is the distance from the vehicle to each of the RSUs. Moreover, the angle with which the vehicle is deviating from each of the RSUs is also included here, in the state definition, as it is important for the agent to learn how likely is for the vehicle to get closer or further away from the RSU in the near future. We assume that the offloading decisions depend on the distance and angle of movement of each vehicles with all RSUs in the network. Incorporating these parameters into the state space enables us to capture the relative dynamics of vehicles' positions and orientations, which play a crucial role in anticipating and responding to varying speeds.

The state space is defined as $s_t = \{distance_{j,t}^{k}, \Omega_{j,t}^{k}, 1 \le j \le M, 1 \le k \le N\}$ in tth time slot, where $distance_{j}^{k}$ represents the distance between kth vehicle and jth RSU and $\Omega_{j,t}^{k}$ represents the angle of the kth vehicle's direction with regard to the jth RSU. By quantifying the angles between vehicles and RSUs, our algorithm gains a detailed understanding of the orientation of each vehicle in relation to the available offloading options. This fine-grained spatial awareness allows the proposed algorithm to intelligently distribute tasks based on not just proximity, but also the direction from which vehicles approach RSUs.

In order to determine which roadside unit should be selected for offloading, we employed a vector-based approach that accounts for both the vehicle's movement direction relative to the east direction and the position of the RSUs. To do so, we first transform the angle of the vehicle's movement to a two-dimensional vector using Eq. (34), where $angle_{vehicle,k}$ shows the clockwise angle of vehicle k from the east direction, expressed in radians

$$\left[x_{angle,k}, y_{angle,k} \right] = \left[-\sin \left(angle_{vehicle,k} \right), \cos \left(angle_{vehicle,k} \right) \right], \tag{34}$$

where $\left[x_{angle,k}, y_{angle,k} \right]$ is related to the direction vector of the kth vehicle, shown by the green vector in Fig. 2. In the next step, the distance vector from the vehicle to the RSU, shown by the red vector in Fig. 2, can be computed as Eq. (35)

$$\left[x_{vehicle,k}^{RSU,j}, y_{vehicle,k}^{RSU,j} \right] = \left[x_j - x_k, y_j - y_k \right], \tag{35}$$

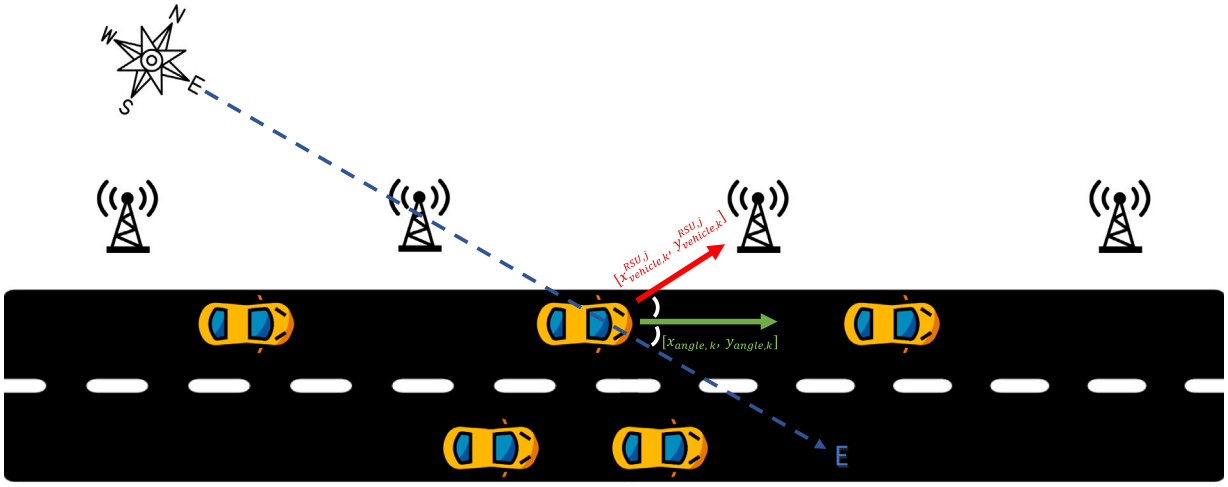


Fig. 2. Calculus of the angle between the vehicle's movement vector and the distance vector.

where $\left[x^{RSU,j}_{vehicle,k}, y^{RSU,j}_{vehicle,k}\right]$ is the distance vector of the kth vehicle to the jth RSU. Finally, the angle between the vehicle direction vector and the distance vector can be computed by Eq. (36).

$$angle^{RSU,j}_{vehicle,k} = \cos^{-1}\left(\frac{x_{angle,k}\cdot x^{RSU,j}_{vehicle,k} + y_{angle,k}\cdot y^{RSU,j}_{vehicle,k}}{\sqrt{x^2_{angle,k}+y^2_{angle,k}}\sqrt{x^{RSU,j\ 2}_{vehicle,k}+y^{RSU,j\ 2}_{vehicle,k}}}\right). \quad (36)$$

In time slot t, and state space s_t, the agent takes an action in the action space. To support the computational requirements of the Rainbow algorithm for vehicular task offloading, we propose utilizing an RSU among all, as the agent, due to its superior computation capabilities compared to vehicles. In the VEC network considered herein, the agent needs to decide whether to process a task locally or to offload it to an RSU. Hence, the action space is defined as $a_t = \{\left(x^j_{k,i}\right)_t \mid k \in N, j \in M, i \in I\}$. It is worth mentioning that in conventional offloading decision scenarios, the typical approach is binary, involving a straightforward choice between offloading tasks to an RSU or conducting local processing. While this binary method simplifies decision-making, it can sometimes fall short in terms of precision and may not fully exploit the potential advantages of offloading. Our algorithm introduces a novel approach by locating the target RSU, enabling us to make nuanced offloading decisions based on unique RSU characteristics, including factors like distance and angle. This context-aware strategy empowers us to intelligently distribute tasks to the most suitable RSU in real-time, which results in improved resource utilization, shorter task processing durations, and overall system performance enhancements.

The reward function is defined to address the objective of the optimization problem. Our objective is to minimize total delay and energy consumption for all vehicles in the network, while the goal of the agent is to achieve the maximum reward. Therefore, the reward function should negatively correlate with the objective function. In our reward formulation, we take into account the concept of task failure, which helps us incorporate deadlines into the decision-making process. When a task's processing time exceeds its specified deadline, we consider it a failed task. This failure is reflected in the reward function to guide the agent toward making decisions that prioritize timely task completion, aligning with our goal of minimizing delays and energy consumption for all vehicles. The agent gains a reward r_t at the tth time slot, by taking the action a_t when in the state s_t, in order to maximize long-term cumulative reward. To this end, we define the reward function as Eq. (37).

$$r_t = \frac{\left(n_{tasks} - n_{failed}\right)}{\Sigma^N_{k=1} Cost_k}, \quad (37)$$

where n_{tasks} denotes the total number of tasks generated in tth time slot, while n_{failed} denotes the total number of tasks failed due to the longer execution time than their deadlines. By employing a global reward mechanism, we incentivize vehicles to prioritize decisions that enhance the overall system's performance, even if it does not always result in reduced costs for individual vehicles in every particular circumstance. The primary aim is to enhance network-wide efficiency and optimize the utilization of resources, which closely aligns with the objectives of VEC. While this strategy may not ensure cost savings for each vehicle in every scenario, it results in more efficient and collaborative task offloading choices that resonate with the broader objectives of VEC environments.

Based on the discussion given in Sections 3 and 4, and the formulation above, we design the task offloading Algorithm 1 based on Rainbow. In the first step, we initialize the replay memory, the number of time slots and episodes, the main network, and the target network (line 1). Then, all generated tasks by vehicles are stored to process (line 4). The next part of the algorithm is devoted to select the best action using the ϵ-greedy technique (line 5). After the action is accomplished, the agent observes a new state and reward, and the tuple (s_i, a_i, r_i, s_{i+1}) is stored as a transition in replay memory in order to train networks (line 6). Then, the agent samples some of the data as a batch from experience replay memory based on their computed complexity to train networks (line 7). The target value y_i is calculated for each transition (line 9). The algorithm minimizes the loss function $L_t(\theta_t)$ by updating the parameters of the main DQN network, θ_t (line 10). For a fixed number of time slots, the algorithm updates the parameters of the target network, θ'_t, to match the current parameters of the main DQN network, θ_t (line 11).

6. Performance evaluation

In this section, the performance of the proposed algorithm is evaluated through a simulation of a real traffic data. For the sake of comparison, various DQN-based algorithms are also implemented and simulated to verify the suitability and superiority of the Rainbow algorithm.

6.1. Experimental settings

Simulations were conducted in a real urban scenario, considering two distinct time periods: from 7:00 a.m. to 9:00 a.m. and from 5:00 p.m. to 7:00 p.m. These time slots were chosen to encompass different traffic conditions and scenarios, providing a comprehensive assessment of the proposed algorithms' performance. The simulations

Algorithm 1: Rainbow DQN-based algorithm

Input:
Initial parameters of the main DQN network and the target network,
The replay memory D.

1. **for** *episode* = *1* **to** E **do**
2. Initialize state s_1
3. **for** $t = 1$ **to** T **do**
4. Store tasks resulted from the application decomposition, produced by vehicles
5. Choose action a_t based on the ϵ-greedy policy
6. Carry out action a_t, and observe the new state s_{t+1} and r_t
7. Calculate complexity of transitions and extract a batch of i transitions from the memory $D\langle s_i, a_i, r_i, s_{i+1}\rangle$
8. **for** all (s_i, a_i, r_i, s_{i+1}) **in** D **do**
9. Calculate the target value y_i
10. Minimize loss function $L_t(\theta_t)$, by updating θ_t
11. Update θ_t' with θ_t in every fixed number of time-slots

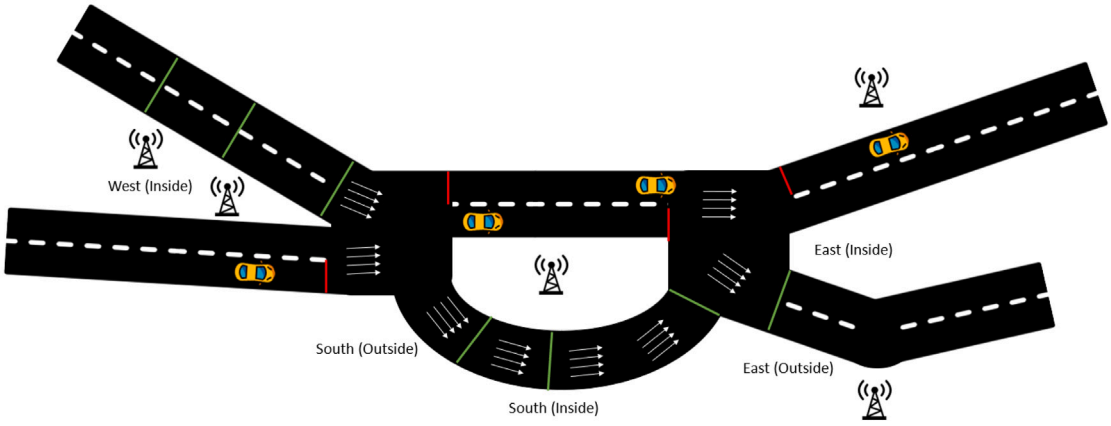


Fig. 3. Vehicles route map.

were carried out on a circular road located in Crete, France (a. Lebre et al., 2015). We show the algorithm's capability to effectively handle scenarios with varying vehicle speeds by using real-world data featuring vehicles of different speeds. As shown in Fig. 3, it includes a roundabout with 4 entrances and exits, multiple two-lane or three-lane roads, one bus road, four-lane change points, and 15 traffic lights. We consider the trajectory of three vehicles, $N = 3$, driving at variable speed. The computation capability of each vehicle is 20 GHz in our experiments. We assume $M = 5$ RSUs are distributed among the road at equal distances with communication range of 125 m, with a computation capability of 100 GHz. These computation capabilities assume typical values used in related research (Liu et al., 2019; Lv et al., 2021). Moreover, the bandwidth is set to 2 MHz. It is also assumed that the computation intensity of each task is 600 cycles/bit, and the input data size follows a uniform distribution within $[1, 8]$ MB. The service latency allowed for each task is 1.7 s. It is worth mentioning that all experiments were carried out on a computer system with an Intel(R) Core(TM) i7- 3610QM 2.30 GHz processor and 4 GB of main memory running an x64 Windows 10 operating system. Rainbow DQN underwent a *1463 seconds* training period, followed by a *68 seconds* test phase. These time statistics shed light on the computational needs during training and emphasize the algorithms' efficiency in practical real-world situations.

The proposed Rainbow DQN-based algorithm together with three state-of-the-art algorithms, namely DQN (Xu et al., 2022), Double DQN (Ardagna et al., 2022), and DRQN (Chen et al., 2020), are implemented by using PyTorch[1]; a machine learning framework based on the Torch library used for applications such as computer vision and natural language processing. We choose these DQN-based algorithms because they are strongly used and accepted in the field of reinforcement learning. The reason behind this choice was the discrete nature of our action space, where the agent's decisions are binary, e.g., deciding whether or not to perform task offloading. This discrete characteristic aligns perfectly with DQN-based methods which are tailored to handle discrete action spaces. The learning rate, discount factor, size of the mini-batch, and the replay buffer are set to 10^{-3}, 0.9, 64, and 2000, respectively. Herein a fixed number of time slots is considered and its value is set to 320. A detailed list of the parameters and their values are provided in Table 2.

6.2. Experimental results

For comparison and analysis purposes, multiple criteria, namely, task failure, time, and energy consumption are considered for different values of the parameters. Results are then depicted for the proposed algorithm and all of the state-of-the-arts. Fast convergence to high reward values is an indicator showing how effectively the iterations of the algorithm are proceeding. The convergence curve of the cumulative

[1] https://github.com/pytorch/pytorch

Table 2
List of parameters and their values in the experiment.

Parameter	Value	Description
M	5	Number of RSUs
N	3	Number of vehicles
I	3	Number of tasks in each time slot
σ	−110 dbm	Gaussian white noise
$\bar{\omega}$	2	Path loss exponent
h	1	Channel fading coefficient
L_k	10^{-27}	Effective switched capacitance coefficient of vehicle k
$P_{tr,k}$	1.3 watts	Transmission power of vehicle k
$P_{idle,k}$	0.2 watts	Power consumption of standby mode of vehicle k
E	150	Episode size
T	185	Number of time slots
B	2 MHz	System bandwidth
$d_{k,i}$	1.7 s	Deadline of task i generated by vehicle k
$a_{k,i}$	1–8 MB	Data size of task i generated by vehicle k
$f_k^{vehicle}$	20 GHz	Computation capability of vehicle k
f_j^{rsu}	100 GHz	Computation capability of RSU j
α_k	0.5	Coefficient of importance of time in vehicle k
β_k	0.5	Coefficient of importance of energy in vehicle k

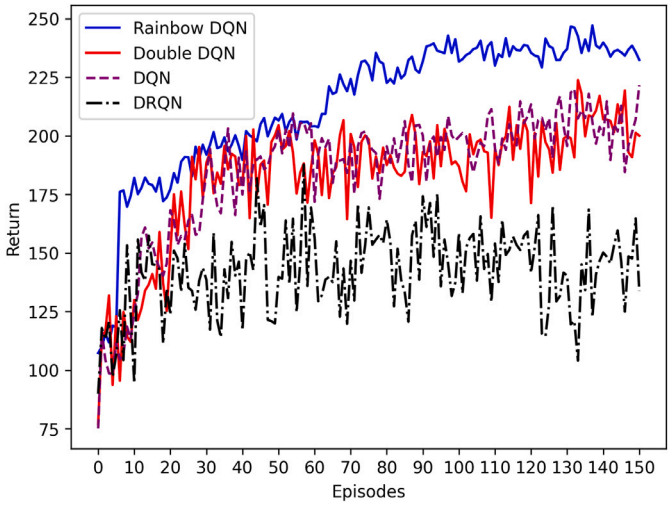


Fig. 4. Convergence of the training stage for the proposed Rainbow DQN-based algorithm and three DQN-based algorithms for the interval 7:00 a.m. to 9:00 a.m.

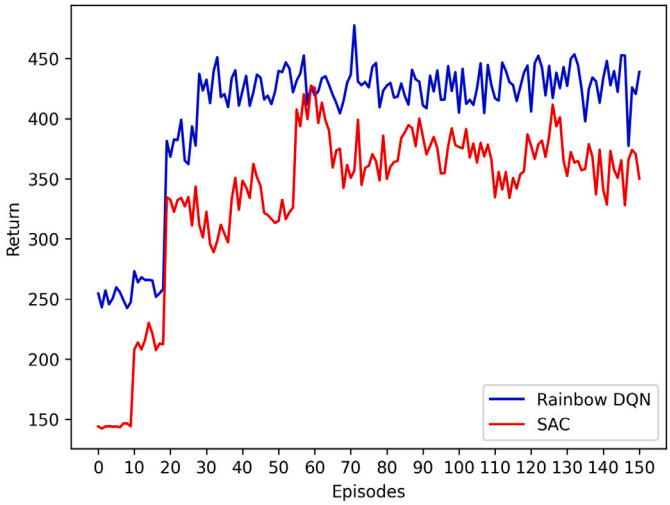


Fig. 5. Convergence of the training stage for the proposed Rainbow DQN-based algorithm and SAC algorithm for the interval 5:00 p.m. to 7:00 p.m.

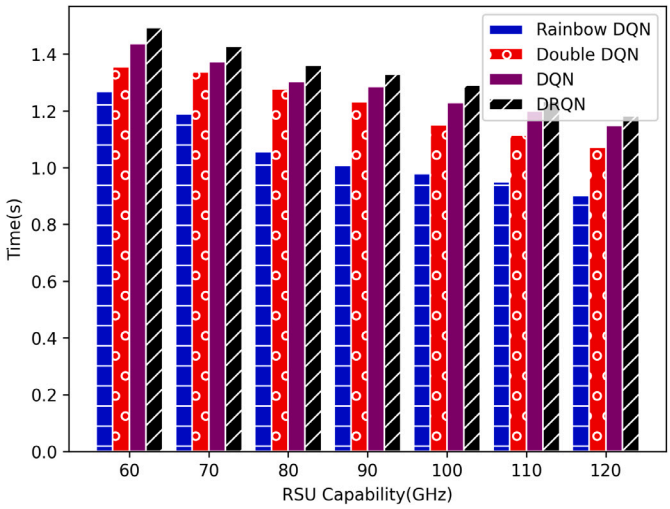


Fig. 6. Total task processing delay against RSU capability.

average reward is plotted for 150 episodes in Fig. 4. As shown in this figure, within 150 episodes, and in the first 10 iterations, the presented Rainbow DQN-based algorithm shows a faster convergence speed and gets the highest reward in comparison with the other benchmarks. DQN and Double DQN follow the same pattern approximately in the learning process. Notably, the performance of the DRQN algorithm may seem less favorable during the depicted training period due to its recurrent layer complexity. Recurrent layers excel at capturing sequential dependencies in data, which is valuable for tasks involving temporal sequences. However, this advantage comes with a trade-off; recurrent networks typically demand more training time to effectively learn these dependencies and reach an optimal policy. Thus, in Fig. 4, DRQN may appear slower to converge or perform less efficiently compared to non-recurrent algorithms like DQN. In Fig. 5, we observe the training convergence of the proposed Rainbow DQN-based algorithm and the SAC algorithm during the interval 5:00 p.m. to 7:00 p.m. Rainbow demonstrates superior performance in terms of both convergence speed and final policy quality. This advantage can be attributed to Rainbow's effective handling of discrete action spaces, enabling it to make precise offloading decisions in the dynamic vehicular environment. The plot highlights Rainbow's capability to quickly adapt and optimize task offloading strategies for improved efficiency.

Next, three main criteria are studied for the algorithms. Figs. 6 to 8, respectively, show the average task execution time, average energy consumption, and task failure rate versus the computation capability of the MEC servers deployed in the RSUs. According to the plots, Rainbow outperforms other algorithms by a significant margin and achieves better outcomes for the objective function. Meanwhile, Double DQN has a better performance compared to DQN due to alleviating overestimation, and DRQN has the highest task execution time compared to the other three methods due to the lack of adequate time for training.

It is important to understand how different algorithms succeed in controlling the task failure rate when deadline intervals are too tight or too loose. This is illustrated in Fig. 9, where Rainbow is performing better than the other DQN-based algorithms, more specifically when the tasks are more time-sensitive. As can be seen in Fig. 9, when deadlines are tight, the superiority of the presented Rainbow DQN-based algorithm against the other algorithms is much clear.

The effect of increasing data size, on delay and energy consumption, is also investigated herein, based on the results depicted in Figs. 10 and 11. Expectedly, the energy consumption and execution time increase with the increase in the data size. For an input data size of 4 MB, the latency and energy of all algorithms are almost the same. The reason is that all schemes prefer to process tasks locally while ignoring capabilities of RSUs. At increased data sizes, Rainbow shows better

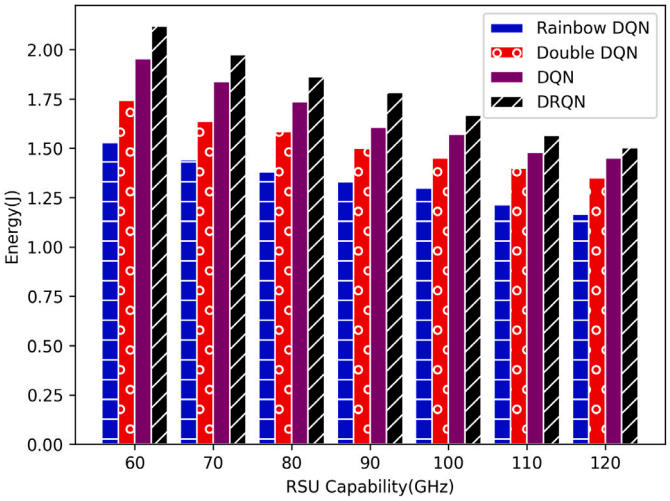


Fig. 7. Total task energy consumption against RSU capability.

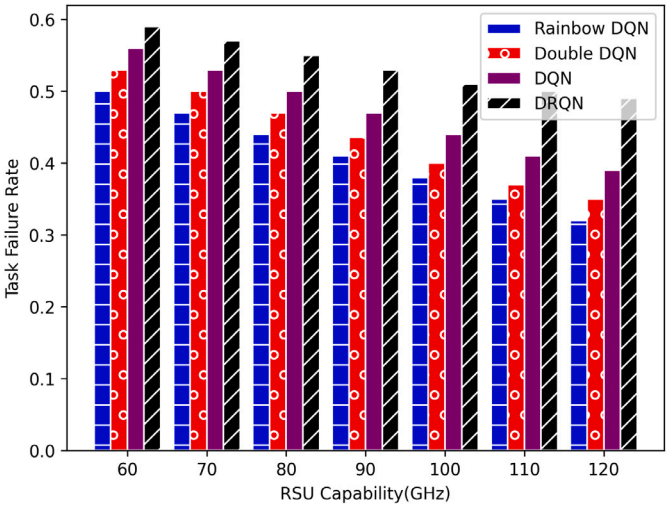


Fig. 8. Task failure rate against RSU capability.

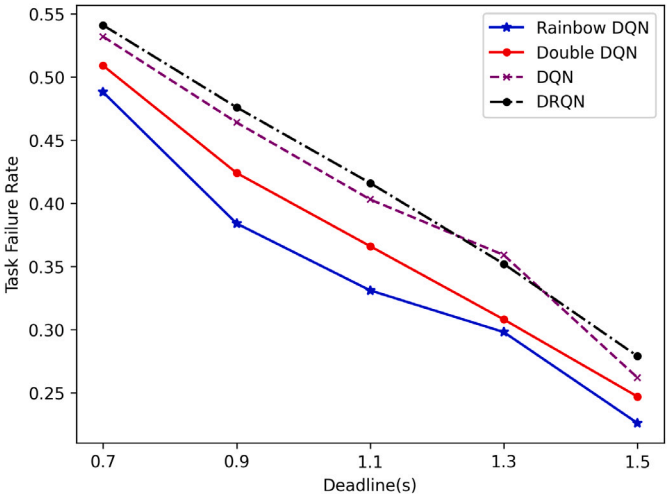


Fig. 9. Task failure rate against the deadline.

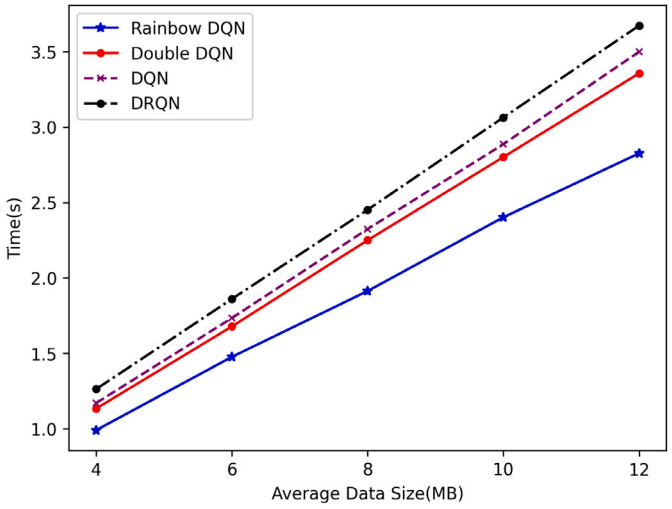


Fig. 10. Total task processing delay against the average data size.

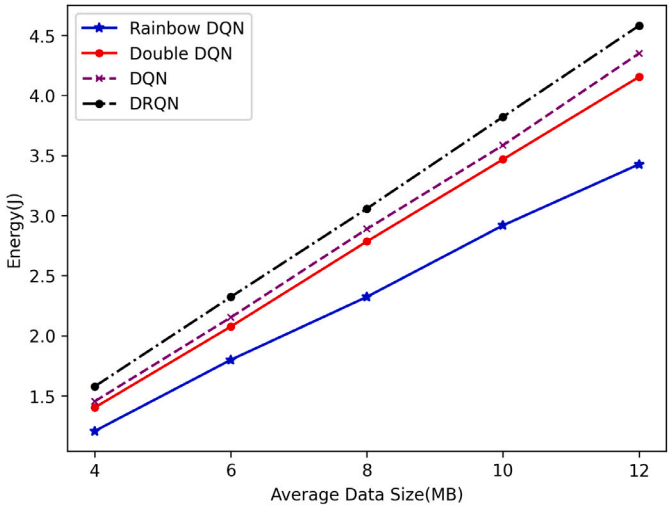


Fig. 11. Energy consumption of task processing against the average data size.

performance and keeps the energy consumption and latency levels lower, even when facing more complex tasks, since its gap with the other methods increases gradually.

Finally, the behavior of the two main factors, energy and time, is assessed when the bandwidth is changing. In Figs. 12 and 13, with an increase of the network bandwidth, less time and energy will be spent on offloading tasks to the roadside units, and this ultimately leads to a reduction in energy consumption and total time in all four algorithms. According to the results, the proposed Rainbow DQN-based algorithm is keeping lower consumption values and more optimal latency levels, even facing the limited bandwidth of 1 MHz.

Table 3 provides a summary of the average energy consumption values observed in our experiments. To offer a more comprehensive perspective on the reliability and variability of these results, we have thoughtfully included 95% confidence intervals alongside the average values. These confidence intervals serve as essential indicators, offering insights into the degree of certainty and dispersion within the energy consumption data, further enhancing the robustness of our findings.

7. Conclusions and future work

In this paper, we focused on designing an efficient task offloading algorithm in VEC networks in order to jointly minimize delay and

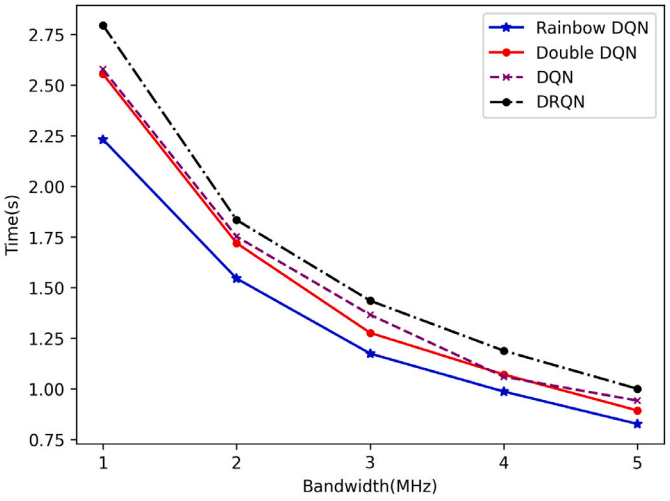


Fig. 12. Total task processing delay against the bandwidth.

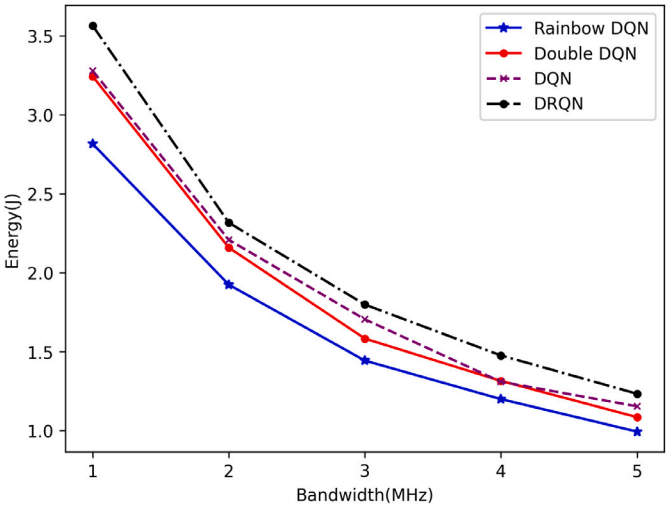


Fig. 13. Energy consumption of task processing against the bandwidth.

Table 3
Energy consumption of task processing (J).

Bandwidth (MHz)	Rainbow	Double DQN	DQN	DRQN
1	2.81 ± 0.18	3.24 ± 0.18	3.27 ± 0.25	3.56 ± 0.27
2	1.92 ± 0.23	2.15 ± 0.25	2.20 ± 0.29	2.31 ± 0.18
3	1.44 ± 0.25	1.58 ± 0.30	1.70 ± 0.32	1.79 ± 0.21
4	1.19 ± 0.26	1.31 ± 0.21	1.30 ± 0.25	1.47 ± 0.24
5	0.99 ± 0.30	1.08 ± 0.50	1.15 ± 0.27	1.23 ± 0.25

energy consumption by taking advantage of MEC servers deployed on RSUs. In order to address the high mobility of vehicles and the complex dynamic vehicular environment, we applied the Rainbow algorithm, which is an integration of independent advances on DQN to obtain a fast learning process and efficient decisions. In order to evaluate the effectiveness of the proposed algorithm, in comparison with state-of-the-art DQN algorithms, extensive experiments were conducted on real-world traffic data. The utilization of real traffic data in this evaluation serves as a bridge between theoretical algorithmic development and the complex realities of vehicular networks. This integration empowers us to conduct a comprehensive analysis of the proposed algorithm's performance, ultimately ensuring its practical relevance and applicability within the dynamic landscape of real-world vehicular environments. The proposed task offloading algorithm shows better performances, 18% and 15% improvement in energy consumption and delay, respectively, compared to the state-of-the-arts. However, the practical application of the Rainbow algorithm in VEC scenarios presents certain challenges. It demands high computational resources, potentially straining edge devices, and relies on extensive and diverse training data collection. Additionally, its exploration strategies and prioritized experience replay can introduce communication overhead, impacting real-time responsiveness as a crucial factor in VEC. Balancing algorithmic performance with resource constraints is a critical trade-off in VEC, where the need for powerful hardware must be weighed against improved decision-making. This decision requires careful optimization to strike the right balance between performance and resource practicality.

While offloading the whole task to the RSU is the more usual case and is completely realistic, in some scenarios the vehicle divides a complex task into multiple sub-tasks and locally processes only a portion of them. In such a situation, it is important to keep track of the task dependency graphs while distributing tasks in order to successfully combine the results. Therefore, a possible future work is to develop a dependency aware task offloading algorithm. In our study on multi-vehicle task offloading, we recognize the potential of load balancing as a promising future direction. While our current emphasis is on optimizing task offloading in dynamic vehicular environments, load balancing presents an opportunity to boost system performance by evenly distributing computational tasks among RSUs. As another guideline for the future work, one can delve deeper into the ablation studies of the Rainbow algorithm to gain a comprehensive understanding of the individual impacts of its components, further enhancing the insights into reinforcement learning.

CRediT authorship contribution statement

Mina Khoshbazm Farimani: Conceptualization, Methodology, Software, Data curation, Writing – original draft. **Soroush Karimian-Aliabadi:** Conceptualization, Methodology, Validation, Writing – original draft. **Reza Entezari-Maleki:** Supervision, Methodology, Validation, Writing – review & editing. **Bernhard Egger:** Methodology, Investigation, Writing – review & editing. **Leonel Sousa:** Methodology, Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

References

Alam, M. Z., & Jamalipour, A. (2022). Multi-agent DRL-based Hungarian algorithm (MADRLHA) for task offloading in multi-access edge computing Internet of Vehicles (IoVs). *IEEE Transactions on Wireless Communication*, *21*, 7641–7652.

Alchalabi, A. E., Shirmohammadi, S., Mohammed, S., Stoian, S., & Vijayasuganthan, K. (2021). Fair server selection in edge computing with Q-value-normalized action-suppressed quadruple Q-Learning. *IEEE Transactions on Artificial Intelligence*, *2*, 519–527.

Ardagna, C. A., Zhao, H., Hua, J., Zhang, Z., & Zhu, J. (2022). Deep reinforcement learning-based task offloading for parked vehicle cooperation in vehicular edge computing. *Mobile Information Systems*.

Chen, X., Chen, T., Zhao, Z., Zhang, H., Bennis, M., & Ji, Y. (2020). Resource awareness in unmanned aerial vehicle-assisted mobile-edge computing systems. In *IEEE 91st vehicular technology conference* (pp. 1–6). Antwerp, Belgium.

Chen, L., Wu, J., Zhang, J., Dai, H.-N., Long, X., & Yao, M. (2022). Dependency-aware computation offloading for mobile edge computing with edge-cloud cooperation. *IEEE Transactions on Cloud Computing*, *10*, 2451–2468.

Chen, M., Yi, M., Huang, M., Huang, G., Ren, Y., & Liu, A. (2023). A novel deep policy gradient action quantization for trusted collaborative computation in intelligent vehicle networks. *Expert Systems with Applications*, *221*, Article 119743.

Fan, W., Su, Y., Liu, J., Li, S., Huang, W., Wu, F., & Liu, Y. (2022). Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes. *IEEE Transactions on Intelligent Transportation Systems*, 1–10. http://dx.doi.org/10.1109/TITS.2022.3230430.

FortuneBusinessInsights (2020). Explosive trace detection market size. https://fortunebusinessinsights.com/explosive-trace-detection-market-104050. (Accessed March 2023).

Hasselt, V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 2094–2100). Phoenix, Arizona, USA.

Hausknecht, M., & Stone, P. (2015). Deep recurrent Q-learning for partially observable MDPs. In *AAAI fall symposium on sequential decision making for intelligent agents*. Arlington, Virginia, USA.

He, Y., Zhao, N., & Yin, H. (2018). Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, *67*, 44–55.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., & Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence* (pp. 3215–3222). New Orleans, Louisiana, USA.

Hu, L., Tian, Y., Yang, J., Taleb, T., Xiang, L., & Hao, Y. (2019). Ready player one: UAV-clustering-based multi-task offloading for vehicular VR/AR gaming. *IEEE Network*, *33*, 42–48.

Jäger, J., Helfenstein, F., & Scharf, F. (2021). Bring color to deep Q-networks: Limitations and improvements of DQN leading to Rainbow DQN. In B. Belousov, H. Abdulsamad, P. Klink, S. Parisi, & J. Peters (Eds.), *Studies in computational intelligence SCI*: *Vol. 883*, *Reinforcement learning algorithms: Analysis and applications* (pp. 135–149). Springer.

Jeremiah, S. R., Yang, L. T., & Park, J. H. (2024). Digital twin-assisted resource allocation framework based on edge collaboration for vehicular edge computing. *Future Generation Computer Systems*, *150*, 243–254.

Jiang, Q., Xu, X., He, Q., Zhang, X., Dai, F., Qi, L., & Dou, W. (2021). Game theory-based task offloading and resource allocation for vehicular networks in edge-cloud computing. In *IEEE international conference on web services* (pp. 341–346). Chicago, Illinois, USA.

Jiang, K., Zhou, H., Li, D., Liu, X., & Xu, S. (2020). A Q-learning based method for energy-efficient computation offloading in mobile edge computing. In *The 29th international conference on computer communications and networks* (pp. 1–7). Honolulu, Hawaii, USA.

Karimi, E., Chen, Y., & Akbari, B. (2022). Task offloading in vehicular edge computing networks via deep reinforcement learning. *Computer Communications*, *189*, 193–204.

Kumar, N., Swain, S. N., & Murthy, C. S. R. (2018). A novel distributed Q-learning based resource reservation framework for facilitating D2D content access requests in LTE-A networks. *IEEE Transactions on Network and Service Management*, *15*, 718–731.

a. Lebre, M., Mouel, F. L., & Menard, E. (2015). On the importance of real data for microscopic urban vehicular mobility trace. In *The 14th international conference on ITS telecommunications* (pp. 22–26). Copenhagen, Denmark.

Lee, Y., Masood, A., Noh, W., & Cho, S. (2022). DQN based user association control in hierarchical mobile edge computing systems for mobile IoT services. *Future Generation Computer Systems*, *137*, 53–69.

Li, M., Gao, J., Zhao, L., & Shen, X. (2020). Deep reinforcement learning for collaborative edge computing in vehicular networks. *IEEE Transactions on Cognitive Communications and Networking*, *6*, 1122–1135.

Li, L., Quek, T. Q., Ren, J., Yang, H. H., Chen, Z., & Zhang, Y. (2021). An incentive-aware job offloading control framework for multi-access edge computing. *IEEE Transactions on Mobile Computing*, *20*, 63–75.

Liao, L., Lai, Y., Yang, F., & Zeng, W. (2023). Online computation offloading with double reinforcement learning algorithm in mobile edge computing. *Journal of Parallel and Distributed Computing*, *171*, 28–39.

Liu, Y., Wang, S., Zhao, Q., Du, S., Zhou, A., Ma, X., & Yang, F. (2020). Dependency-aware task scheduling in vehicular edge computing. *IEEE Internet of Things Journal*, *7*, 4961–4971.

Liu, Y., Yu, H., Xie, S., & Zhang, Y. (2019). Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, *68*, 11158–11168.

Liu, L., Zhao, M., Yu, M., Jan, M. A., Lan, D., & Taherkordi, A. (2022). Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, *24*, 2169–2182.

Lv, B., Yang, C., Chen, X., Yao, Z., & Yang, J. (2021). Task offloading and serving handover of vehicular edge computing networks based on trajectory prediction. *IEEE Access*, *9*, 130793–130804.

Mao, Y., Zhang, J., Song, S. H., & Letaief, K. B. (2016). Power-delay tradeoff in multi-user mobile-edge computing systems. In *IEEE global communications conference* (pp. 1–6). Washington, DC, USA.

Mao, Y., Zhang, J., Song, S., & Letaief, K. B. (2017). Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. *IEEE Transactions on Wireless Communication*, *16*, 5994–6009.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *Proceedings of the 27th international conference on machine learning* (pp. 1–9). Atlanta, GA, USA.

Morra, L., Lamberti, F., Pratticò, F. G., Rosa, S. L., & Montuschi, P. (2019). Building trust in autonomous vehicles: Role of virtual reality driving simulators in HMI design. *IEEE Transactions on Vehicular Technology*, *68*, 9438–9450.

Ning, Z., Zhang, K., Wang, X., Guo, L., Hu, X., Huang, J., Hu, B., & Kwok, R. Y. K. (2020). Intelligent edge computing in Internet of Vehicles: A joint computation offloading and caching solution. *IEEE Transactions on Intelligent Transportation Systems*, *22*, 2212–2225.

Obando-Ceron, J. S., & Castro, P. S. (2021). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *The 38th international conference on machine learning* (pp. 1–11).

Peng, X., Han, Z., Xie, W., Yu, C., Zhu, P., Xiao, J., & Yang, J. (2022). Deep reinforcement learning for shared offloading strategy in vehicle edge computing. *IEEE Systems Journal*, 1–10. http://dx.doi.org/10.1109/JSYST.2022.3190926.

Raza, S., Liu, W., Ahmed, M., Anwar, M. R., Mirza, M. A., Sun, Q., & Wang, S. (2020). An efficient task offloading scheme in vehicular edge computing. *Journal of Cloud Computing: Advances, Systems and Applications*, *9*, 1–14.

Shi, W., Chen, L., & Zhu, X. (2023). Task offloading decision-making algorithm for vehicular edge computing: A deep-reinforcement-learning-based approach. *Sensors*, *23*, 7595.

Shi, J., Du, J., Wang, J., Wang, J., & Yuan, J. (2020). Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, *69*, 16067–16081.

Song, S., Ma, S., Yang, L., Zhao, J., Yang, F., & Zhai, L. (2022). Delay-sensitive tasks offloading in multi-access edge computing. *Expert Systems with Applications*, *198*, Article 116730.

Sun, Y., Guo, X., Song, J., Zhou, S., Jiang, Z., Liu, X., & Niu, Z. (2019). Adaptive learning-based task offloading for vehicular edge computing systems. *IEEE Transactions on Vehicular Technology*, *68*, 3061–3074.

Tan, L., Kuang, Z., Gao, J., & Zhao, L. (2022). Energy-efficient collaborative multi-access edge computing via deep reinforcement learning. *IEEE Transactions on Industrial Informatics*, 1–10. http://dx.doi.org/10.1109/TII.2022.3213603.

Tang, M., & Wong, V. W. S. (2022). Deep reinforcement learning for task offloading in mobile edge computing systems. *IEEE Transactions on Mobile Computing*, *21*, 1985–1997.

Tang, H., Wu, H., Qu, G., & Li, R. (2023). Double deep Q-network based dynamic framing offloading in vehicular edge computing. *IEEE Transactions on Network Science and Engineering*, *10*, 1297–1310.

Xu, X., Shen, B., Ding, S., Srivastava, G., Bilal, M., Khosravi, M. R., Menon, V. G., Jan, M. A., & Wang, M. (2022). Service offloading with deep Q-network for digital twinning-empowered Internet of Vehicles in edge computing. *IEEE Transactions on Industrial Informatics*, *18*, 1414–1423.

Yang, J., Sun, Y., Lei, Y., Zhang, Z., Li, Y., Bao, Y., & Lv, Z. (2022). Reinforcement learning based edge computing in B5G. *Digital Communications and Networks*, *8*, 469–476.

Zhang, F., Ge, J., Wong, C., Li, C., Chen, X., Zhang, S., Luo, B., Zhang, H., & Chang, V. (2019). Online learning offloading framework for heterogeneous mobile edge computing system. *Journal of Parallel and Distributed Computing*, *128*, 167–183.

Zhang, J., Guo, H., Liu, J., & Zhang, Y. (2019). Task offloading in vehicular edge computing networks: A load-balancing solution. *IEEE Transactions on Vehicular Technology*, *69*, 2092–2104.

Zhang, Y., Lan, X., Ren, J., & Cai, L. (2020). Efficient computing resource sharing for mobile edge-cloud computing networks. *IEEE/ACM Transactions on Networking*, *28*, 1227–1240.

Zhao, X., Huang, G., Jiang, J., Gao, L., & Li, M. (2022). Task offloading of cooperative intrusion detection system based on deep Q-network in mobile edge computing. *Expert Systems with Applications*, *206*, Article 117860.

Zhao, P., Tian, H., Qin, C., & Nie, G. (2017). Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. *IEEE Access*, *5*, 11255–11268.