TLCO: Topological Link-Aware Task Co-Offloading Method for Joint V2V and V2I System

Huijun Tang, *Member, IEEE*, Ming Du, Huaming Wu, *Senior Member, IEEE*, Pengfei Jiao, *Member, IEEE*, and Ruidong Li, *Senior Member, IEEE*

***Abstract*— Joint Vehicle-to-vehicle (V2V) and Vehicle-to-Infrastructure (V2I) offloading presents an efficient approach to leverage surplus computing resources from neighboring devices, thereby expanding the coverage of computing resources supply in the context of the Internet of Vehicles. However, many studies overlook the significance of topological communications caused by the rapid movement of vehicles, privacy, and communication intentions. To achieve efficient task offloading when facing various topological link structures, we first propose a novel topological link-aware task co-offloading (TLCO) method designed for partially offloading in the joint V2V and V2I system. Next, we model the sequential subtasks offloading process as the Markov Decision Process (MDP) and utilize the Double Deep Q-Network (DDQN) algorithm to optimize the total delay of the proposed system. Additionally, we put forth a prediction framework named Sliding Time Windows and TLCO algorithm (STW-TLCO) to accurately forecast the computation load at various time windows using pulsed parameters. Extensive experimental results demonstrate the effectiveness and superiority of the proposed TLCO-DDQN algorithm in comparison to other Deep Reinforcement Learning (DRL)-based and Greedy-based approaches. Furthermore, the STW-TLCO algorithm exhibits high accuracy, with an R-squared value exceeding 96%, confirming its predictive capabilities.**

***Index Terms*— Vehicle-to-vehicle, vehicle-to-infrastructure, deep reinforcement learning, task offloading.**

I. Introduction

In recent years, the advancement of communication technologies has led to the emergence of innovative paradigms, e.g., Vehicle-to-Vehicle (V2V) communication establishes direct and decentralized links among nearby vehicles and Vehicle-to-Infrastructure (V2I) communications through which vehicles establish connections and exchange information with infrastructure elements. These paradigms enable vehicles to effectively exchange real-time information, providing a foundation for numerous applications including collision avoidance [1], [2], cooperative driving [3], [4], [5], [6], and traffic management [7], [8].

In high-demand scenarios like traffic congestion or emergencies, vehicles can pool their computing power and work in coordination to handle the increased load, and nearby vehicles and infrastructures can share computing resources [9], [10], [11]. Wang et al. [12] proposed a cluster-based algorithm that regards buses as cluster heads and overcomes the limitation of resources to meet the Quality of Service (QoS) requirements of emergency messages. Zhao et al. [13] optimized the transmission modes and power levels in a V2V and V2I environment to maximize the total capacity of V2I links and ensure the requirements of QoS. He et al. [14] proposed a NOMA-enhanced V2V and V2I collaboration framework to optimize the total power consumption. Nguyen et al. [15] establish a multi-hop path adaptive method to maintain the connectivity and enhance the achieved throughput. Fan et al. [16] utilize Generalized Benders Decomposition and Reformulation Linearization methods to minimize the delays of all vehicles. It is evident that the V2V and V2I collaborative computing paradigm enables efficient utilization of resources, improve response times, and enhance overall system performance.

The rapid movement of vehicles is one of the reasons for changes in network topology and hinders the development of the Internet of Vehicles (IoV) [12]. In addition, some studies also explore other factors affecting link topology, such as security, link duration, and idle computational resources. Kadam et al. [17] attempt to address the challenges related to security and reliability in VANET communications by computing the direct and indirect trust scores of each vehicle belonging to each cluster to form security links. De Souza et al. [18] select service vehicles by considering link duration and distance between nodes, utilizing the idle resources on vehicle nodes to offload computational tasks. By considering factors such as link reliability, distance, available computational resources, and relative velocity, Bute et al. [19] propose a method to select nearby vehicles with available idle computational resources to enable concurrent task processing. Additionally, V2V communication raises concerns regarding potential privacy rights infringements and unauthorized exposure of personal data to external parties, causing some vehicle owners may opt not to participate in V2V communication so that a subset of vehicles is unable to establish connections with others [20]. Therefore, links between vehicles form a certain topology to ensure different

Received 12 February 2024; revised 10 September 2024 and 3 May 2025; accepted 7 July 2025. Date of publication 23 July 2025; date of current version 16 September 2025. This work was supported in part by Zhejiang Provincial Natural Science Foundation of China under Grant LQN25F020016; and in part by the National Natural Science Foundation of China under Grant 62401190, Grant 62071327, and Grant 62372146. The Associate Editor for this article was G. Orosz. *(Corresponding author: Huaming Wu.)*

Huijun Tang, Ming Du, and Pengfei Jiao are with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: tanghuijune@hdu.edu.cn; mdu@hdu.edu.cn; pjiao@hdu.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Ruidong Li is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan (e-mail: liruidong@ieee.org).

Digital Object Identifier 10.1109/TITS.2025.3590519

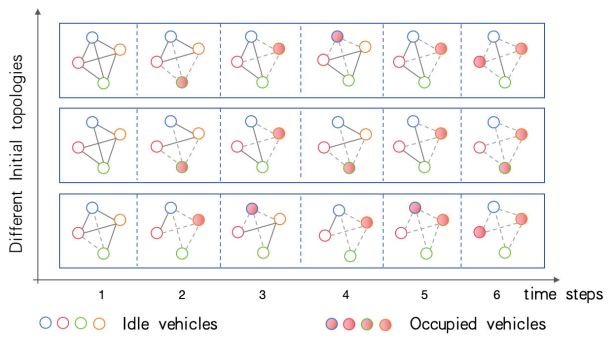


Fig. 1. The changes in topology links under offloading decisions. Different colors represent different vehicles. Hollow circles indicate that the vehicle has available computing resources to execute tasks, while solid circles mean that the vehicle has no available computing resources. Initial topology links influence the preferences for offloading decisions, which in turn affect the system's available computing capacity, thereby impacting the subsequent offloading decisions.

demands on communication quality, reliable communication, and privacy security.

In a static IoV environment, topological links are usually treated as fully connected plus constraints. However, in a dynamic IoV environment, as shown in Fig. 1, full connectivity is just one form of the initial state. Specifically, a solid red dot indicates that the device has no idle computational resources, meaning the vehicle cannot provide computational resources or can only do so with a certain waiting cost. We use a dashed line to indicate a link interruption or additional cost incurred. Thus, at different time steps, the links between devices in the vehicular network system are dynamically changing. We refer to the changes in links due to task computation in the vehicular network as the topological trajectory. Each topology trajectory starts from an initial topology, and the topology at the current time step is only related to the topology of the previous time step. Different link topologies mean different total available computational resources for the system, resulting in significant differences in the topological trajectories from different initial topologies. In conclusion, The initial topological structure will affect the available computational resources of the vehicular network, which in turn affects the offloading decisions, ultimately forming different topological trajectories. However, most studies usually consider dynamic offloading optimization starting from a fully connected initial topology, while neglecting the impact of other initial topological links on the decision-making process.

Aiming to achieve efficient resource allocation and task offloading based on various topological link structures, we propose a topological link-aware co-offloading method to minimize the total delay of the joint V2V and V2I systems that are composed of multiple moving vehicles. The main contributions of this paper are outlined as follows:

- *Topological Link-aware Task Co-Offloading (TLCO) method:* To find the best offloading policies in varying topological link structures, we model the topological link-aware partially offloading process in the joint V2V and V2I system as a Markov Decision Process (MDP) and utilize Double Deep-Q-Network (DDQN) to solve the optimization problem in joint V2V and V2I systems, which encodes the topological structure as a part of the input state of DDQN.
- *Slide time windows method based on TLCO (STW-TLCO):* We further propose an innovative prediction framework based on TLCO with a sliding time windows module STW-TLCO to predict the computational workload of the joint V2V and V2I systems at different time slices under different offloading strategies. This prediction is achieved through the analysis of pulsed parameters, which play a crucial role in determining the arrival time of tasks within the system.
- *Effective Performance:* Extensive experiments have been conducted to evaluate the performance of the proposed algorithm. The simulation results demonstrate the effectiveness and superiority of TLCO, especially compared to other DRL-based methods and greedy-based methods. Furthermore, our experiments validate the accuracy of the prediction results generated by the proposed prediction framework.
- *Cluster Computation Capability Index (3C Index) to evaluate coputational ability of V2I and V2V cluster:* The computing ability of an individual device is an attribute of the device itself, while the computing ability of a V2V and V2I cluster varies depending on the offloading strategy. After the task arrival distribution is known, STW-TLCO predicts the task computation curves for multi-hop partial offloading, whose slope reflects the computing capability of the V2I and V2V cluster, termed the 3C index. The 3C index indicates the computing power of a cluster under a given offloading strategy; a higher 3C index means the offloading method is more effective for the cluster. Experimental results show that TLCO-DDQN achieves a higher 3C index than other methods for the same cluster, indicating it better utilizes cluster computing resources.

The rest of the paper is organized as follows. The related works are shown in Sec. II. The system model and problem formulation are provided in Sec. III. Then, we elaborate on the design of the topological link-aware DDQN-based task co-offloading algorithm and the prediction framework of the amount of computation in Sec. IV. We present the implementation and then evaluate the proposed method in Sec. V. We conclude the paper in Sec. VI.

II. Related Work

Recently, several works have focused on the offloading-decision method on the V2V system. Zhao et al. [28] propose a Vehicle Edge Computing (VEC) offloading scheme where parked vehicles can be used as assisted vehicles of the V2V system and minimize the energy consumption and computation overhead of vehicles communication system under the limited vehicular terminal capacity constraints. Huang et al. [29] designed a V2V- and V2I-based unsignalized intersection collision warning system (UICWS) to enhance driver safety. Saleem et al. [30] proposed DOVEQ algorithm that models the connectivity of vehicles with roadside units (RSUs) and vehicles to estimate

TABLE I
THE QUALITATIVE COMPARISON OF THE CURRENT LITERATURE ON V2V SCHEMES BASED ON RL METHOD

Approaches	**Assisted Vehicle**	**Topological Link-Aware**	**Dynamic Offloading**	**Partial Offloading**	**Task Dependency**	**Method**
Hazarika *et al.* [9]	✓	✗	✓	✗	✗	DDPG
Xu *et al.* [21]	✗	✗	✓	✗	✗	DQN
Gupta *et al.* [22]	✗	✗	✗	✗	✗	DDQN
Alam *et al.* [23]	✓	✗	✓	✗	✗	MADRL
Liu *et al.* [24]	✗	✗	✓	✗	✗	Q-learning
Shi *et al.* [25]	✓	✗	✓	✓	✗	SAC
Liu *et al.* [26]	✗	✗	✓	✓	✗	DDPG
Tang *et al.* [27]	✗	✗	✓	✓	✓	DDQN
Ours	✓	✓	✓	✓	✓	DDQN

the connectivities among vehicles and the offloading capacity by traffic classification, overload control, and admission control. Zhao et al. [31] propose an intelligent partial offloading scheme, namely, Digital Twin-Assisted Intelligent Partial Offloading (IGNITE) to optimize computational delay and vehicle service price. Determining which vehicles provide computing resources based on statistical information of resource demands is impractical in practice because choices made solely on statistical information cannot adapt to time-varying requirements [27], [32], [33], [34]. Therefore, many works utilize reinforcement learning (RL) to optimize the offloading policies in the IoV environment because RL allows vehicles to learn and optimize their communication strategies based on punishment and reward from the environment.

As shown in Table I, some works optimize the V2V scheme using the RL method. To make the edge server in a V2V scenario able to scheme the cooperative perception of the vehicle, Xu and Liu [21] proposed a deep reinforcement learning (DRL) framework that effectively improves perception synergy. Gupta et al. [22] proposed a cluster-head selection algorithm based on double deep Q-network (DDQN) to minimize energy and latency to meet the intelligent transportation system (ITS) requirements. Hazarika et al. [9] proposed a DRL-based algorithm to maximize the mean utility of their proposed V2V framework that consists of assisted vehicles and task vehicles. Shi et al. [35] proposed a DRL-based offloading algorithm for the V2V system to minimize the delay with the consideration of task priority, mobility, and the service availability of vehicles. Alam and Jamalipour [23] proposed a multi-agent DRL-based Hungarian algorithm for solving the dynamic task offloading problem. Liu et al. [24] proposed a Q-learning-based algorithm to choose transmission parameters by interacting with a dynamic vehicular network to ensure reliable delivery. Shi et al. [25] proposed a V2V partial computation offloading algorithm, where tasks are divided into several parts and executed on neighboring vehicles. Li et al. [26] propose a two-tier hybrid partial offloading architecture to meet the computation requirements of vehicle tasks. By employing reinforcement learning techniques, vehicles can learn how to dynamically optimize important factors such as transmission power, channel allocation, and resource allocation, thereby achieving more efficient and reliable V2V communication. Unfortunately, these studies have overlooked the topological links between vehicles, which are crucial in real-world V2V communication scenarios.

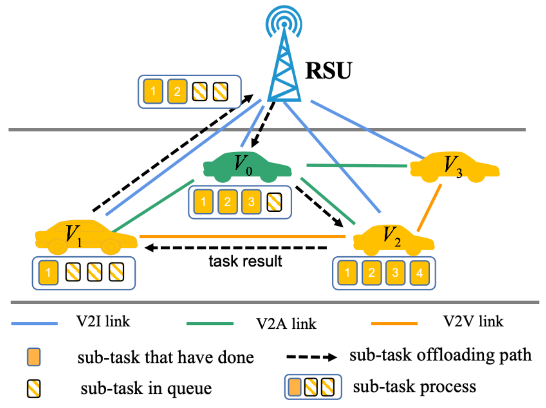


Fig. 2. The overall architecture of the joint V2I and V2V system. After V_1 completes its first subtask, it offloads its second subtask to the RSU. After the RSU completes the second task, V_1's third task is offloaded to the assisted vehicle. After finishing the third task, V_1's fourth task is offloaded to V_2. Finally, the task result of V_1 is transmitted back.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this paper, we consider a joint V2V and V2I system model as depicted in Fig. 2, which is composed of multiple general vehicles $V_1, V_2, \ldots, V_m$, an assisted vehicle V_0, and an RSU, where $m \in \{0, 1, 2, \ldots, M\}$. General vehicles selectively connect with other general vehicles (V2V link), the assisted vehicle can connect with any vehicle (V2A link), and the RSU can connect with any vehicle (V2I link). Within this framework, all vehicles may generate task. Additionally, the RSU and the assisted vehicle are endowed with the capability to execute tasks originating from the assisted vehicle itself, as well as tasks offloaded from the general vehicles. Furthermore, the general vehicles are equipped to execute their designated tasks, alongside tasks offloaded from other connected vehicles. To simplify, we only depict the task offloading decisions of V_1 in Fig. 2, whereas tasks from different vehicles can appear simultaneously.

A. Subtask Offloading Path

V2V partial computation offloading is a promising solution, involving the division of a task into multiple segments that are executed across several neighboring vehicles [25]. Inspired by this approach, we divide the task of vehicle V_m into a sequence of subtasks, each with a length less than K, where $m \in \{0, 1, 2, \dots, M\}$. The subtasks for each vehicle follow a sequential order, with each subsequent subtask generated based on the preceding one. The tasks assigned to each vehicle arrive at random arrival times, with T_m^{start} denoting the arrival time of the task for vehicle V_m, which is equivalent to the arrival time of its initial task. For each vehicle V_m, the set of subtasks is denoted as $Task_m = \{task_{m,1}, task_{m,2}, \dots, task_{m,K}\}$. Specifically, the first subtask, $task_{m,1}$, of vehicle V_m can either be executed within the confines of the vehicle itself or be offloaded to the RSU, the assisted vehicle V_0, or other connected vehicles. Upon completion of the final subtask, $task_{m,K}$, the result must be transmitted back to vehicle V_m.

A task can be rigidly constrained by a sequential order [36], so we divided the task into sequential subtasks. Take the scenario in Fig. 2 as an example. Tasks arrive randomly within a given time, and the start time distribution represents the distribution of initial task arrival times. For the case where $M = 3$, we have partitioned the tasks of the general vehicle V_1 into 4 distinct subtasks each, and the latter subtask can be executed only when its former subtask is done for the execution of a subtask usually depends on the output of other tasks. The first subtask of V_1 executes on local. Subsequently, upon successful completion of the first subtask, the second subtask for the general vehicle V_1 is generated on the current device V_1 and then offloaded to RSU. Furthermore, the third subtask of the general vehicle V_1, which is also generated within the RSU's domain, is offloaded and executed at the assisted vehicle. Lastly, the final subtask of vehicle V_1 is executed within vehicle V_2, and the resulting data is transmitted back to V_1.

In our model, tasks randomly arrive at every vehicle and every task can be divided into a subtasks sequence shorter than K subtasks. Subtasks wander on the topological links and we want to find an offloading decision sequence to minimize the sum of the delay of subtasks, which can be regarded as a paths search process. By thoroughly investigating and optimizing the various offloading paths for these subtasks, we can discern the most advantageous and optimal offloading policies for the integrated V2V and V2I system. Such an analysis aids in identifying the most efficient approach to task allocation and execution in this joint communication framework.

B. Delay Computing Model

The delay computing model consists of three parts: i) The execution time, which depends on the amount of computation and the computing capability of devices; ii) The transmission time, calculated based on the data size and the transmission rate if a link exists; and iii) The waiting time, resulting from the offloading policies of previous subtasks and the limitations posed by the available computing resources.

1) Execution Time: In the proposed joint V2I and V2V system, all devices Dev_l, where $l \in \{0, 1, \dots, M+1\}$, possess the capability to execute subtasks assigned to them. Here, Dev_0 corresponds to the assisted vehicle V_0, Dev_{M+1} corresponds to the RSU and $Dev_1, Dev_2, \dots, Dev_M$ correspond to the general vehicles $V_1, V_2, \dots, V_M$, respectively.

The execution time of $task_{m,k}$ is denoted as $T_{m,k}^{exe}$.

$$T_{m,k}^{exe} = \sum_{l=0}^{M+1} T_{m,k,l}^{exe} = \sum_{l=0}^{M+1} \frac{w_{m,k}}{f_l} I_{m,k,l}^{exe}, \tag{1}$$

where $w_{m,k}$ represents the computation workload of $task_{m,k}$ ($m \in \{1, 2, \dots, M\}$ and $k \in \{1, 2, \dots, K\}$). Additionally, f_l denotes the computing capability of device Dev_l, measured in cycles per second.

Furthermore, we define an indicator $I_{m,k,l}$, which serves to determine whether $task_{m,k}$ is executed on device Dev_l. The calculation of this indicator is given by:

$$I_{m,k,l}^{exe} = \begin{cases} 1 & \text{if } task_{m,k} \text{ is executed on } Dev_l, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

2) Transmission Time: In the proposed system, both the RSU and the assisted vehicle are capable of communicating with all general vehicles. However, communication links between general vehicles are restricted due to privacy concerns, trust authorizations, and communication intentions. These restrictions are represented by $mask_m = [mask_{m,1}, mask_{m,2}, \dots, mask_{m,M}]$, where $m \in \{1, 2, \dots, M\}$.

$$mask_{m,m'} = \begin{cases} 1 & \text{if } V_m \text{ can communicate with } V_{m'}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

In the given context, it is important to note that $task_{m,k}$ is not always transmitted from vehicle V_m, except when $k = 1$. We introduce $CD_{m,k}$ to represent the current device responsible for executing $task_{m,k}$. When $k > 1$, $CD_{m,k}$ is equal to the device that executes $task_{m,k-1}$. Consequently, the transmission time of $task_{m,k}$ is calculated by:

$$T_{m,k}^{tr} = \sum_{l=0}^{M+1} T_{m,k,l}^{tr} = \sum_{l=0}^{M+1} \frac{D_{m,k}}{R_{cur,l}} I_{m,k,l}^{tr} mask_{cur,l}, \tag{4}$$

where cur represents $CD_{m,k}$ and $D_{m,k}$ represents the data size of $task_{m,k}$. Additionally, $I_{m,k,l}^{tr}$ is an indicator value that denotes whether $task_{m,k}$ is transmitted to the target device Dev_l. $R_{cur,l}$ is the transmission rate from the current device $CD_{m,k}$ to the target device Dev_l. Moreover, we have the indicator $mask_{cur,l}$, which specifies whether a link exists from $CD_{m,k}$ to Dev_l. The calculation of $R_{cur,l}$ is given by:

$$R_{cur,l} = B_{cur,l} \log_2 \left(1 + \frac{P|h|^2}{\omega_0 (d_{cur,l})^{\vartheta}}\right), \tag{5}$$

where h is the channel fading coefficient, P is the transmission power, ϑ is the path loss exponent, and ω_0 is the white Gaussian noise power. Furthermore, we have $B_{cur,l}$ and $d_{cur,l}$ which stand for the bandwidth and the distance between the current device $CD_{m,k}$ and the target device Dev_l, respectively.

3) Waiting Time: When the vehicle or the RSU is occupied by the current subtask $task_{m,k}$, other subtasks offloaded to the occupied device need to wait until subtask $task_{m,k}$ is

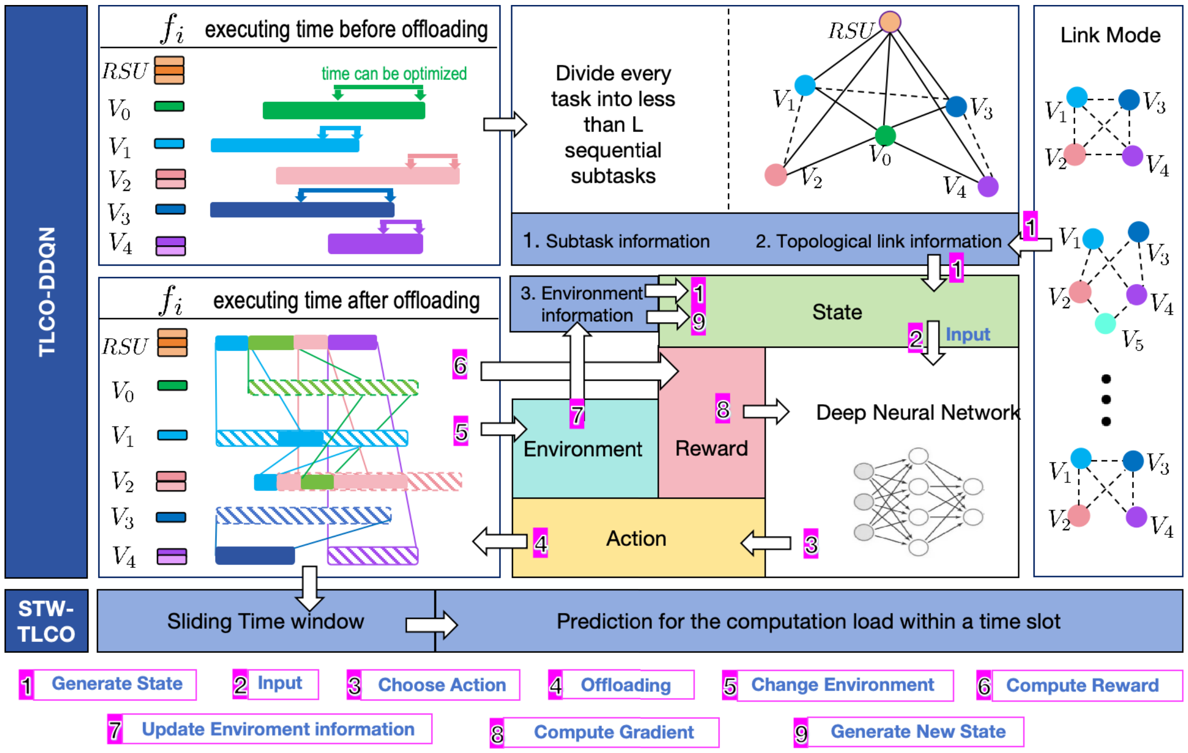


Fig. 3. The overall architecture of the proposed TLCO-DDQN and STW-TLCO frameworks. Vehicle tasks arrive randomly, and each task is divided into fewer than L sequential subtasks. The computation workloads and data size of each subtask constitute its subtask information. Steps 1 to 8 represent the process of completing one subtask. Topological information is encoded by link modes, while environmental information includes vehicle speeds, locations, and task start times. After the proposed TLCO-DDQN framework, we input offloading decisions into the STW-TLCO framework and utilize sliding time windows to compute the system's computation load within a time slot. This enables us to predict the computation load by fitting the relationship between computation load and pulse parameters, which represent various start time distributions of vehicle tasks.

completed. The waiting time $T^{W}_{m,k,l}$ when $task_{m,k}$ chooses to be executed on the device Dev_l is calculated by:

$$T^{W}_{m,k,l} = \max\{T^{occ}_{l} - T^{A}_{m,k,l}, 0\} I^{exe}_{m,k,l}, \ \ \forall l \in \{0, 1, \ldots, M+1\}, \tag{6}$$

where T^{occ}_{l} is the occupied time of the device Dev_l and calculated by $T^{occ}_{l} = \max\{\sum_{j=1}^{M}(T^{start}_{j} + \sum_{i=1}^{k_j} \triangle T_{j,i,l})\}$, k_j is the number of finished subtasks of V_j, and $T^{A}_{m,k,l}$ is the arrival time of $task_{m,k}$ when it arrives at the device Dev_l, which is calculated by $T^{A}_{m,k,l} = T^{start}_{m} + \sum_{i=1}^{k-1}\sum_{l=0}^{M+1} \triangle T_{m,i,l} + T^{tr}_{m,k,l}$. $\triangle T_{m,k,l}$ is the total delay of $task_{m,k}$ executed on the device Dev_l, which is calculated by:

$$\triangle T_{m,k,l} = T^{exe}_{m,k,l} + T^{tr}_{m,k,l} + T^{W}_{m,k,l}. \tag{7}$$

C. Problem Formulation

In order to optimize the utilization of limited computing resources and expedite task completion, we minimize the total delay of tasks by identifying the best offloading policies. To achieve this, we formulate the delay minimization problem for the dynamic joint V2I and V2V system as a path optimization problem, taking into account the possibility of offloading tasks based on their sequential subtask dependencies. The formulation of the problem is as follows:

$$(\mathcal{P}_1) \ \min_{I^{exe}_{m,k,l}, I^{Tr}_{m,k,l}} : \ T^{sum} = \sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{l=0}^{M+1} \triangle T_{m,k,l}, \tag{8}$$

$$\text{s.t.}: \ \sum_{l=0}^{L} I^{exe}_{m,k,l} = 1, \tag{9}$$

$$\sum_{l=0}^{M+1} I^{Tr}_{m,k,l} = 1, \tag{10}$$

$$\sum_{l=0}^{M+1} I^{Tr}_{l,K,m} = 1, \tag{11}$$

$$I^{exe}_{m,k,l}, I^{Tr}_{m,k,l} \in \{0, 1\}, \tag{12}$$

where $m \in \{1, 2, \ldots, M\}$ and $k \in \{1, 2, \ldots, K\}$. Eq. (9) indicates that $task_{m,k}$ can only be executed on one device, Eq. (10) indicates that $task_{m,k}$ can only be transmitted to one device, and Eq. (11) indicates that the last subtask $task_{m,K}$ needs to be transmitted back to its vehicle.

IV. Topological Link-Aware DDQN-Based Task Co-Offloading Algorithm

Fig. 3 illustrates the overall framework of our proposed method, which is composed of TLCO-DDQN and STW-TLCO, where f_i denotes the computing capability of device Dev_l. Tasks arrive at the joint V2V and V2I system randomly and can be divided into less than K sequential subtasks. As shown in Fig. 3, when the computing capability of RSU is tripled then V_0, V_1 and V_3, subtasks of V_0, V_1 and V_3 tend to offload subtasks to devices that have larger computing capability than itself so that the whole tasks can be done as soon as possible. However, a device that has

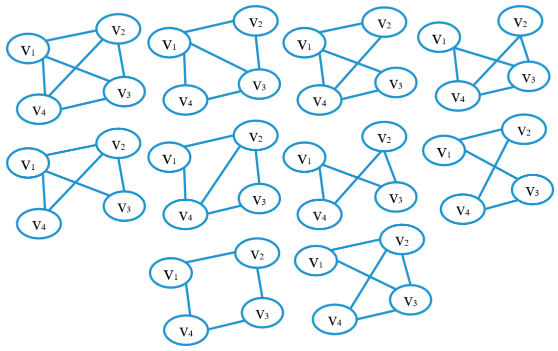


Fig. 4. Possible structures modes of topological links when $M = 4$.

a larger computing capability can only serve one task at one moment for good quality of service (QoS). Therefore, to achieve the lowest delay for the joint V2V and V2I system, collaborative offloading-decision making is necessary. We propose TLCO-DDQN to provide optimal collaborative offloading decisions for the joint V2V and V2I system with topological links and further propose STW-TLCO frameworks to fit the amount of computation curve and predict the computation capacity of the joint V2V and V2I system at any time window. In TLCO-DDQN, we utilize Double Deep-Q Network (DDQN) [37] to determine the optimal offloading policies, which employs two deep neural networks with identical structures as function approximators to create a more stable and accurate estimate of the Q-values, mitigating the overestimation problem of Deep-Q Network (DQN) [38]. In STW-DDQN, we propose a prediction framework for the amount of computation based on TLCO-DDQN.

A. Topological Communication Structure

Due to the rapid movement of vehicles, privacy concerns, and specific communication intentions among the general vehicles, various possible communication link structures can emerge. These link structures are visually depicted in Fig. 4 when $M = 4$ and the degree of each node exceeds two. To denote the set of these topological structure modes, we use $Mode_M = \{MO_1, MO_2, \ldots, MO_{O_M}\}$, where M represents the total number of general vehicles, and O_M denotes the number of modes with nodes having degrees greater than two.

B. TLCO

To address the problem of finding the optimal offloading policies, we formulate the delay minimization problem for sequential subtasks in the joint V2I and V2V system as a path optimization problem. This problem can be modeled as a Markov Decision Process (MDP), where the transition to the next state depends solely on the current state. Leveraging the capabilities of Deep Reinforcement Learning (DRL), we can efficiently identify the best offloading policies. Here, we represent the state, action, and reward at time step t_p as $s(t_p)$, $a(t_p)$, and $r(t_p)$, respectively. By employing DRL, we dynamically learn and update the most effective offloading decisions based on the current state of the system, leading to improved task execution and minimized delays.

1) State Space: In the proposed V2I and V2V system, we define the state of the time step t_p as:

$$s(t_p) = (s_1^p, \quad s_2^p, \quad \cdots, \quad s_M^p) = \begin{pmatrix} w_{1,1}^p & w_{2,1}^p & \cdots & w_{M,1}^p \\ D_{1,1}^p & D_{2,1}^p & \cdots & D_{M,1}^p \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,K}^p & w_{2,K}^p & \cdots & w_{M,K}^p \\ D_{1,K}^p & D_{2,K}^p & \cdots & D_{M,K}^p \\ D_{1,K+1}^p & D_{2,K+1}^p & \cdots & D_{M,K+1}^p \\ MO & MO & \cdots & MO \\ x_1^p & x_2^p & \cdots & x_M^p \\ v_1^p & v_2^p & \cdots & v_M^p \\ T_1^{start} & T_2^{start} & \cdots & T_M^{start} \end{pmatrix} \tag{13}$$

where $D_{m,K+1}^p$ is the result of the last subtask that needs to be transmitted back to the vehicle V_m, $MO \in Mode_M$ is the mode of topological link. x_m^p is the current location of V_m in the p-th time step, and v_m^p is the speed of V_m. $w_{m,k}^p$ and $D_{m,k}^p$ are calculated by:

$$w_{m,k}^p = \begin{cases} 0, & \text{if } task_{m,k} \text{ has been executed before } t_p, \\ w_{m,k}, & \text{otherwise.} \end{cases} \tag{14}$$

$$D_{m,k}^p = \begin{cases} 0, & \text{if } task_{m,k} \text{ has been executed before } t_p, \\ D_{m,k}, & \text{otherwise.} \end{cases} \tag{15}$$

2) Action Space: In each time step, we execute the earliest subtasks and get the vehicle index of the subtask $task_{t_p}$ in t_p by:

$$m_p = \operatorname{argmin} \left\{ T_m^{start} + \sum_{k=1}^{k_m} \sum_{l=0}^{M+1} \triangle T_{m,k,l} \right\}, \tag{16}$$

where k_m is the number of finished subtasks of V_m. We further get the current subtask index by:

$$k_p = k_{m_p} + 1. \tag{17}$$

The action space is $\mathscr{A} = [a_0, a_1, \cdots, a_{M+1}]$, where $a(t_p) = a_0$ means the subtask $task_{m_p,k_p}$ is executed on the assisted vehicle, $a(t_p) \in [a_1, a_2, \ldots, a_M]$ means the subtask $task_{m_p,k_p}$ is executed on a certain general vehicle, and $a(t_p) = a_{M+1}$ means the subtask $task_{m_p,k_p}$ is executed on the RSU.

3) Reward: Significantly, when $t_p > 1$, the tasks $task_{m_p,k_p}$ are generated on CD_{m_p,k_p-1} instead of on V_{m_p}. We denote the device on which $task_{m_p,k_p}$ is generated as $cd_p = CD_{m_p,k_p-1}$. Therefore, it is necessary to check whether the action $a(t_p)$ is present in $mask_{cd_p}$.

We define $Condition$ 1 as follows: If $a(t_p)$ is not in $mask_{cd_p}$ and $k_p = K + 1$, then the result of tasks is preferred to be transmitted back to the V_{m_p}. Similarly, we define $Condition$ 2 as follows: If $a(t_p)$ is not in $mask_{cd_p}$ and $k_p < K + 1$, then $task_{m_p,k_p}$ is preferred to be transmitted

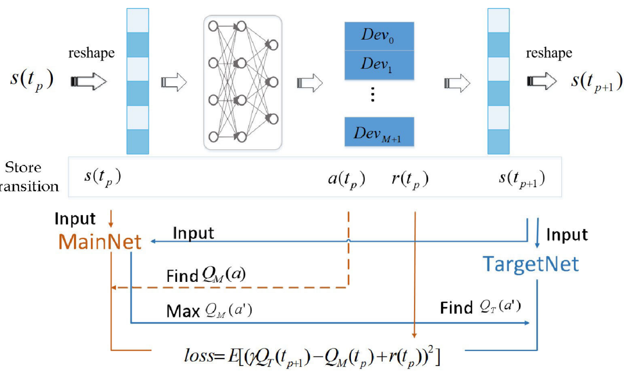


Fig. 5. An illustration of the proposed *TLCO-DDQN* algorithm.

to the connected devices that are in $mask_{cd_p}$.

$$r(t_p) = \begin{cases} -C_1, & \text{if } Condition \text{ 1 holds,} \\ -C_2, & \text{if } Condition \text{ 2 holds,} \\ -\triangle T(t_p) + C_3, & \text{otherwise,} \end{cases} \quad (18)$$

where $\triangle T(t_p) = T^{exe}_{m_p,k_p,a(t_p)} + T^{tr}_{cd_p,k_p,a(t_p)} + T^{W}_{cd_p,k_p,a(t_p)}$, C_1, C_2 and C_3 are three different constants to adjust the punishment and reward based on the preference of the optimization problem.

As shown in Fig. 5, the idea of DDQN is to decouple action selection from action evaluation, effectively mitigating the overestimation bias commonly observed in Q-Learning algorithms. To achieve this, DDQN utilizes two distinct Q-value approximations: MainNet and TargetNet, where MainNet serves as a deep neural network responsible for approximating the Q-values Q_M for various state-action pairs, while TargetNet, another separate deep neural network, provides the target Q-values Q_T used during the training process. DDQN estimates the value of an action based on one set of weights while selecting actions based on another set of weights. The TargetNet operates as an independent copy of the MainNet and undergoes periodic updates with the weights from the main network. The Q-function of MainNet is updated by:

$$\hat{Q}_M(t_p) = r(t_p) + \gamma Q_T\left(s(t_{p+1}), \underset{a(t_{p+1})}{\arg\max}\, Q_M(t_{p+1})\right), \quad (19)$$

where $s(t_{p+1})$ and $a(t_{p+1})$ are the next state and the action, respectively; $\hat{Q}_M(t_p)$ is the prediction of the Q-value of the MainNet in t_p and $Q_T(t_{p+1})$ is the Q-value of the TargetNet when choosing action $a(t_{p+1})$ under state $s(t_{p+1})$. The detailed algorithmic process is as described in **Algorithm 1**.

C. STW-TLCO

In the proposed V2V and V2I system, communication between devices occurs through topological links, and the TLCO algorithm is employed to determine the most optimal offloading policies. To accurately calculate the computation load of the entire V2V and V2I system at different times, sliding time windows are utilized. This approach involves computing the computing capacity within each time window in a pulsed workflow, wherein tasks arrive randomly during a slot. Specifically, when a task from vehicle V_m enters the system, it arrives at time T^{start}_m, where T^{start} lies within the range of $(0, T^{start}_{\max}(\delta))$. Here, δ represents the pulsed parameter, and $T^{start}_{\max}(\delta)$ corresponds to the time slot of the pulsed workflow where tasks from multiple vehicles arrive randomly during the duration of the multiple of δ. To effectively model the entire process, timelines are constructed for pulsed workflows of δ. This facilitates the organization and analysis of tasks and computation loads within the defined time intervals.

Algorithm 1 TLCO: Topological Link-Aware Task Co-Offloading Algorithm Based on DDQN for Joint V2V and V2I System

Input: The initial subtask feature and vehicle feature
Output: optimal offloading policies for input

```
1:  Initialize network parameters $\theta$, update steps $n$, $\mathcal{D}$, structure mode $Mode_M$
2:  for episode do
3:      while not all subtasks are executed do
4:          Get the current vehicle index $m_p$ by Eq. (16).
5:          Get the current subtask index by Eq. (17).
6:          Get $mask_{m_p}$ of vehicle $V_{m_p}$
7:          Calculate the current location $x(m_p)$ of $V_{m_p}$
8:          Input $s(t_p)$ to MainNet and get $Q_M(t_p), a_l)$, $l = 0, 1, 2, \cdots, M+1$, and choose $a(t_p) = \arg\max_a Q_M(t_p)$ according to $\epsilon-$greedy policy
9:          if $k = K$ then
10:             $a(t_p) = a_{m_p}$
11:         end if
12:         Compute $r(t_p)$ by Eq. (18)
13:         Generate the new state $s(t_{p+1})$
14:         Save $[s(t_p), a(t_p), r(t_p), s(t_{p+1})]$ in $\mathcal{D}$
15:         if Training Step then
16:             Sample memories from $\mathcal{D}$
17:             Input $s(t_p)$ to MainNet and get $Q_M(t_p)$
18:             Input $s(t_{p+1})$ to MainNet and get $a(t_{p+1}) = \arg\max_{a_l} Q_M(t_{p+1})$
19:             Input $s(t_{p+1})$ to TargetNet and get $Q_T(t_{p+1})$
20:             Compute the prediction Q-value by Eq. (19)
21:             Update the parameter of the MainNet by minimizing $|Q_M(s,a) - \hat{Q}_M(s,a)|^2$
22:             Copy the parameter of the MainNet to the TargetNet, every $n$ steps
23:         end if
24:     end while
25: end for
26: return optimal offloading decision for input
```

Thus, the start time of subtask $task_{m,k}$ is calculated by:

$$T^{start}_{m,k}(\delta) = \begin{cases} T^{start}_m(\delta), & \text{if } k = 1, \\ T^{start}_m(\delta) + \sum_{k=1}^{k-1} \triangle T_{m,k,a_k}, & \text{otherwise,} \end{cases} \quad (20)$$

where $a_k \in [0, M+1]$ is the result of the TLCO algorithm. $T^{start}_m(\delta) \in [0, C\delta]$ and C is a constant. The end time of subtask $task_{m,k}$ is equal to the start time of

$task_{m,k+1}$:

$$T^{End}_{m,k}(\delta) = T^{start}_{m,k+1}(\delta). \tag{21}$$

Based on the outcomes of the TLCO algorithm, it is determined that $task_{m,k}$ should be executed on Dev_{a_k}, which possesses a computing capability denoted as f_{a_k}. Consequently, the computation executed for $task_{m,k}$ when the sliding time window starts at t_{wd} can be calculated by:

$$Com(t_{wd},\delta)_{m,k} = \begin{cases} f_{a_k}(\min\{T^{End}_{m,k}(\delta), t_{wd}+t_u\} - T^{start}_{m,k}(\delta)), \\ \text{if } t_{wd} < T^{start}_{m,k}(\delta) < t_{wd}+t_u; \\ f_{a_k}(\min\{T^{End}_{m,k}(\delta), t_{wd}+t_u\} - t_{wd}), \\ \text{if } \max\{t_{wd}-t_u, 0\} < T^{start}_{m,k}(\delta) < t_{wd}; \\ 0, \text{otherwise}. \end{cases} \tag{22}$$

To simplify, we consider non-overlapping time windows, so the total computation of the proposed V2V and V2I system is:

$$Com(t_{wd},\delta) = \sum_{m}^{M}\sum_{k}^{K} Com(t_{wd},\delta)_{m,k}. \tag{23}$$

The detailed algorithmic process is as described in **Algorithm 2**.

Algorithm 2 STW-TLCO: Prediction Framework of the Amount of Computation Based on Sliding Time Windows on TLCO for Joint V2V and V2I System

```
Input: The window size $t_u$, $T^{start}_{max}$
Output: The predicted value of the amount of computation
1:  Initialize well-trained model of TLCO, pulsed parameter $\delta$
2:  for $\delta$ do
3:      for episode do
4:          Initialize $t_{wd} = 0$ and the amount of computation $Com(t_0,\delta) = 0$
5:          for window iter do
6:              for subtasks do
7:                  Obtain Optimal offloading decisions $\{a_0, a_1, \ldots, a_K\}$ by TLCO
8:                  Compute $T^{start}_{m,k}(\delta)$ by Eq. (20)
9:                  Compute $T^{task}_{end}(\delta)$ by Eq. (21)
10:                 Compute $Com(t_{wd},\delta)_{m,k}$ by Eq. (22)
11:             end for
12:             Compute $Com(t_{wd},\delta)$ by Eq. (23)
13:             $t_{wd} = t_{wd} + t_u$
14:         end for
15:     end for
16:     Compute the average amount of computation over episodes
17:     Gaussian fitting get Gaussian parameters set $G(\delta) = \{a^\delta, b^\delta, c^\delta\}$
18: end for
19: Predict $\hat{a}^{\delta'}$, $\hat{b}^{\delta'}$ and $\hat{c}^{\delta'}$ for the pulsed workflow of $\delta'$ and further predict the amount curve of computation
20: return The prediction of the amount curve for $\delta'$
```

D. Complexity Analysis

The complexity of the proposed TLCO-DDQN algorithm is $\mathcal{O}(N_{train})$, where N_{train} is the number of training epochs [37]. The complexity of the proposed STW-TLCO algorithm is $\mathcal{O}(N_w \times episode \times M \times L)$, where N_w is the number of time windows. Inference complexity of TLCO-DDQN is $(N_{input} \times N_{hide_1} \times N_{hide_2} \times N_{output})$ FLOPs, where N_{input} is $M \times (2 \times K + 7) + 1$, N_{output} is $M + 2$, and N_{hide_1} and N_{hide_2} are the number of nodes in hidden layers.

V. Performance Evaluation

A. Baselines

We compare the proposed algorithm *TLCO-DDQN* with six other offloading algorithms to validate the superiority of *TLCO-DDQN*, as shown below:

- **TLCO algorithm based on DQN (*TLCO-DQN*)**: DQN has emerged as one of the prominent algorithms for addressing Markovian decision-making processes. However, it differs from the *TLCO-DDQN* algorithm in terms of Q-value computation. Specifically, in DQN, the Q-value is calculated as $\hat{Q}_M(t_p) = r + \gamma \max a_{p+1} Q_T(t_{p+1})$.
- **TLCO algorithm based on DuelingDQN (*TLCO-DuelingDQN*)**: Dueling Deep Q-learning Network (DuelingDQN) is the improved version of DQN, which aims to solve the overestimation issue that may occur in DQN. Compared with *TLCO-DQN*, the output layer of *TLCO-DuelingDQN* is divided into two parts: one is the value of the state $Vs(s_{t_p})$, and the other one is the advantages for each action $Adv(s_{t_p}, a_{t_p})$. The Q-value of the target net is $Q_T(t_p) = Vs(s_{t_p}) + Adv(s_{t_p}, a_{t_p})$ and $Q_M(t_p) = r(t_p) + \gamma \max_{a_{t_{p+1}}} Q_T(s_{t_{p+1}}, a_{t_{p+1}})$.
- **Greedy**: The determination of offloading decisions for subtasks relies on the order in which they are generated. Specifically, we compare the time delay of all available policies for the current subtask $task_{m_p,k_p}$ and select the most optimal policy that incurs the least delay in the current step, taking into account the completion time of all tasks. This process continues iteratively until all tasks are successfully finished.
- **Greedy-Noseg**: Compare the time delay of all available policies for the whole task $task_m$ and select the most optimal policy that incurs the least delay for the vehicle. This process continues iteratively until all tasks are successfully finished. We utilize the Greedy-Noseg method to provide proof of the effectiveness of the sequential partial offloading scheme.
- **All Edge**: This approach enables multiple vehicles to offload their tasks to the RSU in the proposed V2I and V2V system. Subsequently, the RSU processes these tasks and transmits the results back to the respective vehicles.
- **All Local**: To emphasize the significance of the RSU in expediting task completion, we incorporate the *All Local*

TABLE II
PARAMETER SETTINGS

Parameter	Value
v_0	$16, 18, 20$ m/s
v_m	$[15, 20]$ m/s
M	$5, 6, 7$
K	$5, 6, 7$
P	1.3
ϑ	4
ω_0	3×10^{-13}
h	4
w	$[0, 100] \times 10^6$ cycles
D	$[0, 10] \times 10^6$ bits
f_{RSU}	3 GHz
f_m	$[100, 1000]$ MHz
B_{RSU}	4 MHz
B_m	[1,3] MHz
RSU location	100
C_1, C_2, C_3	$6, 5, 2$
t_u	0.02s

algorithm into our baseline comparison, where multiple vehicles execute their own tasks.

- **Random**: To simulate a realistic stochastic environment, we also compare our algorithm with the *Random* offloading algorithm, which involves the random assignment of subtasks to any available device within the network, including the RSU, assisted vehicles, and other connected general vehicles.

B. Parameter Settings

The simulation settings are presented in Table II, which are primarily based on [39]. We assume that the tasks of all vehicles arrive at random moments within 1s, and the task size is randomly generated according to the range specified in Table II. Both the training set and test set are generated in the same manner, with the training set consisting of 10,000 data samples and the test set consisting of 200 data samples. For the training process of DDQN, DQN, and DuelingDQN, we configure the following parameters: a maximum training episode of $Maxiter = 15,000$, a learning rate of $lr = 0.005$, a reward decay rate of 0.99, a ϵ-greedy rate of 0.99, a replace target iter of 350, and a memory size of 13,000. As for the *TLCO-DDQN* architecture, it comprises three layers: the first layer serves as the input layer with dimensions $M \times (2 \times K + 7) + 1$, the subsequent layer is the hidden layer containing 128 nodes, and the last layer is the output layer with $M + 2$ nodes.

C. Experimental Results

1) Impact of the Number of Subtasks: We calculate the average delay of the different methods for different K used to explore the effect of the number of subtasks K on the results. As depicted in Fig. 6, emTLCO-DDQN demonstrates significantly superior performance compared to other methods. For instance, when $M = 5$ and $K = 5$, *TLCO-DDQN* exhibits noteworthy improvements over *TLCO-DQN* by 5.19%, *TLCO-DuelingDQN* by 3.23%, *Greedy* by 22.57%, *Greedy-Noseg* by

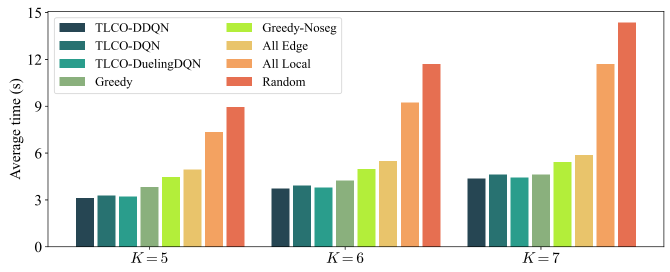


Fig. 6. The average delay of different numbers of subtasks, where $K = [5, 6, 7]$.

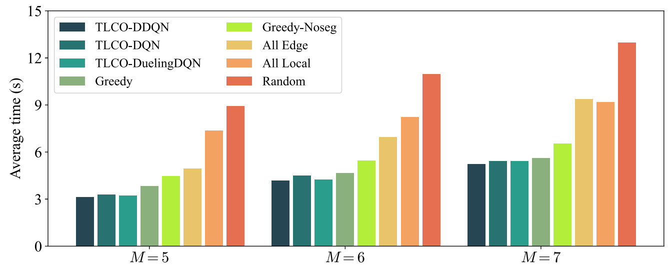


Fig. 7. The average delay of different numbers of vehicles, where $M = [5, 6, 7]$.

43.34%, *All Edge* by 58.72%, *All Local* by 136.12%, and *Random* by 186.82%. As K increases, the average execution time also increases due to the rise in the total computation workload of subtasks. Meanwhile, the total computing capacity of devices remains unchanged, leading to a narrower optimization space. Consequently, the advantages of DRL-based algorithms diminish. Nonetheless, *Greedy* achieves the best offloading decision among the non-DRL-based algorithms, while the three DRL-based algorithms, *TLCO-DDQN*, *TLCO-DQN*, and *TLCO-DuelingDQN*, consistently outperform the non-DRL-based algorithms across varying K values.

2) Impact of the Number of Devices: Fig. 7 shows the results of the average delay of different methods under different M. We set $M = 5, 6, 7$ and we recommend establishing a separate V2V and V2I cluster with other RSUs for scenarios involving more vehicles because of communication qualities and the limitation of the computing resources of RSUs. As depicted in Fig. 7, emTLCO-DDQN achieves significantly better results than other methods. For instance, when the experimental setup is $M = 6$ and $K = 5$, *TLCO-DDQN* achieves remarkable improvements over *TLCO-DQN* by 7.57%, *TLCO-DuelingDQN* by 1.52%, *Greedy* by 11.44%, *Greedy-Noseg* by 30.35%, *All Edge* by 66.47%, *All Local* by 96.91%, and *Random* by 162.31%. As the value of M increases, the average execution time also increases. Interestingly, the results of the DRL-based algorithms show more efficiency improvements compared to when K increases. This is due to the fact that with an increase in M, the available devices for vehicles also increase, thereby expanding the optimization space. The *Greedy* algorithm achieves the best offloading decision among the non-DRL-based algorithms, while the three DRL-based algorithms *TLCO-DDQN*, *TLCO-DQN*, and *TLCO-DuelingDQN* consistently outperform the non-DRL-based algorithms for various M values. The superiority of DRL-based approaches becomes evident as the

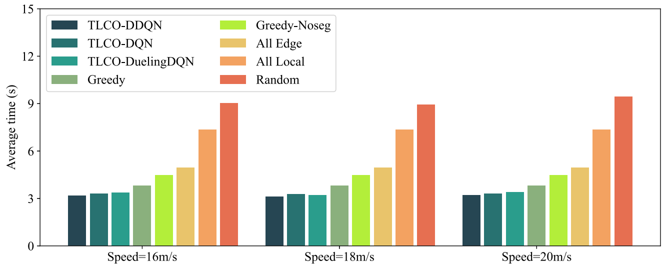


Fig. 8. The average delay of different speeds of the assisted vehicle, where $v_0 = [16, 18, 20]$ m/s.

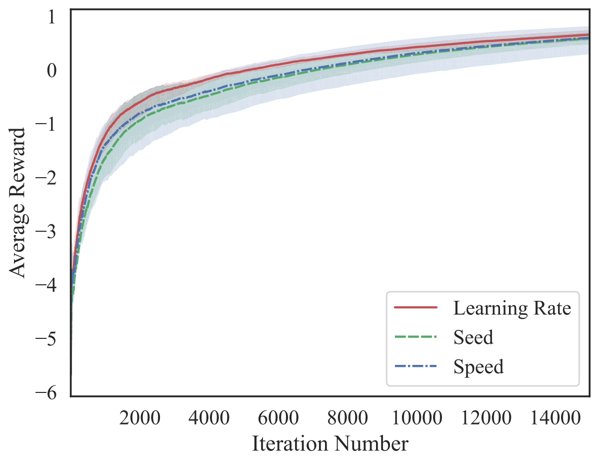


Fig. 9. Comparing the effect of learning rate, seed, and speed on the reward of the training process.

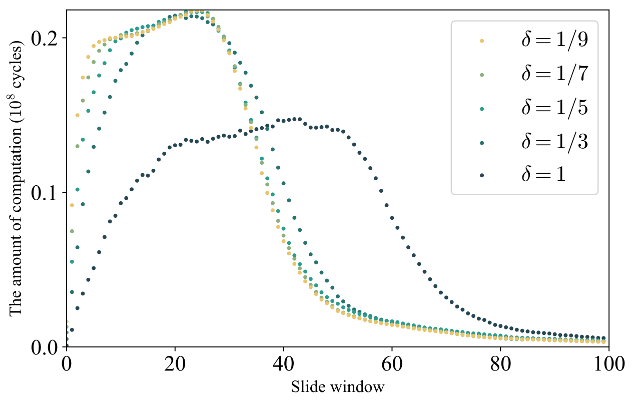


Fig. 10. The amount of computation in varying time windows when $\delta \in [1/9, 1/7, 1/5, 1/3, 1]$.

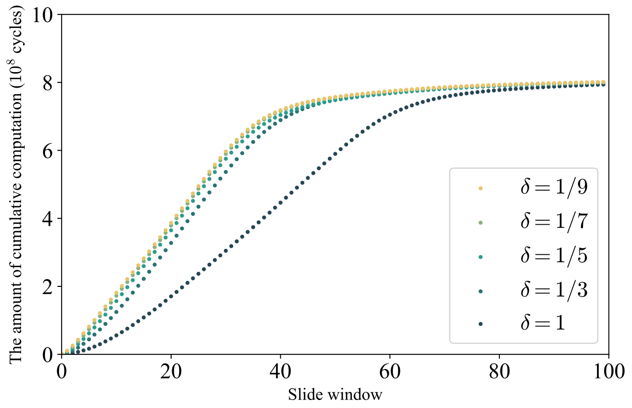


Fig. 11. The amount of cumulative computation in varying time windows when $\delta \in [1/9, 1/7, 1/5, 1/3, 1]$.

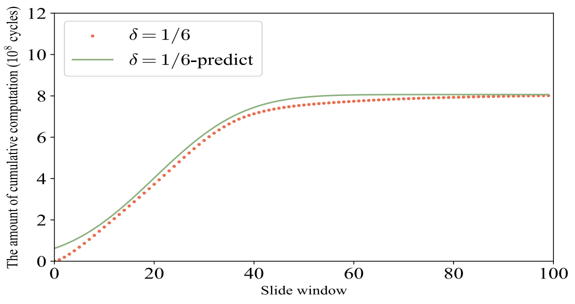
optimization space widens with the increment in available devices, showcasing their effectiveness in handling diverse offloading scenarios within the V2I and V2V system.

3) Impact of the Speed of the Assisted Vehicle: The speeds of the task vehicles are randomly selected from the range [15, 20] m/s, so we conduct a comparison by setting the speeds of the assisted vehicle to 16, 18, and 20 m/s. As depicted in Fig. 8, the variation in the speed of the assisted vehicle has a negligible effect on the experimental delay. This observation demonstrates the robustness of our algorithm in identifying the optimal offloading strategy, irrespective of changes in vehicle speed.

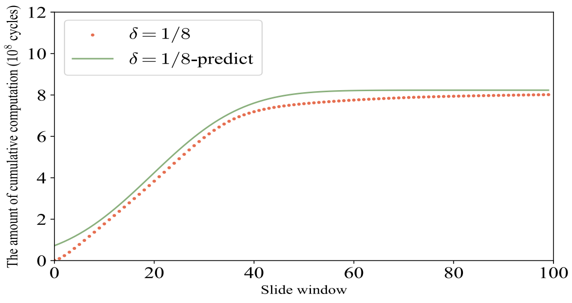
4) Convergence Performance: Considering the randomness of the results caused by different random seeds, we set seeds as 0, 500, and 1000 for *TLCO-DDQN* to reduce the effect of randomness on the experimental results. In addition to this, we also compare the effect of different learning rates and speeds of the assisted vehicle on the experimental results, with its learning rate set at 0.005, 0.007, 0.01 and speed set at 16, 18, and 20 m/s. As depicted in Fig. 9, the different color of the scope indicates the average rewards of the different parameter sets of *TLCO-DDQN*. The narrower the scope, the smaller the effect of different parameters. Different speeds and seeds have almost no effect on the convergence rate and results, while different learning rates only accelerate the convergence rate, and the convergence results are consistent with the different speeds and seeds situation.

5) Impact of the Pulsed Parameter δ: We set the pulsed parameter $\delta \in [1/9, 1/7, 1/5, 1/3, 1]$, which corresponds to the degree of task crowding. As δ increases, the arrival time of tasks is delayed, resulting in a decrease in task density. The time window size is set to 0.1 seconds, with an interval of 0.02 seconds between time windows. As depicted in Fig. 10, decreasing δ results in the wave crest of computation arriving earlier and reaching higher levels, which leads to earlier task completion. For instance, the computation amount drops to 0 around the 100-th slide window for $\delta = 1$, whereas it falls to zero around the 90-th slide window for $\delta = 1/7$. Additionally, the wave crest for $\delta = 1/7$ is higher and arrives earlier. When tasks are sparse, the peak of computation occurs later, while for dense tasks, the peak appears sooner. However, due to limitations in overall computational resources, when the task density becomes sufficiently high, there is no significant difference in computational peaks, although these peaks occur earlier. To better illustrate the impact of different pulse parameters on computation, we report the curve of cumulative computation over the time window in Fig. 11. Curves with smaller pulse parameters exhibit steeper slopes, indicating a faster rate of increase in computation over time. In contrast, curves with larger pulse parameters, such as $\delta = 1$, show more gradual slopes, reaching a maximum value within the 0-20 time window, which implies a slower, more gradual increase in computation.

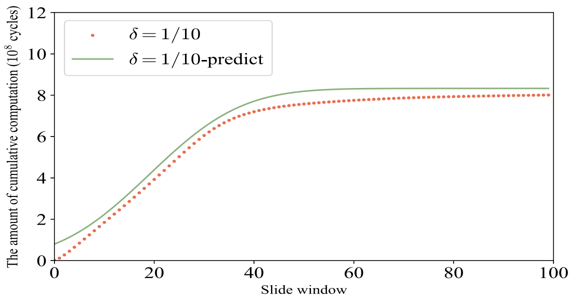
6) Gaussian Curve Fitting: Inspired by the graphics of the amount of computation in Fig. 10, we conjecture that the variation of the amount of computation in the sliding window



(a) $\delta = 1/6$



(b) $\delta = 1/8$



(c) $\delta = 1/10$

Fig. 12. The difference between the predicted and actual value of the amount of computation with varying δ.

conforms to a Gaussian distribution. Thus, we employ Eq. (24) to fit the data and obtain estimates for the Gaussian parameters. We set $\delta \in [1/9, 1/7, 1/5, 1/3, 1]$, along with the parameters a, b, and c for the Gaussian curve. Here, a determines the amplitude of the Gaussian curve corresponding to the peak of computation, b determines the position of the curve's peak corresponding to the sliding window's position, and c determines the width of the Gaussian curve corresponding to the time needed for all tasks to be completed.

$$y = a \cdot \exp\left[-\left(\frac{x-b}{c}\right)^2\right]. \tag{24}$$

7) Prediction for the Computation Load: We make predictions for the parameters of the Gaussian curve corresponding to the amount of computation within the sliding window for different values of $\delta \in [1/10, 1/8, 1/6]$. Specifically, we predict $\delta = 1/6$ by $\delta \in [1/5, 1/3, 1]$, and similarly, $\delta = 1/8$ by $\delta \in [1/7, 1/5, 1/3, 1]$, and $\delta = 1/10$ by $\delta \in [1/9, 1/7, 1/5, 1/3, 1]$. The process of predicting the parameters of the computation curve for the target δ involves fitting the parameters in Eq. 24 by new Gaussian curves.

TABLE III
PREDICTION ACCURACY FOR DIFFERENT δ

δ	#R-Squared	#RMSE
1/6	96.67%	0.0156
1/8	96.36%	0.0165
1/10	96.34%	0.0167

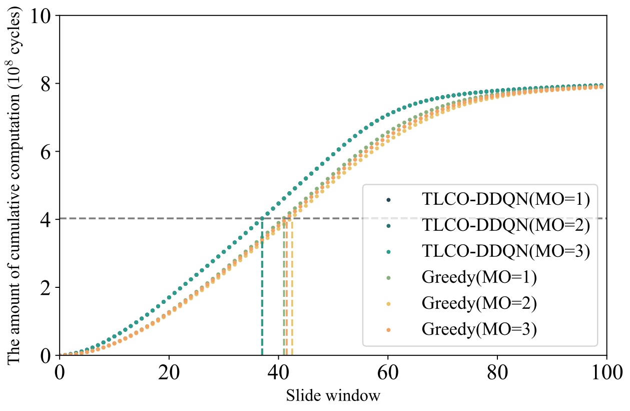


Fig. 13. Comparison of the amount of cumulative computation between TLCO and Greedy algorithms in different modes.

Finally, we draw the computational curve for the target δ based on the predicted parameters. These predicted values are compared to the actual values computed by *TLCO*. The results of these fits for various δ values are presented in Fig. 12. Moreover, we calculate R^2 and RMSE, respectively, as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(\bar{y}_i - y_i)^2}, \tag{25}$$

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}, \tag{26}$$

where R^2 shows the goodness of fit between the predicted values and the corresponding ground truth data. A higher R^2 value approaching 1 indicates a more accurate prediction. RMSE quantifies the deviation between the predicted values and the actual ground truth values. The results are presented in Table III, which indicate that the R-squared values for the fit-based predictions with ground truth at various δ are all consistently higher than 96%. Moreover, the RMSE values also demonstrate excellent results.

8) Impact of the Connection Mode MO: As shown in Fig. 13, we chose three different connection modes for analyzing the effect of different connection modes on *TLCO*. To better illustrate the differences in computational progress among different methods under the same task density, we added horizontal and vertical auxiliary lines, with lines of different colors corresponding to different methods. To better compare the superiority of TLCO, we additionally report the performance of the greedy algorithm under different connection modes. For a clear presentation of the experimental results, we only show the results of three modes that are shown in Fig. 14. The cumulative computation curves of TLCO under three different connection modes nearly overlap,

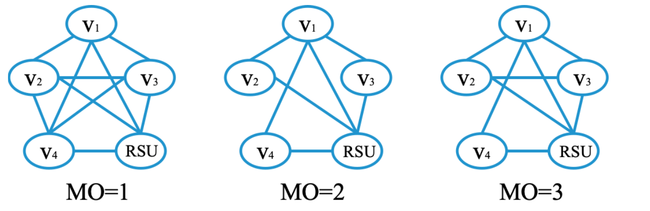


Fig. 14. Connection scenarios for modes 1, 2 and 3, where V_1 represents the assistance vehicle.

TABLE IV
3C INDEX OF DIFFERENT METHODS IN DIFFERENT LINK STRUCTURES

Method	MO=1	MO=2	MO=3
TLCO-DDQN	**1.3718**	**1.3700**	**1.3718**
TLCO-DQN	1.3679	1.369	1.3701
TLCO-DuelingDQN	1.3684	1.3694	1.3716
Greedy	1.3269	1.3589	1.3468
Greedy-Noseg	1.3154	1.3390	1.3213
All Edge	1.3078	1.3078	1.3078
All Local	1.0512	1.0512	1.0512
Random	0.9620	0.9072	0.9389

which indicates that TLCO achieves the optimal offloading strategy for different complex connection modes. In contrast, the greedy algorithm has different cumulative computation curves under different connection modes.

We calculate the slope of the midpoint of the curve in normalized coordinates, referred to as the Cluster Computation Capability Index (3C Index), to indicate the computational ability of V2V and V2I clusters. The results are shown in Table IV. Under the TLCO-DDQN offloading strategy, clusters with different link structures exhibit similar 3C Index values, indicating that computational capability is consistent across varying link structures. In contrast, under the greedy algorithm offloading strategy, clusters with different link structures show varying 3C Index values, suggesting that computational capability is influenced by link structures. These results demonstrate that TLCO is link-aware and effectively utilizes computational resources within V2V and V2I clusters across different link structures, while the greedy algorithm does not achieve the same level of adaptability.

VI. CONCLUSION

To achieve efficient task offloading when facing various topological link structures, we develop a fine-grained partially offloading model for sequential subtasks to optimal co-offloading policies in a dynamic V2I and V2V system with topological links, which allows for efficient utilization of computing resources. Additionally, we propose a topological links-aware offloading algorithm based on DDQN to minimize the overall delay. Through comprehensive experimental evaluations, we demonstrate that the proposed *TLCO-DDQN* algorithm significantly outperforms other DRL-based offloading algorithms and traditional greedy-based offloading approaches. Furthermore, we present a prediction framework based on the proposed TLCO algorithm and sliding time windows. This framework enables us to predict the Gaussian curve for the amount of computation. Consequently, with knowledge of the pulsed parameter δ, we can forecast the amount of computation within different time windows using the predicted Gaussian curve. The experimental results illustrate the high accuracy of the proposed *STW-TLCO* algorithm, with an R-squared value exceeding 96%.

For our future research, we intend to incorporate cloud computing into the joint V2V and V2I system and explore adaptive optimization techniques for vehicle-edge-cloud collaboration among different vehicles, aiming to further enhance the overall system performance.

REFERENCES

[1] A. Hosny, M. Yousef, W. Gamil, M. Adel, H. Mostafa, and M. S. Darweesh, "Demonstration of forward collision avoidance algorithm based on V2V communication," in *Proc. 8th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, May 2019, pp. 1–4.

[2] A. Chtourou, P. Merdrignac, and O. Shagdar, "Context-aware communication for V2V collision avoidance applications," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 1958–1963.

[3] L. Zhao, H. Chai, Y. Han, K. Yu, and S. Mumtaz, "A collaborative V2X data correction method for road safety," *IEEE Trans. Rel.*, vol. 71, no. 2, pp. 951–962, Jun. 2022.

[4] B. Liu et al., "A novel V2V-based temporary warning network for safety message dissemination in urban environments," *IEEE Internet Things J.*, vol. 9, no. 24, pp. 25136–25149, Dec. 2022.

[5] P. Keshavamurthy, E. Pateromichelakis, D. Dahlhaus, and C. Zhou, "Resource scheduling for V2V communications in co-operative automated driving," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[6] M. V. Baumann et al., "Cooperative automated driving for bottleneck scenarios in mixed traffic," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2023, pp. 1–8.

[7] H. Guo, Y. Wang, J. Liu, and N. Kato, "Super-broadband optical access networks in 6G: Vision, architecture, and key technologies," *IEEE Wireless Commun.*, vol. 29, no. 6, pp. 152–159, Dec. 2022.

[8] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[9] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8027–8038, Nov. 2022.

[10] F. Liu, J. Chen, Q. Zhang, and B. Li, "Online MEC offloading for V2V networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 6097–6109, Oct. 2022.

[11] A. Akinsanya, M. Nair, Y. Pan, and J. Wang, "A dynamic resource allocation scheme in vehicular communications," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 127–131.

[12] S. Wang, G. Chen, Y. Jiang, and X. You, "A cluster-based V2V approach for mixed data dissemination in urban scenario of IoVs," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 2907–2920, Mar. 2023.

[13] D. Zhao, H. Qin, B. Song, Y. Zhang, X. Du, and M. Guizani, "A reinforcement learning method for joint mode selection and power adaptation in the V2V communication network in 5G," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 2, pp. 452–463, Jun. 2020.

[14] Y. He, D. Wang, F. Huang, R. Zhang, X. Gu, and J. Pan, "A V2I and V2V collaboration framework to support emergency communications in ABS-aided Internet of Vehicles," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 4, pp. 2038–2051, Apr. 2023.

[15] B. L. Nguyen, D. T. Ngo, N. H. Tran, M. N. Dao, and H. L. Vu, "Dynamic V2I/V2V cooperative scheme for connectivity and throughput enhancement," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1236–1246, Feb. 2022.

[16] W. Fan et al., "Joint task offloading and resource allocation for vehicular edge computing based on V2I and V2V modes," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 4, pp. 4277–4292, Apr. 2023.

[17] M. V. Kadam, V. M. Vaze, and S. R. Todmal, "TACR: Trust aware clustering-based routing for secure and reliable VANET communications," *Wireless Pers. Commun.*, vol. 132, no. 1, pp. 305–328, Sep. 2023.

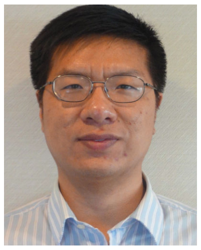
[18] A. B. de Souza, P. A. Leal Rego, and J. N. de Souza, "Exploring computation offloading in vehicular clouds," in *Proc. IEEE 8th Int. Conf. Cloud Netw. (CloudNet)*, Nov. 2019, pp. 1–4.
[19] M. S. Bute, P. Fan, L. Zhang, and F. Abbas, "An efficient distributed task offloading scheme for vehicular edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13149–13161, Dec. 2021.
[20] N. K. Jadav, R. Gupta, and S. Tanwar, "Blockchain and edge intelligence-based secure and trusted V2V framework underlying 6G networks," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2022, pp. 1–6.
[21] H. Xu and X. Liu, "Perception synergy optimization with deep reinforcement learning for cooperative perception in C-V2V scenarios," *Veh. Commun.*, vol. 38, Dec. 2022, Art. no. 100536.
[22] A. Gupta, S. Jaiswal, V. A. Bohara, and A. Srivastava, "Priority based V2V data offloading scheme for FiWi based vehicular network using reinforcement learning," *Veh. Commun.*, vol. 42, Aug. 2023, Art. no. 100629.
[23] Md. Z. Alam and A. Jamalipour, "Multi-agent DRL-based Hungarian algorithm (MADRLHA) for task offloading in multi-access edge computing Internet of Vehicles (IoVs)," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7641–7652, Sep. 2022.
[24] X. Liu, B. St. Amour, and A. Jaekel, "A Q-learning based adaptive congestion control for V2V communication in VANET," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, May 2022, pp. 847–852.
[25] J. Shi, J. Du, J. Wang, and J. Yuan, "Deep reinforcement learning-based V2V partial computation offloading in vehicular fog computing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Jun. 2021, pp. 1–6.
[26] H. Li, C. Chen, H. Shan, P. Li, Y. C. Chang, and H. Song, "Deep deterministic policy gradient-based algorithm for computation offloading in IoV," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 3, pp. 2522–2533, Mar. 2024.
[27] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep Q-Network based dynamic framing offloading in vehicular edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1297–1310, May 2023.
[28] J. Zhao, P. Dong, X. Ma, X. Sun, and D. Zou, "Mobile-aware and relay-assisted partial offloading scheme based on parked vehicles in B5G vehicular networks," *Phys. Commun.*, vol. 42, Oct. 2020, Art. no. 101163.
[29] Y. Huang, Y. Wang, X. Yan, X. Li, K. Duan, and Q. Xue, "Using a V2V- and V2I-based collision warning system to improve vehicle interaction at unsignalized intersections," *J. Saf. Res.*, vol. 83, pp. 282–293, Dec. 2022.
[30] Y. Saleem, N. Mitton, and V. Loscri, "A QoS-aware hybrid V2I and V2V data offloading for vehicular networks," in *Proc. IEEE 94th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2021, pp. 1–5.
[31] L. Zhao et al., "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3386–3400, Nov. 2023.
[32] J. Zhang, S. Chen, X. Wang, and Y. Zhu, "DeepReserve: Dynamic edge server reservation for connected vehicles with deep reinforcement learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2021, pp. 1–10.
[33] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Jul. 2021, pp. 1–10.
[34] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
[35] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.
[36] J. Fang, D. Qu, H. Chen, and Y. Liu, "Dependency-aware dynamic task offloading based on deep reinforcement learning in mobile-edge computing," *IEEE Trans. Netw. Service Manage.*, vol. 21, no. 2, pp. 1403–1415, Apr. 2024.
[37] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 30th Conf. Artif. Intell. (AAAI)*, Mar. 2016, pp. 2094–2100.
[38] V. Mnih et al., "Playing Atari with deep reinforcement learning," *Comput. Sci.*, Jan. 2013.
[39] H. Wang, X. Li, H. Ji, and H. Zhang, "Dynamic offloading scheduling scheme for MEC-enabled vehicular networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, Beijing, China, Aug. 2018, pp. 206–210.

Huijun Tang (Member, IEEE) received the B.Sc. degree from Jinan University, China, in 2016, and the M.S. and Ph.D. degrees from Tianjin University, China, in 2018 and 2022, respectively. She is currently a Lecturer with the School of Cyberspace, Hangzhou Dianzi University. Her research interests include the Internet of Things, mobile edge computing, and deep learning.

Ming Du received the bachelor's degree from Guilin University of Electronic Technology in 2021. He is currently pursuing the master's degree with the School of Cyberspace, Hangzhou Dianzi University. His current research interests include network embedding and graph generation.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from Harbin Institute of Technology, China, in 2009 and 2011, respectively, and the Ph.D. degree (Hons.) in computer science from Freie Universität Berlin, Germany, in 2015. He is currently a Professor at the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, the Internet of Things, deep learning, complex networks, and DNA storage.

Pengfei Jiao (Member, IEEE) received the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 2018. From 2018 to 2021, he was a Lecturer with the Center of Biosafety Research and Strategy, Tianjin University. He is currently a Professor with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. His current research interests include complex network analysis and its applications.



Ruidong Li (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Tsukuba in 2005 and 2008, respectively. He was a Senior Researcher at the National Institute of Information and Communications Technology (NICT), Kanazawa University, Japan, where he is an Associate Professor. His research interests include future networks, big data, intelligent internet edge, the Internet of Things, network security, information-centric networks, artificial intelligence, quantum internet, cyber-physical systems, and wireless networks. He is a member of IEICE. He is the Secretary of IEEE ComSoc Internet Technical Committee (ITC) and is the Founder and the Chair of IEEE SIG on Big Data Intelligent Networking and the IEEE SIG on Intelligent Internet Edge. He also served as the Chair for several conferences and workshops, such as the General Co-Chair for IEEE MSN 2021, AIVR2019, and IEEE INFOCOM 2019/2020/2021 ICCN workshop, and the TPC Co-Chair for IWQoS 2021, IEEE MSN 2020, BRAINS 2020, IEEE ICDCS 2019/2020 NMIC workshop, and ICCSSE 2019. He is an Associate Editor of IEEE INTERNET OF THINGS JOURNAL and the Guest Editor for a set of prestigious magazines, transactions, and journals, such as *IEEE Communications Magazine*, IEEE NETWORK, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.