

RESEARCH ARTICLE

Delay Constrained Hybrid Task Offloading of Internet of Vehicle: A Deep Reinforcement Learning Method

CHENHAO WU¹, (Member, IEEE), ZHONGWEI HUANG², AND YUNTAO ZOU^{1,3}¹Faculty of Innovation Engineering, Macau University of Science and Technology, Macau²International Institute of Next Generation Internet, Macau University of Science and Technology, Macau³Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Yuntao Zou (zouyuntao@hust.edu.cn)

This work was supported in part by the National Key Science and Technology Research Project 2020YFF0305600, and in part by the MUST Faculty Research Grant FRG-21-031-IINGI.


ABSTRACT The rapid development of the Internet of Things (IoTs) has driven the progress of intelligent transportation systems (ITS), which provides basic elements, such as vehicles, traffic lights, cameras, roadside units (RSUs) and their interconnected 5G communications, to constitute the Internet of vehicles (IoVs). In the IoVs, an intelligent vehicle can not only share information with the infrastructures like RSUs by vehicle to infrastructure (V2I) communication but also with vehicles on the road through vehicle-to-vehicle (V2V) communications. We thus expect that vehicles can collaborate with other well-resourced and idling vehicles, making full use of the wasted resources. However, existing approaches cannot achieve this goal due to the increasing strict delay constraints and the dynamic characteristics of the IoVs tasks. To improve the utilization of resources and perform better resource management, in this paper, we propose a hybrid task offloading scheme (HyTOS) based on deep reinforcement learning (DRL), which achieves the vehicle-to-edge (V2E) and V2V offloading by jointly considers the delay constraints and resource demand. To perform optimal offloading decision-making, we introduce a dynamic decision-making method, namely deep Q networks (DQN). To verify the effectiveness of this approach, we choose three baseline offloading approaches (one game theory-based and two single-scenario approaches) and perform a series of simulation experiments. The simulation results demonstrate that, compared to the baseline offloading approaches, our approach can effectively reduce task delay and energy consumption, achieving high-efficiency resource management.

INDEX TERMS Internet of vehicles (IoVs), task offloading, vehicle to edge (V2E), vehicle to vehicle (V2V), deep reinforcement learning (DRL).

I. INTRODUCTION

In the scenario of the Internet of vehicles (IoVs), vehicles are equipped with plenty of sensors, cameras, and computing units to support autonomous vehicle analysis and decision-making. With the implementation of diverse applications, such as augmented reality (AR), image-assisted navigation, intelligent vehicle control, and 3D gaming, there are new requirements and challenges producing in autonomous

vehicles, all of which require massive computation and storage resources, as well as a strict latency requirement. For instance, the latency of the autonomous vehicle steering is hoped less than 100ms [1]. In addition, the real-time operating system of an autonomous vehicle needs to process about 1GB of data per second because there are hundreds of sensors on vehicles generating a sea of data [2], which may easily exhaust the vehicle's onboard resources to process data. Although a more powerful processor such as GPU can be installed in vehicles to support the increasing computation demand, it will also incur higher energy consumption, which

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Mehmood .

affects the energy efficiency and mileage of the vehicle. Therefore, ensuring a good quality of service (QoS) in a resources limited intelligent vehicle is still a challenge.

To overcome the resource limitation of the single-vehicle and avoid the long transmission delay of cloud computing (CC), the deployment of computing capacity at the IoVs edge has become an effective alternative scheme. Multi-access edge computing (MEC) can deliver CC services near the vehicles, bringing the possibility to support computation- and resource-intensive applications while satisfying critical latency requirements [3]. For example, vision-based perception and feature extraction tasks with convolutional neural networks (CNNs) can be processed at the MEC server, i.e., vehicle-to-edge (V2E) offloading. However, due to geographical distribution and the deployment cost constraints, it is hard to deploy the infrastructure in all areas [4]. In this case, collaboration with other vehicles by vehicle-to-vehicle (V2V) communication is an effective solution.

In addition, V2V task offloading is strongly necessary for the IoVs scenarios because 1) The application's stringent latency requirements are often a prerequisite of V2V communication, where round-trip communication from infrastructure or roadside units (RSUs) and task congestion may make the delay-sensitive services invalid; 2) Utilizing network infrastructure like base station (BS) or RSU to deliver and process the IoVs tasks will face higher costs, while the V2V communication technology enables vehicle terminals to directly exchange wireless information with each other without forwarding through the base station, thus having lower cost [5]; 3) The idle or underutilized computing resources can be fully utilized for data processing, collection, and storage, reducing resource and energy waste. 4) Especially, in an urban scenario, vehicles usually gather for a variety of reasons, such as traffic jams, passing through toll booths, waiting for rush hour commute, or being attracted by attractive roadside scenery [6], these situations are quite unavoidable in our daily life. Figure 1 shows a specific example, vehicles move slowly and gather for a short period in a traffic light intersection, these vehicles gather at a relatively short distance and form a "resource pool", idles vehicles may have abundant resources for task execution. Therefore, the task vehicles with services overload can offload computation that cannot be processed locally to these idle service vehicles (i.e., V2V task offloading).

Therefore, we propose a hybrid task offloading scheme (HyTOS) based on deep reinforcement learning (DRL) methods, which achieves the hybrid V2E and V2V task offloading by jointly considering the delay constraints and resource demand of the task. HyTOS improves the utilization of scattered resources and efficiency of resource management. Due to the recent advances in DRL techniques solving the large-scale dynamic decision-making problems, we introduce a lightweight deep Q network (DQN) method to learn the optimal offloading decision. Our main contributions are as summarized follows:

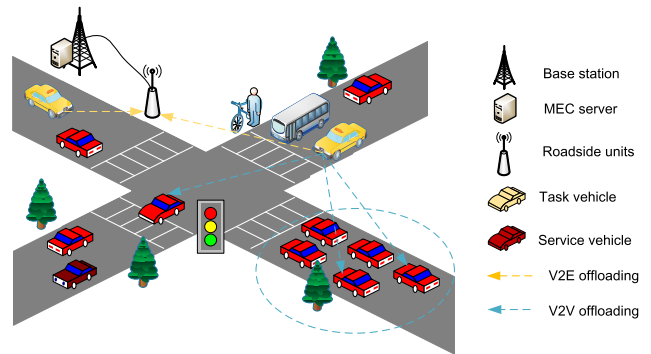


FIGURE 1. Hybrid task offloading scenario of IoVs.

- For urban scenarios, we propose a hybrid task offloading scheme (HyTOS) of computation- and resource-intensive IoVs tasks, which considers the collaboration of V2E and V2V task offloading, making full use of the scattered resources;
- We propose a dynamic offloading method based on DQN to jointly perceive the time-varying characteristics of vehicle tasks and resources distribution to optimize the task delay and energy consumption;
- Conduct comparative experiments with the state-of-the-art game theory-based and single-scenario offloading approaches to evaluate the effectiveness and adaptability of the proposed approach.

The remainder of this paper is organized as follows. We summarize related works in Section II and present the system model and mathematical description in Section III. In Section IV, we first model the hybrid task offloading problem as a Markov decision process (MDP) and propose a DQN-based hybrid offloading method to address this problem. Simulations are conducted in Section V to verify the effectiveness of our proposed method by comparison with benchmarks, finally, we summarize our paper in Section VI.

II. RELATED WORKS

Recent advances in Internet of things (IoTs) have spawned massive computing-intensive and delay-sensitive scenarios, such as virtual reality (VR), autonomous vehicles, healthcare IoTs, etc [7]. Computation offloading is considered a promising technology to cope with these emerging trends. The latest research has involved many new challenges, such as joint low-latency, secure and reliable task offloading which integrate software-defined networking (SDN) and blockchain scheme to the healthcare IoTs [8], joint computation offloading and resource allocation in fog radio access networks enabled IoTs [9], cache-assisted computation offloading in MEC systems to avoid duplicates in offloading [10], and balances multiple system utilities rather than simple objective maximization [11]. More related works can be found in a recent survey [7], and this paper focuses on task offloading in the IoVs scenario.

A. V2E TASK OFFLOADING IN IoVs

Edge computing technology has been increasingly incorporated into the vehicle networks [12], [13], [14], [15], [16], [17]. For example, Luo *et al.* [12] used dynamic programming to represent the tasks offload problem in multiple edge servers, an improved greedy algorithm is proposed to minimize delay. Matching theory is often used to find the appropriate offloading nodes, Gu *et al.* [13] introduced a distributed offloading method based on matching theory, and two heuristics were proposed to minimize the system delay. [14] proposed a collaborative offloading scheme based on RL, in which edge computing and CC are considered together with low communication overhead, however, they didn't take the scattered vehicles resources into consideration. Since game theory methods can solve decision-making problems among multiple players, it has been increasingly adopted for multi-user task offloading problems. Liu *et al.* [17] considered task offloading while ensuring the load balance of MEC servers, making the best offloading decision based on game theory. Similarly, Chen *et al.* [16] took the task offloading problem into a multi-user game-theoretic process and develop a distributed method to solve this problem. However, these studies mainly focus on minimizing delay or cost and do not address the energy consumption problem, especially in electric vehicles (EVs) with limited batteries.

Some researchers take energy consumption as one of the optimization goals in task offloading. For example, [18] optimized the energy consumption of mobile devices and MEC servers in task offloading, and they introduces a meta-heuristics algorithm to get the approximate solution. Yuan *et al.* [19] studies the collaborative computation offloading of distributed CC and MEC, they maximize the profit of service providers including response time, revenue, and penalty costs for each task by optimizing the resources allocation simultaneously. Chen *et al.* [20] considered the time-varying communication and computational resources in sliced wireless access networks (WAN), they formulated the stochastic task offloading as a MDP problem and addressed by reinforcement learning (RL) algorithm. However, the above works are mainly aimed at short-term planning and do not take multi-task and long-term task planning into account, which is inapplicable in a large number of task requests scenario.

B. V2V TASK OFFLOADING IN IoVs

Although MEC and CC or its collaboration can provide ample computing power, this architecture still has inherent limitations. For example, 1) CC faces the high data transmission delay of remote data delivery, and task congestion in the MEC server may fail the delay-sensitive tasks; 2) MEC and CC need to bear high infrastructure deployment costs, not all areas can be covered by infrastructure. Recently, researchers have also turned their attention to V2V offloading strategies, since vehicles can also be allocated with rich computing resources and a lot of vehicle resources are underutilized.

Most studies still consider collaboration task offloading schemes and take the vehicles into account. For example, [21] proposed a collaborative offloading approach that integrates CC, MEC, as well as vehicles. the cooperative optimization problem is addressed based on game-theory methods. Still, this work mainly focuses on CC and MEC collaboration, which is different from our work. Zhang *et al.* [22] also proposed the approximate task offloading that jointly considers the V2V and V2I collaborate scheme, and the edge resource is effectively utilized to reduce the processing delay. However, the above two works only considered single-target optimization and didn't optimize the energy consumption.

On the contrary, some researchers merely consider the V2V task offloading. Chen *et al.* [6] studied the task offloading scenario which purely support by V2V collaboration, they formulated the offloading process as a Min-Max problem and solved by the Particle Swarm Optimization Algorithm. However, purely considering V2V offloading cannot solve the task offloading problem perfectly, due to the limited vehicle resource and service interruption caused by movement. We believe that the task offloading of urban vehicular scenarios must jointly consider the edge and idle vehicle, and make full use of scattered resources.

III. SYSTEM MODEL

A. TASK MODEL

Fig. 1 depicts the IoVs computing offloading of urban scenario that supports V2E and V2V collaboration. In this scenario, vehicles from four directions gather at an intersection waiting for the traffic light. RSU or MEC server is deployed near the vehicles and connected with a BS by a wired link. The red device indicates the service vehicle (SV) that can offer computing assistance. In contrast, yellow indicates the overload task vehicle (TV) that needs to offload the task to relieve its computational load. Both TVs and SVs connect to the MEC server in real-time and send the task requirements as well as available computing resource capacity to the MEC server. Therefore, the MEC server will act as the global controller of the coverage area, combined with the AI algorithm installed on the MEC server to automatically make intelligent offloading decisions.

In this paper, we consider multiple vehicles and a MEC server scenarios. All computing nodes that can provide services are indicated by $N = \{n_0, n_1, n_2, \dots, n_m\}$, where n_0 represents the edge server (i.e., RSU), and we denote its computing resource capacity as F_0 . In addition, there are m vehicles with idle computing resources that can provide computing assistance in the resources pool, namely SVs, which are represented as n_m , and SV n_m with the computing capacity of F_m . The i th computing task generates by task vehicle k at time slot $t \in T$ that needs to be offloaded to the RSU or SV, is represented by a tuple $v_k^i(t) = \{c_i(t), b_i(t), \Theta_i\}$, where $c_i(t)$ indicates the size of the calculation amount (that is, the total CPU cycles) of the tasks v_k^i , $b_i(t)$ represents the data size of the task v_k^i , Θ_i is the maximum delay tolerance

TABLE 1. Symbols used in this paper.

Symbols	Explanation
n_0	the edge server, e.g., RSU, $n_0 \in N$
n_m	the service vehicle, $n_m \in N$
k	task vehicle, $k \in K$
F_0	computing power of the edge server
F_m	computing power of the service vehicle
t	the time slot, $t \in T$
i	the i -th offloading tasks, $i \in V$
$v_k^i(t)$	the i -th task generated by TV k at time t
$c_i(t)$	the size of the CPU calculation amount of task $v_k^i(t)$
$b_i(t)$	the data size of task $v_k^i(t)$
Θ_i	maximum delay tolerance of task v_k^i
$r^{v2e,v}$	the transmission rate between TV and RSU or SV
B^e	the system bandwidth between TV and RSU
B^v	the system bandwidth between TV and SV
N_0	the noise power
p_k^T	transmission power of the TV k
$h_{k,0}^{v2e}$	the channel gain between the TV k and RSU n_0
$h_{k,m}^{v2v}$	the channel gain between the TV k and SV n_m
$d_{m,0}^{-\beta}$	distance between the TV and the RSU
$d_{k,m}^{-\beta}$	distance between the TV and the SV
β	the loss coefficients
f_k^i	the computing resources allocated to task v_k^i by TV k
f_m^i	the computing resources allocated to task v_k^i by SV n_m
λ	the proportion of tasks processed locally
D_i^{local}	the task computing delay of local processing
$D_i^{v2e,v,tran}$	the data transmission delay of V2E or V2V offloading
$D_i^{v2e,v,com}$	the computing delay of V2E or V2V offloading
f_0^i	the computing power allocated to task v_k^i by the RSU n_0
f_m^i	the computing power allocated to task v_k^i by the SV n_m
E_i^{local}	the energy consumption of local processing
E_i^{v2e}	the energy consumption of V2E task offloading
E_i^{v2v}	the energy consumption of V2V task offloading
ε	the switched coefficient
P_0	the transceiver power between the TV k and the RSU n_0
P_m	the transceiver power between the TV k and the SV n_m

of the delay-sensitive task v_k^i , and $k \in K$, K represents the total number of task vehicles, $i \in V$, V is the total number of computing tasks of all vehicles. Each task is regarded as unsuccessful if the task delay tolerance is exceeded after being executed. Table 1 lists the main symbols used in this work.

B. COMMUNICATION MODEL

In this hybrid offloading scenario, we consider that each task can be processed locally, or collaborated with RSU or SV to reduce the task delay and ensure a better user experience. The task data must be transmitted to the corresponding devices as long as the task is offloaded to RSU or SV. The TV and the RSU are connected through the cellular network, and the transmission rate is calculated by:

$$r^{v2e} = \frac{B^e}{|K|} \cdot \log_2 \left\{ 1 + \frac{p_k^T \cdot d_{k,0}^{-\beta} |h_{k,0}^{v2e}|^2}{N_0} \right\}, \quad k \in K, \quad (1)$$

where B^e represents the vehicle to edge system bandwidth and shared by $|K|$ task vehicles, p_k^T represents the transmission power of the TV k , $d_{k,0}^{-\beta}$ indicates the distance between the TV k and the RSU n_0 and β is the loss coefficients, $h_{k,0}^{v2e}$ is the

channel gain between the TV and the RSU, and N_0 represent noise power.

The RSU has the capacity of a relatively limited resource compared to the cloud server, so the task congestion and queuing are still inevitable when RSU is overloaded with excess tasks offloading. The task quality may be degraded and the QoS of tasks cannot be guaranteed. Therefore, offloading the task to SVs is necessary when RSU is overloaded. Similarly, When the TV offloads the computing task to SVs for processing, its data and calculation results are transmitted through V2V communication, the channel transmission rate is calculated by [23]:

$$r^{v2v} = \frac{B^v}{|K|} \cdot \log_2 \left\{ 1 + \frac{p_k^T \cdot d_{k,m}^{-\beta} |h_{k,m}^{v2v}|^2}{N_0} \right\}, \quad k \in K, m \in N. \quad (2)$$

Similarly, B^v represents the vehicle to vehicle system bandwidth and shared by $|K|$ task vehicles, $d_{k,m}^{-\beta}$ indicates the distance between the TV k and the SV n_m , $h_{k,m}^{v2v}$ is the channel gain between the TV and the SV. The transmission rates between TV and RSU, TV and SV all follow Shannon's theorem [24] but have different system bandwidth, distance coverage, etc.

C. COMPUTING DELAY MODEL

The task execution brings a certain delay whether the task is processed locally or offloaded to the RSU or SV, including the data transmission delay and processing delay. When the TV has sufficient computing resources, or the computation pressure of the task is relatively low, the task is considered to be processed locally to reduce data transmission delay. In this case, the overall delay is equal to the computation delay, which is determined by the total CPU cycles of the task v_k^i and the available computing capacity of the TV. The overall delay in local processing of task v_k^i can be calculated by:

$$D_i^{local} = \frac{\lambda b_i c_i}{f_k^i}, \quad i \in V, k \in K, \quad (3)$$

where, $\lambda \in [0, 1]$ indicates the proportion of tasks processed locally, b_i is the data size of task v_k^i , c_i represents the CPU cycles requirement to process a unit of data, and f_k^i is the CPU frequency of TV allocated to process task v_k^i itself.

However, purely local processing can't guarantee the task QoS, as for the computation-intensive or larger amounts of tasks, and all tasks processes locally will bring higher energy consumption compared with transmitting data to other servers, which is not conducive to the battery life of the vehicle. Therefore, the computing-intensive task needs to be offloaded to RSU or other SVs with sufficient resources for processing. The total delay includes the processing delay in the RSU or SVs and the transmission delay of the data, which is calculated by:

$$D_i^{v2e,v} = D_i^{v2e,v,tran} + D_i^{v2e,v,com}, \quad i \in V, \quad (4)$$

where $D_{i,tran}^{v2e,v}$ represents the data transmission delay when offloading to the RSU or SV, i.e., $D_{i,tran}^{v2e}$ or $D_{i,tran}^{v2v}$, which is calculated by the ratio of the data volume to the channel transmission rate:

$$D_{i,tran}^{v2e,v} = \frac{(1-\lambda)b_i}{r^{v2e,v}}, \quad i \in V, \quad (5)$$

where $(1-\lambda)$ indicates the proportion of task offloading. Note that, the size of the calculation result return back is much smaller than that of the raw data, so it can be ignored during transmission [21]. Meanwhile, $D_{i,com}^{v2e,v}$ is the processing delay of the task in the RSU or SV:

$$D_{i,com}^{v2e,v} = \frac{(1-\lambda)b_i c_i}{f_{0,m}^i}, \quad i \in V, \quad m \in N, \quad (6)$$

where $f_{0,m}^i$ denote the CPU frequency allocated to task v_k^i by the RSU n_0 and SV n_m , respectively, i.e., f_0^i and f_m^i .

Finally, we can denote the total delay of the task as follows:

$$D_i^{total} = D_i^{local} + D_i^{v2e,v}, \quad i \in V. \quad (7)$$

D. ENERGY CONSUMPTION MODEL

The energy consumption is also non-negligible during the process of task execution and data transmission. Therefore, the energy consumption optimization is crucial for a hybrid offloading system. The energy consumption of CPU-based computing can be calculated by εf^2 [25], where f is the CPU frequency and ε is the switching coefficient which depends on the CPU architecture. Then, the total energy consumption of tasks processed locally is calculated by:

$$E_i^{local} = \varepsilon (f_k^i)^2 c_i, \quad i \in V, \quad k \in K. \quad (8)$$

Similarly, the energy of task offloading is calculated by (9). In addition to the basic computing energy consumption, it produces extra energy consumption for data transmission:

$$E_i^{v2e,v} = \varepsilon (f_{0,m}^i)^2 \lambda c_i + P_{0,m} \frac{(1-\lambda)b_i}{r^{v2e,v}}, \quad i \in V, \quad m \in N, \quad (9)$$

where $P_{0,m}$ represent the transceiver power between the TV k and the RSU n_0 , and the transceiver power between the TV and the SV n_m , respectively, i.e., P_0 and P_m .

The total energy consumption of task v_k^i is calculated by:

$$E_i^{total} = E_i^{local} + E_i^{v2e,v}, \quad i \in V. \quad (10)$$

E. PROBLEM FORMULATION

This paper aims jointly optimize the delay and energy consumption of the tasks, and control two optimized goals by parameters $\alpha \in [0, 1]$. However, these two optimization objectives have different units and cannot be accumulated directly, needing to be normalized. We first divide the average task delay $D_{ave} = \frac{\sum_{i=1}^V D_i^{total}}{V}$ and average energy consumption $E_{ave} = \frac{\sum_{i=1}^V E_i^{total}}{V}$ by the respective maximum task delay $D_{max} > 0$ and maximum energy consumption $E_{max} > 0$ of all offloaded tasks V in the current time, and get the delay and

energy consumption utility o_d and $o_e \in [-1, 1]$, respectively, as shown in (11) and (12):

$$o_d = \max\{-1, \frac{-D_{ave}}{D_{max}} + 1\} \in [-1, 1], \quad (11)$$

$$o_e = \max\{-1, \frac{-E_{ave}}{E_{max}} + 1\} \in [-1, 1]. \quad (12)$$

Therefore, the objective function is defined as (13), maximizing the weight sum of o_d and o_e by optimizing the offloading decision $\{x_m^i, \lambda\}$, where x_m^i is the task offloading variable as defined in Section IV-A:

$$obj: \arg \max_{\{x_m^i, \lambda\}} \sum_{i=1}^V (\alpha o_d + (1-\alpha) o_e). \quad (13)$$

$$s.t. \sum_{i=1}^V f_{0,m}^i \leq F_{0,m}, \quad i \in V, \quad m \in N. \quad (14)$$

$$0 \leq P_{0,m} \leq P_{0,m}^{max}. \quad (15)$$

$$D_i^{total} \leq \Theta_i, \quad i \in V. \quad (16)$$

The constraints are represented by (14)-(16), where the computing resources and transmission power allocated to tasks v_k^i cannot exceed their resource capacity and maximum transmission power of the edge servers and service vehicles. The delay constraint of task v_k^i is represented by (16), tasks that exceed the delay constraint will be dropped.

IV. DRL-BASED HYBRID OFFLOADING MODEL

A. MARKOV DECISION PROCESS

The RL method is different from traditional supervised learning and unsupervised learning, it learns the optimal strategy through extensively exploring the environment, and constantly improving strategies through rewards and punishments, it's therefore particularly suitable for autonomous decision-making problems. Recently, RL has shown excellent performance and adaptability in controlling and decision-making problems. This paper will realize the hybrid offloading decision through the advanced RL method. The task offloading process can be modeled as a MDP problem [26], [27], which includes three basic elements: $\mathcal{M} = \{\mathcal{S}(t), \mathcal{A}(t), \mathcal{R}(t)\}$, where $\mathcal{S}(t)$ and $\mathcal{A}(t)$ represents the state space set and action space set, $\mathcal{R}(t)$ represents the reward function, and defined as follows:

1) STATE SPACE

The offloading task may be generated by multiple TVs, thus we can define the state $\mathcal{S}(t)$ as follows:

$$\mathcal{S}(t) = \{d(t), u(t), b(t), c(t)\}, \quad (17)$$

where $d(t) = \{d_0(t), d_1(t), d_2(t) \dots\}$ indicates the distance between the TV and RSU or SV, $u(t) = \{u_0(t), u_1(t), u_2(t) \dots\}$ indicate the available resource of the RSU and SVs, $b(t)$ and $c(t)$ indicates the data volume of the task and total CPU cycles requirement, respectively.

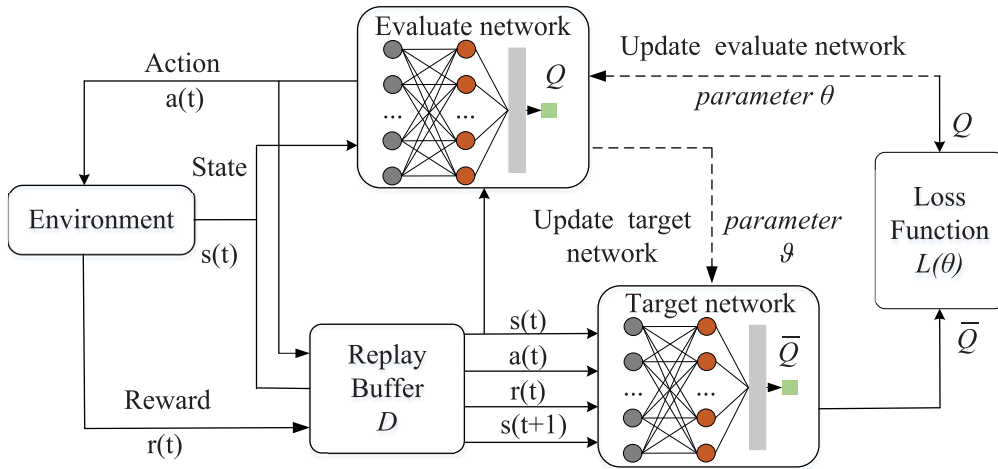


FIGURE 2. The architecture of proposed DQN algorithm.

2) ACTION SPACE

The agent selects the best offloading server (i.e., RSU or SVs) based on task demand and the resources capacity. The selects actions $\mathcal{A}(t)$ according to the current policy at time t is defined as:

$$\mathcal{A}(t) = \{x_m^i, \lambda\}, \quad (18)$$

where $x_m^i = \{-1, 0, 1, 2, \dots, N\}$, $x_m^i = -1$ denotes the task that is processed locally, $x_m^i = 0$ denotes part or all of the tasks that are offloaded to the RSU for execution, $x_m^i = \{1, 2, 3, \dots, N\}$ denotes the task v_k^i that is offloaded to the corresponding SV for execution, and λ denotes the proportion of task offloading. After executing the current action $a(t) \in \mathcal{A}(t)$, the system enters the next state $s(t+1) \in \mathcal{S}(t)$ and obtains a immediate reward $r(t)$. This paper jointly minimize the delay and energy consumption of task offloading. Therefore, in each step, the instant reward uses both two targets for policy optimization as defined in objective function (13). The instant reward of the RL is defined as:

$$r(t) = \alpha o_d + (1 - \alpha) o_e. \quad (19)$$

The DRL method aims to find an optimal policy to maximize the cumulative reward $\mathcal{R}(t) = \sum_{t=1}^T r(t)$ while following the policy.

B. DQN-BASED HYBRID OFFLOADING ALGORITHM

Although traditional RL has a good efficiency in addressing the control problem, it also has some limitations. For example, the RL algorithm based on Q -table faces the curse of dimensionality [28], which is not suitable for large-scale decision-making problems. On the contrary, Deep RL (DRL) uses the deep neural network (DNN) to fit Q -table, solving the disadvantage of table-based reinforcement learning. Currently, DRL has a good application prospect in decision-making and controlling problems. The DQN algorithm has better performance on dealing with discrete action spaces. Compared with multiple neural network based DRL

algorithms, like Double DQN and actor-critic DRL, the DQN algorithm is lightweight and insensitive to hyper-parameters, and more suitable for the low-latency decision-making scenario. Therefore, we introduce a DQN-based hybrid strategy offloading method in this paper. The architecture and learning process of our proposed DQN-based algorithm are shown in Figure 2.

In the RL processing, an action-value function $Q_\pi(s, a)$ is used to represent the expected reward of taking action a at the current state $s(t)$:

$$Q_\pi(s, a) = \mathbb{E}^\pi \left\{ \sum_{t=0}^{\infty} \gamma r(t+1) \mid s(t) = s, a(t) = a \right\}. \quad (20)$$

Further, $Q_\pi(s, a)$ can be rewritten as follow based on the Markov property:

$$Q_\pi(s, a) = \sum_{s' \in \mathcal{S}} P_{s,s'}^a [r(s, a, s')] + \gamma \sum_{a' \in \mathcal{A}} \pi(s'|a') q^\pi(s', s'), \quad (21)$$

where $P_{s,s'}^a$ represents the state transition probability, r is the immediate reward after taking an action a in state s , $\pi(s|a)$ represents the current policy, and γ represents the discount to future rewards.

$Q^*(s, a)$ represents the optimal state-action value function that indicates the optimal strategy a is taken under the state s . Therefore, $Q^*(s, a)$ can be expressed as:

$$Q^*(s, a) = \sum_{s' \in \mathcal{S}} P_{s,s'}^a [r(s, a, s')] + \gamma \max_{a'} q^*(s, s'). \quad (22)$$

DQN integrates an replay buffer D to cache and reuse the experiences, and randomly extracts experiences from D to train the DNN. Afterward, DQN further introduces a target network \bar{Q} to eliminate experience correlation between samples. Finally, DQN uses the temporal-difference (TD) learning method to minimize the loss function of evaluating

network Q and \bar{Q} , and guide the direction of the network parameters θ optimization:

$$L(\theta) = (Q(s, a, \theta) - [r(s, a, s') + \gamma \max_{a'} Q(s', s', \vartheta)]), \quad (23)$$

where ϑ and θ is the parameters of the \bar{Q} and Q , respectively. The parameter ϑ of the \bar{Q} is copied from Q at every fixed step. The specific steps of the algorithm are shown in Algorithm 1.

Algorithm 1 DQN-Based Task Offloading Algorithm

Data: current state: $s(t)$
Result: optimal offloading decision $a(t)$

- 1 Initialize: replay buffer, network parameters: D, θ, ϑ
- 2 **for** each episode **do**
- 3 initialize pre-processed sequenced $\phi_1 = \phi_{s_1}$;
- 4 **for** $t = [1, T]$ **do**
- 5 obtain the environment's state $s(t)$;
- 6 with probability ϵ random select an action $a(t)$ or $a(t) = \arg \max_a Q(s, a, \theta, \vartheta)$;
- 7 executed $a(t)$, obtain $r(t)$, gets next state $s(t+1)$;
- 8 set preprocess $\phi_{t+1} = \phi_{s_{t+1}}$;
- 9 add $\{s(t), a(t), r(t), s(t+1)\}$ into replay buffer D ;
- 10 randomly sample the experience from D ;
- 11 // calculate the target value y_i //
- 12 **if** ϕ_{j+1} is terminal state **then**
- 13 | $y_i = r_j$;
- 14 **else**
- 15 | $y_i = r_j + \max_{a'} Q(s', a', \vartheta)$;
- 16 **end**
- 17 update θ based on loss function $L(\theta)$ of (23);
- 18 update $\vartheta \leftarrow \theta$ at a fixed-interval;
- 19 output the optimal strategy: $\{x_m^i, \lambda\}$;
- 20 **end**
- 21 **end**

C. ALGORITHM COMPLEXITY

Assuming that the DQN-based hybrid offloading algorithm needs to train M episodes to converge, and train the DQN agent for N time steps in each episode, i.e., $T = N$. In algorithm 1, the initialization process (line 1) only runs once, the first “for” loop (lines 2-3) performs $2M$ operations, while the second “for” loop (lines 4-20) performs $12NM$ operations, in which the “if” loop (lines 12-16) performs $2MN$ operations. Therefore, the total complexity is $O(2M + 12MN + 1)$. In our actual training process, the algorithm has converged well after 100 episodes of training, as shown in Fig. 3 and Fig. 4. We trained the DQN agent with 2000 time steps in each episode, i.e., $N = 2000$, and tasks are randomly generated within 2000 time steps. Therefore, M can be considered a

TABLE 2. Experimental settings.

Number of vehicles	[10, 40]
Number of task vehicles	[1, 5]
An RSU coverage	200m
V2E system bandwidth	12 MHz
V2V system bandwidth	8 MHz
Channel gain	$127+30\log d$
Gaussian noise	-174dBm/Hz
Average task CPU requirements	[0.5, 2]Giga cycles
Average data size	[2,10]MB
Transmission power	[5, 38]dBm
Computing power of edge server	[5, 35]GHz
Computing power of service vehicle	[2, 8]GHz

constant, and the total complexity of the proposed algorithm is $O(MN)$, which implies that the proposed algorithm has a relatively small complexity.

V. PERFORMANCE EVALUATION

A. SIMULATION SETTING

In this section we discuss a quasi-static scenario involving a single MEC server and multiple vehicles [29]. In urban scenes, vehicles will inevitably gather, as discussed in section I. In each time slot, assume that the vehicle's position remains the same, and the channel remains stable since the vehicle may still not move for a few seconds or moved for a very short distance in the scenario mentioned above. The RSU is connected to the base station and MEC server by a wire link, and an RSU will cover the vehicles within a radius of 200m. V2V communication adopts the dedicated short-range communication (DSRC) [23] technology. We consider the total number of 10-40 vehicles, in each period, 1-5 TVs generate tasks at time slot t randomly and simultaneously with the mean task interval of 10 time steps, each episode lasts 2000 time steps ($T = 2000$), the rest of the vehicles can be seen as the SVs with different resource capacities. These tasks have various data sizes and computing requirements. The resource capacity of RSU and SVs follows a uniform distribution. Moreover, the channel gain follows $127 + 30\log d$ and the total bandwidth of the system is 20 MHz. Additionally, the Gaussian noise $N_0 = -174\text{dBm/Hz}$ [29]. The detailed simulation settings are summarized in Table 2.

We compare the proposed approach with the local processing approach, single scenario offloading approach, and the advanced hybrid offloading approach to verify the advantages and disadvantages of our proposed approach. The comparison approaches are described as follows:

- Our proposed DQN-based hybrid offloading algorithm using two DNN networks to approximate the Q and \bar{Q} target network, and configures the following hyper-parameters: 1) learning rate = 0.01, 2) discount factor $\gamma = 0.95$, 3) buffer size $|B| = 10000$ with batch size $|b| = 64$, 4) DNN networks with a depth of 64 hidden layers to balance the complexity and training performance.
- Local processing (LC): LC considers tasks are only processed locally, which will reduce the additional transmission delay and energy consumption.

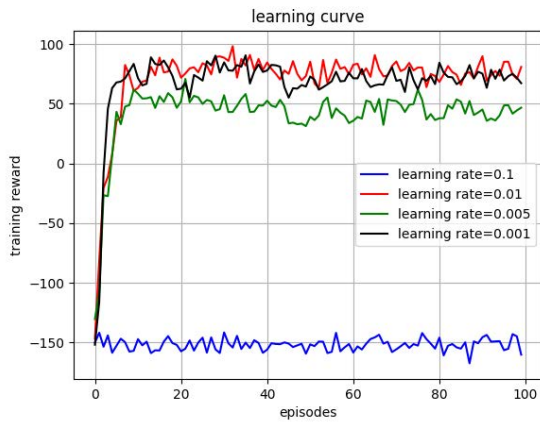


FIGURE 3. The impact of learning rate.

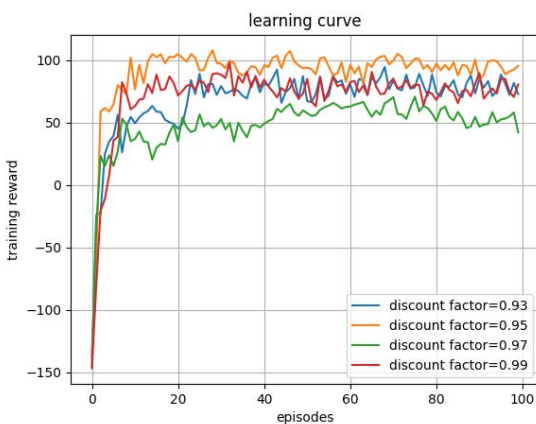


FIGURE 4. The impact of discount factor.

- Single-scenario offloading algorithm: two separate offloading algorithms that consider offloading the task to the edge server (i.e., V2E) or service vehicle for processing (i.e., V2V).
- Game-theory based hybrid offloading algorithm [30] (GT-hybrid): GT-hybrid is a hybrid offloading method, that considers offloading the task to the optimal computing node (i.e., RSU or SV) by getting the Nash equilibrium.

We conduct experiments on a Ubuntu 16.04 server with Intel(R) Core(TM) i7-7820X CPU @ 3.60GHz and a GeForce RTX 2080 Ti GPU. The experiments are running by Python 3.6 and TensorFlow 1.14 and use Adam optimizer to optimize the parameters.

B. PARAMETER ANALYSIS

To make the algorithm converge quickly and get better learning efficiency, we first verify the learning curves of different learning rates and discount factors. Generally, the RL algorithm's converged speed will increase with the learning rate. However, the convergence value cannot be guaranteed to be optimal when the learning rate is too large, as the agent may learn nothing, e.g., learning rate = 0.1 in Figure 3. Conversely, if the learning rate is too small, it is difficult

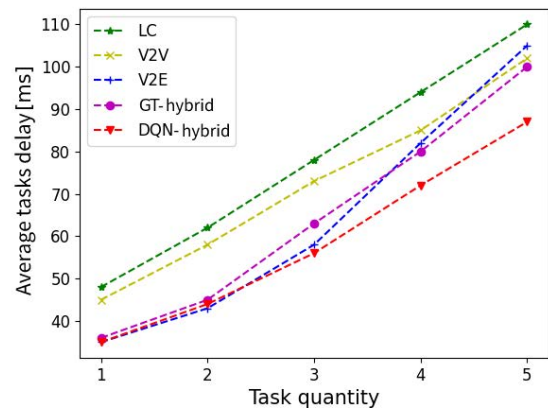


FIGURE 5. The impact of the tasks quantity on task delay.

to obtain the optimal solution within an acceptable time. As shown in Figure 3, DQN has the best learning effect when the learning rate = 0.01 with the default discount factor = 0.99, which means that DQN can learn more strategies.

Except for the learning rate, we also verify the influence of different discount factors. The discount factor is set from 0.93 to 0.99, and learning rate = 0.01. The results are shown in Figure 4. The cumulative reward is smallest when the discount factor = 0.97. When the discount factor = 0.95, the DQN has the best learning performance, and the convergence reward is great than 130. In the subsequent simulations, we set learning rate = 0.01, and discount factor = 0.95.

C. SIMULATION RESULT

1) TASK DELAY

The optimization objective of RL is to optimize the average delay of all tasks, so the task quantity has an important significance to the algorithm performance evaluation. Figure 5 shows the tendency of task delay as the increasing of task quantity. From Figure 5 we can observe that the V2E offloading can process a few tasks well, the task delay is relatively smart when the task quantity is small. However, with the increasing of tasks, the available resources for each task become less, resulting in a rapid increase in the delay of V2E. Similarly, when task quantity is small, the V2V offloading strategy can ensure lower task delay, but there is an obvious effect on the task delay when the task quantity increases due to the limited computing capacity.

In contrast, our approach adopts a hybrid offloading strategy to handle multiple tasks, which better use of distributed resources, and considers the long-term gain of the multi-task offloading decisions. Therefore, the advantages of DQN-hybrid become obvious as the number of tasks increases. The GT-hybrid offloading strategy outperforms the single-scenario offloading approach (i.e., only consider offload task to RSU or SV), while the proposed DQN-hybrid outperforms all the baseline algorithms in terms of task delay. Specifically, the task delay is reduced by 27%, 20%, 16%, and 14%, respectively.

Figure 6 shows the effect of the data volume on the total task delay. Generally, the data transmission delay and

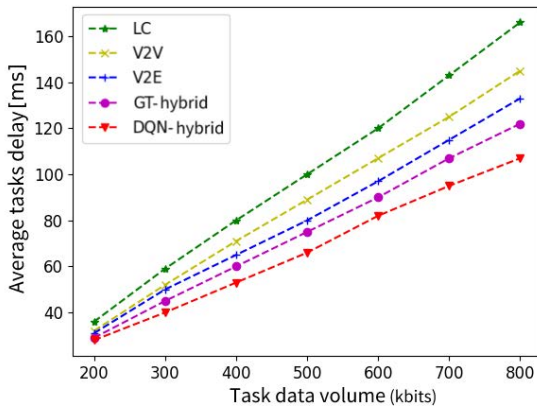


FIGURE 6. The impact of data volume on task delay.

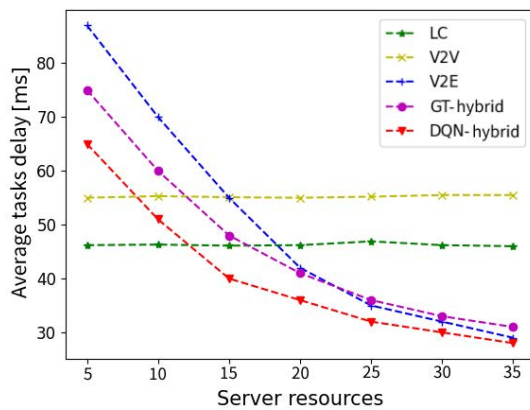


FIGURE 7. The impact of MEC resource changes on task delay.

computing delay will increase corresponding with the data volume, as shown in the Figure. Nevertheless, the advantage of our proposed approach is enhanced with the increase in data volume, which further verifies that the DQN algorithm has a better generalization ability when facing multiple tasks.

Better adaptability to resource changes is also crucial for task offloading algorithms. Figure 7 shows the impact of the available resources change of the MEC server on the task delay. With available resources increasing, the total delay of V2E and hybrid offload mechanisms decreases. When computing resources are insufficient, the total system delay of the V2E algorithm is larger than that of other algorithms. Indicates that V2E offloading is appropriate when the resources of the MEC servers are sufficient. The proposed hybrid offloading algorithm DQN-hybrid selects the appropriate offloading node by jointly considering the resources of RSU and SVs, thus achieving the lowest total task delay regardless of the resources of the edge server are sufficient.

Similarly, Figure 8 shows the impact of changes in the resource of the SVs on task delay. In a real situation, the resource capacity of the vehicle is less than that of the MEC server. The task delay of four algorithms decreases as the resource capacity increase, except for the V2E offloading, since V2E does not utilize the resources of SVs. As expected, when vehicle resources are scarce, V2V offloading has the

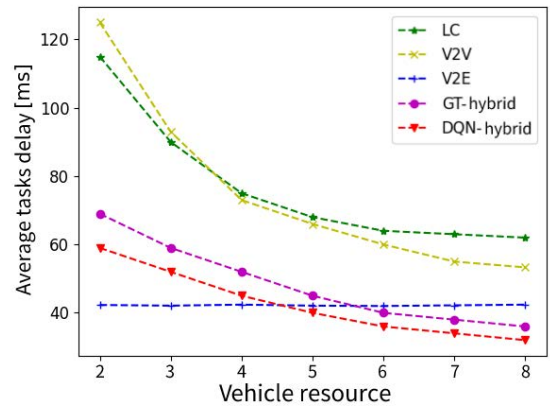


FIGURE 8. The impact of vehicle resource changes on task delay.

highest processing delay, followed by LC. The reason is that V2V task offloading brings additional transmission delay compared to LC. As resources increase, however, V2V task delays are smaller than that of the LC, because the delay of V2V offloading is greatly reduced by parallel processing. As for the hybrid offloading algorithms i.e., GT-hybrid and DQN-hybrid, they have lower task delays because they fully consider different processing nodes. However, the DQN algorithm still has better performance than GT-hybrid, and the delay is further reduced by about 10%.

2) ENERGY CONSUMPTION

With the proposal of energy saving and emission reduction, electric vehicles have become an irresistible trend, electric vehicles are more sensitive to the energy consumption. Figures 9 - 10 show the impact of the task quantity and data volume on the vehicle's energy consumption. Since V2E offloading transmits all the data to RSU to process, the vehicle's energy consumption is only considered by the data transmission, thus the V2E offloading has the smallest energy consumption. On the contrary, V2V offloading incurs the highest energy consumption because no matter where the task offloads the energy consumption of vehicles is necessary, and it's slightly higher than that of LC offloading because the transmission energy consumption is avoided in LC offloading. While hybrid offloading approaches (i.e., GT-hybrid and DQN-hybrid) have relatively low energy consumption. Although sometimes our proposed DQN-hybrid methods are higher than GT-hybrid, considering DQN-hybrid has a better overall delay performance, it still has a better offloading performance than GT-hybrid. In addition, the coefficient α which controls the balance of two optimized goals was set to 0.5 in these two experiments, different degrees of two optimized goals will be obtained by adjusting the parameters.

3) SUCCESSFUL RATE

Figure 11 shows the results of the task successful rates of the all algorithms under the conditions of low, medium, and high resource capacity. As defined in Section III-A, a task can complete before the maximum tolerate delay, we consider

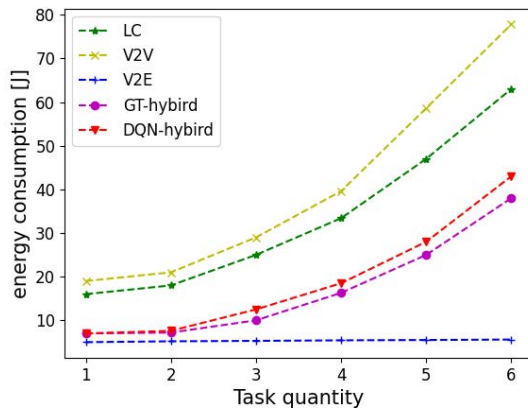


FIGURE 9. The impact of tasks quantity on energy consumption.

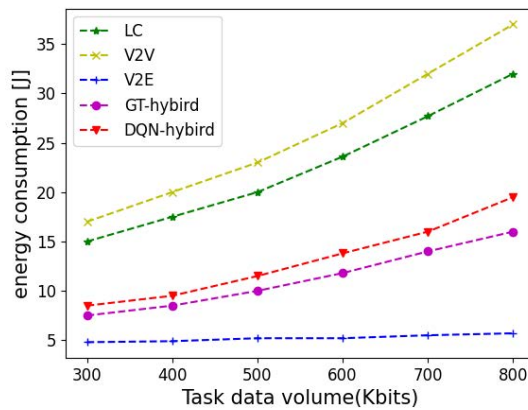


FIGURE 10. The impact of data volumes on energy consumption.

that it a successful service. We can concluded that the hybrid offloading algorithm is better than the LC, V2E, and V2V single-scenario offloading algorithms in different resource capacities. In the case of low resources capacity, the task successful rate of the hybrid offloading strategy is about 30%-40% higher than that of the single-scenario offloading strategy, and the task successful rate improved by the hybrid offloading strategy is about 15% when the resources are sufficient. Our proposed DQN-hybrid offloading strategy is much higher than the single-scenario offloading strategy in terms of task successful rate. Moreover, compared with the state-of-the-art hybrid offloading strategy based on game theory i.e., GT-hybrid, our approach can improve the task successful rate by about 5-8% among three cases.

In short, our proposed DQN-hybrid approach outperforms the single-scenario offload approaches both in delay and energy consumption. Although GT-hybrid is another hybrid offloading approach based on game theory, traditional approaches like GT-hybrid mainly focus on immediate performance, leading to performance degradation in the long term. Therefore, the long-term performance of GT-hybrid is inferior to DQN-hybrid. In detail, the average task delay of DQN-hybrid is about 10-15% lower than that of GT-hybrid, in the case of almost equivalent energy consumption.

Nevertheless, the current offloading method also has some shortcomings. For example, 1) the DQN-based algorithm is limited by the discrete action space, and it is difficult to exert

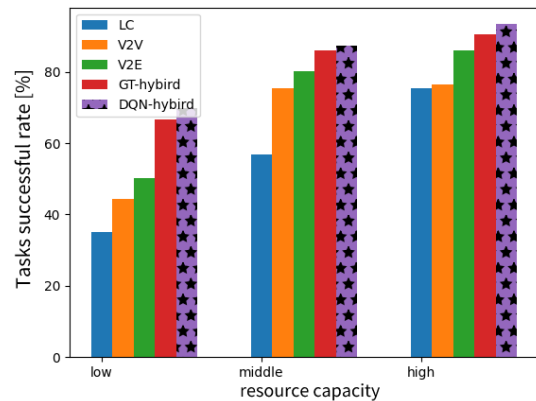


FIGURE 11. The successful ratio of tasks under various resource capacity.

the greatest advantage in continuous action space situations. 2) further exploring in generalization ability of the trained DRL model, as well as the more realistic and complicated scenarios are required, such as the dynamic task offloading scenario that considers the vehicle movement, in which the task must be completed before it switches from its corresponding RSU to another.

VI. CONCLUSION

In this paper, we propose a hybrid task offloading scheme (HyTOS) for the urban IoVs scenario, which jointly considers V2E and V2V offloading to minimize the task delay and energy consumption while making full use of scattered resources of vehicles. We further propose a Deep Q-network (DQN)-based optimal offloading method to satisfy the computing requirements and ensure the delay constraints of the task. The simulation results demonstrated that our approach is significantly better than the single-scenario offloading approaches, and has a better overall performance than the advanced game-theory based hybrid offloading approach in terms of task delay and successfully rate. Our approach has good application prospects in delay-constrained and dynamic IoVs scenarios. Future work is in progress to consider the more dynamic task offloading scenario that consider the vehicle movement.

ACKNOWLEDGMENT

The authors would like to thank Dr. Fang Sen for his participation in discussions and valuable revision comments.

REFERENCES

- [1] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.
- [2] A. B. D. Souza, P. A. L. Rego, T. Carneiro, J. D. C. Rodrigues, P. P. R. Filho, J. N. D. Souza, V. Chamola, V. H. C. D. Albuquerque, and B. Sikdar, "Computation offloading for vehicular environments: A survey," *IEEE Access*, vol. 8, pp. 198214–198243, 2020.
- [3] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. K. Kwok, "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.

- [4] W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, "Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of vehicles," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5839–5852, Apr. 2022.
- [5] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 28–36, Mar. 2019.
- [6] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, and Z. Chen, "Delay-optimized V2 V-based computation offloading in urban vehicular edge computing and networks," *IEEE Access*, vol. 8, pp. 18863–18873, 2020.
- [7] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366.
- [8] J. Ren, J. Li, H. Liu, and T. Qin, "Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT," *Tsinghua Sci. Technol.*, vol. 27, no. 4, pp. 760–776, Aug. 2022.
- [9] G. M. S. Rahman, T. Dang, and M. Ahmed, "Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks," *Intell. Converged Netw.*, vol. 1, no. 3, pp. 243–257, Dec. 2020.
- [10] S. Nath and J. Wu, "Deep reinforcement learning for dynamic computation offloading and resource allocation in cache-assisted mobile edge computing systems," *Intell. Converged Netw.*, vol. 1, no. 2, pp. 181–198, 2020.
- [11] R. Bi, Q. Liu, J. Ren, and G. Tan, "Utility aware offloading for mobile-edge computing," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 239–250, Apr. 2021.
- [12] J. Luo, X. Deng, H. Zhang, and H. Qi, "Ultra-low latency service provision in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
- [13] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 100–106, Sep. 2019.
- [14] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 4, pp. 1155–1165, Dec. 2020.
- [15] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
- [16] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
- [17] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
- [18] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2020.
- [19] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.
- [20] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4005–4018, Jun. 2019.
- [21] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
- [22] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.
- [23] A. Chopra, A. U. Rahman, A. W. Malik, and S. D. Ravana, "Adaptive-learning-based vehicle-to-vehicle opportunistic resource-sharing framework," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 12497–12504, Jul. 2022.
- [24] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.
- [25] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.
- [26] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3296–3309, Mar. 2020.
- [27] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Informat.*, early access, Apr. 19, 2022, doi: [10.1109/TII.2022.3168292](https://doi.org/10.1109/TII.2022.3168292).
- [28] J. Cai, Z. Huang, L. Liao, J. Luo, and W.-X. Liu, "APPM: Adaptive parallel processing mechanism for service function chains," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1540–1555, Jun. 2021.
- [29] J. Cai, H. Fu, and Y. Liu, "Deep reinforcement learning-based multitask hybrid computing offloading for multiaccess edge computing," *Int. J. Intell. Syst.*, vol. 37, no. 9, pp. 6221–6243, Sep. 2022.
- [30] M. Liwang, J. Wang, Z. Gao, X. Du, and M. Guizani, "Game theory based opportunistic computation offloading in cloud-enabled IoV," *IEEE Access*, vol. 7, pp. 32551–32561, 2019.



CHENHAO WU (Member, IEEE) had participated in the National Key Research & Development Project "Science and Technology Winter Olympics," Key Project "Smart Green Service Operation Platform and Application Demonstration in Chongli Winter Olympics Town," and MUST Faculty Research Grants "Edge Intelligence Driven Industrial Service Function Chain Orchestration Mechanism." His research interests include artificial intelligence, Internet of Vehicles, and image processing. He is an IEEE Communications Society Member and a CCF Member of the Faculty of Innovation Engineering, Macau University of Science and Technology.



ZHONGWEI HUANG is currently pursuing the Ph.D. degree with the International Institute of Next Generation Internet (IINGI), Macau University of Science and Technology, Macau, China. He has been a Visiting Scholar of the Shenzhen Guangming Laboratory, since 2022. Recently, he has published several papers on efficient resource allocation and delay constrained services orchestration of edge intelligence empowered networks. His current research interests include artificial intelligence (AI), multi-access edge computing (MEC), and Internet of Vehicles (IoVs). He is a CSIG Member.



YUNTAO ZOU is currently working with the Huazhong University of Science and Technology as a Graduate Supervisor. His research interests include artificial intelligence, big data analysis, pattern recognition, data encryption, robotics, and embedded development. He has successively served as a Senior Member of the China Artificial Intelligence Society, a Senior Member of the China Computer Society, the Vice President of the China Young Scientists Association, the Director of China Vocational Teaching Association, the Executive Chairperson of the Seventh Robot Football Competition, the Vice Chairperson of Wuhan Academic Committee of YOCSEF (2010), an AC Member of YOCSEF Wuhan, from 2011 to 2013, and the MVP of Microsoft.

• • •