

© 2023 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Citation: 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," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2023.3325267.

DOI: <https://doi.org/10.1109/TITS.2023.3325267>

Access to this work was provided by the University of Maryland, Baltimore County (UMBC) ScholarWorks@UMBC digital repository on the Maryland Shared Open Access (MD-SOAR) platform.

Please provide feedback

Please support the ScholarWorks@UMBC repository by emailing scholarworks-group@umbc.edu and telling us what having access to this work means to you and why it's important to you. Thank you.

Deep Deterministic Policy Gradient-Based Algorithm for Computation Offloading in IoV

Haofei Li, Chen Chen^{ID}, *Senior Member, IEEE*, Hangguan Shan^{ID}, *Senior Member, IEEE*,
Pu Li, Yoong Choon Chang, and Houbing Song^{ID}, *Fellow, IEEE*

Abstract—The continuous evolution of cellular networks has resulted in the rapid increase in both mobile applications and devices in the Internet of Vehicles. The introduction of the multi-access edge computing method makes it possible for vehicles in remote areas to offload their computational tasks, which can effectively relieve the computing pressure of local devices and reduce the computational delay as well. Tasks offloading for multi-user is a resource competition problem, especially in dynamic environments, which is difficult to be solved by traditional algorithms. In this article, we propose a two-layer hybrid system with local and edge computing, providing convenient computing and offloading services for vehicle users in dual dynamic scenarios of task generation and vehicle mobility. The delay and queuing situations are considered comprehensively in the formulated optimization problem, which can be solved by the proposed deep deterministic policy gradient-based computation offloading algorithm. The offloading process of the vehicle tasks in dynamic scenarios is transformed into a Markov decision process to obtain the offloading strategy. Simulation results demonstrate the performance advantages of two-tier computing architecture. Compared with random offloading, deep Q network-based offloading, and local computing, the algorithm proposed in this article gains the highest average reward of tasks.

Besides that, numerical results also prove that our algorithm has the lowest average delay under different computing capabilities of edge servers.

Index Terms—Internet of Vehicle, computation offloading, edge computing, deep reinforcement learning.

I. INTRODUCTION

THE emergence of the Internet of Things (IoT) [1] and the development of related state-of-the-art technologies have built a bridge for the integration of many traditional industries and advanced technologies [2], [3]. As a relatively mature field in the IoT, the Internet of Vehicles (IoV) can fully take advantage of the interconnection and interoperability of the Internet of Things [4]. On this basis, related technologies such as in-vehicle communication have also been widely simulated [5] and studied [6], and the transmission efficiency and convenience of the vehicle are significantly improved [7]. Each vehicle in the IoV needs to analyze and fuse massive data from the sensors (approximately 1 GB/s) to make safe decisions. It is predicted that the data generated by the vehicle-only sensors can reach up to 4,000 GB per day [8].

With the rapid development of various emerging technologies [9] and the leading of 6G network technology [10], the application scenarios of IoV are increasingly expanding [11]. This trend indicates that the IoV in the future will have a wider range of applications and stronger technical support. However, the limited resources of vehicles challenge the application of these technologies. The vehicular battery capacity and computing resources are both scarce, which greatly impairs the performance of vehicles executing computation-intensive and latency-sensitive tasks. Multi-access Edge Computing (MEC) is a promising approach with promising application prospects in real-time traffic monitoring in the IoV [12], [13]. By adopting the powerful edge computing paradigm, the efficiency of vehicle processing tasks can be significantly improved, thereby enhancing the driving experience and traffic safety.

The application of Deep Reinforcement Learning (DRL) to resource management has attracted wide attention in recent years [11]. DRL provides new caching and computing strategies for resource management and effectively reduces the long-term average latency of tasks [14]. In the existing literature on DRL for computation offloading problems, the adoption of MEC servers can provide vehicles with context awareness, mobility support, high service responsiveness, scalability, security, and privacy [15], [16]. Therefore, researchers

Manuscript received 27 October 2022; revised 20 May 2023 and 20 September 2023; accepted 21 September 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807500; in part by the National Natural Science Foundation of China under Grant 62072360, Grant 62001357, Grant 62172438, and Grant 61901367; in part by the Key Research and Development Plan of Shaanxi Province under Grant 2021ZDLGY02-09, Grant 2023-GHZD-44, and Grant 2023-ZDLGY-54; in part by the Natural Science Foundation of Guangdong Province of China under Grant 2022A1515010988; in part by the Key Project on Artificial Intelligence of Xi'an Science and Technology Plan under Grant 23ZDCYJSGG0021-2022, Grant 23ZDCYJSGG0008, and Grant 23ZDCYJSGG0002-2023; in part by the Xi'an Science and Technology Plan under Grant 20RGZN0005; and in part by the Proof-of-Concept Fund from the Hangzhou Research Institute of Xidian University under Grant GNYZ2023QC0201. The Associate Editor for this article was H. Wu. (Corresponding authors: Chen Chen; Houbing Song.)

Haofei Li is with the School of Telecommunications Engineering, Xidian University, Xi'an 710071, China (e-mail: lihaofei@stu.xidian.edu.cn).

Chen Chen is with the School of Telecommunications Engineering, Xidian University, Xi'an 710071, China, and also with the Xidian Guangzhou Institute of Technology, Guangzhou 510555, China (e-mail: cc2000@mail.xidian.edu.cn).

Hangguan Shan and Pu Li are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310007, China (e-mail: hshan@zju.edu.cn; 11931084@zju.edu.cn).

Yoong Choon Chang is with the Department of Electrical and Electronic Engineering, Universiti Tunku Abdul Rahman, Kajang 43000, Malaysia (e-mail: ycchang@utar.edu.my).

Houbing Song is with the Department of Information Systems, University of Maryland, Baltimore County (UMBC), Baltimore, MD 21250 USA (e-mail: h.song@ieee.org).

Digital Object Identifier 10.1109/TITS.2023.3325267

1558-0016 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See <https://www.ieee.org/publications/rights/index.html> for more information.

have been exploring how to optimize the application of MEC, to maximize their potential [17], [18]. Most of the existing studies consider a static offloading scenario for vehicle tasks [19], [20]. DDPG is applied in a work related to vehicular computational offloading to demonstrate its effectiveness [21]. In addition, this work uses the graph-weighted convolution network (GWCN) for traffic flow prediction for edge-to-edge models to address the problem of uneven space-time distribution of traffic flows. However, with the increasing popularity of video and other applications that need to continuously receive data in the IoV services, the offloading environment exhibits more complex dynamic characteristics. The existing work can not reflect the realistic situation of the future IoV environment. Therefore, the dynamic vehicle task offloading scenario over continuous time needs to be studied urgently.

To address this challenge, we propose an edge-to-end hybrid system that leverages the computational resources of edge servers to effectively mitigate the problem of limited computational resources of vehicles. For divisible vehicle tasks, we propose an edge offloading algorithm based on deep deterministic policy gradient (DDPG). The algorithm maps the dynamic offloading process of vehicle tasks to a Markov process and makes decisions for each vehicle task to improve the latency in a dynamic offloading environment. The main contributions of this paper are as follows:

- A two-tier hybrid edge computing architecture is proposed for many-to-one offloading scenarios. In the dual dynamic scenario of dynamic task arrivals and vehicle movements, the tasks of the vehicle are segmented and processed by the vehicle itself in parallel with the RSU.
- The objective optimization problem studied in this paper is to meet the computation requirements of vehicle tasks within the RSU coverage range under the latency constraint. The offloading environment is dynamic, including the dynamic generation of vehicle tasks and the movement of vehicles. The system also considers the current queueing situation under task offloading and its impact on the subsequent vehicle offloading.
- To cope with the dynamic offloading environment and random service requests, we propose a computation offloading algorithm based on DDPG to solve the optimization problem. A large number of simulation results demonstrate the performance advantages of the proposed scheme, which achieves lower latency and higher reward than the baseline algorithms.

The rest of this paper is organized as follows. In Section II, we discuss the related work based on heuristics and DRL algorithms. In Section III, we introduce the system model and the offloading problem we proposed. Section IV describes the modeling of MDP and details of our algorithm based on DDPG. In Section V, we present numerical results to illustrate the effectiveness of our algorithm. Finally, this paper ends in Section VI.

II. RELATED WORK

In this section, we first discuss the computation offloading work based on traditional heuristic algorithms. Secondly,

we explore edge computing offloading schemes based on DRL, which are divided into binary offloading and partial offloading.

A. Heuristics Algorithm in Computation Offloading

The IoV has become one of the most promising technological paradigms in the Internet of Things by combining cutting-edge technologies such as big data, artificial intelligence, and digital twins (DT) [22], [23], [24], [25]. The ensuing conflict is between the proliferation of new applications and the limited computing power of local devices. Therefore, in recent years, computing offloading has become a research hotspot in the field of IoV [26], [27]. Based on the Dinkelbach method, an efficient iterative algorithm was designed to solve the computation offloading problem to seek the maximum offloading rate and reduce energy consumption [28]. An energy-saving algorithm for dynamic computation offloading is proposed in [29], which provides an iterative strategy based on the solution of the convex problem within each slot. In addition to this, this work also considers energy harvesting. Reference [30] studied multi-vehicle non-cooperative computation offloading considering vehicle location and mission deadline constraints using a game method. Lyapunov optimization and semi-definite programming are used in [31], and the proposed online mobility-aware offloading and resource allocation algorithms do not require prior knowledge such as user mobility model and channel state information in advance.

B. DRL Algorithm in Computation Offloading

In the research of using DRL to solve the MEC server-assisted computation problem, the current task offloading can be further divided into two parts: binary task offloading and partial task offloading [32]. Many existing DRL-based studies are mainly designed for binary offloading scenarios. In [33], a novel MEC architecture is proposed that leverages heterogeneous servers and pervasive intelligence to provide customized E2E service experience in 6G networks. The computation offloading problem in a high mobility IoVs environment is addressed in [34], and a cooperative three-layer VMEC network is proposed that uses both moving and parked vehicles as fog nodes. Reference [35] studied dynamic computing offloading in a time-varying environment with multiple edge servers, and proposed an end-to-end DRL method to select the optimal offloading decision. Reference [36] proposed a blockchain-enabled edge computing framework based on the Soft Actor-Critic (SAC) algorithm to determine the optimal task assignment strategies in dynamic vehicle environments. The twin delayed deep deterministic policy gradient (TD3) algorithm is utilized to achieve the optimal offloading strategy [37]. Binary offloading is used in this work and the objective of the formulated problem is joint optimization of offloading delay and energy consumption.

Some literature also considers using partial task offloading [38]. Reference [9] proposes a novel method for computation offloading and service caching in MEC-enabled ITS, which uses DT to model the physical world in real-time and adopts a DRL approach to jointly optimize the offloading and caching

problem. For high-dimensional action spaces, [39] developed a CNN-based DDPG algorithm to capture the correlation of the state and action among different zones and minimize serving delay, in which, tasks can be split into multiple parts but the computations can only be performed by either one RSU individually or multiple RSUs concurrently. A DRL scheme combined with federated learning is proposed in [40], where tasks can be partially offloaded to the MEC for simultaneous execution locally.

Heuristic algorithms have high computational complexity and are mostly used for static offloading scenarios of vehicular tasks. For the dynamic vehicular task offloading scenarios in the future IoV environment over continuous time, DRL-based algorithms are more effective. To the best of the authors' knowledge, in the task-divisible vehicular mobility scenarios using DRL methods, the dynamic arrival of task offloading requests and the current task queueing situation and their impact on the subsequent offloading plan are rarely jointly considered. Our work fills this gap.

III. SYSTEM MODEL

In this section, we first introduce our task model and vehicle movement model. We then formulate the proposed computation offloading problem. The key notations are listed in Table I.

The offloading model of tasks is considered in a multi-user scenario in the IoV environment. The Software Defined Network (SDN) controller manages the offloading of vehicle tasks in Road Side Unit (RSU) communication. As shown in Fig. 1, the system model is divided into edge and vehicle terminal layers. There are multiple vehicles in each RSU communication where $\mathcal{I}=\{1, 2, \dots, I\}$ represents the set of vehicles in each RSU communication range. Each vehicle has its unique ID and performs its computing tasks at different times, denoted by $\mathcal{K}_i=\{1, 2, \dots, K_i\}$. Vehicles can offload vehicle computing tasks to RSU via V2I communication, while each RSU is configured with a corresponding Edge Server (ES) and Edge Software Defined Network Controller (ESDNC). The calculation queue of the task is maintained by ES, and ESDNC is mainly responsible for the offloading request queues. RSU allocates tasks to different task processing queues, depending on the message type. It is assumed that the coverage radius of RSU that can provide computing services for vehicles is R_e , the horizontal coordinate of the position of RSUs is x_e and the vertical coordinate is y_e . Tasks in both queues are handled according to the First Come First Service (FCFS).

Fig. 2 shows the steps of the offloading process of the vehicle task. First, The vehicle generates a task and sends the offloading request to RSU. RSU adds it to the offloading request queue in ESDNC. Second, the offloading request queue of the ESDNC follows the vehicles' request sequence and generates an offloading decision for each task in turn. Third, the vehicle receives and performs the task offloading decision made by ESDNC, and divides the task according to this decision. Finally, the vehicle uploads the task that has been split to RSU, while RSU adds the task to the task calculation queue. The ES performs the calculation process and then passes the results back to the corresponding vehicle.

TABLE I
KEY NOTATIONS

Notations	Description
D_i^k	Size of task k for vehicle i
x_i	The horizontal ordinate of the vehicle position
y_i	The vertical ordinate of the vehicle position
C_i	Computing capability of the vehicle i
C^e	Computing capability of the ES i
$T_i^{k,\max}$	Maximum tolerable delay for task k of vehicle i
β	Processing delay by ESDNC for each offloading request
ϵ	Unit conversion constant from million bits to bits
γ	Computational intensity
Num_i^k	Number of tasks to process before the current task
$T_{i,dec}^k$	Decision delay of task k for vehicle i
$T_{i,local}^k$	Local computation delay
$T_{i,wait}^k$	Local waiting delay
$T_{i,local}^k$	Local processing task delay
$T_{i,edge}^k$	Delay of processing tasks in the ES
$T_{i,etran}^k$	Delay of uploading task to the ES
$T_{i,ewait}^k$	Delay of waiting in the ES
$T_{i,ecom}^k$	Delay of computing in the ES
r_i	Upload rate of vehicle i
$L_{e,i}$	The straight-line distance between the vehicle and the RSU
$R_i^{l,k}$	Size of local processing queue for local computing
$R_i^{e,k}$	Size of the task processing queue of the ES
$C_{i,queue}^k$	CPU cycles of the ES for remain task
$T_{i,process}^k$	Total task processing delay of task k of vehicle i
$T_{i,dec}^k$	The decision delay
T_i^k	Total completion delay
v	Vehicular speed
$\alpha(t)$	The angle between the vehicle and the x-direction
η	The smallest divisible size of the task

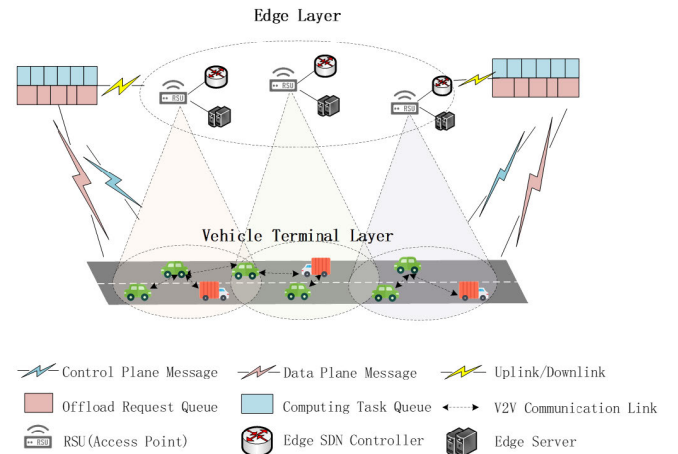


Fig. 1. Two-layer computational offloading queuing model.

A. Task Model

The set κ describes the computing task of each vehicle. The task k of the vehicle i can be described as $\kappa_i^k = \{D_i^k, c_i, T_i^{k,\max}, x_i, y_i\}$, $k \in \mathcal{K}$, $i \in \mathcal{I}$. D_i^k stands for

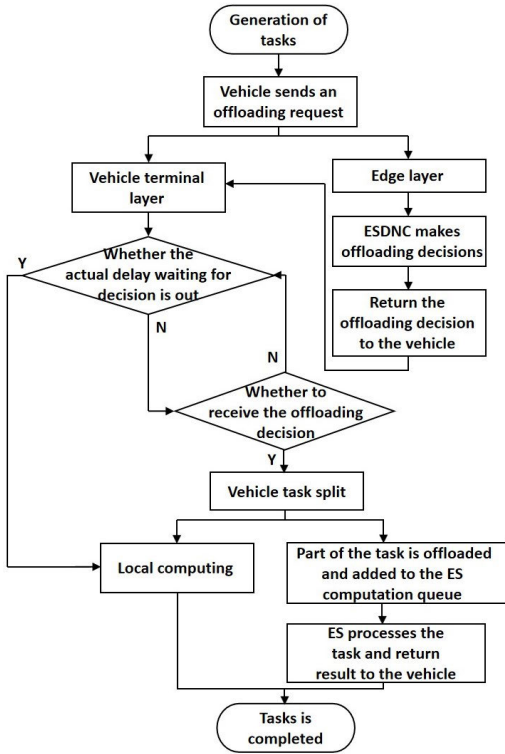


Fig. 2. Task offloading process.

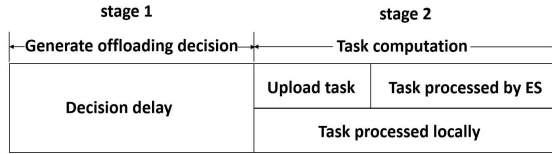


Fig. 3. Task split model.

the size of the task, while x_i and y_i describe the vehicle's location, c_i represents the capability of the vehicle to perform the task, and $T_i^{k,\max}$ is the maximum tolerable delay for a task.

The total delay of the task offload process is shown in Fig. 3, including the decision delay and the task computing delay. Since the task calculation delay includes two parts in parallel, the calculation delay of a complete task is the larger one of the local calculation delay and the edge calculation delay.

1) *Offloading Decision-Making Process*: Considering the different densities of offloading requests in the IoV environment, the queue is handled in three ways, as shown in Fig. 4.

- Case 1: When the offloading request of task $k-1$ from vehicle $i-1$ arrives at the offloading request queue of ESDNC, there is no accumulated offloading request in the queue temporarily. So it can be processed directly.
- Case 2: When the offloading request for task k generated by vehicle i arrives at ESDNC, it is discovered that an offloading request for task $k-1$ from vehicle $i-1$ is already being processed in the offloading request queue, consequently, task k must queue in the line and wait.
- Case 3: When a new offloading request for task $k+1$ generated by vehicle $i-1$ arrives at ESDNC, it is discovered that in addition to an offloading request for task $k-1$ from

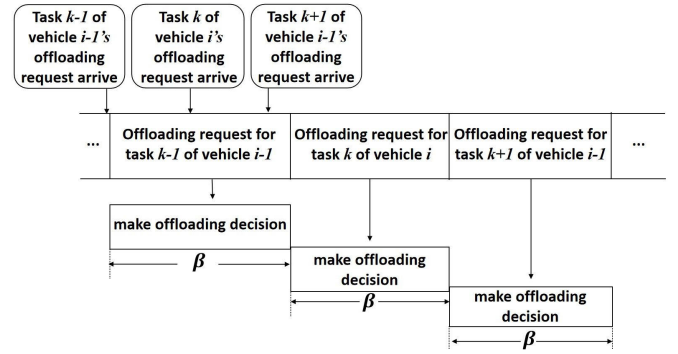


Fig. 4. Offloading request decision model.

the same vehicle currently being processed, an offloading request for task k from vehicle i is also waiting to be processed in the queue.

The ESDNC will process requests from the offloading request queues. In order to simplify the calculation, we assume that the processing time of each offloading request in our model is equal and can be described as β . Under this assumption, Case 1's decision delay is proportional to the length of the task queue. Cases 2 and 3 also include decision waiting delays. It can be expressed as $Num_i^k * \beta$, where the Num_i^k is the number of tasks waiting to be processed in the queue. It should be noted that case 1 satisfies $Num_i^k = 0$. Therefore, for the task i of vehicle k the decision delay $T_{i,dec}^k$ is the sum of the decision processing delay and the decision waiting delay of the offloading request. It can be expressed as

$$T_{i,dec}^k = (Num_i^k + 1) * \beta \quad (1)$$

2) *Task Processing*: $A_i^k = \{a_{i,0}^k, a_{i,1}^k\}$ denotes the offloading decision of task k of the vehicle i . Among them, $a_{i,0}^k$ is the local calculation index of the vehicle task, while $a_{i,1}^k$ is the index of the edge computation, which decides whether the k th task needs to be offloaded to the RSU and use ES's computing resources.

Considering the multi-task situation in dynamic scenes, the local computation of the vehicle also yields a waiting queue, while the local processing delay of the vehicle task should include the local computing delay $T_{i,lcom}^k$ and the local waiting delay $T_{i,wait}^k$. The local computing delay $T_{i,lcom}^k$ is given by

$$T_{i,lcom}^k = \frac{\epsilon D_i^k a_{i,0}^k \gamma}{c_i} \quad (2)$$

where $D_i^k a_{i,0}^k$ denotes the parts of the task k of the vehicle i need to be calculated locally. ϵ is the constant help unit conversion from million bits to bits. The number of CPU cycles required to compute one bit of data can be called computational intensity γ .

Local waiting delay $T_{i,wait}^k$ can be calculated according to the local queue's size $R_i^{l,k}$ and the decision delay $T_{i,dec}^k$. Local queue size $R_i^{l,k}$ represents the task queue remaining in the local CPU while waiting for a task decision. This queue includes part of the current task, and may also include the last task that is not completed. If the calculation of the previous task is

still in progress, there is a local process waiting delay for the current task. $T_{i,dec}^k * c_i$ is the task size calculated by the vehicle i during the waiting period for the offloading decision, which should be subtracted from $R_i^{l,k}$. If the local queue's size $R_i^{l,k}$ is equal to $T_{i,dec}^k * c_i$ it means only this task is in the queue. In other words, the local waiting delay is 0, and the current task can be processed directly. The local waiting delay can be given by

$$T_{i,lwait}^k = \frac{\max(\epsilon R_i^{l,k} \gamma - T_{i,dec}^k * c_i, 0)}{c_i} \quad (3)$$

The vehicle local processing task delay is obtained as

$$T_{i,local}^k = T_{i,lwait}^k + T_{i,lcom}^k \quad (4)$$

For the tasks offloaded, the delay of processing tasks in the ES includes the uploading delay $T_{i,etran}^k$, the waiting delay $T_{i,ewait}^k$, and the computing delay $T_{i,ecom}^k$.

The vehicle can communicate with RSU directly. The vehicle can communicate with RSU directly, and we use the 802.11p protocol in our work. The upload rate of vehicle i is given by:

$$r_i = B \log_2(1 + \frac{(p_i L_{e,i}^{-\delta} h^2)}{\sigma^2}) \quad (5)$$

where σ^2 is the power of the additive Gaussian white noise, p_i is the vehicle's transmission power, and h is the channel fading factor of the communication link between the vehicle and RSU. B is the transmission bandwidth. $L_{e,i}$ represents the straight-line distance between the vehicle and the RSU, which can be obtained as

$$L_{e,i} = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \quad (6)$$

where (x_c, y_c) is the coordinates of the center of RSU. According to Formula (5), the upload time $T_{i,etran}^k$ of the vehicle task can be calculated as

$$T_{i,etran}^k = \frac{D_i^k a_{i,1}^k}{r_i} \quad (7)$$

Here, $D_i^k a_{i,1}^k$ denotes the parts of the task k of the vehicle i that need to be calculated by the ES. Suppose that C^e represents the computing capability of the ESs. The computation delay $T_{i,ecom}^k$ of tasks is given as

$$T_{i,ecom}^k = \frac{\epsilon D_i^k a_{i,1}^k \gamma}{C^e} \quad (8)$$

The task waiting delay $T_{i,ewait}^k$ of the ES for vehicle i task k is calculated based on the CPU cycles of the waiting tasks in the computing queue $C_{i,queue}^k$ and the computing capability C^e of the edge server. Since the ES can still process tasks in the computing queue during the offloading decision-making process and the task transmission process, the amount of the task waiting to be calculated should subtract the amount of the tasks processed by the edge server during this time. Considering the scenario where there are no waiting tasks

in the task computing queue, $C_{i,queue}^k$ should be at least 0. $C_{i,queue}^k$ can be expressed as

$$C_{i,queue}^k = \max(\epsilon R_i^{e,k} \gamma - (\frac{D_i^k a_{i,1}^k}{r_i} + \beta) * C^e, 0) \quad (9)$$

The task waiting delay $T_{i,ewait}^k$ of the ES is given by

$$T_{i,ewait}^k = \frac{C_{i,queue}^k}{C^e} = \frac{\max(\epsilon R_i^{e,k} \gamma - (\frac{D_i^k a_{i,1}^k}{r_i} + \beta) * C^e, 0)}{C^e} \quad (10)$$

The task processing delay $T_{i,edge}^k$ of ES is given by

$$T_{i,edge}^k = T_{i,etran}^k + T_{i,ewait}^k + T_{i,ecom}^k \quad (11)$$

Considering the parallel processing of the vehicle and the ES, the total processing delay of task k of vehicle i is a larger value of the local computation and the edge computation, given as

$$T_{i,process}^k = \max(T_{i,local}^k, T_{i,edge}^k) \quad (12)$$

The total completion delay for each vehicle task T_i^k is obtained by the decision delay and the total task processing delay:

$$T_i^k = T_{i,dec}^k + T_{i,process}^k \quad (13)$$

B. Vehicle Moving Model

In the actual traffic environment, the vehicle's speed will change due to the dynamic condition of the road traffic. Therefore, the vehicle moving model can be constructed for the offloading scenario, as shown in Fig. 5. The speed of each vehicle can be expressed as a function of time, described by the set $\mathcal{V} = \{v_1, v_2, \dots, v_I\}, i \in \mathcal{I}$. The distance of the vehicle traveling in the x and y directions is denoted by $d_{i,x}$ and $d_{i,y}$, respectively, which are given by

$$\begin{cases} d_{i,x,t} = \int_0^t v_i(t) * \cos \alpha(t) dt, i \in \mathcal{I} \\ d_{i,y,t} = \int_0^t v_i(t) * \sin \alpha(t) dt, i \in \mathcal{I} \end{cases}$$

Here $\alpha(t)$ is the angle between the vehicle and the x-direction. While processing vehicle tasks, the vehicle's position changes during this time, and its real-time position coordinates are denoted by x_i and y_i , written as

$$\begin{aligned} x_i &= x_i + d_{i,x,t}, \\ y_i &= y_i + d_{i,y,t}. \end{aligned} \quad (14)$$

C. Problem Formulation

According to the above analysis process, the vehicle task completion delay has two parts, the decision delay, and the

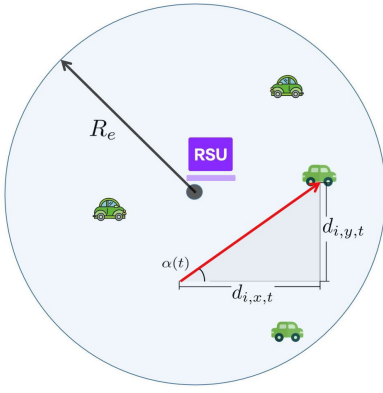


Fig. 5. Vehicle moving model.

task processing delay. Then, the optimization objective is given by Formula (16):

$$\begin{aligned}
 T &= \min_{\{a_{i,0}^k, a_{i,1}^k\}} \sum_{i=1}^I \sum_{k=1}^K T_i^k \\
 &= \min_{\{a_{i,0}^k, a_{i,1}^k\}} \sum_{i=1}^I \sum_{k=1}^K (T_{i,dec}^k + T_{i,process}^k) \\
 s.t. \\
 C1 : &0 \leq a_{i,0}^k, a_{i,1}^k \leq 1 \\
 C2 : &a_{i,0}^k D_i^{l,k} \geq \eta \\
 C3 : &a_{i,1}^k D_i^{e,k} \geq \eta \\
 C4 : &a_{i,0}^k + a_{i,1}^k = 1 \\
 C5 : &T_i^k \leq T_i^{k,max} \\
 C6 : &\sqrt{(x_e - x_i)^2 + (y_e - y_i)^2} \leq R_e \quad (15)
 \end{aligned}$$

where η is the smallest divisible size of the task. Constraint 1 represents the value range of the indicator of the offloading method. Constraints 2 and 3 ensure that the partition size of the task is larger than its minimum divisible size. Constraint 4 ensures that the task is fully handled, while constraint 5 confirms that the task completion delay is less than or equal to the maximum tolerable delay. Constraint 6 ensure that the vehicles move within the coverage range of the RSU.

IV. DDPG-BASED COMPUTATION OFFLOADING ALGORITHM

In this section, we will describe the modeling of the MDP and the details of the DDPG algorithm used in solving the optimization problem we propose.

Following the problem model described in the previous section, we propose a DRL algorithm to offload all vehicle tasks with minimal delay. We adopt a DDPG algorithm that combines value-based and policy-based methods. It is based on the Actor-Critic (AC) framework, which can better learn and select actions in continuous states.

A. Markov Decision-Making Process

To apply the DDPG algorithm in our vehicle task offloading model, we define the MDP as a tuple $\{S_i^k, A_i^k, r_i^k\}$. The three

terms represent actions, states, and rewards, respectively. The specific definitions of these three elements are expressed as follows.

- **State:** From the optimization problem we formulate, we define the state space by a collection of quantities that describe the entire system. The state includes the following five items: computational task size, vehicle computing capability, maximum tolerable delay, and residual task size for ES and local residual task size. These items describe the state of the current offloading environment, which is expressed as $S_i^k = \{D_i^k, c_i, T_i^{k,max}, R_i^{e,k}, R_i^{l,k}\}$.
- **Action:** To make offloading decisions for each task, the action space is defined as $A_i^k = \{a_{i,0}^k, a_{i,1}^k\}$, which satisfies the constraint of Formula (16). It should be noted that in order to dynamically solve the task in the vehicle environment, we define the range of the action to satisfy: $a_{i,0}^k, a_{i,1}^k \in [0, 1]$. The action space is further expanded compared to the binary offloading scheme.
- **Rewards:** The goal of DDPG is to solve the accumulated maximum reward. We proposed the objective function to be optimized in the problem modeling in the previous section. Therefore, we can solve the optimization objective in Formula (16) by transforming it into a reward function in a reasonable way with the help of the solving mechanism of DRL. We denote the reward function by r_i^k , which is given by

$$r_i^k = (T_i^{k,max} - T_i^k) * \xi \quad (16)$$

where ξ is the gain of saving per unit of time. If constraint 5 is not satisfied and the task cannot be completed within the maximum tolerable delay, the task offloading will then fail and $r_i^k = -1$ holds. From the definition of reward, it can be concluded that for a task, the greater the difference between the maximum tolerated delay and the actual completion delay of the task, the greater the reward will be.

- **Agents:** In a dynamic offloading environment, we adopt ESDNC as an agent in the network framework, which makes rational decisions by observing the environment.

The MDP is established according to the state, action, and reward in DDPG. For each offloading decision, the jump process of the offloading environment state is as follows. There we take task k of the vehicle i as an example to illustrate. For convenience, we express it as t -th task. Then, the state can be represented by $S_t = \{D_t, c_t, T_t, R_t^e, R_t^l\}$. Correspondingly, the offloading action is represented by $A_t = \{a_{t,0}, a_{t,1}\}$. After performing the offloading action, the state will jump to the next state.

The queue size for the remaining task R_{t+1}^e of the ES in the next state is the current state R_t^e minus the amount of calculation consumed by the decision process and task upload process. Therefore, the remaining task size for ES is expressed as

$$R_{t+1}^e = \frac{\max(\epsilon R_t^e \gamma - (\frac{D_t a_t}{r_t} + \beta) * C^e, 0)}{\epsilon \gamma} \quad (17)$$

In the next state, the vehicle task size, the maximum tolerable delay of the task, and the vehicle computing capacity

are equal to the corresponding ones for the previous computing task, that is

$$\{D_t, c_t, T_t\} = \{D_{t+1}, c_{t+1}, T_{t+1}\} \quad (18)$$

The local remaining computing task size R_{t+1}^l of the vehicle task is divided into two cases:

- 1) If the task t is not corresponding to the vehicle, the R_{t+1}^l can be described as the current state R_t^l minus the size of the task that the vehicle computes during the decision-making process, given as follows:

$$R_{t+1}^l = \frac{\max(\epsilon R_t^l \gamma - \beta * c_t, 0)}{\epsilon \gamma} \quad (19)$$

- 2) If the task t is corresponding to the vehicle, the R_{t+1}^l needs to add the size of the task assigned to the vehicle's local computation compared to the previous case:

$$R_{t+1}^l = \frac{\max(\epsilon R_t^l \gamma - \beta * c_t, 0)}{\epsilon \gamma} + D_t a_{t,0} \quad (20)$$

B. DDPG Based Offloading Scheme

In the previous sub-section, we gave definitions of actions, states, and rewards to illustrate the Markov decision process for problem modeling. In this section, we will describe how to solve the computational offloading problem of tasks through DDPG-based algorithms.

We consider a dynamic task offloading environment where task generation and offloading requests are random. Random requests can change the current offloading environment. Therefore, for dynamically changing situations, we employ DDPG, an algorithm suitable for solving continuous actions in DRL. The algorithm adopts a deterministic strategy, that is, the actions taken by the same strategy in the same state are uniquely determined. In a continuous action space, the probability of taking different actions in the same state is different. These different probability distributions can help us get the optimal solution. If only the action with the highest probability is taken approximately, the probability distribution of the remaining actions can be ignored. ESDNC acts as an agent to make offloading decisions for tasks. The vehicle first encapsulates the relevant data affecting the offloading decision in the offloading request and uploads it to the ESDNC. ESDNC makes offloading decisions based on offloading requests and then sends these decisions to vehicles. This procession is shown in Fig. 6.

Following the Markov decision process described in the previous section, we add noise to the action space. The Ornstein-Uhlenbeck stochastic process in which the ground noise is time-related is introduced. This increases the exploration space of the model and improves the generalization ability of the model. In order to further improve the stability of the algorithm, DDPG contains two architectures, called main network and target network, respectively. These two architectures include an actor network and a critic network respectively. The actor network is responsible for using a neural network to fit the policy function to obtain actions through the input state. In DDPG, the Q value is used to evaluate whether the actor network can efficiently obtain the

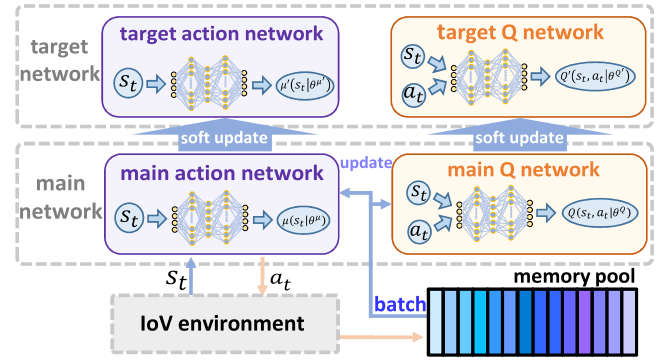


Fig. 6. Proposed algorithm based on DDPG.

policy. The critic network uses a neural network to model the value function and finds the most accurate Q value estimate by inputting actions and states. Since the actor network is mainly used to output actions, and the critic network outputs the Q value, the two networks in our architecture are represented as action network and Q network respectively.

The update process of the Q network is divided into two steps. The first step is to use the Q network of the target network to complete the calculation of the target Q value:

$$y_t = r_t + Q'(S_{t+1}, \mu'(S_{t+1} | \theta^{\mu'}) | \theta^{Q'}) \quad (21)$$

Here, the former is the reward of t th minibatch, the latter is the output for the Q network of the target network. $\mu'(S_{t+1} | \theta^{\mu'})$ is the output for action network of target network. We use the estimated Q value given by the Q network of the main network to calculate the loss between it and the target Q value. The loss is calculated as

$$loss = \frac{1}{N} \sum (y_j - Q(S_j, \mu(S_j, a_j | \theta^Q)))^2 \quad (22)$$

where $Q(S_j, \mu(S_j, a_j | \theta^Q))$ express the estimated Q value. MSE is used as the Loss function for error estimation. It should be noted that the update of the main network is performed by Stochastic Gradient Descent (SGD), while the update of the target network is performed by the soft update algorithm, which can be expressed as

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'} \quad (23)$$

$$\theta^{\mu'} \leftarrow \tau \theta^{\mu} + (1 - \tau) \theta^{\mu'} \quad (24)$$

where θ^Q and θ^{μ} are the parameters of the Q network and the action network for the main network, respectively. $\theta^{Q'}$ and $\theta^{\mu'}$ are the parameters of the two parts in the corresponding target network. Adjustment τ can control the speed of the update.

In the soft update algorithm above, network parameters change little and can slowly approximate the parameters of the main network. This update mechanism further increases the stability of the learning process. The algorithm flow is shown in Algorithm 1.

C. Complexity Analysis

The time complexity of the proposed approach is mainly derived from the training process of DDPG in Algorithm 1 [41]. According to the previous discussion in the article,

Algorithm 1 Algorithm for Vehicle Task Offloading Based on DDPG**Input:** Vehicle Task Parameters and Computing capability: $D_i^k, c_i, T_i^{k,\max}, R_i^{l,k}$ ES computing parameters: $R_i^{c,k}$;**Output:** offload strategy: $A = \{a_i^k\}, k \in \mathcal{K}, i \in \mathcal{I}$;

- 1: Initializing Q network and action network parameters θ^Q and θ^μ ;
- 2: initializing the parameters $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$ of the target Q network and the target action network, and updating interval of the target network parameters
- 3: **for** $i \leftarrow 1$ to EPISODE **do**
- 4: The Initialization State S_1 is determined according to the task sequence and the calculation parameters of the ES
- 5: **for** $t \leftarrow 1$ to STEP **do**
- 6: calculate the output strategy $a_t \leftarrow \mu(S_t|\theta^\mu)$ according to the action network;
- 7: using probability ξ to randomly select action a_t to increase the noise of random distribution, the exploratory action $a_t \leftarrow \mu(S_t|\theta^\mu) + \mathcal{N}$ is obtained
- 8: perform action A to offload the task, then calculate the next state S_{t+1} according to Formula (19), (20) and (21), reward r_t according to Formula (18);
- 9: whether the offloading scheme of the task is judged according to the constraint of Formula (16), if $done_t \leftarrow \text{true}$ is not satisfied;
- 10: Store $(S_t, a_t, r_t, S_{t+1}, done_t)$ in memory pool;
- 11: set $S^{t+1} = S^t$;
- 12: random sample N Size $(S_t, a_t, r_t, S_{t+1}, done_t)$ data sets from memory pool;
- 13: **if** $done$ is true **then**
- 14: Calculate $y_t = r_t + Q'(S_{t+1}, \mu'(S_{t+1}|\theta^{\mu'})|\theta^{Q'})$;
- 15: **else**
- 16: Calculate $y_t = r_t$;
- 17: **end if**
- 18: updating Q network parameters by loss function $loss = \frac{1}{N} \sum (y_j - Q(S_j, \mu(S_j, a_j|\theta^Q)))^2, j \in \{1, \dots, N\}$;
- 19: update the parameters of the action network through the gradient strategy $\nabla_{\theta^\mu} J = \frac{1}{N} \sum (\nabla_a Q(S_j, a_j|\theta^Q) * \nabla_{\theta^\mu} \mu(S_j|\theta^\mu)), j \in \{1, \dots, N\}$;
- 20: update the parameters of the target Q network $\theta^{Q'}$ and the target action network $\theta^{\mu'}$
 $\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$
 $\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$;
- 21: **end for**
- 22: **end for**

DDPG consists of four parts of neural networks, namely two action networks and two Q networks. Assuming the former includes M fully connected layers and the latter includes N fully connected layers, the time complexity of the algorithm can be expressed by:

$$O\left(\sum_{m=0}^{M-1} v_{a,m} v_{a,m+1} + \sum_{n=0}^{N-1} v_{Q,n} v_{Q,n+1}\right) \quad (25)$$

TABLE II
PARAMETERS OF NETWORK

Parameter	Value
Number of input units in action network	5
Number of input units in Q network	7
Number of units in hidden layer 1 of network	20
Number of units in hidden layer 2 of network	10
Action network learning rate	$\{1^{-5}, 5^{-5}, 1^{-4}\}$
Q network learning rate	$\{1^{-5}\}$
Soft update speed control factor	0.001

where $v_{a,m}$ and $v_{Q,n}$ represents the number of neurons in the m layer of action networks and the n layer of Q networks respectively.

The structure of the DQN algorithm is similar to that of the Q network in the algorithm proposed in this paper, hence the expression of time complexity is also similar. However, the difference lies in the fact that the DDPG algorithm can handle continuous actions, while DQN requires the discretization of continuous actions. To approach continuity, the discretization of DQN needs to be fine enough, which, in turn, requires a larger action space for DQN. In the time complexity of DQN, the multiplication term is directly proportional to the dimension of the action space, and is positively correlated with the number of users and the length of the task queue. Therefore, under the same conditions, the time complexity of the algorithm described in this paper is lower than that of DQN.

V. SIMULATION

In this section, numerical simulations are performed to evaluate the performance of the proposed algorithm and the effectiveness of task computation offloading.

A. Simulation Parameters and Indicators

1) *Performance Metrics*: The offloading scenario we consider is that there are multiple vehicles in the coverage area of one ES, and each vehicle will generate multiple tasks one after another. We use the completion delay of tasks and cumulative reward of all offloading tasks as a measure to judge the algorithm's convergence and the performance of offloading strategy.

2) *Simulation Parameters*: According to the section IV, the main network and the target network respectively include two parts, namely the action network and the Q network. The parameters of the action network and Q network are shown by TABLE II. The experimental configuration and environment parameters are given by TABLE III.

B. Simulation Design

For the DDPG-based offloading method we proposed, we investigate the convergence performance of the network at different learning rates. The three curves in Fig. 7 are generated at different learning rates of the action network as 1^{-4} 5^{-5} and 1^{-5} . After the same number of training steps,

TABLE III
MAIN SIMULATION PARAMETERS

Parameter	Value
Number of tasks	[2, 10]
Number of vehicles	[3, 5]
Tasks queue	1000, 4000
Maximum tolerable delay	1s
The size of the task	[1000,2000) Mb
Channel bandwidth	10 MHz
Gaussian white noise power	100 dBm
Unit conversion from Mbits to bits	2^{20}
Computing capability of ES	15 GHz, 20 GHz, 25 GHz
Transmission power of the vehicle	20 dBm
Coverage radius of RSU	300 m
Computing capability of the vehicle	0.5 GHz
Moving speed of vehicles	50.4 km/h

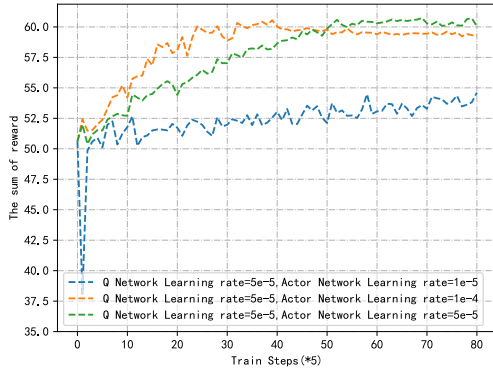


Fig. 7. Cumulative reward for different learning rates.

the cumulative reward curves for training rates 1^{-4} and 5^{-5} gradually flatten out at the end of the training process. This proves the convergence of DDPG at the proper training rate. In contrast, the curve with training rate 1^{-5} is still slowly rising at the end of the curve. This is because, in gradient descent, a small learning rate can lead to slow convergence. For the sum of the cumulative rewards of the task, the learning rate 1^{-4} results in lower results than the learning rate 5^{-5} after 250 train steps. It is because a larger learning rate causes concussion around the optimal value. Therefore, in the subsequent experiments, we set the learning rate as 5^{-5} to obtain a more stable result.

The brief descriptions of the schemes are as follows:

- *Paper offloading algorithm*: Tasks are processed using the computation offloading method based on DDPG proposed in this paper.
- *DQN offloading algorithm*: Generate binary discrete decisions, and use the DQN algorithm to perform task offloading calculations.
- *Random offloading algorithm*: All tasks are randomly chosen to be computed locally on the vehicle or offloaded to the ES for processing.
- *Local computing*: All tasks are computed only by the vehicle's local CPU.

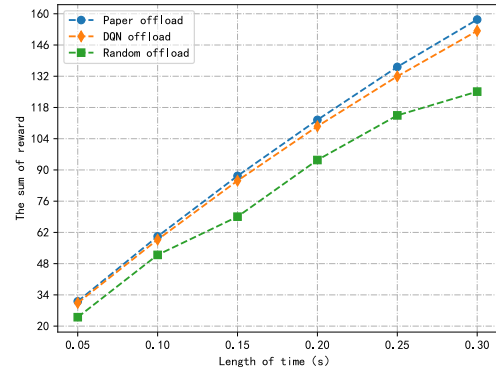


Fig. 8. Cumulative reward of different scenarios.

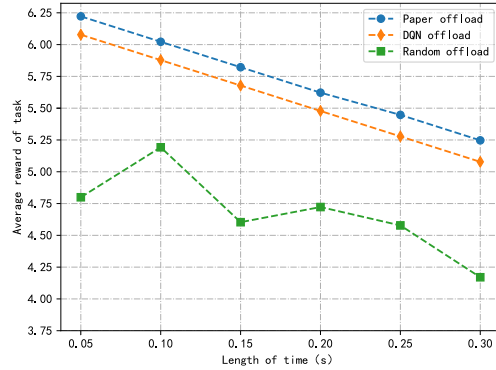


Fig. 9. Average reward of different scenarios.

Fig. 8 shows the cumulative reward sum of vehicle task edge offloading at different time lengths. As the time length increases, the cumulative reward sum gradually increases. This is because the longer the time, the more random tasks are generated, and most of the offloading tasks can be completed within the maximum tolerable delay through offloading decisions. However, the average reward of task processing decreases when the length of time increases, as shown in Fig. 9. This is because the increase in the number of tasks causes the length of the task queue to continue to grow, and tasks need to wait to be processed in sequence.

We also compare the cumulative reward and average reward under different algorithms. It can be seen from Fig. 8 and Fig. 9 that the proposed scheme achieves the best results in terms of both cumulative reward and average reward. The DQN algorithm is mainly aimed at discrete actions, and needs to binarize the values. The performance of the random offloading algorithm is the worst, regardless of the cumulative reward or the average reward. This unintelligent behavior may block computing resources, which leads to the lowest reward. In Fig. 9, the average reward curve of the random offloading algorithm fluctuates significantly over time, which means it is difficult to provide effective services for vehicle edge computing scenarios that require high performance and robustness.

Fig. 10 shows the relationship between the average task delay and different offloading schemes when the computing capability of ES varies from small to large. The uppermost and lowermost black horizontal lines in the boxplot represent the

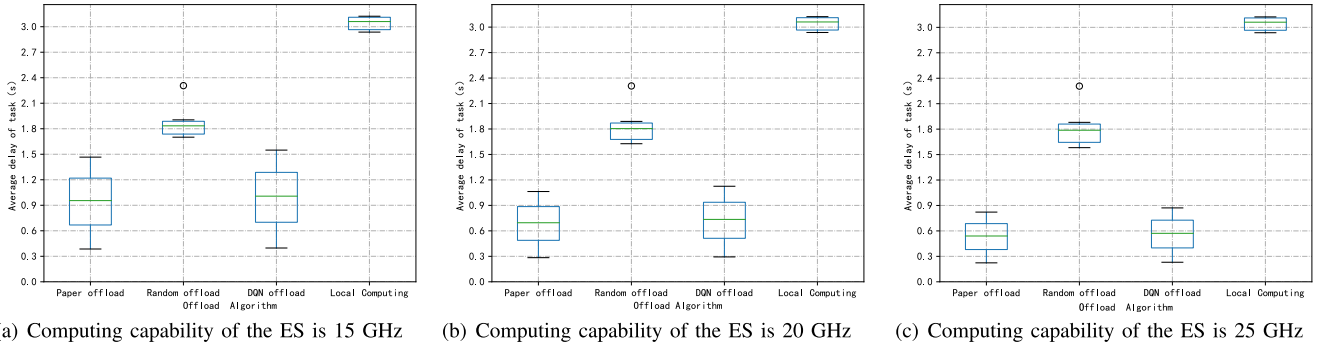


Fig. 10. Average delay of task processing using 4 schemes under different ES computing capability.

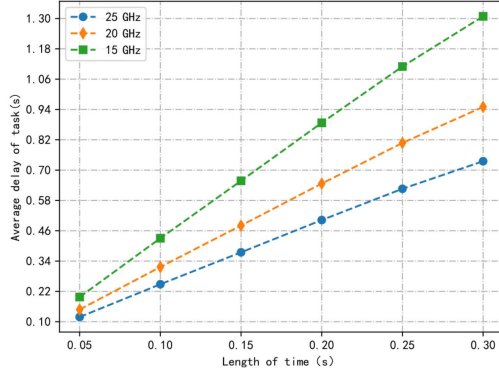


Fig. 11. Average delay of task processing under different computing capability of the ES with the proposed DDPG method.

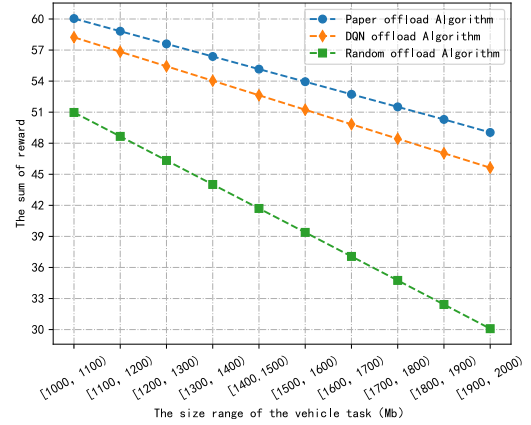


Fig. 12. Sum of reward with different task size under different schemes.

maximum and minimum values of the average delay of tasks, respectively. The bottom and top of the blue boxes represent the 25th and 75th percentiles position in the statistical data arranged from small to large. The green horizontal line is the median of the average delay. By comparing (a), (b), and (c) of Fig. 10, it can be seen that changes in the computing capability of ES will not affect the random offloading scheme and the local computing scheme. This is because the main delay of these two schemes still depends on the local computing delay. For our proposed method and the DQN method, as the computing capability of edge devices increases, the size of tasks that can be processed per unit of time increases, and the average processing delay of vehicle tasks decreases accordingly. Therefore, the results prove that the two DRL offloading algorithms outperform the local computing scheme and the random offloading scheme.

Fig. 11 shows the change in the average task delay under different computing abilities of ES as the task processing time increases under the DDPG scheme. As can be seen from Fig. 11, when a certain ES computing capacity is fixed, as the processing time increases, the value of the average processing delay of the task increases. This is because more tasks arrive dynamically and randomly. However, as the computing capability of ES increases, the slope of the curve decreases, that is, the speed at which the average processing delay of tasks increases gradually decreases. This shows that the increase in the computing capability of ES effectively solves the problem that the increase in randomly arriving tasks makes the waiting time too long.

Fig. 12 shows the effect of the sum of rewards with different task sizes when different processing schemes are used. Although the slopes of the DDPG offloading scheme and the DQN offloading scheme are similar, the sum of rewards for the DDPG scheme is higher than for the DQN offloading scheme. The experimental result of the random offloading scheme is the worst, and the sum of the reward is almost half of the other two schemes when the task size is in the range [1900, 2000). This demonstrates the effectiveness of the DDPG scheme in the continuous dynamic task offloading scenario.

VI. CONCLUSION

In this paper, a hierarchical architecture is proposed for multi-task multi-vehicle scenarios in IoV with random arrival of vehicular offloading tasks. For the dual dynamic offloading environment of task generation and vehicle mobility, we split the task into two parallel computing parts, which are processed by the local computing resources of the vehicle and the edge computing devices on the RSU. When a task offloading request arrives, we first use a Markov decision process to generate an offloading decision. Then we use the decision to parallelize the vehicular task and optimize the task latency. In addition to meeting the latency requirements of vehicular tasks within the RSU coverage area, we also pay attention to the current queueing situation of task offloading and its impact on subsequent vehicular offloading plans. Extensive simulation

results show that compared with other baseline algorithms, our proposed algorithm can effectively reduce the task completion latency while adapting to the dynamic offloading environment.

In our future work. We will consider multi-hop edge offloading of vehicles based on the global network topology of SDN. Besides, We will use the non-terrestrial network to further extend the edge offloading range of vehicular tasks, to make the computation offloading of vehicular tasks more reliable and efficient.

REFERENCES

- [1] N. Chen, T. Qiu, M. Daneshmand, and D. O. Wu, "Robust networking: Dynamic topology evolution learning for Internet of Things," *ACM Trans. Sensor Netw.*, vol. 17, no. 3, pp. 1–23, Aug. 2021.
- [2] J. Li et al., "A federated learning based privacy-preserving smart healthcare system," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 2021–2031, Mar. 2022.
- [3] N. Lv et al., "A hybrid-attention semantic segmentation network for remote sensing interpretation in land-use surveillance," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 2, pp. 395–406, Feb. 2023.
- [4] C. Chen, G. Yao, C. Wang, S. Goudos, and S. Wan, "Enhancing the robustness of object detection via 6G vehicular edge computing," *Digit. Commun. Netw.*, vol. 8, no. 6, pp. 923–931, Dec. 2022.
- [5] M. Mukhtaruzzaman and M. Atiquzzaman, "Cloud based VANET simulator (CVANETSIM)," 2021, *arXiv:2101.11147*.
- [6] A. Ihsan, W. Chen, S. Zhang, and S. Xu, "Energy-efficient NOMA multicasting system for beyond 5G cellular V2X communications with imperfect CSI," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 10721–10735, Aug. 2022.
- [7] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and Industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.
- [8] W. Xu et al., "Internet of Vehicles in big data era," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 19–35, Jan. 2018.
- [9] X. Xu, Z. Liu, M. Bilal, S. Vimal, and H. Song, "Computation offloading and service caching for intelligent transportation systems with digital twin," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20757–20772, Nov. 2022.
- [10] D. Zhu, M. Bilal, and X. Xu, "Edge task migration with 6G-enabled network in box for Cybertwin-based Internet of Vehicles," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4893–4901, Jul. 2022.
- [11] X. Zhou et al., "Edge computation offloading with content caching in 6G-enabled IoV," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 31, 2023, doi: [10.1109/TITS.2023.3239599](https://doi.org/10.1109/TITS.2023.3239599).
- [12] T. Xiao, C. Chen, Q. Pei, and H. H. Song, "Consortium blockchain-based computation offloading using mobile edge platoon cloud in Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17769–17783, Oct. 2022.
- [13] F. Jia, C. Chen, J. Li, L. Chen, and N. Li, "A BUS-aided RSU access scheme based on SDN and evolutionary game in the Internet of Vehicle," *Int. J. Commun. Syst.*, vol. 35, no. 12, p. e393, Aug. 2022.
- [14] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4925–4934, Jul. 2021.
- [15] J. Liu et al., "RL/DRL meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8315–8338, Jun. 2022.
- [16] X. Wu, X. Xu, and M. Bilal, "Toward privacy protection composition framework on Internet of Vehicles," *IEEE Consum. Electron. Mag.*, vol. 11, no. 6, pp. 32–38, Nov. 2022.
- [17] C. Chen, G. Yao, L. Liu, Q. Pei, H. Song, and S. Dustdar, "A cooperative vehicle-infrastructure system for road hazards detection with edge intelligence," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5186–5198, May 2023.
- [18] Y. Ju et al., "Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5555–5569, May 2023.
- [19] K. Bok, J. Lim, S. Hong, and J. Yoo, "A multiple RSU collaborative scheduling scheme for data services in vehicular ad hoc networks," *Cluster Comput.*, vol. 20, no. 2, pp. 1167–1178, Jun. 2017.
- [20] G. G. M. Nawaz Ali, P. H. J. Chong, S. K. Samantha, and E. Chan, "Efficient data dissemination in cooperative multi-RSU vehicular ad hoc networks (VANETs)," *J. Syst. Softw.*, vol. 117, pp. 508–527, Jul. 2016.
- [21] X. Xu, C. Yang, M. Bilal, W. Li, and H. Wang, "Computation offloading for energy and delay trade-offs with traffic flow prediction in edge computing-enabled IoV," *IEEE Trans. Intell. Transp. Syst.*, early access, Nov. 23, 2022, doi: [10.1109/TITS.2022.3221975](https://doi.org/10.1109/TITS.2022.3221975).
- [22] 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.
- [23] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, "Collaborate edge and cloud computing with distributed deep learning for smart city Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8099–8110, Sep. 2020.
- [24] J. Guo, M. Bilal, Y. Qiu, C. Qian, X. Xu, and K.-K. R. Choo, "Survey on digital twins for Internet of Vehicles: Fundamentals, challenges, and opportunities," *Digit. Commun. Netw.*, Jun. 2022, doi: [10.1016/j.dcan.2022.05.023](https://doi.org/10.1016/j.dcan.2022.05.023).
- [25] D. Zhu, M. Bilal, and X. Xu, "Edge task migration with 6G-enabled network in box for cybertwin-based Internet of Vehicles," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4893–4901, Jul. 2022.
- [26] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 4, pp. 2131–2165, 4th Quart., 2021.
- [27] M. Afrin, J. Jin, A. Rahman, A. Rahman, J. Wan, and E. Hossain, "Resource allocation and service provisioning in multi-agent cloud robotics: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 842–870, 2nd Quart., 2021.
- [28] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "Multiuser computation offloading and downloading for edge computing with virtualization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4298–4311, Sep. 2019.
- [29] M. Merluzzi, P. D. Lorenzo, S. Barbarossa, and V. Frascolla, "Dynamic computation offloading in multi-access edge computing via ultra-reliable and low-latency communications," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 6, pp. 342–356, 2020.
- [30] Y. Wang et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.
- [31] H. Hu, Q. Wang, R. Q. Hu, and H. Zhu, "Mobility-aware offloading and resource allocation in a MEC-enabled IoT network with energy harvesting," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17541–17556, Dec. 2021.
- [32] Y. C. Cheng and T. G. Robertazzi, "Distributed computation with communication delay (distributed intelligent sensor networks)," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 24, no. 6, pp. 700–712, Nov. 1988.
- [33] X. Song et al., "Everyone-centric heterogeneous multi-server computation offloading in ITS with pervasive AI," *IEEE Netw.*, vol. 37, no. 7, pp. 62–68, Mar./Apr. 2023.
- [34] 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.
- [35] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 3, pp. 881–892, Sep. 2021.
- [36] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 7, pp. 3882–3897, Jul. 2023.
- [37] L. Yao, X. Xu, M. Bilal, and H. Wang, "Dynamic edge computation offloading for Internet of Vehicles with deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 22, 2022, doi: [10.1109/TITS.2022.3178759](https://doi.org/10.1109/TITS.2022.3178759).
- [38] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.

- [39] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.
- [40] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9282–9293, Sep. 2021.
- [41] A. Gao, T. Geng, S. X. Ng, and W. Liang, "A continuous policy learning approach for hybrid offloading in backscatter communication," *IEEE Commun. Lett.*, vol. 25, no. 2, pp. 523–527, Feb. 2021.



Haofei Li received the B.E. degree in communication engineering from Hohai University, Nanjing, China, in 2019. She is currently pursuing the Ph.D. degree with Xidian University, Xi'an, China. Her research interests include wireless communications, mobile edge computing, and satellite terrestrial networks.



Pu Li received the B.Eng. degree from Xi'an Polytechnic University in 2005 and the M.Sc. degree from Northwest A&F University in 2013. Since 2016, he has been the Vice President of ECARX (Hubei) Technology Company Ltd., Hangzhou, China. Now, he is also with the College of Information Science and Electronic Engineering, Zhejiang University, as a Ph.D. candidate. His main research interests include deep learning, the Internet of Things, and edge computing.



Yoong Choon Chang received the B.Eng. degree in electrical and electronic engineering from the Northumbria University at Newcastle, U.K., and the M.Eng.Sc. and Ph.D. degrees from Multimedia University. He is currently an Associate Professor and the Head of the Department of Electrical and Electronic Engineering, Lee Kong Chian Faculty of Engineering and Science, Universiti Tunku Abdul Rahman (UTAR), Malaysia. Over the past ten years, his research work has been funded by the Embassy of France, Kuala Lumpur; the Asia-Pacific Telecommunity; the Malaysian Communications and Multimedia Commission; Telekom Malaysia, Motorola, Malaysia; and the Ministry of Science, Technology and Innovation, Malaysia. He has more than 40 scientific publications in international journals and conferences. His research interests include video signal processing and digital home.



Chen Chen (Senior Member, IEEE) received the B.Eng., M.Sc., and Ph.D. degrees in telecommunication from Xidian University, Xi'an, China, in 2000, 2006, and 2008, respectively. He was a Visiting Professor with the Department of EECS, The University of Tennessee, and the Department of CS, University of California. He is currently a Professor with the School of Telecommunications Engineering, Xidian University. He is also the Director of the Xi'an Key Laboratory of Mobile Edge Computing and Security and the Intelligent Transportation Research

Laboratory, Xidian University. He has authored/coauthored two books and more than 130 scientific papers in international journals and conference proceedings. He has contributed to the development of five copyrighted software systems and invented more than 100 patents. He is a Distinguished Member of the China Computer Federation (CCF) and a Senior Member of the China Institute of Communications (CIC). He serves as the general chair, the PC chair, and the workshop chair or a TPC member for a number of conferences.



Houbing Song (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in August 2012. He is currently a Professor, the Director of the NSF Center for Aviation Big Data Analytics (Planning), the Associate Director for Leadership of the DOT Transportation Cybersecurity Center for Advanced Research and Education (Tier 1 Center), and the Director of the Security and Optimization for Networked Globe Laboratory (SONG Lab), University of Maryland, Baltimore County (UMBC),

Baltimore, MD. He is the author of more than 100 articles and an inventor of two patents (U.S. & WO). His research has been featured by popular news media outlets, including IEEE GlobalSpec's Engineering360, USA Today, U.S. News & World Report, Fox News, Association for Unmanned Vehicle Systems International (AUVSI), *Forbes*, WFTV, New Atlas, The Washington Times, Battle Space, and Defense Daily. His research interests include cyber-physical systems, cybersecurity and privacy, the Internet of Things, edge computing, AI/machine learning, big data analytics, unmanned aircraft systems, connected vehicle, smart and connected health, and wireless communications and networking. He is a Distinguished Member of ACM. He has been a Highly Cited Researcher identified by Clarivate™ (2021, 2022). He received Research.com Rising Star of Science Award in 2022, 2021 Harry Rowe Mimno Award bestowed by IEEE Aerospace and Electronic Systems Society, and ten+ Best Paper Awards from major international conferences, including IEEE CPSCOM-2019, IEEE ICII 2019, IEEE/AIAA ICNS 2019, IEEE CBDCOM 2020, WASA 2020, AIAA/IEEE DASC 2021, IEEE GLOBECOM 2021 and IEEE INFOCOM 2022. He serves as an Associate Editor for IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE (TAI) (2023-present), IEEE INTERNET OF THINGS JOURNAL (2020-present), IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (2021-present), and IEEE JOURNAL ON MINIATURIZATION FOR AIR AND SPACE SYSTEMS (J-MASS) (2020-present). He was an Associate Technical Editor for IEEE *Communications Magazine* (2017-2020). He is an Editor of eight books, including *Aviation Cybersecurity: Foundations, Principles, and Applications* (Scitech Publishing, 2022); *Smart Transportation: AI Enabled Mobility and Autonomous Driving* (CRC Press, 2021); *Big Data Analytics for Cyber-Physical Systems: Machine Learning for the Internet of Things* (Elsevier, 2019); *Smart Cities: Foundations, Principles, and Applications* (Hoboken, NJ: Wiley, 2017); *Security and Privacy in Cyber-Physical Systems: Foundations, Principles, and Applications* (Chichester, U.K.: Wiley-IEEE Press, 2017); *Cyber-Physical Systems: Foundations, Principles and Applications* (Boston, MA: Academic Press, 2016); and *Industrial Internet of Things: Cybermanufacturing Systems* (Cham, Switzerland: Springer, 2016). He is an ACM Distinguished Speaker.



Hanguan Shan (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2004, and the Ph.D. degree in electrical engineering from Fudan University, Shanghai, China, in 2009. From 2009 to 2010, he was a Post-Doctoral Research Fellow with the University of Waterloo, Waterloo, ON, Canada. Since 2011, he has been with the College of Information Science and Electronic Engineering, Zhejiang University, where he is currently an Associate Professor. He is also with the Zhejiang

Provincial Key Laboratory of Information Processing and Communication Networks and SUTD-ZJU IDEA, Zhejiang University. His current research interests include modeling and optimization for wireless networks, machine learning for wireless networks, and the quality-of-service provisioning in wireless networks. He has served as a technical program committee member for various international conferences. He was a recipient of the Best Industry Paper Award from IEEE WCNC 2011, Quintana Roo, Mexico. He was an Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.