sensors

MDPI

Article

Multi-User Computation Offloading and Resource Allocation Algorithm in a Vehicular Edge Network

Xiangyan Liu [1,*], Jianhong Zheng [1], Meng Zhang [2], Yang Li [3], Rui Wang [1,4] and Yun He [1]

[1] School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; zhengjh@cqupt.edu.cn (J.Z.); d190101012@stu.cqupt.edu.cn (R.W.); heyun@cqupt.edu.cn (Y.H.)
[2] State Key Laboratory of Block Chain and Data Security, Zhejiang University, Hangzhou 310058, China; zhangmengyang@zju.edu.cn
[3] Cyberspace Security Key Laboratory of Sichuan Province, Chengdu 610043, China; liyang.yalee@gmail.com
[4] Department of Electronic Communication Engineering, Yuxi Normal University, Yuxi 653100, China
* Correspondence: xiangyan.leo@gmail.com

Abstract: In Vehicular Edge Computing Network (VECN) scenarios, the mobility of vehicles causes the uncertainty of channel state information, which makes it difficult to guarantee the Quality of Service (QoS) in the process of computation offloading and the resource allocation of a Vehicular Edge Computing Server (VECS). A multi-user computation offloading and resource allocation optimization model and a computation offloading and resource allocation algorithm based on the Deep Deterministic Policy Gradient (DDPG) are proposed to address this problem. Firstly, the problem is modeled as a Mixed Integer Nonlinear Programming (MINLP) problem according to the optimization objective of minimizing the total system delay. Then, in response to the large state space and the coexistence of discrete and continuous variables in the action space, a reinforcement learning algorithm based on DDPG is proposed. Finally, the proposed method is used to solve the problem and compared with the other three benchmark schemes. Compared with the baseline algorithms, the proposed scheme can effectively select the task offloading mode and reasonably allocate VECS computing resources, ensure the QoS of task execution, and have a certain stability and scalability. Simulation results show that the total completion time of the proposed scheme can be reduced by 24–29% compared with the existing state-of-the-art techniques.

Keywords: Vehicular Edge Computing Network (VECN); computation offloading; resource allocation; deep reinforcement learning

Citation: Liu, X.; Zheng, J.; Zhang, M.; Li, Y.; Wang, R.; He, Y. Multi-User Computation Offloading and Resource Allocation Algorithm in a Vehicular Edge Network. *Sensors* **2024**, *24*, 2205. https://doi.org/10.3390/s24072205

Academic Editors: Sabur Baidya and Yu-Jen Ku

Received: 13 March 2024
Revised: 28 March 2024
Accepted: 28 March 2024
Published: 29 March 2024

1. Introduction

The emergence of various intelligent on-vehicle applications in the Internet of Vehicles (IoV), such as autonomous driving, online games, augmented reality, intelligent guidance of traffic behavior, and voice-based dynamic human-vehicle interaction, makes resource-constrained vehicles face significant challenges in supporting these intelligent services [1–5]. Vehicular Edge Computing Networks (VECNs) extend computation capability to the edge of the wireless network by providing additional computation resources close to mobile vehicles, which can ease the burden on vehicles. Moreover, VECNs make it possible to take full advantage of ubiquitous computation resources in the system. Computing offloading is used to realize computation-intensive and delay-sensitive applications processed in the ubiquitous computation resources, which frees Task Vehicles (TaVs) from complex tasks, helps to reduce service delay, effectively alleviates the problem of limited computation capability of TaVs, and provides better Quality of Service (QoS) for vehicle users [6,7]. However, the mobility of vehicles and the diversity of edge computing nodes and vehicle offloading modes bring challenges to task offloading services [8].

Computation tasks can be offloaded to Service Vehicles (SeVs) via vehicle-to-vehicle (V2V) links to use computation resources in the system. It can also be offloaded to a Vehicular Edge Computing Server (VECS) connected to a Road Side Unit (RSU) or a Base Station (BS) via a vehicle-to-infrastructure (V2I) link. Thus, a vehicle offloading mode mainly includes Local execution mode (Loc), Local + SeV execution mode (Loc + SeV), Local + VECS execution mode (Loc + Edge), and Local + SeV + VECS execution mode (Loc + Sev + Edge).

The remainder of this paper is organized as follows: In Section 2, related works are discussed. The system model and problem formulation are formulated in Section 3. The multi-user computing offloading and resource allocation method based on Deep Deterministic Policy Gradient (DDPG) is given in Section 4. We conduct the simulation results and analysis of the proposed algorithm in Section 5. The conclusion and future work are given in Section 6.

2. Related Work

V2V links are used to offload tasks to SeVs [9,10]. Platooning vehicles are considered in work [9], where platoon members can only communicate with the platoon leader and use the resources via V2V communication links. More factors are considered in [10] when SeVs are selected, such as the caching factor, energy factor, and location factor of vehicles to offload non-real-time traffic to V2V networks.

V2I links are used to offload tasks to VECS in the way of partial offloading [11–16]; namely, parts of tasks are processed locally, and others are offloaded to VECS. The environmental settings and optimization objectives distinguish these works. TaVs should pay for the services provided by VECS in [14]. Analytical offloading schemes for some special VECNs are proposed in [15], including the cases of one TaV with one VECS, one TaV with two VECSs, and two TaVs with one VECS. The mobility of vehicles is not considered in some works [11–15]. However, the mobility of TaVs while TaVs move in a random direction at a specific rate is considered in [16].

Both V2V and V2I links are leveraged to offload tasks to SeVs and VECSs, which can fully use the system's ubiquitous computation resources [17–20]. VECSs deployed at RSUs are regarded as fixed VECSs, while the mobile vehicles are regarded as mobile VECSs. The two types of VECSs cooperate to provide additional computing resources for TaVs [17]. Except for mobile vehicles, parked vehicles also can be treated as SeVs [18,19]. Based on this, the work in [18] proposes a dynamic pricing strategy to maximize the revenue of the computing service provider. In contrast, the work in [19] organizes RSUs and roadside parked vehicles into parking clusters to make up for the computation resource bottleneck caused by insufficient infrastructure construction. Furthermore, considering the importance of the matching between TaVs and corresponding processing terminals, the authors in [20] propose a four-lane dual carriageway model to simulate the urban traffic environment and use the Kuhn-Munkras algorithm to realize the matching of TaVs and service providers.

Some of the literature adopts traditional methods for computing offloading [21–25]. For instance, a queue-based improved multi-objective particle swarm optimization algorithm to solve the problem of multi-dependent task offloading in multi-access edge computing is proposed in [21]. The author in [22] divides and conquers the goal into two phases: VECS selection and offloading decision. For the VECSs selection phases, TaVs are grouped into one BS, considering their physical distance and workload. After VECS selection, the original problem is divided into parallel multi-user-to-one-server offloading decision subproblems and a distributed offloading strategy based on a binary-coded genetic algorithm is used to obtain an adaptive offloading decision. Considering the heterogeneity of communication modes and computing capabilities of network computing points in ubiquitous networks, a distributed multi-hop computing task offloading framework based on an improved genetic algorithm is proposed in [23] so that tasks could be recursively offloaded among computing points in the ubiquitous network. Benders decomposition technology is used to realize task offloading in [24]. The author in [25] considers the quasi-

static channel model during task offloading, wherein the channel remains constant during the offloading period but may change during different offloading periods; a two-stage Stalberg game is then used to solve the optimization objective.

Computing offloading via V2V communication and V2I communication can make full use of the ubiquitous computing resources of the system and improve the performance of mobile edge computing [26]. However, due to the mobility of vehicles and dynamic wireless channel conditions, the formulation of computing offloading strategies has high-dimensional and time-varying characteristics. Most of the optimization-based computation offloading schemes lack the ability to adapt to dynamic environments. Fortunately, deep reinforcement learning in artificial intelligence can solve such high-dimensional time-varying feature problems with limited and inaccurate information [27,28]. Deep reinforcement learning algorithms for task offloading management are used in some of the literature [29–32]. Based on this, the computation tasks of TaVs are offloaded to edge vehicles and cloud networks to acquire more computation resources [29]. The problem of computation offloading and resource allocation for tasks offloading to VECS through V2I links is addressed in [30]. Tasks are offloaded hierarchically in [31]. A vehicle may have multiple tasks, and the author in [32] considers offloading these tasks to multiple vehicles, nearby pedestrians that use mobile phones or tablets, other vehicles that can provide computing services, and VECSs. The characteristics, pros, and cons revealed in the recent research are provided in Table 1. For simplicity, M^0, M^1, M^2, M^3 are used to denote the four modes of task execution, where the M^0, M^1, M^2, and M^3 modes represent Loc mode, Loc + Sev mode, Loc + Edge mode, and Loc + Sev + Edge mode, respectively.

As seen in Table 1, this literature is based on the three execution modes, M^1, M^2, and M^3, which all involve the local execution mode M^0. Most of the research in the M^1 and M^2 modes adopts the partial offloading mode [9,10,14–16,25,29], and most of the research in the M^3 mode adopts the 0–1 offloading mode [19,20,23,31,32]. Furthermore, we have studied the use of V2V and V2I links to extend the system's computing resources [33,34]. However, the computing offloading and the resource allocation of VECSs in dynamic environments have not been fully considered. Based on this, this paper comprehensively considers TaV's preference for the Loc mode, Loc + Sev mode, Loc + Edge mode, and Loc + Sev + Edge mode in a dynamic environment and the impact of task offloading and resource allocation on offloading delay. The computation offloading and resource allocation problem is modeled as a Mixed Integer Nonlinear Programming (MINLP) problem. Then, considering the advantage of DDPG for environmental dynamics, a method based on DDPG is proposed; namely, the multi-user computation offloading and resource allocation scheme (MCORA). The main contributions of this paper are summarized as follows:

1. To solve the problem of task execution time being difficult to acquire because of the mobility of vehicles and the dynamic of channel state information, a computing offloading and resource allocation optimization scheme is proposed for multiple TaV, which adopts the best mode from four execution modes; namely, Loc mode, Loc + Sev mode, Loc + Edge mode, and Loc + Sev + Edge mode. Leveraging these four modes, we can analyze the complex task execution process more simply and acquire the task execution time.
2. To minimize the total task execution time by choosing the adaptive mode and allocating the computation resources of a VECS, the optimization objective is established according to the delay of task execution, and then the computing offloading mode chosen with the resource allocation problem is transformed into a MINLP problem. This can be described as a Markon Decision Processes (MDP), and the MCORA algorithm is proposed to solve it.
3. To solve the non-convexity and the discontinuity of the offloading mode selection and the resource allocation of this problem, DDPG is considered because it can deal with continuous and discontinuous actions, so the MCORA scheme is based on DDPG.
4. To verify the effectiveness of our scheme, three baseline schemes are compared with our scheme; namely, Offloading in Loc + Sev mode (OLSM) [10], Offloading in

Loc + Edge mode (OLEM) [14], and Offloading in Random Mode (ORM). Simulation results show that compared with the existing schemes, the proposed scheme can significantly reduce the delay of computation offloading and resource allocation.

Table 1. Comparison with the latest related studies.

Ref.	Year	Mode	Mobility	Method	Advantages	Shortcomings
[9]	2023	M^1	✓	DDPG	Both computation offloading and power allocation are considered.	Only the resources of the platoon leader are shared.
[10]	2022	M^1	✓	Q-learning	Jointly consider the cache factor, energy factor, and position factor.	Non-real-time traffic is offloaded into the V2V network.
[14]	2022	M^2	✗	Deep Q-network	Considering the limited capability of calculating access points and users' budgets.	The mobility of vehicles is not considered.
[15]	2022	M^2	✗	Analytical offloading scheme	Multiple computational access points can help vehicular users compute tasks.	Only several scenarios are considered.
[16]	2023	M^2	✓	DDPG	The trade-off optimization of delay and energy consumption is considered.	Action encoding is used to replace actions in continuous action space.
[19]	2021	M^3	✓	Heuristics algorithm	Parked vehicles and TaVs driving trajectory prediction are considered.	Each uploaded task is assumed to be performed by only one edge server (0–1 offloading).
[20]	2021	M^3	✓	Greedy matching	Both the resources of the RSU and nearby vehicles are considered.	0–1 offloading is adopted.
[23]	2023	M^3	✗	Genetic algorithm	Dispersed computing is considered, including each mobile device, edge, and cloud server.	The solution space dimension is significant, and 0–1 offloading is considered.
[25]	2023	M^2	✗	Stackelberg game-based scheme	Reasonable prices are designed for computing resources.	Single-server is considered.
[29]	2023	M^2	✓	Primal-dual DDPG	A multi-tier computation offloading network structure is considered.	The resources of nearby vehicles are not used.
[30]	2023	M^2	✓	TD3	Considering real-time decision-making and prediction.	0–1 offloading is considered.
[31]	2021	M^3	✓	DDPG-based	The prioritized experience replay and the stochastic weight averaging mechanisms are considered.	0–1 offloading is considered.
[32]	2022	M^3	✓	SAC	Both the priority and the size of the tasks are considered.	One TaV and 0–1 offloading is considered.
Proposed		M^3	✓	DDPG	Several execution modes are considered	The energy consumption is not considered.

3. System Model and Problem Formulation

3.1. System Model

Four communication modes can be adopted; namely, Loc mode, Loc + SeV mode, Loc + Edge mode, and Loc + SeV + Edge mode. The architecture of the VECNs is shown in Figure 1.

Four modes are depicted in Figure 1. It was Loc mode when TaV executed all tasks locally, Loc + SeV mode when TaV executed some tasks locally and offloaded the others to SeV, Loc + Edge mode when TaV executed some tasks locally and offloaded the others to VECS, and Loc + SeV + Edge mode when tasks were executed in the three terminals.

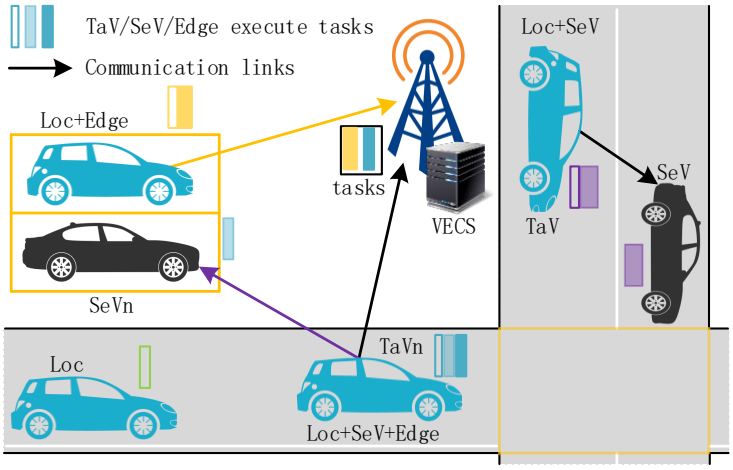


Figure 1. System model.

Set $\mathcal{N} = \{V_1, V_2, \cdots, V_n \cdots, V_N\}$ represents N TaVs randomly distributed on an urban traffic environment, and all TaVs are connected to a BS located in the center. A VECS is deployed at the BS. TaVn has a task $I_n = \{D_n, App_n\}$ to be processed, where D_n (in bits) represents task sizes, and App_n (in CPU cycles/bit) represents the processing density of TaVn. Based on this, $C_n = D_n App_n$ represents the CPU resources required to complete this task. These tasks are all arbitrarily divisible, and their maximum processing delay is $t_{\max}$, which is also the processing period of these tasks. We divide the time into T equal time slots $\mathcal{T} = \{1, 2, \cdots, t, \cdots, T\}$, and the time in each time slot is $\tau = t_{\max}/T$. At any time slot, the task can be executed in any execution mode.

3.2. Communication Model

Based on the available literature [16], this paper further considers the relative position and the task offloading between TaVs and SeVs. Then, the communication model is established as follows: TaVn and SeVn are moving at a certain speed v_n^{tav} and v_n^{sev}, respectively, assuming that the distance between them changes in a uniformly distributed range $d_{\text{tav}}^{\text{sev}}$. Moreover, V2I links and V2V links are all adopting orthogonal frequency division multiplexing technology [20]. Channel power gains of the V2V link from TaVn to SeVn and the V2I link from TaVn to VECS are denoted as $g_{n,t}^{\text{v2v/v2i}} = \alpha_{n,t}^{\text{v2v/v2i}} \cdot h_{n,t}^{\text{v2v/v2i}}$, where $\alpha_{n,t}^{\text{v2v/v2i}}$ and $h_{n,t}^{\text{v2v/v2i}}$ represent the large-scale fading and small-scale fading of the V2V links and V2I links, respectively. The small-scale fading is exponentially distributed. $\alpha_{n,t}^{\text{v2v/v2i}}$ includes path loss and shadow fading. The path losses of the V2V links and V2I links are calculated as follows [35]:

$$PL_{\text{v2v}}^{t}(d_{\text{tav}}^{\text{sev}}) = \begin{cases} 22.7\log_{10}3 + 41 + 20\log_{10}(\dfrac{freq}{5}), d_{\text{tav}}^{\text{sev}} \le 3 \\ 22.7\log_{10}(d_{\text{tav}}^{\text{sev}}) + 41 + 20\log_{10}(\dfrac{freq}{5}), d_{\text{tav}}^{\text{sev}} \le \dfrac{4freq(H^{\text{veh}}-1)^2}{c} \\ 40\log_{10}(d_{\text{tav}}^{\text{sev}}) + 9.45 - 17.3\log_{10}((H^{\text{veh}}-1)^2) + 2.7\log10(\dfrac{freq}{5}), \text{else} \end{cases}, \tag{1}$$

and

$$\text{PL}_{\text{v2i}}^{t}(d_n^{\text{edg}}) = 128.1 + 37.6\log_{10}(\frac{d_n^{\text{edg}}}{1000}), \tag{2}$$

where H^{veh} represents the antenna height of the vehicle, $freq$ denotes the carrier frequency, and d_n^{edg} denotes the distance between TaVn and VECS. The updated formula of shadow fading is as follows [35]:

$$S_t^{\text{v2v/v2i}} = S_{t-1}^{\text{v2v/v2i}} \cdot e^{-\frac{(\Delta_{\text{tav}}^t + \Delta_{\text{sev}}^t)}{10}} + S^{\text{v2v/v2i}} \cdot \sqrt{1 - e^{-\frac{2(\Delta_{\text{tav}}^t + \Delta_{\text{sev}}^t)}{10}}}, \tag{3}$$

where Δ_{tav}^{t} and Δ_{sev}^{t} denote the distance traveled by TaV and SeV in the tth time slot. Then, the data transmission rate of the V2V links and V2I links can be expressed as follows:

$$R_{n,t}^{\text{v2v/v2i}} = B\log_2(1+\gamma_{n,t}^{\text{v2v/v2i}}), \tag{4}$$

where $\gamma_{n,t}^{\text{v2v/v2i}} = P_n g_{n,t}^{\text{v2v/v2i}}/\delta^2$. P_n represents the transmission power of TaVn.

3.3. Mode Selection and Task Offloading Delay Computing in Edge Networks

At time slot t, if TaVn chooses to execute tasks locally, the number of bits that can be processed can be expressed as follows:

$$U_{n,t}^{\text{loc}} = \frac{\tau \cdot f_n^{\text{loc}}}{App_n}, \tag{5}$$

where f_n^{loc} denotes the processing capacity of TaVn. If TaVn chooses to execute tasks at SeVn, the task needs to be offloaded to SeVn at first, and the task in time slot t includes not only the time to transmit to SeVn but also the time to execute the task at SeVn. Let $U_{n,t}^{\text{sev}}$ represent the number of bits that can be completed in time slot t. Then, according to the transmission time sev_{tr}=$U_{n,t}^{\text{sev}}/R_n^{\text{v2v}}$, the computation time $sev_{\text{e}} = U_{n,t}^{\text{sev}} \cdot App_n/f_n^{\text{sev}}$, and the equality $sev_{\text{tr}} + sev_{\text{e}} = \tau$, $U_{n,t}^{\text{sev}}$ can be obtained in time slot t as follows:

$$U_{n,t}^{\text{sev}} = \frac{\tau \cdot R_n^{\text{v2v}} \cdot f_n^{\text{sev}}}{f_n^{\text{sev}} + App_n \cdot R_n^{\text{v2v}}}. \tag{6}$$

Similarly, when tasks of a TaV are selected to be executed at VECS through the V2I link, according to the transmission time $edg_{\text{tr}} = U_{n,t}^{\text{edg}}/R_n^{\text{v2I}}$, the computing time $edg_{\text{e}} = U_{n,t}^{\text{edg}} \cdot App_n/(\rho_n^t \cdot F^{\text{edg}})$, and $edg_{\text{tr}} + edg_{\text{e}} = \tau$, the number of bits that can be processed in time slot t is obtained as follows:

$$U_{n,t}^{\text{edg}} = \frac{\tau \cdot R_n^{\text{v2I}} \cdot \rho_n^t \cdot F^{\text{edg}}}{\rho_n^t \cdot F^{\text{edg}} + App_n \cdot R_n^{\text{v2I}}}. \tag{7}$$

$M_n \in \{M^0, M^1, M^2, M^3\}$ is used to denote the four modes of task execution. When TaVn chooses M^0 mode to execute tasks, the number of bits that can be processed in time slot t is equal to the number of bits that can be executed locally:

$$U_{\text{total},n}^{t} = U_{n,t}^{\text{loc}} = \frac{\tau \cdot f_n^{\text{loc}}}{App_n}. \tag{8}$$

When TaVn chooses mode M^1 to execute tasks, the number of bits that can be completed in time slot t is equal to the sum of the number of bits that can be executed locally and in SeVn:

$$U_{\text{total},n}^{t} = U_{n,t}^{\text{loc}} + U_{n,t}^{\text{sev}} = \frac{\tau \cdot f_n^{\text{loc}}}{App_n} + \frac{\tau \cdot R_n^{\text{v2v}} \cdot f_n^{\text{sev}}}{f_n^{\text{sev}} + App_n \cdot R_n^{\text{v2v}}}. \tag{9}$$

Similarly, in M^2 mode, the number of bits that can be processed is the sum of the tasks that can be processed locally and by VECS:

$$U_{n,t}^{\text{total}} = U_{n,t}^{\text{loc}} + U_{n,t}^{\text{edg}} = \frac{\tau \cdot f_n^{\text{loc}}}{App_n} + \frac{\tau \cdot R_n^{\text{v2I}} \cdot \rho_n^t \cdot F^{\text{edg}}}{\rho_n^t \cdot F^{edg} + App_n \cdot R_n^{\text{v2I}}}. \tag{10}$$

In M^3 mode, the number of bits that can be processed is the sum of the bits of the three terminals:

$$U_{n,t}^{\text{total}} = U_{n,t}^{\text{loc}} + U_{n,t}^{\text{sev}} + U_{n,t}^{\text{edg}} = \frac{\tau \cdot f_n^{\text{loc}}}{App_n} + \frac{\tau \cdot R_n^{\text{v2v}} \cdot f_n^{\text{sev}}}{f_n^{\text{sev}} + App_n \cdot R_n^{\text{v2v}}} + \frac{\tau \cdot R_n^{\text{v2I}} \cdot \rho_n^t \cdot F^{\text{edg}}}{\rho_n^t \cdot F^{edg} + App_n \cdot R_n^{\text{v2I}}}. \tag{11}$$

Since a TaV must choose an execution mode to execute its tasks as long as it is not completed, the completion time t_n^{total} of TaVn's tasks can be expressed as the minimum number of time slots spent for the cumulative maximum execution bits of its tasks:

$$t_n^{\text{total}} = t \cdot \tau,\ t = \min_t \max \sum_{t=1}^{T} U_{n,t}^{\text{total}}. \tag{12}$$

3.4. Problem Formulation

The execution time of all completed tasks in time slot t can be expressed as follows:

$$P_t(M(t), \rho(t)) = \sum_{t=1}^{T} t_n^{\text{total}} \tag{13}$$

where $M(t) = \{M_n(t) | n \in \mathcal{N}\}$ represents the mode chosen by TaVn in time slot t. $\rho(t) = \{\rho_n(t) | n \in \mathcal{N}\}$ represents the proportion of computation resources allocated to TaVn by VECS in time slot t. Our goal is to minimize the execution time of all tasks:

$$\begin{aligned} \mathcal{P} \min_{M(t),\rho(t)} P_t(M(t), \rho(t)) &= \sum_{t=1}^{T} \min_{M(t),\rho(t)} t_n^{\text{total}} \\ \text{s.t. C1} &: M_n(t) \in \{M^0, M^1, M^2, M^3\}, n \in \mathcal{N} \\ \text{C2} &: 0 \leq \sum_{n \in \mathcal{N}} \rho_n(t) \leq 1 \\ \text{C3} &: 0 \leq \rho_n(t) \leq 1, n \in \mathcal{N} \\ \text{C4} &: t_{total,n} \leq t_{\max}, n \in \mathcal{N} \end{aligned}$$

where C1 denotes the mode chosen by TaVn, which is one of the four modes, and C2 indicates that the VECS resources allocated to all TaVs do not exceed the total computation resources of the VECS. C3 limits the proportion of the VECS's computation resources allocated to each TaV. C4 means that the execution time does not exceed the maximum delay of TaVn.

4. Multi-User Computing Offloading and Resource Allocation Method Based on DDPG

In order to solve the formulated objective, we propose a DDPG-based computing offloading and resource allocation scheme. Considering the actual VECNs environment, the objective $\mathcal{P}$ is described as MDP. Then, the DDPG algorithm is designed to solve it, and the key issues are the normalization of states and actions and the design of the reward function. Finally, the task offloading algorithm based on DDPG was used to solve the optimization objective.

4.1. Markov Decision Processes for Mode Selection and Computing Offloading

MDP is a mathematical framework for describing sequential decision-making problems with stochastic properties [36]. A Markov model can be represented as a quadruple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. The elements inside represent the set of states, the set of actions, the states' transition probability, and the immediate reward function for performing the actions.

(1) State space. System state $s_t \in \mathcal{S}$ can be expressed as follows:

$$s_t = (R(t), O(t), Ti), \tag{14}$$

where (a) $R(t) = \{[R_1^{\text{v2v}}(t)..., R_n^{\text{v2v}}(t), ..., R_{\text{N}}^{\text{v2v}}(t)], [R_1^{\text{v2i}}(t)..., R_n^{\text{v2i}}(t), ..., R_{\text{N}}^{\text{v2i}}(t)]\}$ denotes the data transmission rate of V2V/V2I links at time t; (b) $O(t) = \{[O_1(t)..., O_n(t), ..., O_{\text{N}}(t)]\}$

represents the proportion of TaVn's tasks remaining to be processed; (c) $Ti = \{Ti(t)\}$ denotes the remaining processing time.

(2) Action space. $a_t \in \mathcal{A}$ can be expressed as follows:

$$a_t = (M(t), \rho(t)), \tag{15}$$

where (a) $M(t) = \{[M_1^m(t), ..., M_N^m(t)]\}$ denotes the mode selected by each TaVs in time slot t; (b) $\rho(t) = \{[\rho_1(t), ..., \rho_N(t)]\}$ denotes the proportion of computation resources allocated to TaVn by VECS at time slot t.

(3) Reward function.

$$R_t^{\text{im}}(s_t, a_t) = \begin{cases} 1 \text{ if } \pi(t) = N, \\ 1 - (P_t(M(t), \rho(t))) \text{ else}, \end{cases} \tag{16}$$

where $\pi(t)$ represents the number of tasks completed in time slot t. If all tasks are completed, the reward is assigned to 1 immediately; otherwise, the average remaining available time is assigned to the reward function.

4.2. DDPG-Driven Computation Resource Offloading and Resource Allocation Strategies

The DDPG-based deep reinforcement learning algorithm is used to solve the joint computing offloading and resource allocation problem. As shown in Figure 2, the algorithm includes three modules: main network, target network, and experience replay memory. The policy of the main network is to produce action a_t based on current state s_t. The main network includes two parts, the main actor depth neural network (DNN) $\pi(s_t|\theta_\pi)$ and the main critic DNN $Q(s_t, a_t|\theta_Q)$. The target network, aiming to train the network in the target, has the same structure as the main network. The parameters can be expressed as $\pi'(s_t|\theta'_\pi)$ and $Q'(s_t, a_t|\theta'_Q)$. The experience replay memory is used to store the resulting experience tuples.

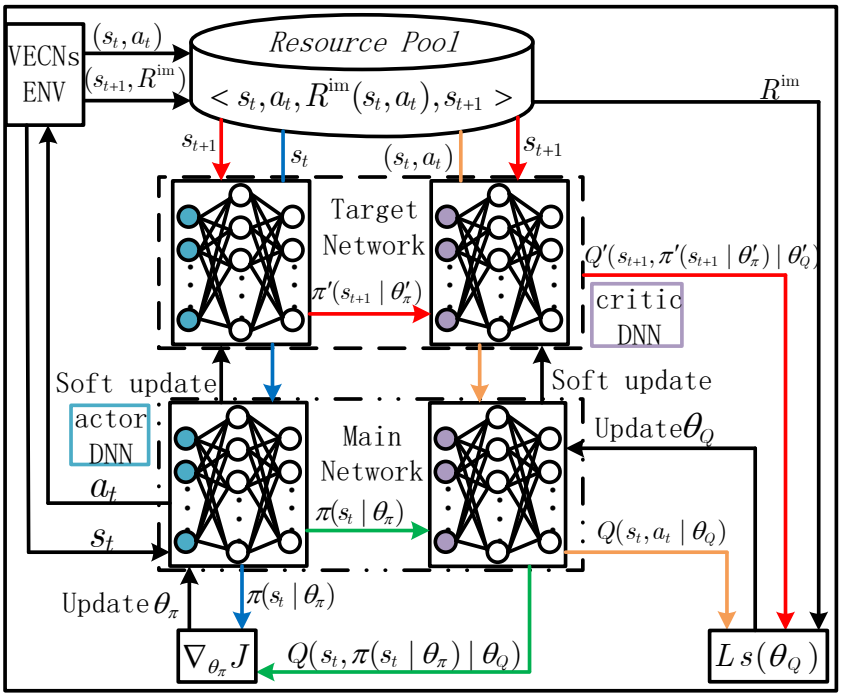


Figure 2. DDPG structure and its update process.

(1) Main actor DNN training. The explored policy can be defined as a function with parameter θ_π, which maps the current state to an action $\hat{a}_t = \pi(s_t|\theta_\pi)$, where $\hat{a}_t$ is obtained by mapping, and $\pi(s_t|\theta_\pi)$ is the model selection and computation resource allocation

policy obtained by the exploration of actor DNN. The added noise n_t follows Gaussian distribution $n_t \sim (\mu_t, \sigma_t^2)$. Then, the refactoring action can be expressed as follows:

$$a_t = \text{clip}(\pi(s_t|\theta_\pi) + n_t, a_{\text{low}}, a_{\text{high}}), \tag{17}$$

where the clip function limits the range of action values to a_{low} and a_{high}, and the main actor DNN uses sampled policy gradients to update the network parameters:

$$\nabla_{\theta_\pi} J \approx [\nabla_a Q(s_t, a_t|\theta_Q) \nabla_{\theta_\pi} \pi(s_t|\theta_\pi)], \tag{18}$$

where $Q(s_t, a_t|\theta_Q)$ is an action-value function. At each step of the training process, θ_π is updated by a batch of experience $< s_t, a_t, R_t^{\text{im}}, s_{t+1} >$:

$$\theta_\pi = \theta_\pi - \frac{\alpha_\pi}{V} \sum_{t=1}^{V} [\nabla_a Q(s_t, a_t|\theta_Q) \nabla_{\theta_\pi} \pi(s_t|\theta_\pi)], \tag{19}$$

where α_π represents the learning rate of the main actor DNN.

(2) Main critic DNN training. The main critic DNN evaluates the performance of the selected action based on the action-value function. The action-value function is computed based on the Bellman optimality equation, which can be expressed as follows:

$$Q(s_t, a_t|\theta_Q) = [R_t^{\text{im}}(s_t, a_t) + \varepsilon Q(s_{t+1}, \pi(s_{t+1})|\theta_Q)], \tag{20}$$

where the main critic DNN considering the current state s_t and the next state s_{t+1} is used to calculate each state-action value $Q(s_t, a_t|\theta_Q)$. The main critic DNN updates the network parameters θ_Q by minimizing the loss function $Ls(\theta_Q)$:

$$Ls(\theta_Q) = \left[(y_t - Q(s_t, a_t|\theta_Q))^2 \right], \tag{21}$$

where y_t is the target value, which can be expressed as follows:

$$y_t = R_t^{\text{im}}(s_t, a_t) + \varepsilon Q'(s_{t+1}, \pi'(s_{t+1}|\theta'_\pi)|\theta'_Q), \tag{22}$$

$Q'(s_{t+1}, \pi'(s_{t+1}|\theta'_\pi)|\theta'_Q)$ is obtained by the target network which is the network with parameters θ'_π and θ'_Q. The gradient computation of $Ls(\theta_Q)$ is expressed as follows:

$$\nabla_{\theta_Q} \text{Ls} = \left[2(y_t - Q(s_t, a_t|\theta_Q)) \nabla_{\theta_Q} Q(s_t, a_t) \right]. \tag{23}$$

In each training step, θ_Q is updated by a batch of experience $< s_t, a_t, R_t^{\text{im}}, s_{t+1} >$ as follows:

$$\theta_Q = \theta_Q - \frac{\alpha_Q}{\text{V}} \sum_{t=1}^{\text{V}} [2(y_t - Q(s_t, a_t|\theta_Q)) \nabla_{\theta_Q} Q(s_t, a_t)], \tag{24}$$

where α_Q represents the learning rate of the main actor DNN.

(3) Target network training. The target network can be regarded as an older main network version with different parameters θ'_π and θ'_Q. In each iteration, the parameters θ'_π and θ'_Q are updated according to (25):

$$\begin{array}{l} \theta'_\pi = \omega\theta_\pi + (1-\omega)\theta'_\pi \\ \theta'_Q = \omega\theta_Q + (1-\omega)\theta'_Q \end{array}, \tag{25}$$

where $\omega \in [0, 1]$.

The computation offloading and resource allocation algorithm based on DDPG is shown in Algorithm 1. Firstly, parameter θ_π is used to initialize the computation offloading and resource allocation strategy $\pi(s|\theta_\pi)$ of main actor DNN, and parameter θ_Q is used to initialize the action-value function of critic DNN $Q(s_t, a_t|\theta_Q)$. The parameters θ'_π and θ'_Q of

the target network are initialized at the same time. Then, the main actor DNN generated action a_t according to the current policy $\pi(s|\theta_\pi)$ and state s_t. Based on the observed reward $R_t^{\text{im}}(s_t, a_t)$ and the next state s_{t+1}, the tuple $< s_t, a_t, R^{\text{im}}(s_t, a_t), s_{t+1} >$ is constructed and stored in an experience replay memory. The memory is stored in a first-in-first-out manner, and if the memory is about to overflow, the oldest experience will be deleted and updated to the latest experience. Based on the mini-batch technique, the algorithm updates the DNN network of the main critic DNN by minimizing the function $Ls(\theta_Q)$ and updates the main actor DNN by using the sampled policy gradient. After a period of training, the parameters of the target network are updated according to (25).

Algorithm 1 Multi-user computation offloading and resource allocation algorithm

Initialization:
1. Leverage parameters θ_π and θ_Q to initialize $\pi(s|\theta_\pi)$ and $Q(s, a|\theta_Q)$;
2. Leverage parameters ${\theta'}_\pi \leftarrow \theta_\pi$ and ${\theta'}_Q \leftarrow \theta_Q$ to initialize $\pi'(s|{\theta'}_\pi)$ $Q'(s, a|{\theta'}_Q)$;
3. Initialize experience replay memory;

for each episode **do:**
  Initialize system environment setup;
  for each time slot t **do:**
    Acquire action a_t according to (17);
    Obtain immediate reward $R^{\text{im}}(s_t, a_t)$ with (16) and accumulated reward, update next state s_{t+1};
    if experience replay memory is not full **do:**
      Store tuple $< s_t, a_t, R^{\text{im}}(s_t, a_t), s_{t+1} >$ into experience replay memory;
    else:
      A batch tuple V is randomly drawn from the experience replay memory;
      The target value y_t is calculated based on (22);
      Parameters θ_Q are updated by minimizing the loss function based on (21);
      Parameters θ_π are updated according to the sampled policy gradient based on (18);
      Parameters θ'_π and θ'_Q are updated based on (25);
    End if
  End for
End for

5. Simulation Results and Analysis

5.1. Simulation Environment

The experiment was carried out on the Windows10 operating system with the processor Intel Core i7-6700 CPU @3.40GHz (Santa Clara, CA, USA), while the software used was Python3.7.9 and TensorFlow1.15.0. The urban IoV simulator, including the vehicle, lane, and wireless communication network model defined in Appendix A of 3GPP TR 36.885 [35] is adopted. The main simulation parameters are shown in Table 2. The actor and critic networks of the DDPG agent both consist of three fully connected hidden layers consisting of 64, 16, and 4 neurons, respectively. ReLU is used as the activation function, and Adam is used as the optimizer to train and update the weights of the neural network iteratively. The algorithm was trained for a total of 2000 episodes, and the exploration probability was annealed by linear annealing algorithm from 1 at the beginning to 0.01 at 1600 episodes, and then remained unchanged in the following training steps [37]. Unless otherwise specified, the simulation parameters in this chapter are executed according to Table 2, and the results are the average values of the last 100 episodes.

Table 2. Simulator parameters.

Parameter	Value
Wireless bandwidth of the links (B)	2 MHz
The number of TaVs (N)	20
Transmit power of TaVn (P_n)	23 dBm
Noise power (δ^2)	-114 dBm
CPU cycle frequency of TaVn (f_n^{loc}) or SeVn (f_n^{sev})	[1, 2] GHz
The speed of TaVn (v_n^{tav}) or SeVn (v_n^{sev})	[10, 15] m/s
The distance between TaVn and SeVn	[50, 100] m
CPU cycle frequency of the VECS (F^{edg})	40 GHz
Data size of a task (D_n)	[5, 15] Mbits
The required CPU cycles per bit of a task (App_n)	[50, 150] CPU cycles/bit

5.2. Baseline Algorithms

1. OLSM [10]: TaVs choose to offload part of the tasks to corresponding SeVs via V2V links.
2. OLEM [14]: TaVs choose to offload part of the tasks to the edge server via V2I links.
3. ORM: TaVs choose the offloading mode randomly.

5.3. Simulation Results

In this section, the convergence of the proposed algorithm is first analyzed. Then, the cumulative reward and performance of the proposed algorithm are verified and compared with the baseline algorithms in four aspects: the number of TaVs, task size, required computation resources per bit, and the computing capability of vehicles and VECS.

Figure 3 shows the changing trend of different algorithms' rewards with the number of iterations. When the number of iterations is 500, the proposed MCORA and OLSM algorithms converge. In contrast, the cumulative rewards of the OLEM and ORM algorithms are relatively stable throughout the process. Due to the change in network topology and channel fading caused by vehicle mobility, the fluctuation of TaVs' task size and processing density, and the change in system computing capability, the convergence value of cumulative rewards will fluctuate. It is seen that MCORA has the best cumulative reward.

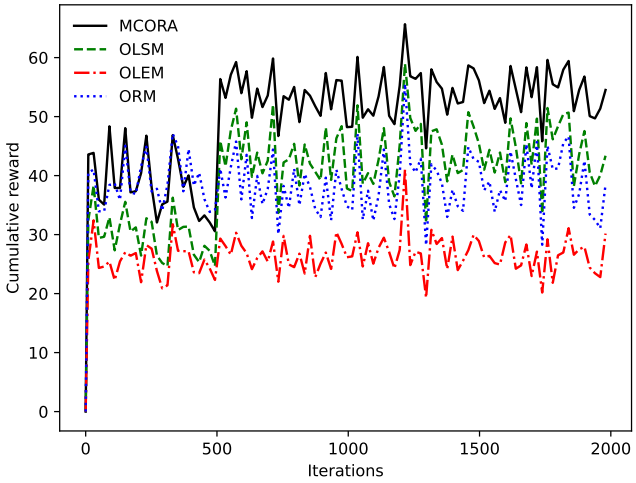


Figure 3. Cumulative rewards of different algorithms vs. the number of iterations.

Figure 4 shows the cumulative rewards of different algorithms as the number of TaVs changes. As the number of TaVs increases, the cumulative rewards of all algorithms decrease. Due to the limitation of VECS resources, the cumulative reward of the OLEM algorithm decreases sharply, and the decreasing trend of the ORM algorithm is faster than that of the OLSM algorithm because part tasks in the ORM algorithm choose Loc + Edge mode. Moreover, the decreasing trend of the cumulative reward of the MCORA algorithm

and the OLSM algorithm is relatively stable, and the cumulative reward of the MCORA algorithm is always the largest.

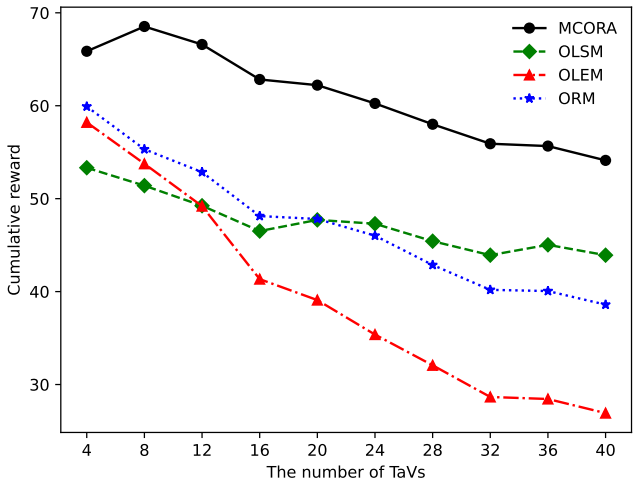
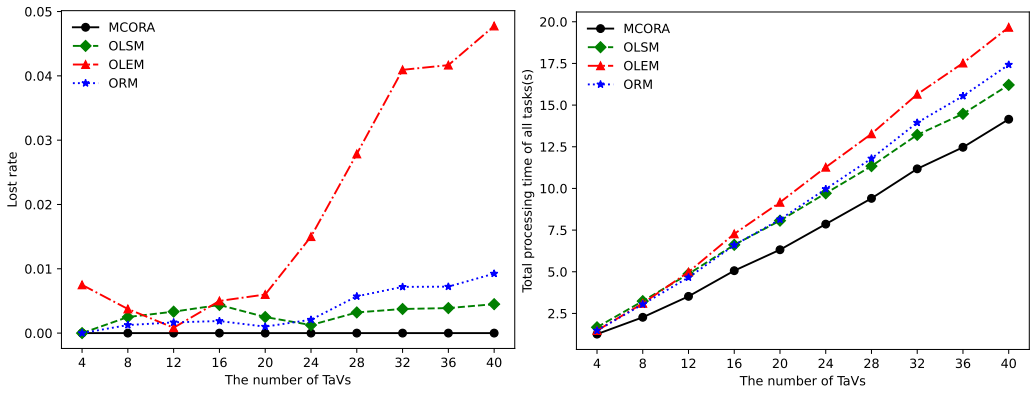


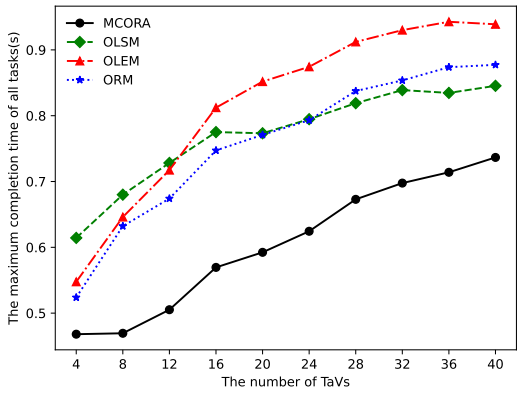
Figure 4. Cumulative rewards of different algorithms vs. the number of TaVs.

Figure 5 shows the performance of different algorithms when the number of TaVs increases, and the performance comparison mainly includes the loss rate, the total completion time of tasks, and the maximum task completion time. The loss rate is defined as the ratio of tasks that cannot be completed with limited delay. As shown in Figure 5a, the OLEM algorithm has the highest loss rate, which reaches nearly 5% when the number of tasks is 40. However, the other three algorithms, MCORA, OLSM, and ORM, are all below 1%, with MCORA almost 0%. As shown in Figure 5b, when the number of tasks is 40, compared with the OLSM algorithm, OLEM algorithm, and ORM algorithm, the total task completion time of the MCORA algorithm is reduced by 13%, 28%, and 19%, respectively. From Figure 5c, it can be seen that the maximum task time of the OLEM algorithm has a significant upward trend. When the number of tasks is small, the maximum task execution time of the OLEM and ORM algorithm is smaller than that of the OLSM algorithm due to the abundant VECS resources. As the number of TaVs and tasks increases, compared with the OLEM and ORM algorithms, the maximum task completion time of the OLSM algorithm rises slowly. When the number of tasks is greater than 12 and 20, the maximum task completion time of the OLEM algorithm and the ORM algorithm is gradually close to and greater than the OLSM algorithm.



(**a**) (**b**)

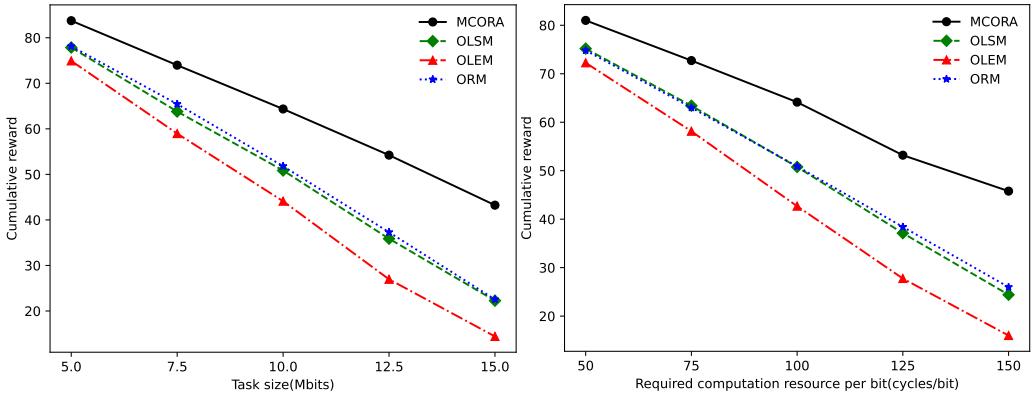
Figure 5. *Cont.*



(**c**)

Figure 5. Performance comparison of different algorithms vs. the number of TaVs. (**a**) Loss rate vs. the number of TaVs. (**b**) Total processing time vs. the number of TaVs. (**c**) Tasks' maximum completion time vs. the number of TaVs.

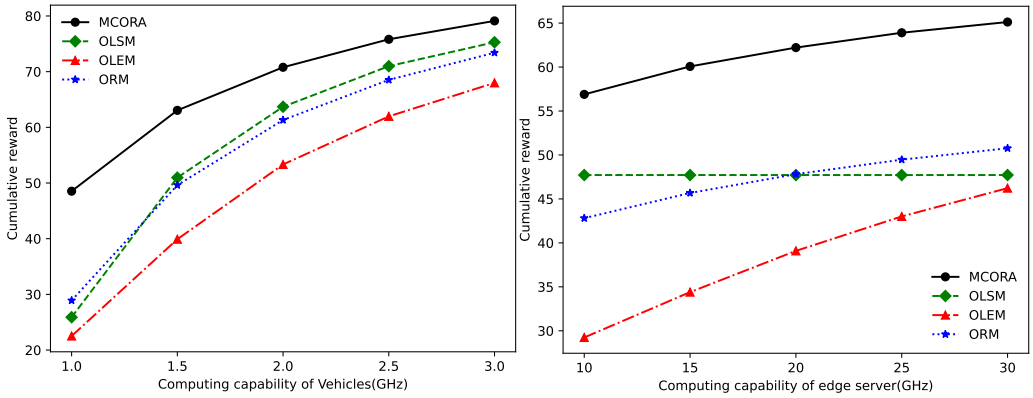
When the task size of TaVs and the computation resource required per bit varied, the cumulative rewards of different algorithms are shown in Figure 6a and Figure 6b, respectively. As the task size and computing density increase, the computation resources required by the TaVs gradually increases, and the cumulative reward decreases.



(**a**) (**b**)

Figure 6. Cumulative rewards of different algorithms as task size and required computation resource per bit vary. (**a**) Cumulative rewards of different algorithms vs. task size. (**b**) Cumulative rewards of different algorithms vs. required computation resource per bit.

When the computation capability of vehicles and VECS are changed, the cumulative rewards of different algorithms are shown in Figure 7. It can be seen from Figure 7a that the increases in vehicle computation capability gradually increase the cumulative rewards of all algorithms. The growth trend of these algorithms is similar and stable because they use their computation resources to process tasks, and improving vehicle computing performance is bound to increase the cumulative rewards. However, as shown in Figure 7b, as the VECS performance increases, the cumulative reward of the OLSM algorithm remains unchanged because the OLSM algorithm does not use the VECS resources. The other three algorithms have steadily increased cumulative rewards.



(**a**) (**b**)

Figure 7. Cumulative rewards of different algorithms as task size and required computation resource per bit vary. (**a**) Cumulative rewards of different algorithms vs. computing capability of TaVs. (**b**) Cumulative rewards of different algorithms vs. computing capability of edge server.

Figure 8a,d, Figure 8b,e, and Figure 8c,f show the comparison of the loss rate, the total task completion time, and the maximum completion time of a single task as the TaVs' task size and computing density varies, respectively. When the task size or computing density is small, it can be seen from Figure 8a,d that the loss rate of all algorithms is close to 0%. With increased task size or computing density, the loss rate of the OLSM algorithm, OLEM algorithm, and ORM algorithm increases due to insufficient utilization of system computation resources. It can be seen from Figure 8b,e that the growth trend of the total task completion time is similarly under the influence of these two variables.

Meanwhile, as the tasks' number is 20, the performance of the OLSM and OLEM algorithm is almost the same, which can be confirmed in Figures 4 and 5. When the single task size is 15 Mbits, the total task completion time of the MCORA algorithm is reduced by 18%, 21%, and 20%, respectively, compared with the OLSM algorithm, OLEM algorithm, and ORM algorithm. When the required computation resource per bit is 150 cycles/bit, the total task completion time of the MCORA algorithm is reduced by 20%, 24%, and 20%, respectively, compared with the OLSM algorithm, OLEM algorithm, and ORM algorithm.

Figure 9a–c and Figure 9d–f show the performance comparison with the change in vehicle computing capability and VECS computing capability, respectively. When the computing capacity of the vehicle is small, TaVs and SeVs can only provide fewer computing resources, and the system computing resources are relatively scarce, which leads to the larger loss rate, task completion time, and the maximum single task completion time of the OLSM algorithm, OLEM algorithm, and ORM algorithm. When the vehicle computation capability exceeds 1.5GHz, all algorithms can complete all tasks according to the regulations. When the vehicle computation capability is 3 GHz, the total task completion time of the MCORA algorithm is reduced by 13%, 30%, and 19% compared with the OLSM algorithm, OLEM algorithm, and ORM algorithm, respectively. From Figure 9d, we can see that the loss rate of the OLEM algorithm is less than 1% only when the VECS computation capability is greater than 20 GHz, while the loss rate of the other three algorithms is always in a low range because when the number of tasks is 20, according to the average vehicle computation capability, the computation capability that vehicles can provide is 20×15, namely, 30 GHz. It has certain advantages to make full use of ubiquitous vehicle resources reasonably. It can be seen from Figure 9b that when the computation capability of VECS is 30 GHz, the total task completion time of the MCORA algorithm is reduced by 26%, 29%, and 24%, respectively, compared with the OLSM algorithm, OLEM algorithm, and ORM algorithm.

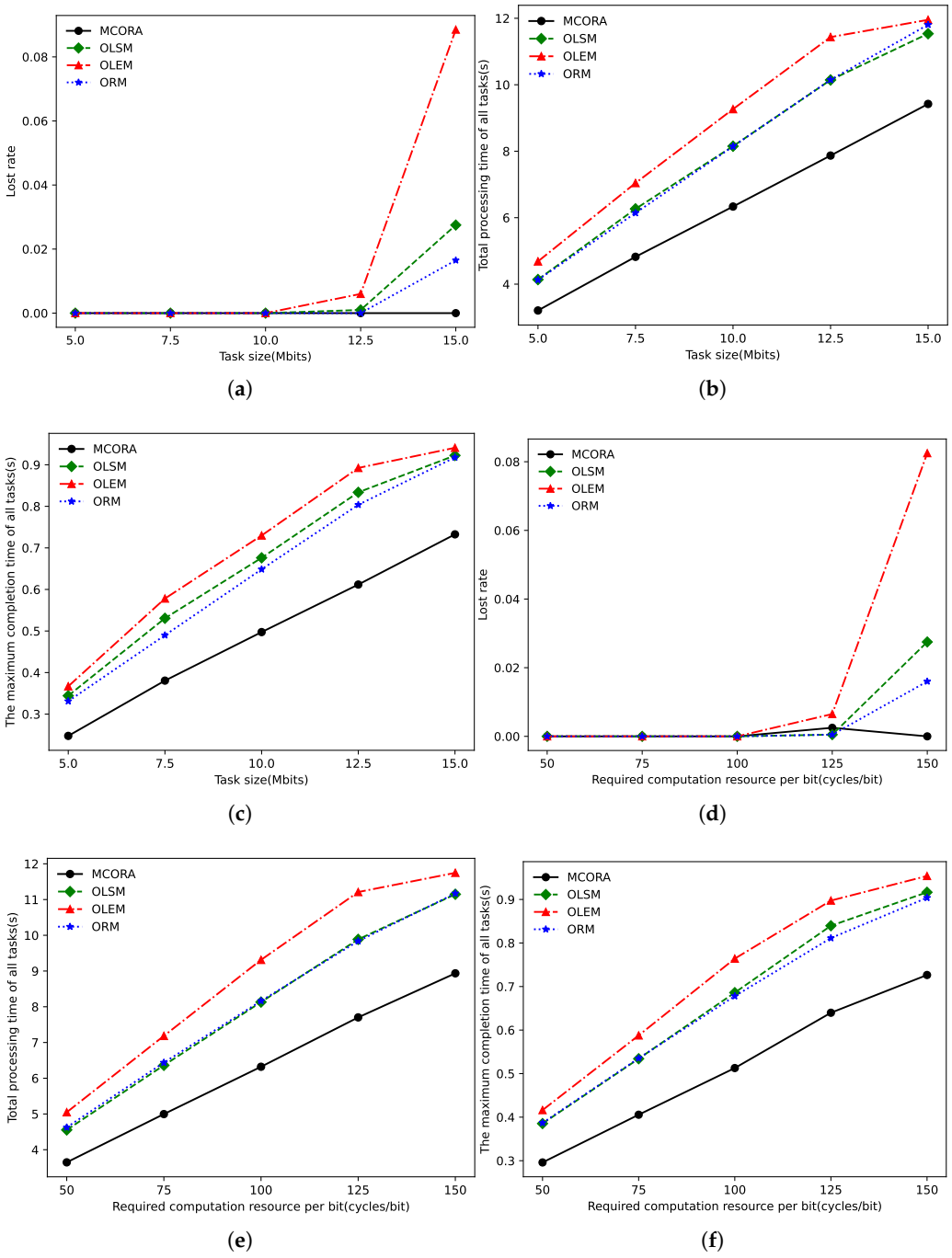


Figure 8. Performance comparison of different algorithms as task size and required computation resources per bit vary. (**a**) Loss rate vs. task size. (**b**) Total processing time vs. task size. (**c**) Tasks' maximum completion time vs. task size. (**d**) Loss rate vs. required computation resource per bit. (**e**) Total processing time vs. required computation resource per bit. (**f**). Tasks' maximum completion time vs. required computation resources per bit.

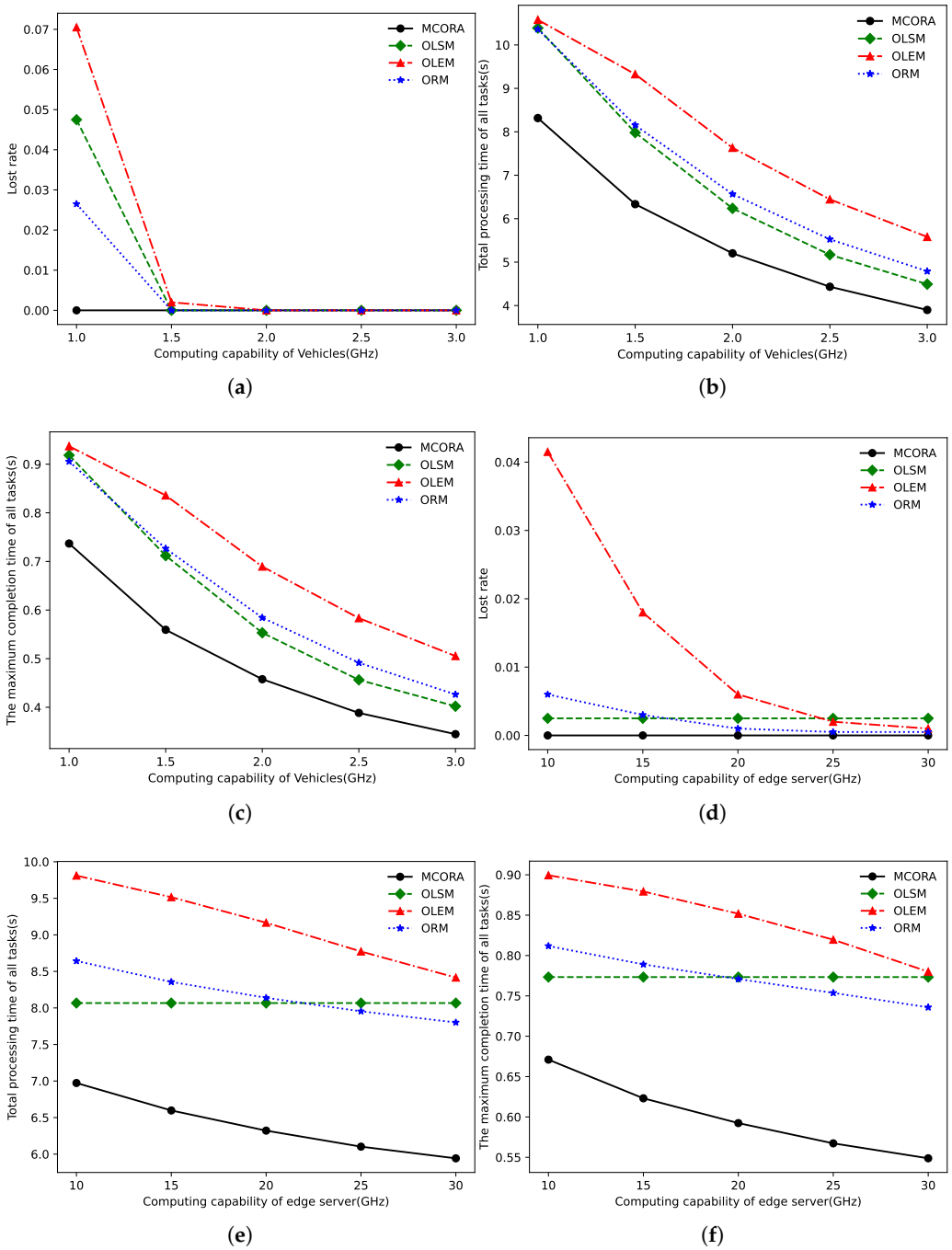


Figure 9. Performance comparison of different algorithms as task size and required computation resource per bit vary. (**a**) Loss rate vs. computing capability of TaVs. (**b**) Total processing time vs. computing capability of TaVs. (**c**) Tasks' maximum completion time vs. computing capability of TaVs. (**d**) Loss rate vs. computing capability of edge server. (**e**) Total processing time vs. computing capability of edge server. (**f**) Tasks' maximum completion time vs. computing capability of edge server.

These simulation results show that, compared with the other three schemes, the MCORA scheme can effectively reduce the total delay of task execution, guarantee the QoS of TaVs, and have a certain scalability and stability.

5.4. Discussion, Comparison, and Limitations

According to the number of bits each TaV can execute in different modes, the DDPG-based MCORA algorithm is used to select the appropriate task execution mode for each TaV in each time slot τ. Meanwhile, the computing resources of VECS are allocated. Compared with the OLSM and OLEM algorithms, the proposed MCORA algorithm can fully use ubiquitous communication and computing resources in VECNs. Although four execution modes are considered, the simulation results are carried out regarding the number of TaVs, task size, required computation resource per bit, and the computing capability of vehicles and VECS. However, the following shortcomings and limitations still exist: 1. The delay for each task in our proposed system is tmax, regardless of the diversity of tasks; 2. The limited latency of tasks is considered, but the overhead of energy consumption is ignored; 3. DDPG is more challenging to deploy.

6. Conclusions

This paper proposes a MCORA optimization model based on DDPG reinforcement learning for the computing task offloading environment of the IoV. Reinforcement learning is used to allocate task offloading modes and VECS computation resources, aiming to solve the problem of the insufficient utilization of system resources in the dynamic environment of VECNs. The proposed method can quickly obtain the approximate optimal solution in a time-varying environment and achieve a low total task completion delay with almost no task lost. The proposed method has better stability and scalability than the existing algorithms. VECNs are gradually developed and improved with the development of cellular networks, and it occupies a certain proportion in the development of 5G and 6G. It can be used for autonomous driving, smart city, and digital twin construction in the future. Additionally, more effective offloading strategies deserve to be formulated by combining task execution and energy consumption because energy saving is essential [38–40].

Author Contributions: Conceptualization, X.L. and J.Z.; methodology, X.L. and M.Z.; software, X.L. and Y.L.; investigation, X.L.; resources, J.Z.; writing—original draft preparation, X.L., R.W. and Y.H.; writing—review and editing, X.L., J.Z., M.Z. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

VECNs	Vehicular Edge Computing Networks
QoS	Quality of Service
VECS	Vehicular Edge Computing Server
DDPG	Deep Deterministic Policy Gradient
MINLP	Mixed Integer Nonlinear Programming
IoV	Internet of Vehicles
TaVs	Task Vehicles
SeVs	Service Vehicles
V2V	Vehicle-to-Vehicle
RSU	Road Side Unit
BS	Base Station
V2I	Vehicle-to-Infrastructure
Loc	Local Execution

Loc + SeV	Local + SeV execution
Loc + Edge	Local + VECS execution
Loc + Sev + Edge	Local + SeV + VECS execution
MCORA	Multi-user computation offloading and resource allocation
OLSM	Offloading in Loc + Sev mode
OLEM	Offloading in Loc + Edge mode
ORM	Offloading in Random Mode

References

1. Ren, Y.; Yu, X.; Chen, X.; Guo, S.; Qiu, X.-S. Vehicular Network Edge Intelligent Management: A Deep Deterministic Policy Gradient Approach for Service Offloading Decision. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 905–910. [CrossRef]
2. Raja, G.; Anbalagan, S.; Senthilkumar, S.; Dev, K.; Qureshi, N.M.F. SPAS: Smart Pothole-Avoidance Strategy for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 19827–19836. [CrossRef]
3. Sohail, R.; Saeed, Y.; Ali, A.; Alkanhel, R.; Jamil, H.; Muthanna, A.; Akbar, H. A Machine Learning-Based Intelligent Vehicular System (IVS) for Driver's Diabetes Monitoring in Vehicular Ad-Hoc Networks (VANETs). *Appl. Sci.* **2023**, *13*, 3326. [CrossRef]
4. Sohail, H.; Hassan, M.u.; Elmagzoub, M.A.; Rajab, A.; Rajab, K.; Ahmed, A.; Shaikh, A.; Ali, A.; Jamil, H. BBSF: Blockchain-Based Secure Weather Forecasting Information through Routing Protocol in Vanet. *Sensors* **2023**, *23*, 5259. [CrossRef] [PubMed]
5. Jamil, F.; Cheikhrouhou, O.; Jamil, H.; Koubaa, A.; Derhab, A.; Ferrag, M.A. PetroBlock: A Blockchain-Based Payment Mechanism for Fueling Smart Vehicles. *Appl. Sci.* **2021**, *11*, 3055. [CrossRef]
6. Dai, Y.; Xu, D.; Zhang, K. Deep Reinforcement Learning for Edge Computing and Resource Allocation in 5G Beyond. In Proceedings of the IEEE 19th International Conference on Communication Technology (ICCT), Xi'an, China, 16–19 October 2019; pp. 866–870. [CrossRef]
7. He, X.; Lu, H.; Huang, H.; Mao, Y.; Wang, K.; Guo, S. QoE-Based Cooperative Task Offloading with Deep Reinforcement Learning in Mobile Edge Networks. *IEEE Wirel. Commun.* **2020**, *27*, 111–117. [CrossRef]
8. He, X.; Lu, H.; Mao, Y.; Wang, K. QoE-driven Task Offloading with Deep Reinforcement Learning in Edge intelligent IoV. In Proceedings of the IEEE Global Communications Conference (GLOBECOM), Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
9. Wang, L.; Liang, H.; Zhao, D. Deep Reinforcement Learning-Based Computation Offloading and Power Allocation within Dynamic Platoon Network. *IEEE Internet Things J.* **2023**, *11*, 10500–10512. [CrossRef]
10. Lu, Y.; Wang, X.; Li, F.; Yi, B.; Huang, M. RLbR: A reinforcement learning based V2V routing framework for offloading 5G cellular IoT. *IET Commun.* **2022**, *16*, 303–313. [CrossRef]
11. Geng, L.; Zhao, H.; Liu, H.; Wang, Y.; Feng, W.; Bai, L. Deep Reinforcement Learning-based Computation Offloading in Vehicular Networks. In Proceedings of the 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/the 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), Washington, DC, USA, 26–28 June 2021; pp. 200–206. [CrossRef]
12. Sun, H.; Ma, D.; She, H.; Guo, Y. EC-DDPG: DDPG-Based Task Offloading Framework of Internet of Vehicle for Mission Critical Applications. In Proceedings of the IEEE International Conference on Communications Workshops (ICC Workshops), Rome, Italy, 28 May–1 June 2023; pp. 984–989. [CrossRef]
13. Shan, Y.; Liao, P.; Wang, Z.; An, L. Partial Computation Offloading and Resource Allocation via Deep Deterministic Policy Gradient. In Proceedings of the International Conference on Networking and Network Applications (NaNA), Urumqi, China, 3–5 December 2022; pp. 376–383. [CrossRef]
14. Zhang, L.; Zhou, W.; Xia, J. DQN-based mobile edge computing for smart Internet of vehicle. *EURASIP J. Adv. Signal Process* **2022**, *2022*, 45. [CrossRef]
15. Lu, J.; Chen, L.; Xia, J. Analytical offloading design for mobile edge computing-based smart internet of vehicle. *EURASIP J. Adv. Signal Process* **2022**, *2022*, 44. [CrossRef]
16. Li, Z.; Yu, Z. A Multi-user Computation Offloading Optimization Model and Algorithm Based on Deep Reinforcement Learning. *J. Electron. Inf. Technol.* **2023**, *45*, 1–12. [CrossRef]
17. Lin, J.; Huang, S.; Zhang, H.; Yang, X.; Zhao, P. A Deep-Reinforcement-Learning-Based Computation Offloading with Mobile Vehicles in Vehicular Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 15501–15514. [CrossRef]
18. Liao, Y.; Qiao, X.; Yu, Q.; Liu, Q. Intelligent dynamic service pricing strategy for multi-user vehicle-aided MEC networks. *Future Gener. Comput. Syst.* **2021**, *114*, 15–22. [CrossRef]
19. Ma, C.; Zhu, J.; Liu, M.; Zhao, H.; Liu, N.; Zou, X. Parking Edge Computing: Parked-Vehicle-Assisted Task Offloading for Urban VANETs. *IEEE Internet Things J.* **2021**, *8*, 9344–9358. [CrossRef]
20. Tian, S.; Deng, X.; Chen, P.; Pei, T.; Oh, S.; Xue, W. A dynamic task offloading algorithm based on greedy matching in vehicle network. *Ad Hoc Netw.* **2021**, *123*, 102639. [CrossRef]
21. Ma, S.; Song, S.; Yang, L.; Zhao, J.; Yang, F.; Zhai, L. Dependent tasks offloading based on particle swarm optimization algorithm in multi-access edge computing. *Appl. Soft Comput.* **2021**, *112*, 107790. [CrossRef]

22. Liao, Z.; Peng, J.; Xiong, B. Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm. *J. Cloud Comput.* **2021**, *10*, 15. [CrossRef]
23. Liu, H.; Niu, Z.; Du, J.; Lin, X. Genetic algorithm for delay efficient computation offloading in dispersed computing. *Ad Hoc Netw.* **2023**, *142*, 103109. [CrossRef]
24. Alameddine, H.A.; Sharafeddine, S.; Sebbah, S.; Ayoubi S.; Assi, C. Dynamic Task Offloading and Scheduling for Low-Latency IoT Services in Multi-Access Edge Computing. *IEEE J. Sel. Area Commun.* **2019**, *37*, 668–682. [CrossRef]
25. Tong, Z.; Deng, X.; Mei, J.; Dai, L.; Li, K.; Li, K. Stackelberg game-based task offloading and pricing with computing capacity constraint in mobile edge computing. *J. Syst. Archit.* **2023**, *137*, 102847. [CrossRef]
26. Jiang, W.; Feng, D.; Sun, Y.; Feng, G.; Wang, Z.; Xia, X.-G. Joint Computation Offloading and Resource Allocation for D2D-Assisted Mobile Edge Computing. *IEEE Trans. Serv. Comput.* **2023**, *16*, 1949–1963. [CrossRef]
27. Dai, Y.; Xu, D.; Zhang, K. Deep Reinforcement Learning and Permissioned Blockchain for Content Caching in Vehicular Edge Computing and Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 4312–4324. [CrossRef]
28. Gueriani, A.; Kheddar, H.; Mazari, A.C. Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey. In Proceedings of the 2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM), Medea, Algeria, 28–29 November 2023; pp. 1–7. [CrossRef]
29. Zhang, H.; Feng, L.; Liu, X.; Long, K.; Karagiannidis, G.K. User Scheduling and Task Offloading in Multi-Tier Computing 6G Vehicular Network. *IEEE J. Sel. Area Commun.* **2023**, *41*, 446–456. [CrossRef]
30. Yao, L.; Xu, X.; Bilal, M.; Wang, H. Dynamic Edge Computation Offloading for Internet of Vehicles with Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 12991–12999. [CrossRef]
31. He, X.; Lu, H.; Du, M.; Mao, Y.; Wang, K. QoE-Based Task Offloading With Deep Reinforcement Learning in Edge-Enabled Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2252–2261. [CrossRef]
32. Hazarika, B.; Singh, K.; Biswas, S.; Li, C.-P. DRL-Based Resource Allocation for Computation Offloading in IoV Networks. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8027–8038. [CrossRef]
33. Liu, X.; Zheng, J.; Zhang, M. A novel D2D–MEC method for enhanced computation capability in cellular networks. *Sci. Rep.* **2021**, *11*, 16918. [CrossRef]
34. Liu, X.; Zheng, J.; Zhang, M.; Li, Y.; Wang, R.; He, Y. A Game-Based Computing Resource Allocation Scheme of Edge Server in Vehicular Edge Computing Networks Considering Diverse Task Offloading Modes. *Sensors* **2024**, *24*, 69. [CrossRef]
35. 3GPP. *3rd Generation Partnership Project; Technical Specification Group Radio Access Network;* Study on LTE-based V2X Services (Release 14); Technical Report TR 36.885; 3GPP: Sophia Antipolis, France, 2016.
36. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018; pp. 47–50.
37. Fang, W.; Wang, Y.; Zhang, H. Optimized communication resource allocation in vehicular networks based on multi-agent deep reinforcement learning. *J. Beijing Jiaotong Univ.* **2022**, *46*, 64–72. [CrossRef]
38. Naqvi, S.S.A.; Jamil, H.; Faseeh, M.; Iqbal, N.; Khan, S.; Kim, D.H. A comprehensive review on development strategies of integrated electronic control units in IoEVs for energy management. *Internet Things* **2024**, *25*, 101085. [CrossRef]
39. Nasir, T.; Raza, S.; Abrar, M.; Muqeet, H.A.; Jamil, H.; Qayyum, F.; Cheikhrouhou, O.; Alassery, F.; Hamam, H. Optimal Scheduling of Campus Microgrid Considering the Electric Vehicle Integration in Smart Grid. *Sensors* **2021**, *21*, 7133. [CrossRef]
40. Jamil, H.; Naqvi, S.S.A.; Iqbal, N. Analysis on the Driving and Braking Control Logic Algorithm for Mobility Energy Efficiency in Electric Vehicle. *Smart Grids Energy* **2024**, *9*, 12. [CrossRef]