Learning-Based Load-Aware Heterogeneous Vehicular Edge Computing

Lei Zhu[*‡], Zhizhong Zhang[†], Peng Lin[†], Omair Shafiq[‡], Yu Zhang[†], and F. Richard Yu[‡]
[*]School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[†]School of Electronic and Information Engineering,
Nanjing University of Information Science and Technology, Nanjing 210044, China
[‡]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, K1S 5B6, Canada

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001192

***Abstract*—Vehicular edge computing is an emerging enabler to support vehicular-based computation-intensive tasks. By reason of the time-varying vehicular wireless environments and the stochastic task generation, the dynamically unbalanced task load distribution among resource-constrained edge infrastructures leads to the performance bottleneck and low efficiency of computation resource utilization. We employ an aerial relay station that can establish relay connections between vehicles and nearby heterogeneous edge infrastructures to relieve this situation. The computation offloading strategy design in the multi-vehicle multi-edge infrastructure scenario that is closely linked to system latency performance will be particularly complicated. To address this issue, a model-free multi-agent reinforcement learning is adopted, and we propose a practical constraint in the problem formulation. Simulation experiments show that the proposed strategy can guarantee load balancing among edge infrastructures.**

***Index Terms*—Vehicular edge computing, load balancing, multi-agent reinforcement learning, long-term optimization.**

I. Introduction

Rapid developments in sensor technologies and wireless communications are enabling the proliferation of Internet of Vehicles (IoV), which can achieve a comfortable ecosystem integrated by “human,” “vehicle,” and “thing” [1]. Camera-based and radar-based sensors embedded in intelligent and connected vehicles (ICVs) can collect surrounding driving environment contexts and real-time operating states [2]. Such sensing devices generate a huge mass of raw data to perform enormous computation-intensive, bandwidth-consuming, and latency-sensitive vehicular applications, such as smart anti-collision warning, autonomous driving, and high-precision navigation [3], [4].

Vehicular edge computing (VEC) infrastructures, such as roadside units (RSUs) and macro base stations (MBSs), are envisioned to be furnished with onboard sensing, communication and computing facilities. They can form heterogeneous small-scale cloudlets [5], [6]. Offloading tasks to such edge cloudlets can result in extra energy and time consumption and may not always bring benefits [7], [8]. Thus computation offloading decision is quite crucial to satisfy the quality of service (QoS) of VEC applications and improve the resource utilization efficiency in resource-constrained edge systems.

Extensive existing works have formulated the VEC offloading decision as single-objective optimization problems or joint optimization problems considering latency, energy consumption, QoS, and revenue. The authors in [9] focused on vehicle-to-vehicle computation offloading and designed an adaptive learning-based offloading algorithm to minimize the average offloading latency in a distributed manner. In [10], the idle neighboring vehicles, RSUs, and base stations formed an edge computing directed acyclic graph to serve the vehicles with computing tasks. The authors in [11] formulated the edge resources allocation in a generalized vehicular edge computing paradigm as an auction mechanism to perform coded distributed computing tasks. The authors in [12] integrated the high altitude platform station into the VEC system and then formulated the system delay minimization problem considering the offloading policy, caching strategy as well as resource allocation. Due to the advantages of agility and good line-of-sight (LOS) connection, aerial relay stations (ARSs) were deployed to provide effective relay connectivity and coverage to improve the overall computation capacity [13].

Although the aforementioned works have illustrated significant efforts in the VEC performance improvement, there are still some problems to be solved: *i*. due to the ongoing vehicular movement and stochastic task arrival, the task load distribution among the resource-constrained edge infrastructures is generally unbalanced, which leads to the performance bottleneck and low resource efficiency; *ii*. as far as we know, the online offloading decision problem in the ARS-assisted multi-vehicle multi-edge infrastructure system has not been explored in existing works. The main contributions of this work are summarized as follows:

- We propose a novel ARS-enabled heterogeneous edge computing paradigm for IoV. The ARS with wide coverage and good channel condition is served as a relay to alleviate the unbalanced load distribution among edge nodes.

This work is supported in part by the Basic Science (Natural Science) Research Project of Jiangsu Colleges and Universities of China under Grant 22KJB510003 and in part by the open research fund of National Mobile Communications Research Laboratory, Southeast Universityunder Grant 2022D11. (Corresponding Author: Zhizhong Zhang)

- We propose a practical constraint named offloading gain and formulate the offloading decision with dynamic candidate edge infrastructure set as a system computation latency minimization problem. The problem is challenging because of the online decision requirement with exponentially large search space.
- We transform the optimization problem based on a decentralized partially observable Markov decision process (Dec-POMDP). Then, a learning-based algorithm in a centralized training and distributed execution (CTDE) manner is employed to learn the long-term real-time optimal state-action mapping. Extensive simulation experiments show that the proposed offloading scheme outperforms the existing baseline schemes*.

II. System Model and Problem Statement

A. System Architecture

As illustrated in Fig. 1, we consider a heterogeneous vehicular edge computing (HVEC) scenario, where a set of ICVs, denoted by $\mathcal{J} = \{1, 2, ..., j, ..., J\}$, are driven along the expressway. A set of RSUs equipped with edge servers, denoted by $\mathcal{H} = \{1, 2, ..., h, ..., H\}$. The ICV-RSU communication connections are operating on the mmWave band, and each RSU can establish no more than N^{R} mmWave beams at the same time. The other communication links in the scenario are operating on the Sub-6G band. The MBS equipped with an edge server, denoted by M, is deployed away from the expressway to serve multiple types of mobile users. The ICVs are unable to establish acceptable connections with the MBS due to signal blockage and shadowing. We denote the maximal CPU-cycle frequency of each ICV, each RSU and the MBS by F^{I}, F^{R} and F^{M}, respectively. The ARS with energy capacity E^{A}, denoted by A, hovers at a fixed coordinate above the ground to provide relay service, and it needs to recharge at the ground charging station when the available energy reaches a threshold value.

The communication among edge servers is realized through the core network. Furthermore, due to the large size of raw task data, transmitting the raw task data among edge servers can distinctly degrade the core network's transmission efficiency, and thus the load distribution among edge servers will be unbalanced. To relieve this issue, by deploying the ARS with wide coverage and good LOS channel condition, the tasks of ICVs can be offloaded to nearby RSUs or the MBS with low load. Then, an ARS-assisted HVEC network is formed.

The location of ICV j in the range of RSU h is denoted by $\Theta_{j^h}(t) = \left[x_{j^h}(t), y_{j^h}(t), z_{j^h}(t)\right]$. The velocity of ICVs follow a truncated Gaussian distribution, which is widely used to model the practical vehicular movement [14]. Therefore, the equivalent velocity of an ICV is denoted by

$$v_{j^h} = \frac{1}{\int_{V_{\min}}^{V_{\max}} \frac{\tilde{f}_v(v)}{v} dv} = \frac{erf\left(\frac{V_{\max}-\mu}{\sigma\sqrt{2}}\right) - erf\left(\frac{V_{\min}-\mu}{\sigma\sqrt{2}}\right)}{\frac{2}{\sigma\sqrt{2\pi}} \int_{V_{\min}}^{V_{\max}} \frac{\exp\left(-\frac{(v-\mu)^2}{2\sigma^2}\right)}{v} dv}, \tag{1}$$

*The source code for the proposed offloading method is available at https://github.com/HuangHanzhong/HDLOA.

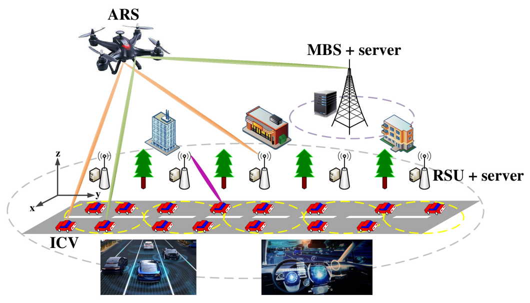

Fig. 1. An illustration of the HVEC architecture.

where $\tilde{f}_v(v)$ is the truncated Gaussian probability density function with mean μ and standard deviation σ; $erf(\cdot)$ is error function; and $V_{\min} = \mu - 3\sigma$ and $V_{\max} = \mu + 3\sigma$ are the minimal and maximal velocities, respectively.

B. Computation Offloading Model

In this HVEC system, we discretize the time horizon into decision epochs and each epoch is indexed by $t \in \mathbb{N}_+$. At the beginning of decision epoch t, ICV j has a periodical computation-intensive indivisible task to be processed in time, which is modeled as $\Omega_{j^h}(t) = \left\{\lambda_{j^h}(t), \rho_{j^h}(t)\right\}$. Here, let $\lambda_{j^h}(t)$ denote the raw task data size. The processing density, i.e., the number of CPU cycles required to process a unit of data, is denoted by $\rho_{j^h}(t)$.

We consider a binary offloading scenario. Each ICV chooses to compute the task locally or one of the edge infrastructures. The offloading decision set of ICV j is denoted by $\mathbf{\Psi}_{j^h}(t) \in \left\{\psi^j_{j^h}, \psi^1_{j^h}, ..., \psi^h_{j^h}, ..., \psi^H_{j^h}, \psi^M_{j^h}\right\}$. In what follows, we will introduce the offloading models with latency analyses.

1) Local Computing: When the task is executed locally, i.e., $\psi^j_{j^h} = 1$, the task computing delay is

$$D^{\mathrm{lc}}_{j^h}(t) = \frac{\lambda_{j^h}(t)\rho_{j^h}(t)}{F^{\mathrm{I}}}. \tag{2}$$

2) Direct Edge Computing: As for the computation offloading process, firstly, the raw task information is offloaded to the edge server. Whereafter, the edge server performs the computing and transmits the results to the vehicle. Considering that the size of the results is minimal, the downlink delay can be neglected in this paper.

We assume the mmWave communication links for ICV-RSU are interference-constrained, and the impact of multipath fading and non-line-of-sight (NLOS) transmission is negligible [15], [16]. Thus, the LOS channel power gain is below

$$g^{\mathrm{IR}}_{j^h,h}(t) = g^{\mathrm{fspl}}(f_c, d_0) + 10\varphi^{\mathrm{IR}}\log_{10}\left(\frac{d^{\mathrm{IR}}_{j^h,h}}{d_0}\right) + X_{\sigma^{\mathrm{sf}}}, \tag{3}$$

where $g^{\mathrm{fspl}}(f_c, d_0) = 20\log_{10}\left(4\pi d_0 f_c / c_{\mathrm{light}}\right)$ denotes the free space path loss, and it is a function of carrier frequency f_c at the reference distance $d_0 = 1$ m, and where c_{light} is the speed of light; φ^{IR} stands for the path-loss exponent; $d^{\mathrm{IR}}_{j^h,h}(t) = \|\Theta_{j^h}(t) - \Theta_h\|$ is the propagation distance from ICV

j to RSU h and is assumed to be a constant, and where $||\cdot||$ denotes the Euclidean norm of a vector; and $X_{\sigma^{\mathrm{sf}}}$ denotes the lognormal shadowing, which is a zero mean Gaussian random variable with standard deviation σ^{sf}.

Based on (3), we can derive the uplink transmission rate of ICV-RSU link as follows

$$r^{\mathrm{IR}}_{j^h,h}(t) = B^{\mathrm{mmW}}\left(1 + \frac{p_{j,h}G^{\mathrm{I}}_{\mathrm{tx}}G^{\mathrm{R}}_{\mathrm{rx}}g^{\mathrm{IR}}_{j^h,h}(t)}{N_0 B^{\mathrm{mmW}}}\right), \tag{4}$$

where B^{mmW} is the bandwidth of mmWave band, $G^{\mathrm{I}}_{\mathrm{tx}}$ and $G^{\mathrm{R}}_{\mathrm{rx}}$ are antenna gains for the ICV and RSU, respectively [16]. The transmission delay is given by $D^{\mathrm{tr}}_{j^h,h}(t) = \lambda_{j^h}(t)/r^{\mathrm{IR}}_{j^h,h}(t)$.

We assume that the beam training and alignment mechanisms are executed in advance between the RSU and ICVs, and thus the appropriate beams can be established for the data transmission.

To perform multiple computation tasks simultaneously, the dynamic frequency and voltage scaling technique which can dynamically adjust the CPU frequency is adopted in VEC servers [9]. We allocate the computing resources for tasks according to the computing requirements. We define $f^h_{j^h}(t) = \frac{\lambda_{j^h}(t)\rho_{j^h}(t)}{\zeta_h(t)} * F^{\mathrm{R}}$ as the CPU-cycle frequency allocated to ICV j by RSU h, where $\zeta_h(t)$ is the whole computing resource requirements of tasks served by RSU h. The computation delay is defined by $D^{\mathrm{co}}_{j^h,h}(t) = \lambda_{j^h}(t)\rho_{j^h}(t)/f^h_{j^h}(t)$.

Therefore, the offloading delay of task $\Omega_{j,h}(t)$ can be expressed as

$$D^{\mathrm{IR}}_{j^h,h}(t) = D^{\mathrm{tr}}_{j^h,h}(t) + D^{\mathrm{co}}_{j^h,h}(t). \tag{5}$$

3) Relay Edge Computing: It is assumed that the received signal mainly depends on the ICV-ARS distance for simplicity [13]. We allocate the communication resources for tasks uploading according to the transmission requirements. The whole data size of tasks served by ARS is defined as $\lambda_{\mathrm{A}}(t)$, and then we define $b^{\mathrm{A}}_{j^h}(t) = \frac{\lambda_{j^h}(t)}{\lambda_{\mathrm{A}}(t)} * B^{\mathrm{sub}}$ as the uplink bandwidth allocated to ICV j by the ARS, where B^{sub} is the Sub-6G bandwidth. The transmission rate of the ICV-ARS link is

$$r^{\mathrm{IA}}_{j,h}(t) = b^{\mathrm{A}}_{j^h}(t)\left(1 + \frac{p^{\mathrm{I}}G^{\mathrm{I}}_{\mathrm{tx}}G^{\mathrm{A}}_{\mathrm{rx}}g^{\mathrm{IA}}_{j^h}(t)}{N_0 b^{\mathrm{A}}_{j^h}(t)}\right), \tag{6}$$

where $G^{\mathrm{R}}_{\mathrm{rx}}$ is the receiving antenna gain for the ARS; $g^{\mathrm{IA}}_{j^h}(t) = \rho_0 - 10\log_{10}\varphi^{\mathrm{IA}}d^{\mathrm{IA}}_{j^h}(t)$ is the LOS channel power gain from the ICV j to the ARS, and where ρ_0 denotes the channel power gain at the reference distance $d_0 = 1$ m, and φ^{IA} represents the path-loss exponent; and $d^{\mathrm{IA}}_{j^h}(t)$ is the distance from ICV j to the ARS.

We assume that there is no intra-cell interference in the uplink transmissions of ICVs to the ARS (e.g., due to orthogonal channel access technology used by different ICVs)[†]. Then the uplink transmission delay from ICV j to the ARS is computed by the ratio of task input data size and the associated uplink transmission data rate, which is defined by $D^{\mathrm{tr}}_{j^h,A}(t) = \lambda_{j^h}(t)/r^{\mathrm{IA}}_{j^h}(t)$.

When the task data is transmitted to the ARS successfully, the ARS starts to relay the task data of vehicle j in RSU h to the nearby RSU $k (k \neq h; k, h \in \mathcal{H})$ or the MBS. We denote the transmission rate of the relay link by $r^{\mathrm{AR}}_{j^h,k}(t)$. Then the transmission delay from the ARS to RSU k is given by $D^{\mathrm{tr}}_{A,k}(t) = \lambda_{j^h}(t)/r^{\mathrm{AR}}_{j^h,k}(t)$. The computation delay of RSU server k on task j is given by $D^{\mathrm{co}}_{j^h,k}(t) = \lambda_{j^h}(t)\rho_{j^h}(t)/f^k_{j^h}(t)$. Therefore, the offloading delay of task $\Omega_{j^h}(t)$ can be expressed as

$$D^{\mathrm{IAR}}_{j^h,k}(t) = D^{\mathrm{tr}}_{j^h,A}(t) + D^{\mathrm{tr}}_{A,k}(t) + D^{\mathrm{co}}_{j^h,k}(t). \tag{7}$$

Similarly, we can obtain the relay offloading delay to the MBS $D^{\mathrm{IAM}}_{j^h,M}(t)$. Note that the MBS server can serve not only ICVs but also some other communication devices (e.g., IoT sensors, mobile terminal), so the computation capacity of the MBS server should be time-varying. Mathematically, we define $f^{\mathrm{M}}_{\mathrm{avail}}(t) = F^{\mathrm{M}} - f^{\mathrm{M}}_{\mathrm{occu}}(t)$ as the available CPU-cycle frequency of the MBS server, where $f^{\mathrm{M}}_{\mathrm{occu}}(t)$ is the occupied computation resources for other devices and is modeled as an independent identically distributed (i.i.d.) Poisson process.

To sum up, based on (2), (5), and (7), we formulate the offloading latency $D_{j^h}(t)$ of ICV j as follows

$$D_{j^h}(t) = \psi^j_{j^h}D^{\mathrm{lc}}_{j^h} + \psi^h_{j^h}D^{\mathrm{IR}}_{j^h,h} + \psi^k_{j^h}D^{\mathrm{IAR}}_{j^h,k} + \psi^M_{j^h}D^{\mathrm{IAM}}_{j^h,M}. \tag{8}$$

C. Problem Formulation

If ICV j offloads the task to one of the edge infrastructures with delay $D^{\mathrm{eg}}_{j^h}(t)$, we define a practical evaluating indicator named offloading gain by $\varpi_{j^h}(t) = D^{\mathrm{lc}}_{j^h}(t) - D^{\mathrm{eg}}_{j^h}(t)$. Only when $\psi^j_{j^h}(t) = 0$ and $\varpi_{j^h}(t) > 0$, the offloading decision is applicable for ICV j. To our best knowledge, we are the first to consider this practical constraint in the problem formulation.

To keep the load balancing among all the computation anchors, the optimization objective is to minimize the weighted sum of the task delay of ICVs by optimizing $\mathbf{\Psi}(t) = \left\{\mathbf{\Psi}_1(t), \mathbf{\Psi}_2(t), ,..., \mathbf{\Psi}_j(t), \mathbf{\Psi}_J(t)\right\}$, where $\mathbf{\Psi}_j(t) = \left\{\mathbf{\Psi}_{j^h}(t), \forall h \in \mathcal{H}\right\}$. Then we define

$$\begin{aligned} \mathrm{P} : & \min_{\mathbf{\Psi}(t)} \sum_{\forall j \in \mathcal{J}} D_{j^h}(t) \\ & \mathrm{C1} : \sum \mathbf{\Psi}_j(t) \in \{0, 1\},\ \forall j \in \mathcal{J}, \\ & \mathrm{C2} : \varpi_{j^h}(t) > 0,\ \forall j \in \mathcal{J}\ \&\ \psi^j_{j^h} = 0, \\ & \mathrm{C3} : n^{\mathrm{R}}(t) \leq N^{\mathrm{R}}, \end{aligned} \tag{9}$$

where C1 is the offloading indicator constraint; C2 guarantees that ICVs should get positive offloading gains when offload to edge infrastructures; and C3 denotes the maximum served ICV number constraints of RSUs in mmWave band.

To derive a satisfying offloading decision, one needs to search in a space including $(H + 2)^J$ possible offloading decisions. Traditional "one-shot" solutions such as [9], [11] require iteratively adjusting the decisions towards the optimum, which is fundamentally infeasible for real-time optimization under fast-moving vehicular environments. Model-

[†]It is worth noting that our scheme is also generic and applicable in the non-orthogonal channel access system, thus can be integrated with some practical interference models

free deep reinforcement learning (DRL), is envisioned as a promising paradigm to learn the optimal offloading strategy through continually interacting with the stochastic vehicular environments without any prior environment knowledge [17]. Moreover, centralized DRL algorithms are inapplicable when the action space grows exponentially with the number of ICVs and fully distributed DRL algorithms are also difficult to converge due to the communication problems among multiple agents. To solve this problem, a CTDE-based multi-agent DRL (MADRL) algorithm will be invoked in the next section to find the long-term optimal solution from the perspective of reducing the action and state space.

III. Learning-Based Computation Offloading Algorithm

Based on a modeled Markov decision process (MDP) defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ [18], the goal of the DRL agent is to derive an optimal policy π^* that maximizes long-term expected return by searching the optimal state-action values (Q-values) $Q^*(s,a)$, which is defined by $Q^*(s_\tau, a_\tau) = \max\limits_{\pi} \mathbb{E}\left[r(s_\tau, a_\tau) + \sum_{k\in\mathcal{K}} \gamma^k r_{\tau+1+k} | s_\tau, a_\tau, \pi\right]$, where $\gamma \in [0,1]$ is the discount factor depicting the balancing between immediate reward and future rewards.

The optimal Q-value function is derived by minimizing the temporal difference between the target Q-value and the current Q-value, i.e.,

$$\sigma = r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta), \tag{10}$$

where target Q-value $y(s,a) = r + \gamma \max\limits_{a'} Q(s', a'; \theta^-)$ is the approximate objective of the current Q-value. The loss function of the neural network is decribed as $J(\theta) = 0.5 \sum \sigma^2$.

A. CTDE-based Learning Architecture

CTDE, a promising hybrid multi-agent RL framework, can utilize the global state information to train agents in a centralized manner, and then the agents can use the local observation to execute in a decentralized manner [19]. However, the main challenge of CTDE is how to extract distributed strategies after centralized training. The value-decomposition networks (VDNs) algorithm is introduced to solve this challenge [12], and a hybrid distributed learning-based offloading algorithm is designed.

We model the formulated problem above as a cooperative MADRL task where each ICV acts as an agent, and actions of agents can be coordinated to achieve a shared team reward together. Then the task can be modeled as a Dec-POMDP, which can be defined as a Markov game with incomplete information and denoted by a tuple $\langle \mathcal{J}, \mathcal{S}, O, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. The environment state $s_t \in \mathcal{S}$ can not be fully observed in the POMDP scenarios, agent j can only get an observation $o_t^j \in O$ which is extracted from the state function $s_t \in \mathcal{S}$.

Independent Q-learning (IQL) cannot cope with the severe environment non-stationarity problem caused by the changing policies of the multiple agents [12]. Thus, CTDE is designed to learn a joint action-value function $Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t)$ with global state information in the process of centralized training, where $\boldsymbol{\tau}_t = (\boldsymbol{\tau}_t^1, \boldsymbol{\tau}_t^2, ..., \boldsymbol{\tau}_t^J)$ denotes a turple of agent histories, and $\boldsymbol{\tau}_t^j = a_1 o_1 r_1, ..., a_{t-1} o_{t-1} r_{t-1}$ is a joint action-observation history for agent j, and $\boldsymbol{a}_t = (a_t^1, a_t^2, ..., a_t^J)$ is a joint action for all the agents. Then, by employing the learned $Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t)$, the distributed agents can choose more reasonable independent actions based on local observations only compared with fully distributed algorithms.

VDNs were proposed to embed the learned centralized $Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t)$ into the distributed execution process. Specifically, VDNs can decompose $Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t)$ into individual Q-value function $Q_j(\boldsymbol{\tau}_t^j, \boldsymbol{a}_t^j)$, $j \in \mathcal{J}$, i.e.,

$$Q_{tot}(\boldsymbol{\tau}_t, \boldsymbol{a}_t) \approx \sum\nolimits_{j\in\mathcal{J}} Q_j(\boldsymbol{\tau}_t^j, \boldsymbol{a}_t^j; \theta_t^j), \tag{11}$$

where θ_t^j is the neural Q-network parameter of agent j.

It is worth mentioning that the strict coupling relationship for VDNs should be $Q_{tot}(\boldsymbol{s}_t, \boldsymbol{a}_t) \approx \sum_{j\in\mathcal{J}} Q_j(\boldsymbol{s}_t^j, \boldsymbol{a}_t^j; \theta_t^j)$. However, the state function s_t is unavailable for agents in Dec-POMDP, and thus o_t^j and a_t^j are not sufficient to model acurate $Q_j(\boldsymbol{s}_t^j, \boldsymbol{a}_t^j; \theta_t^j)$. To solve this issue, we store historical observations $\boldsymbol{\tau}_t^j$ in a long short term memory network as additional valuable information to enhance the Q-function model for agents.

We denote the target Q-value of agent j by Q_j^-. Similarly, $Q_{tot}^-(\boldsymbol{\tau}_t, \boldsymbol{a}_t) \approx \sum_{j\in\mathcal{J}} Q_j^-(\boldsymbol{\tau}_t^j, \boldsymbol{a}_t^j; \theta_t^j)$, so the time difference target can be derived as $y_{tot}^n = r^n + \gamma \max\limits_{a'} \bar{Q}_{tot}^-(\boldsymbol{\tau}', a')$. Then the cost function of Q_{tot} is shown as

$$L(\theta) = \frac{1}{N} \sum_{n=1}^{N} (y_{tot}^n - Q_{tot}^n(\boldsymbol{\tau}^n, \boldsymbol{a}^n))^2, \tag{12}$$

where N is the mini-batch size of the Q-network.

By updating the gradient of $L(\theta)$ with backpropagation, $Q_j(\boldsymbol{\tau}^j, \boldsymbol{a}^j)$, $j \in \mathcal{J}$ can be updated implicitly rather than from any reward signal from agent j. To reduce the training complexity, the agent networks are allowed to share weights.

B. Problem Transformation

We formulate our offloading decision problem as a Dec-POMDP. The details of the formulation are as follows.

Local Observation O: The local observation of agent $j \in \mathcal{J}$ includes five parts, i.e., the coordinate of ICV j, the raw task size of ICV j, the task computation resource requirement of ICV j, the local computing latency of ICV j, and the offloading gain of ICV j at last decision epoch. For simplicity, we ignore the index t in the first four parts. Thus the local observation is as follows

$$O_j(t) = [y_{j^h}^{\mathrm{I}}, \lambda_{j^h}, \lambda_{j^h}\rho_{j^h}, D_{j^h}^{\mathrm{lc}}, \varpi_{j^h}(t-1)]. \tag{13}$$

Action Space $\mathcal{A}$: The action space of agent j is defined as the candidate offloading infrastructure set, i.e.,

$$A_j(t) = [\psi_{j^h}^j, \psi_{j^h}^1, ..., \psi_{j^h}^h, ..., \psi_{j^h}^H, \psi_{j^h}^M]. \tag{14}$$

Due to the vehicular movement and randomly available computation resources of the MBS, the candidate offloading infrastructure set is dynamic in different decision epochs for different ICVs.

Reward $\mathcal{R}$: In this cooperative task, all the agents share the team reward, and we set a negative penalty when $\delta_{j^h}(t) < 0$. Then we construct the reward based on the objective function and constraints to shape the training process, i.e.,

$$R(t) = -\sum_{\forall j \in \mathcal{J}} D_{j^h}(t). \tag{15}$$

IV. Simulation Results

We consider a two-way road network where $J = 18$ and $H = 3$. The ARS hovered in the air at an altitude of 200 m. The radius of RSUs is 200 m. The bandwidth of mmWave and sub-6G is 200 MHz and 20 MHz, respectively. We set $\sigma^{\text{sf}} = 3.5$ dB, and $p_0 = -65$ dB. The transmission power of ICVs and the ARS is 20 dBm. We set $\mu = 60$ and $\sigma = 21^{0.5}$ for the truncated Gaussian probability distribution. The computation capacities F^{I}, F^{R} and F^{M} are set as 0.5 GHz, 6 GHz, and 10 GHz, respectively. The Poisson factor is 3. The task bit size $\lambda_{j^h} \in \{1, 1.5, 2\}$ Mbits, and the task computation density $\rho_{j^h} \in \{50, 100, 150\}$ CPU cycles/bit. We set $N^{\text{R}} = 4$. To reduce the action space and simplify the simulation complexity, we choose N^{R} ICVs with largest computation resource requirement to offload to the belonging RSU when $\eta_h(t) > N^{\text{R}}$. The RMSProp optimizer is applied for updating the neural networks. Table I list the hyper-parameters of the MADRL. To demonstrate the results distinctly, we handle the original results data with a moving average model.

TABLE I
Training Hyper-parameters

Parameter	Value	Parameter	Value
Discount factor	0.9	Learning rate	0.0003
Exploration factor	1/0.05	Target update interval	100
Buffer size	2600	Batch size	64
Neuron network layer number	2	Neuron number	64/32

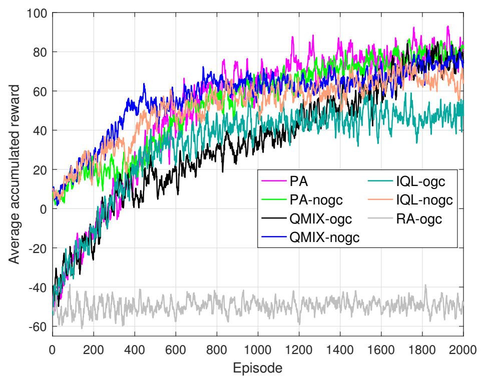

Fig. 2. The average accumulated reward comparision.

Fig. 2 shows the convergence performance of the proposed algorithm (PA) and six baseline approaches. The PA-nogc scheme in Fig. 2 is the proposed algorithm without considering the positive offloading gain constraints. The QMIX-ogc and QMIX-nogc are QMIX-based schemes considering and without considering the positive offloading gain constraints, respectively [20]. The IQL-ogc and the IQL-nogc are IQL-based schemes considering and without considering the positive offloading gain constraints, respectively. The simulation results illustrate that the PA, PA-nogc, QMIX-ogc and QMIX-nogc schemes have a similar convergent reward value around 80, and the convergent reward values of all the four schemes are higher than the two IQL-based schemes. However, from Fig. 3, we can see the convergent average system delay (around 2400 ms) for the PA scheme is the lowest. What's more, by performing multiple simulation comparisons for the QMIX-based schemes, we find that the learning process of QMIX-based schemes is unstable, which means that the variance of accumulated rewards is large. The authors in [20] also point out the VDN gets a little bit better and more stable performance than QMIX when the Q-value weight of each agent is approximately equal, which is applicable in our multi-agent systems. As for the two IQL-based approaches, the environment non-stationarity caused by independent updating policies of the multiple agents can deteriorate the offloading decision learning for IQL-based approaches.

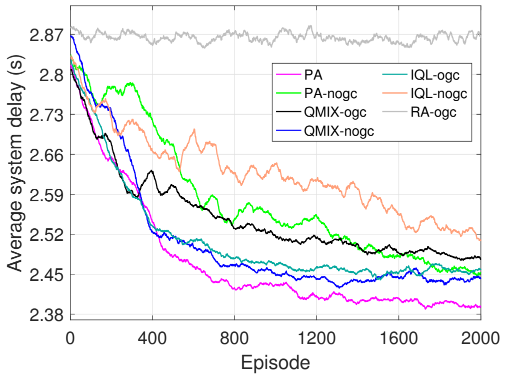

Fig. 3. The average system delay comparison.

The two schemes without considering the offloading gain constraints, including PA-nogc and QMIX-nogc obtain similar convergent rewards with PA and QMIX-ogc. The reason is that when there exist negative offloading gains, the PA and QMIX-ogc schemes send agents a penalty while the nogc schemes send agents a positive reward, so the accumulated reward values of (PA/QMIX-ogc) and (PA-nogc/QMIX-nogc) are similar. Besides, the agents obtain many penalties at the beginning of the training process when the offloading gain constraints are violated. Thus the start points of the curves for PA-nogc, QMIX-nogc and IQL-nogc schemes are both higher than PA, QMIX-ogc and IQL-ogc schemes. The ICVs offloaded to the edge with negative offloading gains can get higher unreasonable offloading latency, and thus edge computing resources are wasted. Consequently, other ICVs with positive offloading gains also get higher offloading latency.

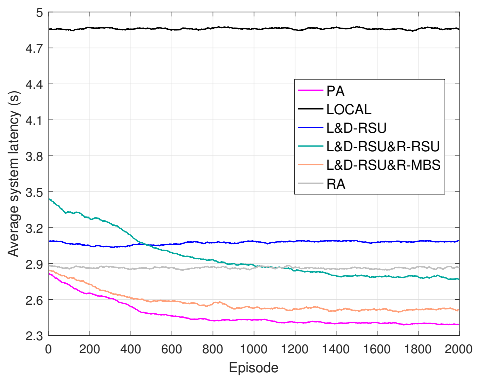

Fig. 4. The average system delay comparison for different offloading modes.

To sum up, we can derive that the proposed offloading gain constraints in our problem formulation can help agents make better decisions and relieve the edge resource waste.

In Fig. 4, we perform the average system latency comparison of six offloading schemes. The average system latency decreases with episodes at the beginning of the learning process for PA, and we can observe that the curve of the proposed approach can be stable after around 800 episodes. The proposed approach obtains better performance than the local computing scheme (LOCAL), local and direct RSU computing scheme (L&D-RSU), local, direct RSU and relay RSU computing scheme (L&D-RSU&R-RSU), as well as local, direct RSU and relay MBS computing scheme (L&D-RSU&R-MBS). Besides, the performance of the LOCAL, L&D-RSU and L&D-RSU&R-RSU are all worse than L&D-RSU&R-MBS. The reason behind this observation is that the proposed approach can relay the computation workload of ICVs in the range of RSUs with heavy workloads to the empty or idle RSUs, and the MBS with randomly available computation capacity. Although the occupied computation resource of the MBS is random and modeled as an independent identically distributed (i.i.d.) Poisson process, the available resource of the MBS is still higher than the RSUs in the long run.

V. Conclusion

We propose a heterogeneous edge computing architecture for the IoVs, where the ARS can establish two relay connections to access the idle computing resources of nearby RSUs and unconnectable MBS. We minimize the system computing latency of all the ICVs under some practical constraints and design the solution based on a CTDE-based MADRL algorithm to learn the load-aware computation offloading scheduling. The simulation results confirm that the system latency is greatly affected by the computation offloading scheduling, and thus the idle computing resources of nearby RSUs and unconnectable MBS can be fully utilized.

References

[1] Z. Lv, D. Chen, and Q. Wang, "Diversified technologies in internet of vehicles under intelligent edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2048–2059, Apr. 2021.

[2] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 740–759, Feb. 2022.

[3] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Commun. Mag.*, vol. 27, no. 4, pp. 126–132, Aug. 2020.

[4] Y. He, Y. Wang, Q. Lin, and J. Li, "Meta-hierarchical reinforcement learning (MHRL)-based dynamic resource allocation for dynamic vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 4, pp. 3495–3506, Apr. 2022.

[5] L. Liu, Z. Zhang, G. Chen, and H. Zhang, "Resource management of heterogeneous cellular networks with hybrid energy supplies: A multi-objective optimization approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 7, pp. 4392–4405, Jul. 2021.

[6] L. Liu, Z. Zhang, N. Wang, H. Zhang, and Y. Zhang, "Online resource management of heterogeneous cellular networks powered by grid-connected smart micro grids," *IEEE Trans. Wireless Commun.*, pp. 1–15, Apr. 2022.

[7] Q. Ren, J. Chen, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and F. R. Yu, "An application-driven nonorthogonal-multiple-access-enabled computation offloading scheme," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1453–1466, Feb. 2021.

[8] P. Lin, Q. Song, D. Wang, F. R. Yu, L. Guo, and V. C. M. Leung, "Resource management for pervasive-edge-computing-assisted wireless VR streaming in industrial internet of things," *IEEE Trans. Industr. Inform.*, vol. 17, no. 11, pp. 7607–7617, Nov. 2021.

[9] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[10] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.

[11] J. S. Ng, W. Y. B. Lim, Z. Xiong, D. Niyato, C. Leung, and C. Miao, "A double auction mechanism for resource allocation in coded vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 1832–1845, Feb. 2022.

[12] Q. Ren, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and J. Chen, "Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems," *IEEE Trans. Wireless Commun.*, pp. 1–1, May 2022.

[13] Y. Liu, J. Zhou, D. Tian, Z. Sheng, X. Duan, G. Qu, and V. C. M. Leung, "Joint communication and computation resource scheduling of a UAV-assisted mobile edge computing system for platooning vehicles," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–16, Jun. 2021.

[14] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[15] Y. Zhang, Y. Huo, D. Wang, X. Dong, and X. You, "Channel estimation and hybrid precoding for distributed phased arrays based MIMO wireless communications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12 921–12 937, Nov. 2020.

[16] M. Sana, A. De Domenico, W. Yu, Y. Lostanlen, and E. Calvanese Strinati, "Multi-agent reinforcement learning for adaptive user association in dynamic mmwave networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 10, pp. 6520–6534, Oct. 2020.

[17] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless VR-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15 749–15 761, Nov. 2021.

[18] D. Wang, B. Li, B. Song, Y. Liu, K. Muhammad, and X. Zhou, "Dual-driven resource management for sustainable computing in the blockchain-supported digital twin iot," *IEEE Internet of Things Journal*, pp. 1–1, Apr. 2022.

[19] Y. He, Y. Wang, F. R. Yu, Q. Lin, J. Li, and V. C. M. Leung, "Efficient resource allocation for multi-beam satellite-terrestrial vehicular networks: A multi-agent actor-critic method with attention mechanism," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2727–2738, Mar. 2022.

[20] Y. Yang, J. Hao, B. Liao, K. Shao, G. Chen, W. Liu, and H. Tang, "Qatten: A general framework for cooperative multiagent reinforcement learning," *arXiv preprint arXiv:2002.03939*, 2020.