Multi-Agent Deep Reinforcement Learning-Based Cooperative Perception and Computation in VEC

Liang Zhao, *Member, IEEE,* Longjia Li, Zhiyuan Tan, *Senior Member, IEEE,* Ammar Hawbani, Qiang He, *Member, IEEE,* and Zhi Liu, *Senior Member, IEEE*

***Abstract*—Connected and autonomous vehicles (CAVs) are an important paradigm of intelligent transportation systems. Cooperative perception (CP) and vehicular edge computing (VEC) enhance CAVs' perception capacity of the region of interest (RoI) while alleviating the pressure of intensive computation on onboard resources. However, existing CP and computation schemes are based on inefficient broadcast communications and still face challenges such as highly dynamic communication link channel conditions caused by vehicle mobility, and limited computing resources in VEC environments. Considering the delay sensitivity of CAVs' perception tasks and the need for enhanced perception, we propose a unicast-based cooperative perception and computation scheme to achieve more efficient resource utilization and perception task execution in VEC scenarios. Our goal is to maximize CP gain and minimize task execution delay by optimizing the decision of each ego CAVs. To solve the sequential decision-making problem of multi-objective optimization, we propose a solution based on improved multi-agent proximal policy optimization deep reinforcement learning, where CAVs agents make adaptive decisions distributed based on partial observations. Simulation results show that compared with the baseline algorithm, our proposed scheme effectively reduces the execution delay of ego CAVs perception tasks and ensures a high perception gain.**

***Index Terms*—Connected and autonomous vehicles, Cooperative perception, Vehicular edge computing, Multi-agent deep reinforcement learning.**

I. Introduction

With the rapid advancement of the Internet of vehicles and artificial intelligence, connected and autonomous vehicles (CAVs) have attracted increasing attention as a key component of intelligent transportation systems [1]–[4]. To achieve safe and reliable autonomous driving, CAVs must capture comprehensive environmental information through various onboard sensors and process these sensor data in real time to perceive the surrounding environment, including obstacles and traffic participants [5]. However, a single vehicle can experience observation blind spots or insufficient perception accuracy due to line-of-sight (LoS) occlusion and the sparse observation characteristics of the sensors at long distances [6]. Moreover, relying solely on the limited computing resources of CAVs, makes it challenging to complete various computationally intensive and delay-sensitive tasks, including perception tasks [7]–[9]. As promising solutions, cooperative perception (CP) and vehicular edge computing (VEC) are proposed to address the aforementioned issues. Specifically, using vehicle-to-everything (V2X) wireless communication technology [10], perception information sharing and distributed computing can be achieved between CAVs, as well as between CAVs and road infrastructure. This integration not only provides CAVs with accurate and real-time situational awareness, but is also expected to improve traffic efficiency and safety.

CP enables the exchange of perceived environmental information among CAVs and roadside units (RSUs), thereby compensating for the limitations of single-vehicle perception system. Depending on the type of shared information, CP is divided into early fusion of shared raw sensor data [11], intermediate fusion of shared data characteristics [12], [13], and late fusion of shared perception results [14], [15]. Among them, early fusion can obtain the most detailed environment sensing data and is robust to timestamp misalignment, thus achieving the best perception performance [16], but there are still many challenges. Firstly, the communication link between CAVs and other nodes is unstable due to vehicle mobility, which can cause significant transmission delays [17]. Secondly, the spectrum efficiency of broadcast-based collaborative mechanism is low, making it challenging to scale in dense traffic and limited bandwidth scenarios. Sidelink unicast, supported as a new feature in 5G new radio (NR) V2X, helps stabilize the connection and improve spectrum utilization [18]. Therefore, it is necessary to evaluate the cooperative weight of the cooperative nodes and collaborate with the node that offers greater benefits to its own perception [19], thereby achieving an efficient unicast-based CP. In addition, existing studies on CP primarily aim to improve the detection accuracy of a single CAV, while ignoring the time-varying computing resources. High computational loads faced by CAVs make it challenging to meet the strict delay constraints of real-time perception tasks [20]. To solve this problem, it is necessary to select appropriate communication node locations for sensing data. The task offloading technology of VEC is expected to make full use of the computing resources at the network edge.

VEC provides context-aware storage and distributed computing at the network edge close to the sensors of CAVs and RSUs, thus offering solutions to task execution delay and communication bandwidth limitations [21]. Specifically, CAVs offload computational tasks to nearby CAVs or edge servers

Liang Zhao, Longjia Li, and Ammar Hawbani are with the School of Computer Science, Shenyang Aerospace University, Shenyang 110136, China (e-mail: lzhao@sau.edu.cn; lilongjia@stu.sau.edu.cn; anmande@ustc.edu.cn).

Zhiyuan Tan is with the School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, U.K. (e-mail: z.tan@napier.ac.uk).

Qiang He is with the School of Medicine and Biological Information Engineering, Northeastern University, Shenyang 110169, China (e-mail: heqiang@bmie.neu.edu.cn).

Zhi Liu is with the Department of Computer and Network Engineering, The University of Electro-Communications, Tokyo, Japan (e-mail: liu@ieee.org).

Qiang He and Ammar Hawbani are the corresponding authors.

with abundant computing resources for processing, thereby reducing the execution delay and energy consumption of tasks [22]. However, in task offloading, computational tasks generated by vehicles are usually considered independent, ignoring the collaborative characteristics of CP tasks. For CP tasks, sensor data from multiple sources must be fused before model inference, which increases the complexity of task offloading. Therefore, it is essential to study efficient cooperation and task offloading to enable CAVs to take advantage of the powerful perception and computing capabilities of edge service nodes. Furthermore, due to the dynamic changes of VEC networks and the complexity and diversity of traffic scenarios, deep reinforcement learning (DRL) has emerged as an effective and feasible solution algorithm for solving task offloading in dynamic environments [23]. However, existing studies typically deploy DRL algorithms in RSUs to implement centralized task offloading strategies, requiring continuous network connection to obtain vehicle status in real time, which can limit the scalability of vehicle network systems [10]. Multi-agent deep reinforcement learning (MADRL) has been applied to vehicle applications as an emerging distributed solution [24], [25], but its application in CP tasks is still in its early stages.

Considering the complex association of CP tasks in VECs and the limited communication and computing resources, a unicast-based cooperative perception and computation scheme is proposed, aiming to achieve a lower delay in the execution of the perception task and a higher gain in perception of objects in the region of interest (RoI). First, to enable unicast-based cooperation, CAVs acquire the basic information of other CAVs and RSUs within the communication range by broadcasting lightweight cooperative awareness messages (CAMs), and then use indicators such as cooperative weight as edge service node selection criteria. Second, considering the complex dynamic interactions between CAVs and edge service nodes, a solution based on an improved MADRL algorithm is used to optimize the offloading strategy of CP tasks and enhance the scalability of the system. Finally, to further stabilize the offloading decisions generated by the CAVs agent using a distributed policy network, the loss function is improved. The weights are also adjusted to adapt to different traffic environments according to varying requirements of task execution delay and perception gain. The main contributions of this paper are summarized as follows:

- To fully utilize the computing resources in the VEC environment, an infrastructure-assisted cooperative perception and computation scheme is proposed, combining vehicle-to-infrastructure and vehicle-to-vehicle task offloading frameworks. To implement the unicast-based cooperation mechanism, CAMs are used to evaluate the cooperative weight of edge service nodes and assist in selecting appropriate cooperation nodes, with the intersection degree between the interest area and the node perception range used to calculate the cooperative weight.
- The multi-objective optimization problem of joint task execution delay and CP gain for multiple CAVs is formulated as a Markov decision process. The cooperative weight evaluation, computational load, and channel conditions of edge service nodes are comprehensively considered to minimize delay and maximize CP gain.
- For complex state spaces, an improved multi-agent proximal policy optimization algorithm is proposed, using a loss function with stricter Kullback-Leibler (KL) divergence constraints to obtain more stable strategies. Simulation experiments in multiple scenarios verify that the algorithm can provide CAVs with a higher CP gain and a lower system latency.

The subsequent sections of this paper are arranged as follows. Section II introduces the related work. Section III delineates the system model and problem formulation. Section IV describes the details of the cooperative perception and computation solution based on MADRL. Simulation results and discussion are provided in section V, and conclusions are given in section VI.

II. Related Work

In this section, we review the research on CP and task offloading based on V2X wireless communication technology and summarize the shortcomings. A detailed comparison of these works from various aspects is presented in Table I.

A. Cooperative Perception

To address single CAV viewpoint occlusion and sparse observation data, researchers have extensively studied V2X-based CP and designed CP algorithms from different fusion levels to improve object perception accuracy and system robustness. Late fusion [26] involves sharing detection results, including the 3D bounding box and category of objects. Mao Shan et al. [14] consider the cross-correlation between the perception results and propose a trajectory-to-trajectory fusion method based on the covariance intersection (CI) algorithm. A CP framework based on vehicle-to-vehicle (V2V) communication is proposed in [27] to share the vehicle's own state and the trajectory of other objects detected by the sensor. These late fusion schemes are more sensitive to errors in positioning and objects position estimation [15], and may introduce large errors even with complex fusion algorithms. Intermediate fusion shares the extracted data features. The V2VNet proposed by Tsun-Hsuan Wang et al. [28] uses a spatial perception graph neural network to aggregate data intermediate representations from surrounding CAVs. Following a similar transmission paradigm, a delay compensation module is proposed in [29] to achieve feature synchronization to mitigate the impact of communication transmission delays, thereby improving the robustness of CP. Runsheng Xu et al. propose CP frameworks using only V2V in [30] and combined with vehicle-to-infrastructure (V2I) in [31], both of which employ multi-scale attention modules to capture local and global spatial feature relationships between different agents or views.

B. Task Offloading

In VEC, through V2V and V2I task offloading frameworks, CAVs can utilize the resources of edge servers nodes

TABLE I
COMPARISON OF SOLUTIONS IN RELATED WORK

Reference	Algorithm	Offloading	Considered Factors				Optimization Objective
			CP	V2V	V2I	Distributed	
[14]	CI		✓	✓		✓	Minimize uncertainty of position estimates
[27]	IMM		✓	✓		✓	Minimize errors in the estimation of the motion states of vehicles
[28]	V2VNet		✓	✓		✓	Maximize the perception accuracy of CAVs
[29]	SyncNet		✓	✓		✓	Maximize the perception accuracy of CAVs
[30]	CoBEVT		✓	✓		✓	Maximize the perception accuracy of CAVs
[31]	V2X-ViT		✓	✓	✓	✓	Maximize the perception accuracy of CAVs
[32]	GK-DDPG	Partial		✓	✓		Minimize delay and service price.
[33]	DTP-DIESEL	Partial		✓	✓		Minimize execution delay and energy consumption
[35]	LSTM-DDQN	Binary			✓		Minimize the ratio of dropped tasks and average delay.
[36]	PPO	Binary	✓	✓	✓		Minimize total priority of the discarded tasks
[37]	TDGC		✓		✓		Maximize the detection range and perception accuracy of CAVs
[38]	MADDPG	Binary	✓	✓		✓	Maximize computing efficiency gain
[34]	MAPPO	Binary				✓	Minimize delay and rental price
Our work	IMAPPO	Binary	✓	✓	✓	✓	Minimize execution delay and Maximize CP gain

to perform computational tasks. Collaborating with nodes that have sufficient computing resources and better views to perform perception tasks can reduce response delays and improve detection accuracy. Zhao et al. [32] propose a digital twin-assisted VEC task offloading framework, which jointly optimizes delay and service prices. To adapt to the dynamic changes of network links, [33] predicts the communication distance of CAVs based on their historical trajectories before offloading, reduces the decision space of V2V offloading, and uses heuristic algorithms to optimize system energy consumption and delay. S. S. Hassan et al. [34] offloaded the tasks of intelligent transportation systems nodes to a satellite network that supports edge computing, jointly optimized delay and rental price, and applied MADRL to distributed decide task offloading. These works show excellent results in reducing delay and energy consumption, but they cannot be directly applied to CP tasks, and arbitrary division of tasks is unrealistic. In our work, we formulate a binary task offloading problem [35] to optimize the multi-objective problem of task execution delay and cooperative perception gain.

C. Cooperative Perception Assisted by VEC

Recent studies have attempted to apply task offloading to CP to unleash the potential of VEC. Pin Lv et al. [36] propose a centralized task offloading scheme that defines the perception task weight by the number of overlaps between the perception range and the surrounding vehicle RoI. However, the optimization objective of maximizing the task weight completed by the server is difficult to reflect the perception gain. To solve the cooperative task allocation and enable ego CAV to achieve perception within the RoI, a joint cooperative task allocation and offloading optimization problem is formulated in [39], and a two-layer binary intelligent firefly heuristic method is used to minimize the task execution delay. In [37], an edge-assisted multi-vehicle CP system called EdgeCooper is proposed. This system centrally fuses light detection and ranging (LiDAR) point cloud data at the edge server based on voxel units in PointPillars, and then performs object detection on the merged view. In [40], an attention mechanism is introduced in the offloading task to dynamically focus on relevant information from the input state and joint actions, so that CAVs can make wise decisions while considering the behavior of other agents. [38] uses MADRL to determine when each CAV pair switches between cooperative and standalone perception, and uses an auxiliary model to solve the cooperative CAV pair communication and computing resource allocation subproblems to maximize the computing efficiency gain under delay constraints.

D. Summary

In reviewing the existing research, we can summarize the shortcomings of CP and task offloading. Firstly, late fusion has limited perception gain and may fail to detect objects even after fusion. The data features extracted by different CAVs in intermediate fusion may lack universality. In contrast, early fusion not only improves accuracy but also allows CAVs to perform distributed fusion and computation of raw sensor data. Secondly, most existing works overlook the limitations of communication and computing resources, and the broadcast cooperation mechanism poses significant challenges to existing V2X networks. Thirdly, centralized scheme places more stringent requirements on the network, as CAVs needs to maintain a constant network connection to transmit data. Processing the perception task entirely on the infrastructure also wastes the CAV's own resources. Our approach uses unicast to transmit the raw point cloud and employs an improved multi-agent proximal policy optimization algorithm to enable the CAV to make decentralized offloading decisions based on local observations.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we first outline the system scenario, then introduce the cooperative perception task and gain, communication model, and edge computing model in detail. Finally, we formulate the joint optimization problem of cooperative perception gain and task execution delay. For clarity, a summary of the primary notations used throughout this paper is provided in Table II.

A. System Overview

We consider the intersection traffic scenario involving infrastructure-assisted perception and computation for CAVs.

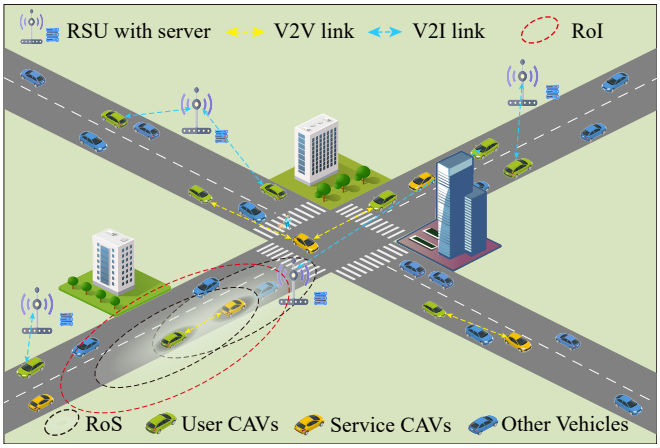

Fig. 1. Overview of the scenario.

As shown in Fig. 1, the CAVs and RSUs establish communication connections using NR V2X unicast links. There are N user CAVs, represented by $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, and the user CAVs need to offload perception tasks to detect the objects within their RoI. The service CAVs and RSUs with sufficient idle resources act as edge service nodes to provide cooperative perception and computation support for the user CAVs. The set of M edge service nodes is represented by $ES = \{es_1, es_2, \ldots, es_M\}$. In each time slot $t \in \mathcal{T} = \{1, 2, \ldots, T\}$, each user CAV makes decentralized offloading decisions based on the real-time road status. Specifically, at the start of each time slot t, the communication nodes exchange essential information, including location, vehicle status, and RSU status, via lightweight CAMs. When a user CAV decides to compute the perception task locally, it operates in standalone perception (SP) mode. Conversely, if it decides to offload the perception task, the user CAV transmits its sensing data to an edge service node, which can be either a static RSU or a mobile service CAV, and switches to CP mode. The edge service node fuses and computes the data offloaded by the user CAVs, and then returns the enhanced detection results to the user CAVs. Since CAVs pay more attention to objects around them, our scheme focuses on the cooperative perception gain within the RoI range of each CAV.

B. Cooperative Perception Task Model

LiDAR point clouds offer the advantage of depth information and is easy to fuse after coordinate transformation, we opt for early CP based on point clouds. In time slot t, CAVs and RSUs utilize their sensors to scan and generate point cloud data $P_k = (x_k^i, y_k^i, z_k^i, r_k^i)$, which represent the coordinates and intensity values of the reflection points respectively. Upon offloading to the edge service node, the user CAV's point cloud undergoes transformation through the rotation matrix R, $R = R_z(\alpha)R_y(\beta)R_x(\theta)$, where $R_z(\alpha)$, $R_y(\beta)$, $R_x(\theta)$ are the basic rotation matrices in the three dimensions of z, y and x axis respectively. α, β and θ are the differences between the yaw angle, pitch angle and roll angle of the sender node n and the receiver node m respectively. The PointPillars detection pipeline is used for target detection. The perception task of user CAVs n is defined as $ST_t^{(n)} = \{D_n, C_n, T_n\}$. In the SP mode, D_n denotes the size of the single frame data, and in the CP mode, it denotes the size of the fused data. C_n denotes the CPU cycles required to complete the task computation, calculated as $C_n = D_n \cdot q_n$, where q_n denotes the computational density (unit: CPU cycle/bit). T_n denotes the maximum tolerable delay of the perception task.

TABLE II
SUMMARY OF NOTATIONS

Notation	Description
n, N	Index and number of user CAVs
m, M	Index and number of edge service nodes
t, T	Index and number of time slots
ST_n	Perception task of user CAV_n
P_k	Point cloud data generated by node k
D_n	Data size of user CAV_n's perception task
C_n	CPU cycles required for the perception task
T_n	Maximum tolerable delay of the perception task
q_n	Computational density of the perception task
r_S	Benefit of standalone perception
r_I	Benefit of cooperative perception
$\mathcal{G}_n$	Cooperative perception gain for user CAV_n
R_{nm}	Transmission rate between nodes n and m
B_{nm}	Wireless channel bandwidth between nodes n and m
Loc_k	Position of communication node k
d_{nm}	Distance between nodes n and m
T_n^{loc}	Local execution delay of the perception task for user CAV_n
f_n	Computational capacity of CAV_n
f_m	Computational capacity of edge service node m
T_{nm}^{up}	Transmission delay of perception data
T_{nm}^{fu}	Fusion delay of perception data
T_{nm}^{edge}	Edge computing delay of the perception task
T_{nm}^{down}	Result return delay
$\mathcal{D}_n^{ego}$	Execution delay of the perception task for user CAV_n
ω_1	Weight of cooperative perception gain
ω_2	Weight of task execution delay
η	Penalty term associated with T_n
S	State space
A	Action space
r_n	Reward function for user CAV_n
$R(t)$	Reward of user CAV at time t

Each user CAV maintains two sets of objects O_S and O_I, where O_S contains objects in its region of sense (RoS) and O_I contains objects in its region of interest (RoI), and $O_S \subseteq O_I$. For simplicity, the CAV's concerned RoI is assumed to be a circular area with a radius R_{RoI} , and the sensing range RoS's radius is R_{RoS}. The detection result for the object $o_j \in O_I$ is expressed as $\Phi(o_j) \in \{0, 1\}$, $\Phi(o_j) = 1$ indicating that the object o_j is accurately located and classified, otherwise $\Phi(o_j) = 0$. Due to the varying importance of objects to the user CAVs, the benefit of SP considering the objects importance weight is calculated by (1), where $\omega_j^{(t)}$ is the importance weight of object j at time slot t, which is related to the distance from the user CAV according to [19]. Objects that lie outside the RoS but within the RoI are detected by

performing CP, and the benefit of CP is calculated by (2).

$$r_S(t) = \sum_{j \in O_S} \omega_j^{(t)} \Phi(o_j) \tag{1}$$

$$r_I(t) = \sum_{j \in O_I} \omega_j^{(t)} \Phi(o_j) \tag{2}$$

The benefit gain from CP is contingent upon the cooperating nodes, and the cooperative perception gain of user CAV_n is calculated as (3).

$$\mathcal{G}_n^{ego}(t) = r_I(t) - r_S(t) = \sum_{j \in O_I - O_S} \omega_j^{(t)} \Phi(o_j) \tag{3}$$

In summary, the cooperative perception gain is the cumulative weighted benefit of objects detected through CP. This gain is obtained when the detection process is completed within a single time slot.

C. Communication Model

User CAVs make distributed offloading decisions from the perspective of ego. First, the location $Loc_m(t) = [x_m(t), y_m(t), z_m(t)]^{\mathrm{T}}$ of the service CAVs and RSUs within the communication range is identified through the CAMs. Given that CAMs are lightweight messages, their communication overhead is negligible. However, for point clouds with a frame size of 200 KB to several MB, broadcast-based sharing will result in a data transmission volume that is $(N+M)(N+M-1)$ times larger. The delay of broadcast exceeds 100 ms under the existing communication bandwidth, while unicast can selectively establish links based on channel quality, improving bandwidth efficiency. Fluctuations in channel conditions, influenced by distance and occlusion, dictate the transmission rate between communication nodes. NR V2X sidelink communication between nodes employs orthogonal frequency division multiple access (OFDMA). This advanced spectrum resource allocation technology minimizes the interference between wireless channels. Based on Shannon's theorem, the data transmission rate from node n to node m at time slot t is given by (4), where B_{nm} denotes the bandwidth allocated to the channel between the sender node n and the receiver node m, P_n denotes the transmission power of the uplink of the sender node n, h_{nm} is the small scale fading coefficient of the Rayleigh distribution, $h_{nm} \sim c\mathcal{N}(0,1)$, d_{nm} denotes the distance between sender and receiver, γ is the path loss exponent, and σ^2 is the noise power spectral density at the receiver node.

$$R_{nm}(t) = B_{nm}(t)\log_2\left(1 + \frac{P_n |h_{nm}|^2(t) d_{nm}^{-\gamma}(t)}{\sigma^2}\right) \tag{4}$$

Assuming that the system remains stable within the t time slot, the distance at time t is expressed as (5), where $Loc_n(t)$ is the location of user CAV_n.

$$d_{nm}(t) = \sqrt{|Loc_n(t) - Loc_m(t)|^2} \tag{5}$$

Beyond LoS and non-LoS (NLoS) channel states, we have also incorporated the non-LoS vehicle (NLoSv) state in accordance with the 3GPP NR V2X 37.885 standard [41]. This state describes a situation where the direct path is obstructed by vehicles, resulting in additional path blocking loss compared to the LoS state.

D. Edge Computing Model

To elucidate the edge computing model, we initially delineate the attributes of user CAV and edge service node. User CAV is represented as $v_n = \{ST_n(t), f_n(t), Loc_n(t), f_n^v\}$, which denote the perception task, the computational capacity at time t, location at time t and the maximum computational capacity of user CAV_n respectively. Similarly, edge service node is represented as $es_m = \{ST_m(t), f_m(t), Loc_m(t), f_m^{es}\}$. The trained neural network models are deployed on each communication node to detect the categories and 3D bounding boxes of the objects from single viewpoint or fused data. The decision variable of user CAV_n is $a_{nm} \in \{0,1\}, \forall n \in \mathcal{V}, \forall m \in ES$.

1) Local computing model: When user CAV_n decision variable $a_{nm} = 0$, it indicates that the user CAV processes the single viewpoint data locally, which is referred to as SP mode. In this scenario, the task execution delay is expressed by the local computing delay as (6), where D_n denotes the sensing data size, q_n denotes the CPU cycle required to process the unit byte data, and $f_n(t)$ denotes the computational capacity (CPU cycles per second) of user CAV_n at time t.

$$T_n^{loc}(t) = \frac{D_n \cdot q_n}{f_n(t)} \tag{6}$$

2) Task offloading model: When the computational load of user CAV_n is high, it elects to offload the task to edge service node m for cooperative computation, that is, $a_{nm} = 1$. In this scenario, the task execution delay is articulated by the data transmission delay, data fusion delay, edge computing delay, and result return delay.

The data upload delay for user CAV_n is given by (7), where $D_n(t)$ represents the sensing data size of user CAV_n, and $R_{nm}(t)$ is the transmission data rate between user CAV_n and edge service node m at time t. When the perception task of CAV_n is offloaded to edge service node m, there may be cooperation (data fusion) with other user CAVs, so the transmission delay depends on the maximum value of the data upload delay of the user CAVs that offload the task to this node. Therefore, the transmission delay for user CAV_n is defined as (8), where a_{nm} is the offloading decision variable of user CAV_n.

$$t_{nm}^{up}(t) = \frac{D_n(t)}{R_{nm}(t)} \tag{7}$$

$$T_{nm}^{up}(t) = \max\left\{a_{1m} t_{1m}^{up}(t), a_{2m} t_{2m}^{up}(t), \ldots, a_{nm} t_{nm}^{up}(t)\right\} \tag{8}$$

The data from user CAV_n is fused with the data on edge service node m, resulting in $D_{nm}(t) = a_{nm} \cdot D_n(t) \cup D_m(t)$. The data fusion delay is expressed as (9), where ζ denotes the ratio of data fusion delay to data size.

$$T_{nm}^{fu}(t) = \zeta D_{nm}(t) \tag{9}$$

Subsequently, edge service node m performs inference computation utilizing the fused data, and the edge computing delay is expressed as (10), where D_{nm} denotes the size of the fused perceived data, and f_m denotes the computational capacity of edge service node m at time t.

$$T_{nm}^{edge}(t) = \frac{D_{nm}(t) \cdot q_m}{f_m(t)} \tag{10}$$

The detection result is returned to user CAV_n, and the result return delay is expressed as (11), where $\varpi D_{nm}(t)$ is the size of the result.

$$T_{nm}^{down}(t) = \frac{\varpi D_{nm}(t)}{R_{nm}(t)} \tag{11}$$

When the perception task of user CAV_n is executed on the edge service node m, the task edge execution delay is expressed as (12).

$$T_{nm}^{exe}(t) = T_{nm}^{up}(t) + T_{nm}^{fu}(t) + T_{nm}^{edge}(t) + T_{nm}^{down}(t) \tag{12}$$

Based on (6), (8), (9), (10), (11), and (12) above, the task execution delay can be represented as (13).

$$\mathcal{D}_n^{ego}(t) = \begin{cases} T_n^{loc}(t), & \text{if } a_{nm} = 0 \\ T_{nm}^{exe}(t), & \text{if } a_{nm} = 1 \end{cases} \tag{13}$$

E. Problem Definition

Within the VEC framework, the objective is to optimize the decision-making process from the ego perspective of each user CAV, maximizing the cooperative perception gain while minimizing task execution delay under highly dynamic conditions such as occlusion, computing resources, and communication conditions. We define the single-step optimization indicator of user CAV_n in time slot t as (14), where ω_1 and ω_2 are adjustment parameters that control the trade-off between cooperative perception gain and task execution delay.

$$\Omega_n^{ego}(t) = \omega_1 \mathcal{G}_n^{ego} + \omega_2(-\mathcal{D}_n^{ego}) \tag{14}$$

Notable is the fact that $\omega_1 + \omega_2 = 1$, and $\omega_1 \geq \omega_2$ indicates a preference for maximizing cooperative perception gain, while $\omega_1 < \omega_2$ indicates a focus on reducing task execution delay. The objective of the optimization problem is to maximize the long-term system optimization indicator of all user CAVs. Overall, we formulate the problem as follows:

$$\mathbf{P:} \max_{a_{nm}} \quad \sum_{t=1}^{T} \sum_{m=1}^{M^+} \sum_{n=1}^{N} \Omega_n^{ego}(t) \tag{15a}$$

$$\text{s.t.} \quad C_1: \ a_{nm} \in \{0,1\}, \forall n \in N, \forall m \in M^+, \tag{15b}$$

$$C_2: \ \sum_{n \in N} a_{nm} = 1, \forall m \in M^+, \tag{15c}$$

$$C_3: \ D_n^{ego} \leq T_n, \forall n \in N, \tag{15d}$$

$$C_4: \ f_{\min}^{es_m} \leq f_m(t) \leq f_{\max}^{es_m}, \forall m \in M, \tag{15e}$$

$$C_5: \ f_{\min}^{v_n} \leq f_n(t) \leq f_{\max}^{v_n}, \forall n \in N, \tag{15f}$$

$$C_6: \ \sum_{n \in N} a_{nm} B_{nm}(t) < B_{total}, \forall m \in M. \tag{15g}$$

The optimization problem incorporates constraints C_1 and C_2 to model the decision variables of user CAVs, specifying the selection of data computation location among M edge service nodes or local computation. Consequently, the offloading decision in time slot t is represented as an $(M+1)$-dimensional one-hot vector, signifying that each task can either be computed locally or offloaded to an edge service node, thereby preventing the redundant execution of tasks. Constraint C_3 ensures that each perception task is completed within the maximum tolerable delay T_n specific to the task. Constraints C_4 and C_5 stipulate that the computational capacity provided by both user CAVs and edge service nodes must lie within the range of their maximum and minimum computational capacities. Lastly, constraint C_6 accounts for the bandwidth limitations across all communication channels.

IV. Proposed Solution

This section presents the specifics of the cooperative perception and computation framework. First, the optimization problem is formulated as a Markov decision process (MDP). Next, we present an in-depth explanation of the multi-agent proximal policy optimization (MAPPO) algorithm and its improvements. Additionally, we introduce the dynamic weight adjustment method. Finally, we conduct a comprehensive analysis of the proposed algorithm. Details on preliminaries of the scheme are provided on supplementary file.

A. Formulation of Markov Decision Process

Since the formulated objective problem can be transformed into a sequential decision problem, we standardize it based on the MDP and solve it utilizing our improved MAPPO algorithm. Each user CAVs observes the local state from the environment, and subsequently employs a distributed policy network to determine actions. Both the policy network and the value network are trained on historical data, enabling users CAVs to progressively refine their strategies to achieve optimal decision-making. The MDP framework, which encompasses the state space, action space, and reward function, is delineated as follows.

1) State space: The state space at time slot t, denoted by $S(t)$, consists of the local states of all user CAVs. It is represented as $S(t) = \{s_1(t), s_2(t), \ldots, s_N(t)\}$, where N denotes the number of user CAVs. The local state of a user CAV_n consists of its own state as well as the states of the edge service nodes it can observe. This local state is defined as $s_n(t) = \{ST_n(t), f_n(t), Loc_n(t), f_n^v, \{es_n(t)_{m \in M}, \omega_m^{CP}(t)\}\}$, where $ST_n(t)$ denotes the perception task generated by user CAV_n, $f_n(t)$ denotes the computational capacity at time t, $Loc_n(t)$ denotes the current geographical coordinates, and f_n^v represents the total computational capacity that can be provided. Moreover, the $\{es_m(t)_{m \in M}\}$ denotes the state information of the edge service nodes observed by user CAV m at time t, which is obtained through lightweight CAMs, and the details are shown in the system overview Section III-A. To enhance the local observation of each user CAV and to obtain more cooperative perception gain, each user CAV calculates the cooperative weight of the edge service node m using an overlap level, which is expressed as (16).

$$\omega_m^{CP}(t) = \begin{cases} 0, & d_{nm}(t) > R_n^{RoI} + R_m^{RoS} \\ 1 - \dfrac{d_{nm}(t)}{R_n^{RoI} + R_m^{RoS}}, & \text{otherwise} \end{cases} \tag{16}$$

2) Action space: User CAVs choose actions from the action space based on the current state and their strategy. This involves deciding whether to offload computations and, if so, to which edge service node. The action of user CAV_n is

denoted by $a_n(t) = [a_{n1}, a_{n2}, \ldots, a_{nm+1}]$, which is a one-hot encoded vector. A value of 1 at the first position signifies local computation, while a 1 at any other position indicates that the computation is offloaded to the corresponding edge service node. This node could be either a service CAV or a static RSU, as specified by the index. The collective action space for all user CAVs at time t is represented by $A(t) = \{a_1(t), a_2(t), \ldots, a_N(t)\}$.

3) Reward function: The design of reward function is closely related to the optimization objectives. Upon executing an action, the user CAV agent obtains corresponding rewards. This reward serves as feedback from the environment, quantifying the efficacy of the chosen action. Our overarching objective is to maximize the cumulative rewards for all user CAVs, which entails enhancing the cooperative perception gains across the entire operational cycle while concurrently reducing task execution delay. Therefore, we define the immediate reward as the weighted sum of the negative task delay and the cooperative perception gain for user CAV_n during each time slot t, which is associated with the single-step optimization indicator, as shown in (14). The reward function for user CAV_n can be represented as (17).

$$r_n(t) = \begin{cases} {\Omega^{ego}}_n(t), & \text{if } \mathcal{D}_n < T_n \\ {\Omega^{ego}}_n(t) - \eta, & \text{otherwise} \end{cases} \tag{17}$$

When the task execution delay satisfy the specified constraint, the corresponding reward is applied; otherwise, a corresponding penalty term is incorporated into the reward function. The penalty term, denoted by η, is associated with the maximum tolerable delay for the perception task. Since the result of the perception task is crucial to subsequent autonomous driving tasks, η is invoked when the task execution delay of the perception task surpasses the maximum tolerable threshold. Based on the reward $R(t) = \{r_1(t), r_2(t), \ldots, r_N(t)\}$, each CAV agent updates its strategy to converge towards the optimal strategy.

B. Cooperative Computing Based on Improved MAPPO

In the original MAPPO algorithm, excessive update amplitude can result in an unstable optimization process and suboptimal strategy. To address this, we improve the loss function, thereby ensuring that the algorithm benefits from the monotonic improvement lower bound theorem by restricting the KL divergence more strictly. In the following, we refer to this improved MAPPO algorithm as IMAPPO, and Fig. 2 illustrates the foundational framework of IMAPPO. IMAPPO employs a centralized training approach coupled with distributed execution. Each user CAVs intelligent agent encompasses two networks: the policy network (actor network) and the value network (critic network). The actor network is tasked with mapping the observed environment state to an action distribution from which actions are sampled. During the training process, the actor network first interacts with the environment for T time steps to obtain the trajectory τ. The trajectory is represented as $\tau = (s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$, and is stored in a buffer for subsequent parameter updates. The policy parameter update aims to maximize the expected cumulative reward. The cumulative discounted reward is represented as (18), where γ denotes the discount rate, a value that lies within the interval $[0, 1)$ and is employed to weigh the immediate and future rewards.

$$R_t = \sum_{k=0}^{\infty} \gamma^k r(t+k) \tag{18}$$

The parameters of the critic network are denoted by ϕ. This network is used to approximate the mapping function from the current state to the expected reward, which is known as the state value function, defined as (19). The state value function serves as the foundation for determining the advantage function, which assesses the differential expected reward of selecting a specific action in a given state over alternative actions. The advantage is articulated by the formula (20).

$$V_\phi(s_t) = \mathrm{E}_{a_t, s_{t+1}, \ldots}[R_t | s_t] \tag{19}$$

$$\delta(t) = R(t) + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \tag{20}$$

To balance the variance and bias caused by the state value estimate and ensure the stability of the training process, we employ the general practice method of generalized advantage estimation (GAE), incorporating both advantage normalization and value clipping. The GAE is given by (21), where ϱ denotes the GAE discount factor. When ϱ is 0, GAE degenerates into a single-step temporal difference, reducing the variance; when ϱ is 1, GAE is equivalent to using monte carlo return, which tends to reduce the deviation. The expression of the value function through GAE is articulated by (22).

$$\hat{A}(t) = \delta(t) + (\gamma\varrho)\delta(t+1) + \cdots + (\gamma\varrho)^{T-t+1}\delta(T-1) \tag{21}$$

$$\hat{V}_{\pi_\varphi}(s_t) = \hat{A}(t) + V_{\pi_\varphi}(s_{t+1}) \tag{22}$$

To mitigate the estimation error inherent in GAE, the value functions of all agents are subjected to normalization. Subsequently, the loss function for the critic network is articulated as (23), where $\tilde{V}_\phi(s_{n,t})$ denotes the normalized state value.

$$J(\phi) = \frac{1}{NT} \sum_{t=1}^{T} \sum_{n=1}^{N} \left(V_\phi(s_{n,t}) - \tilde{V}_\phi(s_{n,t})\right)^2 \tag{23}$$

Here, gradient descent is employed to minimize the root mean square error, thereby optimizing the critic network. For the actor network, parameterized by φ, the optimization target is given by (24), where $H[\pi_\varphi]$ denotes the cross entropy between different action probability distributions and μ denotes entropy coefficient.

$$J(\varphi) = \frac{1}{NT} \sum_{t=1}^{T} \sum_{n=1}^{N} \{\mu H[\pi_\varphi(a_{n,t}|s_{n,t})] + \chi_n(\varphi, t)\hat{A}_n(t) D[KL(\pi_\varphi, \pi_{\varphi old})]\} \tag{24}$$

In the original MAPPO algorithm, the ratio $\chi_n(\varphi, t) = \pi_\varphi(a_{n,t}|s_{n,t}) / \pi_{\varphi old}(a_{n,t}|s_{n,t})$ is constrained within the interval $[1-\epsilon, 1+\epsilon]$ using a clipping function. However, this approach does not effectively cap the KL divergence. The KL divergence measures the dissimilarity between two probability distributions, effectively gauging the divergence in actor

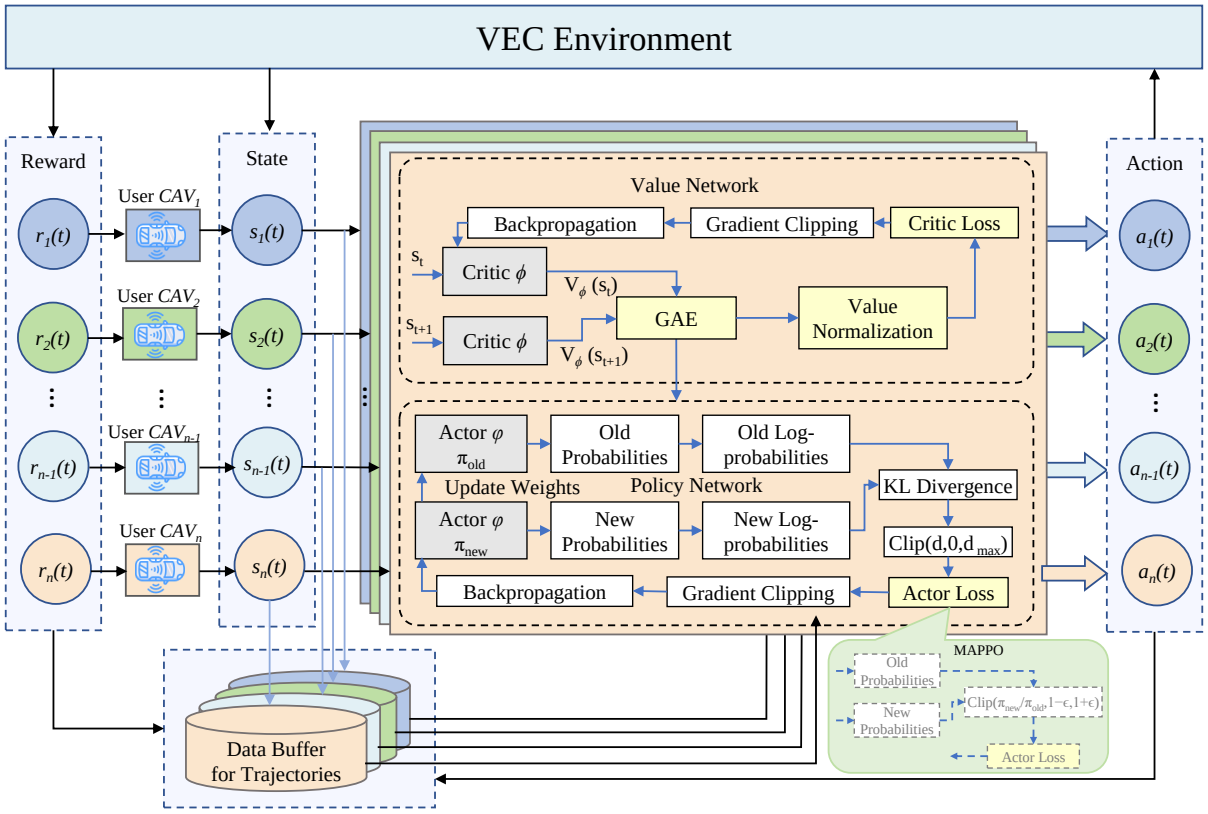

Fig. 2. IMAPPO framework and its improvements compared to MAPPO. The original MAPPO uses policy clipping to avoid unstable updates, but the bounded probability ratio cannot strictly limit the KL divergence. IMAPPO achieves stricter constraints by clipping the KL divergence.

network updates. Excessive KL divergence can destabilize the training process. Thus, in our IMAPPO algorithm, a stringent KL penalty (25) is integrated into the loss function (24).

$$D[KL] = \begin{cases} 1, & KL(\pi_\varphi, \pi_{\varphi old}) < d_{\max} \\ \dfrac{d_{\max}}{KL(\pi_\varphi, \pi_{\varphi old})}, & \text{otherwise} \end{cases} \tag{25}$$

By setting the fractional component of KL divergence, the KL divergence is limited during the parameter update process. When the KL divergence is less than the set threshold, the trust region constraint is met, thereby achieving stable strategy optimization. When the KL divergence is greater than the set threshold, the fractional form enables the strategy network to reduce the KL divergence as much as possible to maximize the expected return. Furthermore, the IMAPPO algorithm leverages mini-batches to reuse trajectory samples stored in the buffer during agent-environment interactions, which mitigates sample correlation and bolsters sample utilization efficiency.

Building upon the preceding discussions, we summarize the edge-assisted multi-CAVs cooperative perception and computation framework based on the IMAPPO in Algorithm 1. The algorithm begins by elucidating the input and output, culminating in the optimized decision-making model. At line 1, the algorithm initializes the actor and critic network parameters in an orthogonal manner, which mitigates gradient-related issues during training, such as gradient disappearance or gradient explosion. By line 4, a data buffer is established to archive trajectory samples of duration T, collected by the CAV agent in each iteration, with each user CAV maintaining a corresponding buffer. From lines 5 to 12, during each time slot, each user CAV independently observes the environment, generates an action probability distribution based on the actor network, samples a specific action from this distribution, and calculates the state value. From lines 13 to 18, each user CAV executes the sampled action and obtains a new local state and reward from interacting with the environment. Notably, the CAV obtains specific data on the task execution delay and cooperative perception gain after completing the detection task. The user CAV interacts with the VEC environment over T time steps to collect trajectory data τ. Between lines 19 and 24, the advantage function $\hat{A}$ is calculated using GAE based on collected trajectory data samples. Following this, the normalized value function V_ϕ is calculated. The sequence of data within the buffer is then randomized and re-indexed, an intervention that diminishes data correlation, thereby stabilizing the training process. Lines 26 to 31 extract small batches of data from the buffer, divide the data into numerous mini batches, and train them multiple times within a single epoch, thereby enhancing data utilization efficiency. Concluding with lines 32 and 33, the algorithm updates the actor parameter φ and critic parameter ϕ using the Adam optimizer. The variable E denotes the number of training epochs, while T specifies the trajectory length interacting with the environment.

Algorithm 1 IMAPPO-Based Cooperative Perception and Task Offloading

Input: Simulation parameters for the VEC network.
Output: The optimal IMAPPO agent policy π_φ.

Initialization: Orthogonal initialize actor network π's parameters φ and critic network V's parameters ϕ.
Set learning rate l_r.
for episode = $1, 2, \ldots, E$ **do**
  Initialize data buffer D.
  for batch size $i = 1, 2, \ldots, B$ **do**
    Initialize empty list τ.
    for time slot $t = 1, 2, \ldots, T$ **do**
      for user CAV $n = 1, 2, \ldots, N$ **do**
        $P_n(t) = \pi_\varphi(a_{n,t}|s_n(t))$.
        Sample $a_{n,t}$ from $P_n(t)$.
        $V_n(t) = V_\phi(s_t)$.
      end for
      Execute action $a(t)$.
      Get instant rewards from VEC Environment.
      Calculate CP weight of edge service nodes by (16).
      Observe s_{t+1} based on VEC environment.
      $\tau += [s_t, a_t, V(t), r(t), s_{t+1}]$.
    end for
    Calculate advantage estimate $\hat{A}$ by (21) on τ.
    Calculate $\hat{V}_\phi$ by (22) on τ and normalize.
    Split trajectory τ into chunks of length L.
    for $l = 0, 1, \ldots, T//L$ **do**
      Append chunk $\tau[l : l+T]$, $\hat{A}[l : l+L]$, and $\hat{R}[l : l+L]$ to buffer D.
    end for
  end for
  for mini-batch $k = 1, \ldots, K$ **do**
    b $\leftarrow$ random mini-batch from D with all agent data.
    for each data chunk c in mini-batch b **do**
      Update V based on first hidden state in data chunk.
    end for
  end for
  Update φ on $J(\varphi)$ with mini-batch b using Adam.
  Update ϕ on $J(\phi)$ with mini-batch b using Adam.
end for

C. Dynamic Weight Adjustment

To adapt to varying traffic scenarios, it is essential to dynamically adjust the weights assigned to cooperative perception gain (ω_1) and task execution delay (ω_2). The two weights are interrelated by the constraint $\omega_1 + \omega_2 = 1$, and the dynamic adjustment method primarily adjusts ω_2, indirectly modifying ω_1. This dynamic adjustment method takes into account two critical factors: the time slot occupancy rate of task execution and the current computational load of user CAVs. The time slot occupancy rate is defined in (26), where Δ is the time slot size. For example, when the time slot occupancy rate is large, it means that the task execution delay is high, so the component ρ increases the percentage of weight ω_2 and the importance of optimizing the task execution delay is higher.

$$\rho = 1 - \frac{\Delta - D_n^{ego}}{\Delta} \tag{26}$$

Concurrently, the computational load of user CAVs is evaluated, primarily based on the computational capacity available in the current time slot. The evaluation of the computational load is represented by a sigmoid function as depicted in (27), where k is a parameter that controls the steepness of the sigmoid curve, f_{th} is a threshold value used to determine load level, and $f_n(t)$ represents the computational capacity at time t.

$$\varsigma = \frac{1}{1 + \exp(-k(f_{th} - f_n(t))} \tag{27}$$

When the available computational capacity $f_n(t)$ of the user CAV is small, it means that the computational load is high and the weight ω_2 is larger, which makes the CAV prioritize the task execution delay to ensure the timely processing of the task. The task execution delay weight considering the above factors is expressed as (28).

$$\omega_2 = \rho \cdot \varsigma \tag{28}$$

Consequently, an increase in both the time slot occupancy rate and the computational load leads to a proportional rise in ω_2, prompting user CAVs to prioritize achieving a lower task execution delay. When the time slot occupancy rate and computational load are low, ω_2 is smaller and ω_1 is larger, enhancing the emphasis of cooperative perception gain.

D. Computational Complexity

We analyze the time complexity of the IMAPPO algorithm, which is directly proportional to the number of neurons in the neural network. Fully connected neural networks are used to implement the actor and critic networks. For the training process, each user CAV agent completes the interaction in parallel, so N does not contribute to the total running time. Consequently, the time complexity is delineated as $\mathcal{O}\left(E \cdot B \cdot T \cdot (\sum_{j=1}^{L_{ac}-1} X_j^{ac} X_{j-1}^{ac} + \sum_{j=1}^{L_{cr}-1} X_j^{cr} X_{j-1}^{cr})\right)$, where L_{ac} and L_{cr} represent the number of layers in the actor and critic networks, respectively, and X_j^{ac} and X_j^{cr} represent the number of neurons in the j-th layer of the respective networks. For the distributed inference process, the time complexity is $\mathcal{O}\left(\sum_{j=1}^{L_{ac}-1} X_j^{ac} X_{j-1}^{ac} + \sum_{j=1}^{L_{cr}-1} X_j^{cr} X_{j-1}^{cr}\right)$. The design of the input and output layers of the neural network can be found in Section IV-A, and the various training hyperparameters are detailed in Section II-A in supplementary file.

V. Performance Evaluation

In this section, the effectiveness of the proposed solution is verified through rigorous experiments. Firstly, we introduce the simulation tool and detail the experimental parameter settings. Secondly, we compare the IMAPPO algorithm with other baseline algorithms across multiple metrics, and evaluate its adaptability across different scenarios. Lastly, the practical relevance, limitations, and future work of our scheme are discussed. For the effect of the KL penalty on KL divergence and distribution entropy, please see Section II-A in supplementary file.

A. Simulation Tools and Parameter Settings

Simulation data is generated using OpenCDA [42], an open-source cooperative driving automation framework that integrates simulation of urban mobility (SUMO) traffic simulator

TABLE III
SYSTEM PARAMETERS IN SIMULATION

Parameters	Value
Number of user CAVs (N)	$[4, 20]$
Number of service CAVs	$[5, 10]$
Number of RSUs	5
Center frequency (f_c)	5.9 GHz
Noise power (σ^2)	-104 dBm
Transmit power (p_n)	23 dBm
Bandwidth (B_{total})	20 MHz
CAVs maximum local CPU frequency (f_n^v)	4 GHz
CAVs sensing range radius (R_{RoS})	100m
CAVs maximum speed	16.7m/s
RoI radius (R_{RoI})	150m
RSUs maximum CPU frequency (f_n^{es})	2×4 GHz
RSUs sensing range radius	200m
Maximum delay tolerance (T_n)	100 ms
Computational density (q_n)	30 cycles/bit
Batch size (B)	1
Mini-batch number (K)	4
Total training epochs (E)	1860
Trajectory length (T)	105

and car learning to act (CARLA) autonomous driving simulator. The PointPillars objects detection algorithm and the multi-agent reinforcement learning algorithm are implemented using PyTorch. The algorithm training was conducted on a 6-core CPU with a maximum frequency of 4.4GHz, delivering 240 Giga-FLOPS. It also utilized a 22.06 TFLOPS GPU and 32GB of RAM. CAVs and RSUs data are generated at a frequency of 10Hz in the map Town06. The Town06 road network consists of four main roads, each with four to six lanes, along with multiple branches and intersections. A total of 100 vehicles are generated, including 4 to 20 user CAVs, 5 to 10 service CAVs, and the remainder being human-driven vehicles, along with 5 RSUs. For the communication model, parameters are designed to simulate a 20 MHz 5G NR V2X system. The transmission power is set to 23dBm, and the noise power is set to -104 dBm. In an urban traffic environment, the path loss of LOS and NLOSv channels is calculated by $PL_{LOS} = 38.77 + 16.7log_{10}d + 18.2log_{10}f_c$, while the path loss of NLOS channel is given by $PL_{NLOS} = 36.85 + 30log_{10}d + 18.9log_{10}f_c$, where d denotes the distance between sender and receiver, and f_c denotes the center frequency. The computational density of the perception task is set to 30 cycle per bit, and the size of the sensor data frame is determined by the actual file size generated. The weight of objects is calculated depending on the speed of the user CAVs and the distance to the objects, as described in [19]. The CAVs sensing range radius is 100 meters, the RoI radius is 150 meters, and the RSUs sensing range is 200 meters. The main parameters and values are shown in Table III. In addition, different computational load levels under the setting of parameters $k = 1$ and $f_{th} = 2$ GHz are quantitatively presented in Table IV.

B. Comparative Performance and Adaptability Evaluation

1) Evaluation metrics: We employ the following metrics to evaluate the effectiveness of different schemes:

TABLE IV
COMPUTATIONAL LOAD CLASSIFICATION

	Low	**Medium**	**High**
Load Percentage	$[0, 32.1\%]$	$(32.1\%, 67.9\%]$	$(67.9\%, 100\%]$
$f_n(t)$ (GHz)	$[2.75, 4]$	$[1.25, 2.75)$	$[0, 1.25)$

Note: parameters $k = 1$ and $f_{th} = 2$ GHz.

(a) Total reward (TR): This metric signifies optimization goal of the system. The convergence value and convergence speed of the total reward per episode are indicative of the strategy's quality and the training efficiency of the intelligent algorithm.
(b) Average task execution delay (ATED): In the context of VEC, this metric serves as a critical indicator, directly reflecting both the computing efficiency of the chosen computing nodes and the quality of communication over unicast links.
(c) Average cooperative perception gain (ACPG): This metric reflects the perceptual benefits that user CAVs obtain from detecting objects within their RoI and encourages them to collaborate with edge service nodes that have higher cooperative weight.

2) Counterparts: For the effectiveness of the proposed scheme, we compare it with the following baselines, and the baseline time complexity analysis is given in Section II-B1 of the supplementary file.

(a) Random: In this strategy, each user CAV independently samples actions from the action space for offloading decisions. This independent approach allows for an assessment of the performance lower bound, as CAVs operate without coordination.
(b) Greedy [19]: Each user CAV selects the edge service node with the highest confidence upper limit based on historical perception gains. This approach prioritizes individual benefit maximization without accounting for the broader implications on resource contention and cooperative vehicle interactions.
(c) DQN [33]: A centralized solution featuring a central controller (CC) as a single agent, this approach aims to approximate the optimal value function using a DNN and selects the action with the highest value.
(d) MADDPG [38]: A multi-agent deep deterministic policy gradient algorithm that uses a replay buffer to store experience and performs soft updates of parameters based on a delayed replicated target actor and critic network.
(e) MAPPO [34]: A distributed multi-agent proximal policy optimization algorithm that based on the actor-critic framework. The MAPPO utilizes policy clipping and has significantly high algorithm operation efficiency than MADDPG with limited computing resources [43].

3) Comparative Analysis of Performance: Our proposed IMAPPO algorithm is benchmarked against its counterparts based on evaluation metrics and its adaptability to different scenarios is also analyzed. For the backbone overhead of different algorithms, please see Section II-B2 of supplementary file.

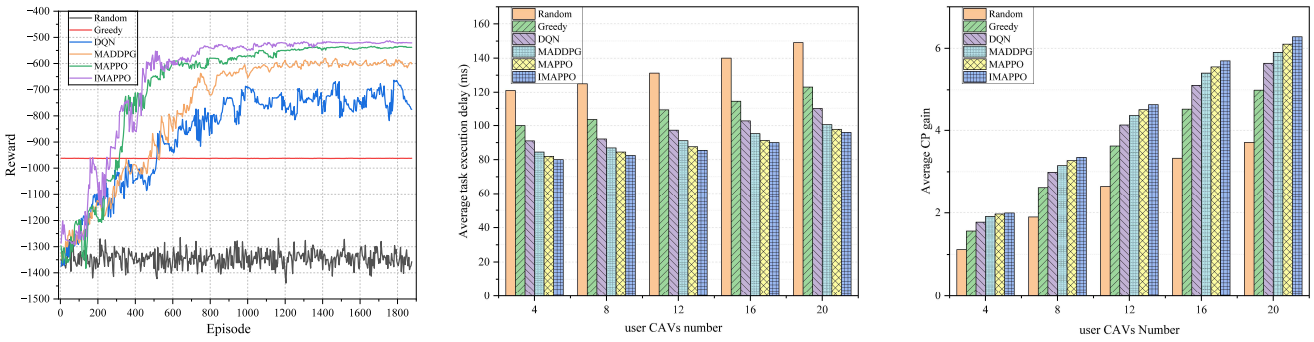

Fig. 3. The performance of the different algorithm. (a) Total reward with different Episode. (b) ATED with different user CAVs number. (c) ACPG with different user CAVs number.

Fig. 3(a) illustrates a comparison of the total rewards obtained in each episode by our proposed IMAPPO algorithm and the baseline algorithms. Reinforcement learning-based intelligent algorithms rely on empirical data to optimize their strategies. As the strategies are iteratively optimized, the total rewards of each episode progressively increase. In Fig. 3(a), the rewards initially fluctuate and gradually rise during the early stages of training, eventually converging as network parameters are updated. Our proposed IMAPPO algorithm demonstrates a more rapid growth in rewards after approximately 500 episodes and converges when the episode count exceeds 1400. Notably, the total rewards achieved after convergence are higher than those obtained by the clipped MAPPO, MADDPG, DQN, random strategy, and greedy algorithm. Among these, the greedy algorithm does not require iterative strategy optimization and shows better performance in the first 200 episodes. However, it performs worse than reinforcement learning-based methods in later episodes because it focuses solely on the immediate optimal solution while not considering long-term returns. Compared with MAPPO, MADDPG and DQN, the total rewards of IMAPPO are increased by 3.1%, 15.4% and 25.8% respectively. Specifically, the actor-critic architecture employed in IMAPPO proves more effective in handling large state spaces than the DQN, and IMAPPO provides smoother policy updates compared to the soft-updating MADDPG. Furthermore, compared to the clipped MAPPO algorithm, the improved loss function in IMAPPO imposes explicit restrictions to prevent excessive divergence in KL divergence during policy updates, thereby enhancing stability. Consequently, IMAPPO attains the best overall performance.

In Figs. 3(b) and 3(c), we test the impact of different numbers of user CAVs on both the ATED and ACPG of each user CAV's perception tasks, thereby verifying the scalability of our scheme. As shown in the figures, the ATED across various schemes increases with an increasing user CAVs count, a trend mirrored by the ACPG. The primary reasons are as follows: first, the delay performance of the CP task is influenced by multiple variables, with the most important factor being the number of cooperative data sources, that is, the number of user CAVs. Within a given system time slot, the edge service node must await the completion of all sensor data transmissions prior to undertaking data fusion. Consequently, an augmentation in the cooperative node count not only prolongs the data fusion delay but also increases the transmission delay. Second, as number of user CAVs grows, the data enhancement achieved through data fusion leads to a more comprehensive environmental perception, consequently boosting the ACPG. Furthermore, it is evident that our proposed IMAPPO algorithm surpasses other methods in terms of these performance metrics. This superiority stems from its enhanced strategy, thereby validating the efficacy of the IMAPPO algorithm.

In Figs. 4(a) and 4(b), we demonstrate the impact of sensor data quality on ATED and ACPG. The horizontal axis indicates the maximum number of scan points contained in each frame of the simulated LiDAR data. A higher number of scan points corresponds to a more detailed perception of the surrounding environment. As observed in the figures, across various algorithms, an increase in data quality correlates with a higher ACPG. This is because more detailed data frames enhance the detection accuracy of the model, albeit with diminishing returns. Specifically, at a maximum number of scan points of 80,000, the enhancement of ACPG becomes more pronounced. In this scenario, IMAPPO demonstrates a notable improvement, showing an approximate increase of 2.9% in ACPG compared to other algorithms. Simultaneously, the ATED also increases, as larger data frames demand additional time for transmission and computation. Nevertheless, our enhanced algorithm, through an optimized collaboration and offloading strategy, achieves the lowest task execution delay when compared to other algorithms.

In Fig. 4(c), we examine the impact of varying computational capacity of RSUs on the ATED across different algorithms. The findings indicate that the ATED progressively diminishes with an increase in the computational capacity of RSUs serving as edge service nodes. This trend underscores the advantage of edge service nodes with greater computational resources, which can offer expedited data processing. User CAVs can leverage the powerful computing capability of these edge service nodes to enhance the performance of task processing. Moreover, our solution consistently achieves the lowest ATED across various levels of edge computational

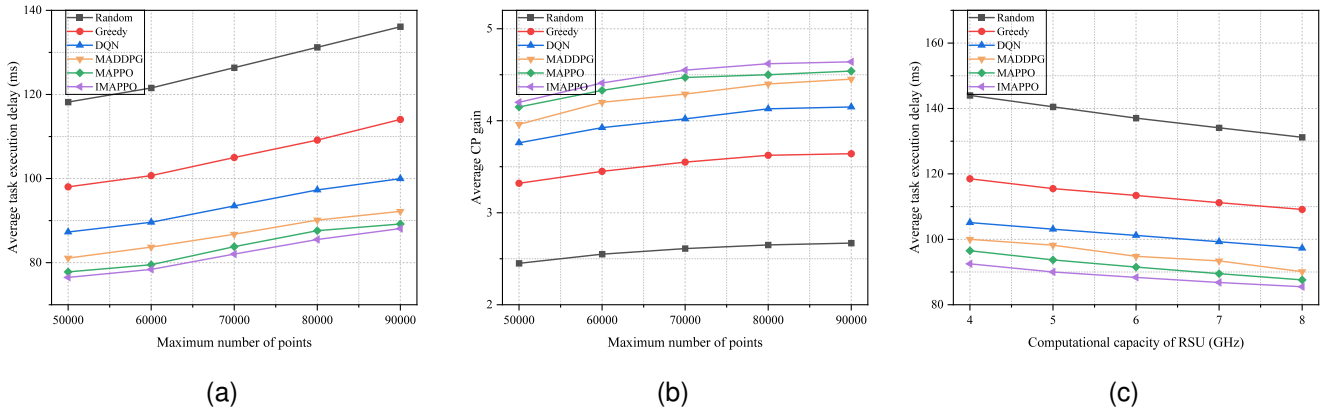

Fig. 4. (a) ATED with different maximum number of points. (b) ACPG with different maximum number of points. (c) ATED with different computational capacity of RSU.

capacity, demonstrating the adaptability of the IMAPPO optimization strategy to diverse system parameters.

To assess the efficacy of our dynamic weight adjustment method, we design scenarios with varying computational loads: high, medium, and low. In these scenarios, the impact of weight adjustments on the ATED and ACPG is systematically compared under varying numbers of user CAVs, as illustrated in Fig. 5. When the computational load is low, a lower weight is assigned to task execution delay, thereby enhancing the emphasis on cooperative perception gain, which in turn boosts the perceived gains for user CAVs. Conversely, under high computational loads, task execution delay is prioritized, prompting user CAVs to engage with service nodes that possess sufficient computational resources and superior communication conditions. Although this strategy may marginally diminish cooperative perception gain, it ensures that the task completes the computation within the maximum tolerable delay. In scenarios with medium computational loads, a more balanced strategy is taken, optimizing both task execution delay and cooperative perception gains. The experimental results demonstrate that our dynamic weight adjustment method can adeptly balances task execution delay and cooperative perception gain in real-world dynamic scenarios.

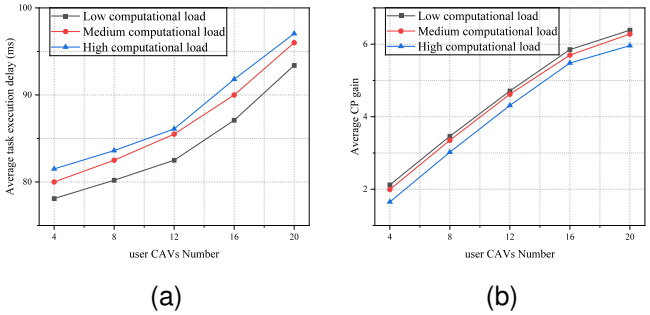

Fig. 5. Impact of weight adjustment on ATED and ACPG. (a) ATED with different user CAVs number. (b) ACPG with different user CAVs number.

C. Discussion

1) Practical Relevance: Applying this approach to real-world VEC environments requires considering the differences in data distribution between simulation and reality. To improve the generalization of the model, domain adaptation or transfer learning can be used. Domain adaptation helps the model adapt to the differences between simulated and real-world data, while transfer learning can fine-tune the pre-trained model based on real-world data, thereby reducing the need for large real-world datasets and improving practicality.

2) Limitations: The current simulation has several limitations that need to be further addressed:

- Network connection variation: Vehicle movement leads to unstable connections, which causes packet loss, and the current simulation weakens this issue.
- Strict synchronization assumption: The simulation assumes perfect time slot synchronization, which may be challenging for synchronization of multiple devices in the real world.
- Data characteristics: The model does not consider packet loss, sensor failure, or malicious attacks, which reduces the reliability of the system.
- RAM limitations: The model does not consider the impact of RAM limitations on processing efficiency, especially in the case of high memory usage.

3) Future Research: Future research can focus on the following areas:

- Diverse configurations: Test the system with various vehicle and edge server configurations to ensure adaptability and scalability.
- RAM impact: Incorporate RAM-related metrics to analyze how memory limitations affect performance.
- Large-scale simulations: Test in more complex and larger-scale scenarios to evaluate scalability and adaptability.
- Realistic communication models: Include packet loss, communication interruptions, and data corruption to improve system reliability.
- Environmental factors: Simulate the impact of weather, traffic density, and other external factors to enhance system robustness.

VI. Conclusion

In this paper, we introduce a unicast-based cooperative perception and computation scheme for CAVs, assisted by edge infrastructure, to tackle the challenge of real-time transmission and computation of massive perception data under limited communication and computing resources. We pose a multi-objective optimization problem focusing on task execution delay and cooperative perception gain, and develop an improved multi-agent proximal policy optimization algorithm to train decentralized decision-making agents. Moreover, our adaptive weight adjustment method is designed to dynamically respond to varying computational loads, effectively balancing the trade-offs between task execution delay and cooperative perception gain. Our comprehensive simulation results demonstrate that the proposed scheme achieves a reduction in average task execution delay by 1.8% to 2.4% and an improvement in average cooperative perception gain by 1.2% to 2.9% over the baseline algorithm. In summary, these results prove the effectiveness and superiority of our scheme in minimizing task execution delay and maximizing cooperative perception gain.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant 62372310, and in part by the Liaoning Province Applied Basic Research Program under Grant 2023JH2/101300194, and in part by the LiaoNing Revitalization Talents Program under Grant XLYC2203151. Large language model is used to proofread this paper.

References

[1] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (cavs)," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6240–6259, 2022.

[2] B. Lu, X. Huang, Y. Wu, L. Qian, D. Niyato, and C. Xu, "Cooperative perception aided digital twin model update and migration in mixed vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2024.

[3] M. Ahmed, M. A. Mirza, S. Raza, H. Ahmad, F. Xu, W. U. Khan, Q. Lin, and Z. Han, "Vehicular communication network enabled cav data offloading: A review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 7869–7897, 2023.

[4] J. Liu, Y. Zhou, and L. Liu, "Communication delay-aware cooperative adaptive cruise control with dynamic network topologies-a convergence of communication and control," *Digital Communications and Networks*, 2023.

[5] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang, "Vips: real-time perception fusion for infrastructure-assisted autonomous driving," in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, ser. MobiCom '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 133–146.

[6] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 514–524.

[7] X. Xu, K. Liu, P. Dai, F. Jin, H. Ren, C. Zhan, and S. Guo, "Joint task offloading and resource optimization in noma-based vehicular edge computing: A game-theoretic drl approach," *Journal of Systems Architecture*, vol. 134, p. 102780, 2023.

[8] Z. Lu, T. Wu, J. Su, Y. Xu, B. Qian, T. Zhang, and H. Zhou, "Toward edge-computing-enabled collision-free scheduling management for autonomous vehicles at unsignalized intersections," *Digital Communications and Networks*, 2024.

[9] X. Huang, P. Li, H. Du, J. Kang, D. Niyato, D. I. Kim, and Y. Wu, "Federated learning-empowered ai-generated content in wireless networks," *IEEE Network*, vol. 38, no. 5, pp. 304–313, 2024.

[10] R. Chen, S. Sun, Y. Liu, X. Hu, Y. Hui, and N. Cheng, "Proactive effects of c-v2x-based vehicle-infrastructure cooperation on the stability of heterogeneous traffic flow," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 9184–9197, 2024.

[11] Y. Jia, R. Mao, Y. Sun, S. Zhou, and Z. Niu, "Online v2x scheduling for raw-level cooperative perception," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 309–314.

[12] Y. Li, S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang, "Learning distilled collaboration graph for multi-agent perception," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 541–29 552, 2021.

[13] Y. He, L. Ma, Z. Jiang, Y. Tang, and G. Xing, "Vi-eye: semantic-based 3d point cloud registration for infrastructure-assisted autonomous driving," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 573–586.

[14] M. Shan, K. Narula, S. Worrall, Y. F. Wong, J. Stephany Berrio Perez, P. Gray, and E. Nebot, "A novel probabilistic v2x data fusion framework for cooperative perception," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 2013–2020.

[15] Z. Song, F. Wen, H. Zhang, and J. Li, "A cooperative perception system robust to localization errors," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–6.

[16] X. Ye, K. Qu, W. Zhuang, and X. Shen, "Accuracy-aware cooperative sensing and computing for connected autonomous vehicles," *IEEE Transactions on Mobile Computing*, pp. 1–15, 2023.

[17] S. Ren, Z. Lei, Z. Wang, M. Dianati, Y. Wang, S. Chen, and W. Zhang, "Interruption-aware cooperative perception for v2x communication-aided autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 4, pp. 4698–4714, 2024.

[18] M. Harounabadi, D. M. Soleymani, S. Bhadauria, M. Leyh, and E. Roth-Mandutz, "V2x in 3gpp standardization: Nr sidelink in release-16 and beyond," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 12–21, 2021.

[19] Y. Jia, R. Mao, Y. Sun, S. Zhou, and Z. Niu, "Mass: Mobility-aware sensor scheduling of cooperative perception for connected automated driving," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14 962–14 977, 2023.

[20] W. Fan, Y. Zhang, G. Zhou, and Y. Liu, "Deep reinforcement learning-based task offloading for vehicular edge computing with flexible rsu-rsu cooperation," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2024.

[21] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6206–6221, 2022.

[22] L. Zhao, E. Zhang, S. Wan, A. Hawbani, A. Y. Al-Dubai, G. Min, and A. Y. Zomaya, "Meson: A mobility-aware dependent task offloading scheme for urban vehicular edge computing," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4259–4272, 2024.

[23] J. Liu, M. Ahmed, M. A. Mirza, W. U. Khan, D. Xu, J. Li, A. Aziz, and Z. Han, "Rl/drl meets vehicular task offloading using edge and vehicular cloudlet: A survey," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8315–8338, 2022.

[24] L. Wu, J. Qu, S. Li, C. Zhang, J. Du, X. Sun, and J. Zhou, "Attention-augmented maddpg in noma-based vehicular mobile edge computational offloading," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[25] Z. Gao, L. Yang, and Y. Dai, "Fast adaptive task offloading and resource allocation via multiagent reinforcement learning in heterogeneous vehicular fog computing," *IEEE Internet of Things Journal*, vol. 10, no. 8, pp. 6818–6835, 2023.

[26] Z. Y. Rawashdeh and Z. Wang, "Collaborative automated driving: A machine learning-based method to enhance the accuracy of shared information," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3961–3966.

[27] D. D. Yoon, B. Ayalew, and G. G. M. Nawaz Ali, "Performance of decentralized cooperative perception in v2v connected traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6850–6863, 2022.

[28] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "V2vnet: Vehicle-to-vehicle communication for joint perception and prediction," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 605–621.

[29] Z. Lei, S. Ren, Y. Hu, W. Zhang, and S. Chen, "Latency-aware collaborative perception," in *European Conference on Computer Vision*. Springer, 2022, pp. 316–332.

[30] R. Xu, Z. Tu, H. Xiang, W. Shao, B. Zhou, and J. Ma, "Cobevt: Cooperative bird's eye view semantic segmentation with sparse transformers," *arXiv preprint arXiv:2207.02202*, 2022.

[31] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, and J. Ma, "V2x-vit: Vehicle-to-everything cooperative perception with vision transformer," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 107–124.

[32] L. Zhao, Z. Zhao, E. Zhang, A. Hawbani, A. Y. Al-Dubai, Z. Tan, and A. Hussain, "A digital twin-assisted intelligent partial offloading approach for vehicular edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3386–3400, 2023.

[33] L. Zhao, T. Li, E. Zhang, Y. Lin, S. Wan, A. Hawbani, and M. Guizani, "Adaptive swarm intelligent offloading based on digital twin-assisted prediction in vec," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2023.

[34] S. S. Hassan, Y. M. Park, Y. K. Tun, W. Saad, Z. Han, and C. S. Hong, "Satellite-based its data offloading & computation in 6g networks: A cooperative multi-agent proximal policy optimization drl with attention approach," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4956–4974, 2024.

[35] M. Tang and V. W. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2022.

[36] P. Lv, W. Xu, J. Nie, Y. Yuan, C. Cai, Z. Chen, and J. Xu, "Edge computing task offloading for environmental perception of autonomous vehicles in 6g networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1228–1245, 2023.

[37] G. Luo, C. Shao, N. Cheng, H. Zhou, H. Zhang, Q. Yuan, and J. Li, "Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 1, pp. 207–222, 2024.

[38] K. Qu, W. Zhuang, Q. Ye, W. Wu, and X. Shen, "Model-assisted learning for adaptive cooperative perception of connected autonomous vehicles," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2024.

[39] Z. Xiao, J. Shu, H. Jiang, G. Min, H. Chen, and Z. Han, "Perception task offloading with collaborative computation for autonomous driving," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 457–473, 2023.

[40] L. Wu, J. Qu, S. Li, C. Zhang, J. Du, X. Sun, and J. Zhou, "Attention-augmented maddpg in noma-based vehicular mobile edge computational offloading," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[41] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1972–2026, 2021.

[42] R. Xu, Y. Guo, X. Han, X. Xia, H. Xiang, and J. Ma, "Opencda: An open cooperative driving automation framework integrated with co-simulation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 1155–1162.

[43] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," 2022.

Liang Zhao (Member, IEEE) is a Professor at Shenyang Aerospace University, China. He received his Ph.D. degree from the School of Computing at Edinburgh Napier University in 2011. Before joining Shenyang Aerospace University, he worked as associate senior researcher in Hitachi (China) Research and Development Corporation from 2012 to 2014. He is also a JSPS Invitational Fellow (2023-2024). He was listed as Top 2% of scientists in the world by Standford University (2022 and 2023). His research interests include ITS, VANET, WMN and SDN. He has published more than 170 articles. He served as the Chair of several international conferences and workshops, including 2022 IEEE BigDataSE (Steering CoChair), 2021 IEEE TrustCom (Program Co-Chair), 2019 IEEE IUCC (Program Co-Chair), and 2018-2022 NGDN workshop (founder). He is Associate Editor of Frontiers in Communications and Networking and Journal of Circuits Systems and Computers. He is/has been a guest editor of IEEE Transactions on Network Science and Engineering, Springer Journal of Computing, etc. He was the recipient of the Best/Outstanding Paper Awards at 2015 IEEE IUCC, 2020 IEEE ISPA, 2022 IEEE EUC and 2013 ACM MoMM.

Longjia Li is a student at Shenyang Aerospace University, Shenyang, China. He is currently studying for M.S. degree in Computer Science, Shenyang Aerospace University. His research interests mainly include V2X, Vehicular Edge Computing, Internet of Things, and Intelligent Transportation Systems.

Zhiyuan Tan (Senior Member, IEEE) received his Ph.D. degree from the University of Technology Sydney (Australia) in 2014. He was a Postdoctoral Researcher at the University of Twente (Netherlands) from 2014 to 2016, and is now an Associate Professor with the School of Computing, Engineering & the Built Environment at Edinburgh Napier University (UK). Dr Tan is also a Senior IEEE Member, and ACM Member. His recent research interest includes Cyber Security, Machine Learning, Cognitive Computing, and Intelligent Transportation. His research has been funded by the Royal Society (UK), the Scottish Informatics & Computer Science Alliance, the ENU Development Trust, the Commonwealth Scientific and Industrial Research Organisation (Australia), etc. He is an Associate Editor of IEEE Transactions on Reliability, and an Academic Editor of Security and Communication Networks.

Ammar Hawbani is a full Professor at the School of Computer Science at Shenyang Aerospace University. He received the B.S., M.S. and Ph.D. degrees in Computer Software and Theory from USTC, in 2009, 2012 and 2016, respectively. From 2016 to 2019, he worked as Postdoctoral Researcher in the School of Computer Science and Technology at USTC. Later, he worked as an Associate Researcher in the School of Computer Science and Technology at USTC from 2019 to 2023. His research interests include IoT, WSNs, WBANs, WMNs, VANETs, and SDN.

Qiang He (Member, IEEE) received the Ph.D. degree in computer application technology from the Northeastern University, Shenyang, China in 2020. He also worked with School of Computer Science and Technology, Nanyang Technical University, Singapore as a visiting PhD researcher from 2018 to 2019. He is currently an Associated Professor at the College of Medicine and Biological Information Engineering, Northeastern University, Shenyang, China. His research interests include machine learning, social network analytic, data mining, health care, infectious diseases informatics, etc. He has published more than 70 journal articles and conference papers, including IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Cybernetics, IEEE Transactions on Cloud Computing, IEEE Transactions on Computational Social Systems, IEEE Transactions on Cognitive and Developmental Systems.

Zhi Liu (Senior Member, IEEE) received the B.E. degree from the University of Science and Technology of China, China, and the Ph.D. degree in informatics from the National Institute of Informatics. He is currently an Associate Professor with The University of Electro-Communications, Japan. His research interests include video network transmission, vehicular networks, and mobile edge computing. He was a recipient of the IEEE StreamComm 2011 Best Student Paper Award, the 2015 IEICE Young Researcher Award, and the ICOIN 2018 Best Paper Award. He is an Editorial Board Member of Wireless Networks (Springer) and IEEE Open Journal of the Computer Society.