

# Digital Twin-Assisted Task Offloading and Resource Allocation in ISAC-Enabled Internet of Vehicles

Shanhao Zhan, Zhang Liu, Lianfen Huang, Shaowei Shen, Ziyang Bai, Zhibin Gao,  
and Dusit Niyato, *Fellow, IEEE*

**Abstract**—The convergence of the Internet of vehicles (IoV) and 6G networks is driving the evolution of next-generation intelligent transportation systems. However, IoV networks face persistent challenges, including low spectral efficiency in vehicular communications, difficulty in achieving dynamic and adaptive resource optimization, and the need for long-term stability under highly dynamic environments. In this paper, we study the problem of digital twin (DT)-assisted task offloading and resource allocation in integrated sensing and communication (ISAC)-enabled IoV networks. The objective is to minimize the long-term average system cost, defined as a weighted combination of delay and energy consumption, while ensuring queue stability over time. To address this, we employ an ISAC-enabled design and introduce two transmission modes (i.e., raw data transmission (DataT) and instruction transmission (InstrT)). The InstrT mode enables instruction-level transmission, thereby reducing data volume and improving spectral efficiency. We then employ Lyapunov optimization to decompose the long-term stochastic problem into per-slot deterministic problems, ensuring long-term queue stability. Building upon this, we propose a Lyapunov-driven DT-enhanced multi-agent proximal policy optimization (Ly-DTMPPPO) algorithm, which leverages DT for global state awareness and intelligent decision-making within a centralized training and decentralized execution (CTDE) architecture. Extensive simulations verify that Ly-DTMPPPO achieves superior performance compared with existing benchmarks.

**Index Terms**—Integrated sensing and communications, digital twin, resource allocation, task offloading, Lyapunov optimization.

## I. INTRODUCTION

### A. Background Research

WITH the commercial deployment of 5G and the ongoing research toward 6G, the Internet of vehicles (IoV) has emerged as a cornerstone technology for intelligent transportation and autonomous driving. However, IoV faces pressing demands for low latency, highly reliable communications, accurate environmental sensing, and intelligent computing [1].

Shanhao Zhan ([shanhao@stu.xmu.edu.cn](mailto:shanhao@stu.xmu.edu.cn)) is with the Department of Informatics and Communication Engineering, Xiamen University, Fujian 361102, China. Zhang Liu ([zhangliu@xmu.edu.cn](mailto:zhangliu@xmu.edu.cn)) is with the Department of Computer Science and Technology, Xiamen University, Fujian 361102, China. Lianfen Huang ([lfhuang@xmu.edu.cn](mailto:lfhuang@xmu.edu.cn)) is with the Key Laboratory of Intelligent Manufacturing Equipment and Industrial Internet Technology, Fujian Provincial Universities, and with School of Information Science and Technology, Xiamen University Tan Kah Kee College, and with the Department of Informatics and Communication Engineering, Xiamen University, Fujian 361102, China. Shaowei Shen ([shenshaowei@stu.xmu.edu.cn](mailto:shenshaowei@stu.xmu.edu.cn)) and Ziyang Bai ([23320230157375@stu.xmu.edu.cn](mailto:23320230157375@stu.xmu.edu.cn)) are with the Department of Informatics and Communication Engineering, Xiamen University, Fujian 361102, China. Zhibin Gao ([gaozhibin@jmu.edu.cn](mailto:gaozhibin@jmu.edu.cn)) is with Navigation Institute, Jimei University, Xiamen 361021, China. D. Niyato ([dniyato@ntu.edu.sg](mailto:dniyato@ntu.edu.sg)) is with the College of Computing and Data Science, Nanyang Technological University, Singapore. (Corresponding author: Zhibin Gao)

Many IoV applications are computation-intensive and latency-sensitive (e.g., autonomous driving, high-definition video analytics, and cooperative vehicular perception), where multi-sensor data from on-board cameras, LiDAR, and millimeter-wave radars must be processed in real time [2]. The deep neural networks (DNNs) that underpin these applications are typically large-scale; for example, ResNet-152 requires roughly 22.6 billion operations for a single image classification, and processing high-definition video sequences can exceed 30 trillion operations [3]. Relying solely on on-board computing for such workloads leads to prohibitive inference delays, undermining safety and user experience.

To mitigate this bottleneck, vehicular edge computing (VEC) has been proposed as an effective task-offloading paradigm [4], [5]. Unlike centralized cloud computing, VEC pushes computation and storage to the network edge. Thus, tasks can be partitioned and jointly executed across roadside units (RSUs) via vehicle-to-infrastructure (V2I) links and nearby vehicles via vehicle-to-vehicle (V2V) links. This edge-proximal architecture significantly reduces end-to-end latency and alleviates backhaul congestion, enabling timely execution of computation-intensive and latency-sensitive IoV applications [6].

Furthermore, the limited spectrum resource has become a major bottleneck for IoV, which must support ultra-reliable communications and high-throughput data exchange for real-time services. For instance, modern autonomous vehicles rely on a large number of onboard sensors that can produce over 8 Gbps of data [7]. Integrated sensing and communication (ISAC) has been recognized as a key enabler for 6G to address this limitation, as it performs both communications and environment sensing within the same spectrum [8]. Unlike the traditional paradigm of “sensing–communication separation,” ISAC enables nodes to perform both communication and sensing, which not only improves spectrum efficiency but also expands the effective communication bandwidth, thereby increasing data transmission rates [9].

However, ISAC-assisted IoV still suffers from reactive decision-making due to the lack of prediction capability. While ISAC can sense the current communication and environmental conditions, it cannot foresee future mobility and workload variations. As a result, task offloading and resource scheduling remain prone to inefficiency under high mobility. Digital twin (DT), as a pivotal tool for cyber–physical integration, addresses this gap by offering global visibility and predictive intelligence [10]. By constructing high-fidelity virtual replicas of vehicles, RSUs, and network resources, DT enables real-time synchronization of physical states and supports predictive

modeling of task dynamics. For instance, DT can predict vehicle trajectories and task load distributions, thereby facilitating proactive optimization of offloading and resource allocation [11]. Consequently, combining DT with ISAC provides a more comprehensive network optimization framework for next-generation intelligent transportation systems.

### B. Motivation and Main Challenges

Although integrating DT and ISAC technologies into the IoV offers significant potential, several fundamental challenges remain unresolved.

- **First, effectively exploiting ISAC to enhance communication in IoV is a critical challenge.** The tight coupling between sensing and communication creates nontrivial trade-offs between throughput and perception accuracy, making resource allocation highly state-dependent and more difficult to stabilize [12]. These issues are further exacerbated in dense, high-mobility scenarios, where simultaneous demands for high-rate transmission and precise sensing significantly increase scheduling uncertainty [13]. To address this, we leverage ISAC's dual functionality to support two transmission modes: raw data transmission (DataT) and instruction transmission (InstrT), where RSUs reconstruct sensory data using their own perception. This mechanism effectively reduces transmission overhead and latency while ensuring task execution accuracy.
- **Second, minimizing latency and energy consumption while ensuring long-term stability poses a fundamental challenge.** In many practical deployments, both vehicles and RSUs are subject to limited energy budgets [14], [15]. Under highly dynamic mobility and time-varying task arrivals, excessive short-term energy consumption can easily lead to queue backlogs and degrade service reliability. Without foresight of future system dynamics, making consistent resource allocation decisions across time slots becomes extremely difficult, especially when simultaneously pursuing low delay and energy efficiency [16]. To address this challenge, we employ a Lyapunov optimization framework that decomposes the long-term stochastic control problem into tractable per-slot decisions, dynamically balancing delay reduction and energy consumption while guaranteeing queue stability over time. This design ensures that vehicles and RSUs can sustain continuous operation without sacrificing responsiveness in highly dynamic IoV environments.
- **Third, achieving efficient and adaptive task offloading in highly dynamic IoV environments remains a major challenge.** Vehicular tasks are heterogeneous and delay-sensitive, while high mobility leads to frequent topology variations and fluctuating workloads [10], [17]. Traditional optimization methods (e.g., convex optimization and heuristic algorithms) struggle to make real-time decisions in such environments. The deep reinforcement learning (DRL) provides adaptability through environment interaction mechanisms, but agents relying solely on local observations may produce unstable or short-sighted actions under partial observability. To tackle this

challenge, we integrate DT into the DRL framework under the centralized training and decentralized execution (CTDE) paradigm, where synchronized global states are provided during training. This improves policy awareness of system dynamics and enhances decision robustness in rapidly changing scenarios (as validated in Sec. VII).

### C. Summary of Contributions

In this paper, motivated by the above challenges, we investigate DT-assisted task offloading and resource allocation in ISAC-enabled IoV. Our objective is to minimize the long-term system cost, defined as a weighted sum of the average delay and energy consumption, while ensuring queue stability in highly dynamic vehicular environments. The main contributions of this work are summarized as follows:

- **Framework:** We formulate a long-term joint task offloading and resource allocation problem in ISAC-enabled IoV, where vehicles and RSUs adopt dual transmission modes (i.e., DataT and InstrT), with the latter reducing data volume and improving spectral efficiency. We model the problem as a mixed-integer nonlinear programming (MINLP) formulation, which is known to be NP-hard. It is particularly challenging to solve under high vehicle mobility and time-varying task arrivals.
- **Solution:** To address this challenge, we first employ the Lyapunov optimization technique to transform the original long-term coupled objective into a sequence of tractable per-slot subproblems, ensuring queue stability and sustainable energy consumption. Building on this transformation, we propose a Lyapunov-driven DT-enhanced multi-agent proximal policy optimization (Ly-DTMPO) algorithm under the CTDE paradigm. The algorithm integrates DT-enhanced global visibility into on-policy reinforcement learning and embeds Lyapunov stability awareness into the reward design, achieving faster convergence, superior stability, and improved adaptability compared with baselines.
- **Validation:** We conduct extensive simulations to validate the proposed framework. The results show that Ly-DTMPO achieves faster convergence, better stability, and greater reductions in both delay and energy consumption compared with Lyapunov-driven multi-agent baselines, thereby attaining the lowest overall system cost. Moreover, the comparison with its DT-absent counterpart confirms that the proposed DT module significantly enhances decision stability and improves the effectiveness of computation offloading and resource scheduling in dynamic vehicular environments.

### D. Paper Organization

The remainder of this paper is organized as follows. Sec. II reviews related works on DT-assisted ISAC in IoV and Lyapunov-based optimization approaches. Sec. III introduces the system model, including DT-assisted vehicular scenarios and ISAC-enabled communication, computing, and queue models. Sec. IV formulates the joint optimization problem with latency and energy constraints. Sec. V presents the

Lyapunov-based problem decomposition method. Sec. VI develops the proposed Ly-DTMPO algorithm under a multi-agent reinforcement learning framework. Sec. VII evaluates system performance through simulations and comparative analysis. Finally, Sec. VIII concludes the paper and discusses future research directions.

## II. RELATED WORK

Henceforth, we review the most relevant studies and identify the research gaps that motivate the present work.

### A. ISAC-Enabled Vehicular Networks

As a fundamental technology for 6G, ISAC enables simultaneous communication and sensing over shared spectrum. The authors in [18] developed an ISAC-enabled V2X MEC framework to reduce long-term service delay via joint offloading and resource allocation. A three-tier ISCC architecture utilizing MEC and cloud servers was proposed in [19] to improve latency and energy performance for sensing tasks. In addition, a data-fusion-based ISAC approach was designed in [20] to optimize long-term delay–energy trade-offs, and [21] proposed an ISAC-driven association strategy to expand sensing range and reduce communication latency in ad hoc networks.

Despite these advancements, existing studies largely concentrate on physical-layer signal processing or short-term trade-off optimization. The interaction between ISAC and higher-layer decision-making, such as cross-layer resource allocation, task scheduling, and long-term system stability, remains insufficiently addressed.

### B. DT-Assisted ISAC in IoV

DT has emerged as a key technology for the IoV, offering global visibility and predictive modeling of network states. The authors in [11] proposed a DT-driven edge–end scheduling scheme that improves heterogeneous task completion via accurate resource estimation. The work in [22] examined the trade-off between maintaining DT models and executing tasks, highlighting DT’s role in balancing scarce computing resources. A collaborative driving architecture leveraging DT for real-time vehicular interaction was presented in [23]. Additionally, Li et al. [24] investigated DT-assisted integrated sensing–communication–computation, where DT predictions assist in offloading decisions to balance sensing accuracy and computational efficiency.

However, most existing DT-assisted IoV studies focus primarily on architecture design or predictive scheduling, while only a limited number explicitly address the tight coupling among sensing, communication, and computing resources in ISAC-enabled environments [25]. The integration of DT and ISAC for stable long-term resource allocation under high mobility remains largely underexplored.

### C. Resource Allocation for ISAC-Enabled IoV

Efficient resource allocation is essential for ISAC-enabled IoV, where spectrum, computing, and power resources must be coordinated to support diverse and latency-critical tasks.

The authors in [26] optimized SDMA-based scheduling to maximize throughput in ISAC-enabled IoV. Chen et al. [27] introduced the ISCC paradigm with TSFC resource decoupling managed through A2GNN. The work in [18] jointly optimized offloading and resource allocation to reduce long-term delay, and [20] leveraged Lyapunov optimization to balance queue latency and energy in cooperative perception.

Although effective, these studies largely emphasize instantaneous or simulation-driven optimization and often simplify the sensing–communication coupling, leaving long-term queue and energy stability underexplored. This gap motivates stability-aware, cross-layer resource allocation tailored to DT-enabled ISAC-aided IoV.

### D. Lyapunov Optimization-Based Approaches on IoV

Lyapunov optimization is widely used in IoV to guarantee queue stability while optimizing long-term metrics. The authors in [28] applied it to ISAC networks for joint bandwidth and power allocation to reduce latency and improve sensing. In the MEC context, [13] adopted a Lyapunov-based multi-agent approach for task and resource coordination, while [16] integrated Lyapunov optimization with GCN-based RL to manage interdependent subtasks. Furthermore, [29] presented LySAC, combining Lyapunov with soft actor–critic to jointly optimize delay and energy.

However, these studies typically rely on simplified assumptions or address only isolated communication–computation trade-offs. The coupling among sensing, communication, and computation, along with the need for predictive global visibility, remains largely unexplored. This motivates our deeper investigation into DT-assisted ISAC-enabled IoV systems using Lyapunov optimization to achieve long-term stability.

## III. SYSTEM MODEL

In this section, we present the DT-assisted ISAC-enabled IoV system model. We first describe the DT-assisted vehicular network architecture and ISAC-enabled communication and computing scenarios, followed by the modeling of task offloading, resource allocation, and queue dynamics. They together lay the foundation for the Lyapunov-based optimization and learning framework developed later.

### A. DT-Assisted IoV Scenarios

As shown in Fig. 1, we introduce a vehicular network scenario that integrates communication, sensing, and computation, and consists of several RSUs deployed along a two-way straight city road segment, where at regular intervals (dependent on the RSU range). These RSUs, equipped with ISAC capabilities and integrated with the MEC server [29], offering computational capacities surpassing those of mobile vehicles, and sensors (e.g. high-definition cameras, LiDAR, and millimeter-wave radar) [30] to sense the information of environment in real time. Adjacent RSUs, interconnected through optical fibers, have the capability to create a DT space to enable a global situational awareness of communication, sensing, and computation states, thereby facilitating accurate

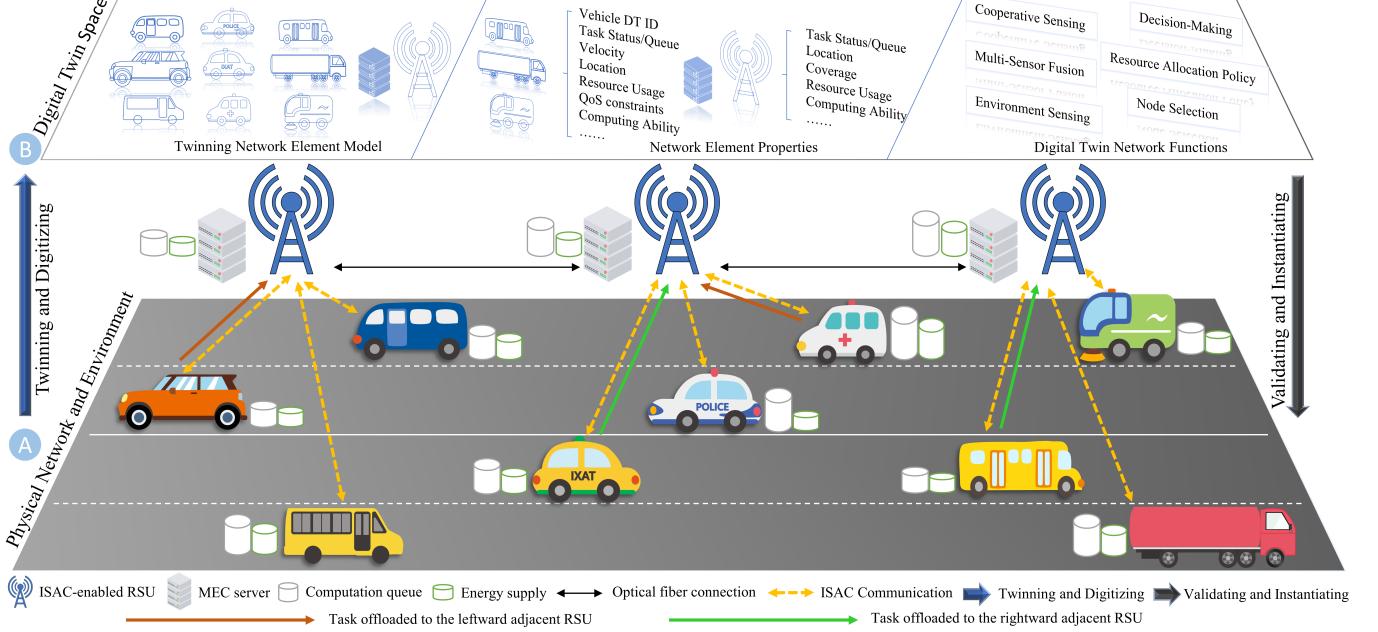


Fig. 1. DT-Assisted ISAC vehicular network scenario. Part A illustrates the physical layer, where ISAC-enabled vehicles and RSUs cooperate in sensing, communication, and computation. Therein, ISAC communication is performed between the vehicle and the RSU. Part B presents the DT layer, which mirrors the physical environment through synchronized digital replicas, supporting predictive control and proactive resource optimization.

and informed decision-making. In our paper, to simultaneously optimize both time-domain and frequency-domain resource allocation, the orthogonal frequency division multiple access (OFDMA) technique is employed to enable concurrent communication between CVs and RSUs. As a result, the interference caused by signal overlapping can be neglected in the subsequent analysis [3], [13]. Moreover, we adopt a time-division duplexing (TDD) mode for the uploading and feedback process.

We adopt a discrete time-slot computational model where the total duration is divided into  $T$  equal time slots, and the duration of each time slot is denoted as  $\tau$ . The set of time slots are represented as  $\mathcal{T} = \{1, \dots, T\}$ . The set of RSUs can be described as  $\mathbb{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_K\}$  ( $k \in \{1, \dots, K\}$ ). Some vehicles carry a computation-intensive and latency-intensive task that takes environmental data as input. These vehicles, regarded as Client Vehicles (CVs), can be described as  $\mathbb{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_I\}$  ( $i \in \{1, \dots, I\}$ ). The details of the vehicle and RSU DT are as follows:

1) *CV-Twin Body*: CV digital twins include task data volume  $G_i^{\text{TasD}}$  and computation instruction data volume  $G_i^{\text{InsD}}$ , computation capability  $C_{CV_i}^{\text{cpu}}$ , vehicle DT ID  $DT - \text{Number}_{CVs}$ , coordinates  $L_i^{\text{cv}}$ , the speed of vehicle  $v_i^{\text{cv}}$ , and the maximum tolerable latency  $T_i^{\text{max}}$ , where  $\mathcal{C}_i \in \mathbb{C}$ .

2) *RSU-Twin Body*: RSU digital twins include RSU DT ID  $DT - \text{Number}_{\text{RSU}}$ , coordinates  $L_k^{\text{rsu}}$ , coverage area (i.e. radius)  $Range_k^{\text{rsu}}$ , computing resources  $F_k^{\text{rsu,max}}$ , where  $\mathcal{R}_k \in \mathbb{R}$ .

Unlike conventional IoV models where vehicles can only access local information, the DT space in our framework provides a virtualized global view of the system. Specifically, the DT synchronizes the physical states of vehicles and

RSUs in real time and maintains virtual queues that mirror the task backlog and resource utilization in RSUs. These queues maintained by the DT are not directly observable by vehicles in practice, but become available to the intelligent decision-making agents through the DT space. This enables predictive modeling of mobility and task dynamics, allowing the reinforcement learning algorithm to exploit global context information for more accurate task offloading and resource allocation.

Accordingly, the DT not only records static parameters (e.g., location and computing capacity) but also functions as a dynamic estimator and predictor, which bridges the gap between locally observable states and globally optimal scheduling strategies. In our practical implementation, the DT further provides the actor network in the CTDE framework with a global perspective, enabling agents to make more informed and coordinated decisions. The effectiveness of this DT-assisted design is validated in our experiments, as detailed in Sec. VII-C. For ease of reference, Table I summarizes the key notations used in the system model.

### B. ISAC-Enabled IoV Computing Scenarios

In Sec. III-A, with the continuous updates, this twin body updates in real-time to reflect the state of the physical object [25]. Therefore, RSU-twin bodies and vehicles-twin bodies in this paper can be regarded as ISAC nodes. Both RSUs and vehicles that are equipped with sensors, their have the ability to get environmental information. Therefore, twin bodies of neighboring RSUs adjacent to a CV may collect similar environmental sensing data. In this case, CVs and RSUs can jointly provide both communication and sensing functions,

TABLE I  
LIST OF KEY NOTATIONS

Symbol	Description
$b(t)$	Bandwidth allocation decision
$\mathcal{C}$	The vehicle set
$C_{CV_i}^{\text{cpu}}$	Local computing capacity of vehicle $C_i$
$\mathcal{L}(t)$	The system-level cost function
$d_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)$	The Euclidean distance between CV $C_i$ and RSU $\mathcal{R}_k$
$D_i^{\text{loc}}(t)$	The local processing latency in CV $C_i$
$D_{\text{rsu}}(t)$	The edge processing latency in RSU $\mathcal{R}_k$
$D_i^{\text{queue,n}}(t)$	The queueing delay
$E_i^{\text{loc}}(t)$	The local processing energy consumption in CV $C_i$
$E_{i,k}^{\text{DataT-up}}(t)$	The energy consumption during task uploading in DataT mode
$E_{i,k}^{\text{co-tra}}(t)$	The energy consumption of the RSU execute coordinate transformation
$E^{\text{task}}(t)$	The total energy consumed by all nodes
$F_{\text{RSU}_k}^{\text{cpu}}(t)$	Computational resource of the RSU allocated to its offloading task at time slot $t$
$G_i^{\text{TasD}}$	Data volume of the task
$G_i^{\text{InsD}}$	Data volume of computation instruction
$I_c^{\text{task}}$	The number of CPU cycles required to process a bit
$\ell(t)$	Task assignment decision
$Q_i^{\text{loc}}(t)$	The task queue length of vehicle $C_i$ at time slot $t$
$Q_k^{\text{rsu}}(t)$	The task queue length of the RSU $\mathcal{R}_k$ at time slot $t$
$\mathbb{R}$	The RSU set
$R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)$	The V2I transmission rate from CV $C_i$ to RSU $\mathcal{R}_k$
$\mathcal{T}$	Index set of time slots
$T_i^{\text{max}}$	The maximum tolerated delay of the task completion
$T_i^{\text{task}}(t)$	The overall latency of all tasks executed in parallel
$V_i(t)$	The virtual energy queue at time slot $t$
$\Psi_i^{\text{CV}}$	Task of vehicle $C_i$ at time slot $t$
$\Theta$	Communication gains
$\eta_i^{\text{rsu}}(t)$	The mode selector

which further enable diverse computing scenarios in ISAC-assisted IoV systems. This redundancy enables cooperative mechanisms where CVs do not necessarily need to upload raw data for every task.

Specifically, we introduce two available transmission modes for CVs to offload their tasks to RSU-twin bodies:

- **DataT Mode:** the CV directly uploads the full raw sensory data  $G_i^{\text{TasD}}(t)$  (e.g., high-definition video and LiDAR point clouds) to the RSU. The offloading time depends on the transmission rate and the size of the raw task data. After the RSU finishes computation, the output results are sent back to the CV, where the feedback delay depends on the size of the result data and the transmission rate.
- **InstrT Mode:** instead of transmitting massive raw data, the CV only uploads computation instructions with significantly smaller size  $G_i^{\text{InsD}}(t)$ . Since the RSU itself collects similar environmental data from its own sensors, it can reconstruct the task input using its local perception data after receiving the instructions [25]. However, due to the different perspectives between the CV and RSU, the raw environmental data need to be aligned through coordinate transformation. This process can be realized by matrix operations using the coordinate information of the CV contained in the computation instructions, as suggested in [25], [31]. The corresponding computational intensity for this preprocessing is denoted by  $I_c^{\text{co-task}}$ . Once aligned, the RSU executes the task using its own

perception data, and returns the result to the CV.

By leveraging ISAC's dual capability, the InstrT mode can significantly reduce transmission overhead and latency compared to DataT, while still ensuring accurate task execution. This ISAC-assisted collaborative mechanism allows tasks to be executed in parallel across RSU-twin bodies, thereby improving both resource utilization and service efficiency in dynamic vehicular environments. In this paper, we focus on scenarios where CVs generate tasks, while RSUs serve as the primary service nodes for task execution. The choice between DataT and InstrT modes becomes a crucial part of the task offloading strategy, and will be jointly optimized with resource allocation in the proposed Lyapunov-driven framework.

### C. Sensing-Enhanced Communication Model

In this paper, we consider V2I communication models. Moreover, similar to [29], we consider the collaborative benefits between communication and sensing functions (i.e. sensing-enhanced communication model). Let the enhancement factor  $\Theta = R_{\text{sen-enhance}}/R_{\text{normal}}$  represent the ratio of the communication rate under sensing-enhanced to the normal communication rate.

**V2I Communication:** Let  $R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)$  be the data transmission rate from CV  $C_i$  to the RSU  $\mathcal{R}_k$  at time slot  $t$ :

$$R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) = \Theta b_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \mathcal{B} \log_2 \left( 1 + \frac{P_i \left| h_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \right|^2 g_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)}{\sigma_0^2} \right), \quad (1)$$

where  $b_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \mathcal{B}$  denotes the communication bandwidth between the CV  $C_i$  and RSU  $\mathcal{R}_k$ ,  $b_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \in [0, 1]$  is the bandwidth allocation ratio from CV  $C_i$  to RSU  $\mathcal{R}_k$  at time slot  $t$ ,  $P_i$  is the transmit power of CV  $C_i$ , and  $\sigma_0^2$  is the additive white gaussian noise (AWGN). Additionally,  $h_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \sim \mathcal{CN}(0, 1)$  is the small-scale fading component between CV  $C_i$  and RSU  $\mathcal{R}_k$  at time slot  $t$ . In addition,  $g_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)$  is the path loss of the large-scale fading component between CV  $C_i$  and RSU  $\mathcal{R}_k$ , calculated as  $-38.4 - 21.0 \log_{10}(d_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t))$  [32], where  $d_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)$  represents the time-varying distance between CV  $C_i$  and RSU  $\mathcal{R}_k$  at time slot  $t$ .

In addition, due to the high mobility of vehicles, we model the real-time 3D Euclidean distance between vehicle and RSU, to capture the spatial and temporal variation in V2I distance. The Euclidean distance between CV  $C_i$  and RSU  $\mathcal{R}_k$  at time slot  $t$  is given by

$$d_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) = \sqrt{|x_k^{\text{rsu}} - x_i(t)|^2 + (y_i^{\text{cv}} - l_k)^2 + H_k^2}, \quad (2)$$

where  $x_i(t) = x_i + v_i^{\text{cv}} t$  is the CV's longitudinal position.  $y_i^{\text{cv}}$  denotes the lateral position determined by the lane it occupies. The coordinate of RSU  $\mathcal{R}_k$  is described as  $L_k^{\text{rsu}}$  (i.e. a tuple  $(x_k^{\text{rsu}}, l_k, H_k)$ ), where  $x_k^{\text{rsu}}$ ,  $l_k$ , and  $H_k$  represent the RSU's longitudinal position, RSU' lateral offset from the road centerline, and the height of RSU, respectively. Similar to [25], we can ignore the relative moving distance between CV  $C_i$  and RSU  $\mathcal{R}_k$ . Therefore, the Euclidean distance between CV  $C_i$  and RSU  $\mathcal{R}_k$  can be updated by

$$d_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) = \sqrt{|x_k^{\text{rsu}}|^2 + (y_i^{\text{cv}} - l_k)^2 + H_k^2}. \quad (3)$$

#### D. Computing Model

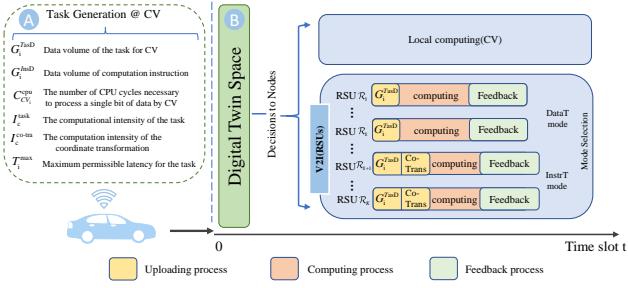


Fig. 2. Total task completion latency. Here, “Co-Trans” stands for coordinate transformation. Part A illustrates task generation at the CV, where data volume and computational attributes are defined. Part B shows the DT space that decides task-offloading ratios between local and RSU nodes to achieve low-latency cooperative computation.

As shown in Fig. 2, at the beginning of each time slot  $t$ , CVs generate a task, i.e.

$$\Psi_i^{\text{CV}} = \{G_i^{\text{TasD}}(t), G_i^{\text{InsD}}(t), C_{CV_i}^{\text{cpu}}, I_c^{\text{task}}, I_c^{\text{co-task}}, T_i^{\text{max}}\}, \quad (4)$$

where  $G_i^{\text{TasD}}(t)$  indicates the data volume of the task for CV  $C_i$  at the time slot  $t$ .  $G_i^{\text{InsD}}(t)$  represents the data volume of computation instruction.  $C_{CV_i}^{\text{cpu}}$  (in cycles/s) denotes the number of CPU cycles per second that the CV  $C_i$  can execute.  $I_c^{\text{task}}$  (in cycles/bit) denotes the computational intensity of a task (i.e., the number of CPU cycles necessary to process a single bit of task data).  $I_c^{\text{co-task}}$  (in cycles/bit) is the computation intensity of the coordinate transformation.  $T_i^{\text{max}}$  represents the maximum tolerable latency for the task completion.

According to Sec. III-B, compared with data volume of the task, the task description and corresponding decisions are very small. As a result, these two transmission time can be negligible [25]. As shown in Fig. 2, tasks generated by CV will be computed in parallel on the local and RSU.

1) *Local Processing*: At time slot  $t$ , when a subtask is executed locally, the local processing latency is the local computation time. The local processing latency  $D_i^{\text{loc}}(t)$  and the energy consumption of locally executed tasks  $E_i^{\text{loc}}(t)$ , are given by

$$D_i^{\text{loc}}(t) = \frac{\ell_i^{\text{loc}}(t) G_i^{\text{TasD}}(t) I_c^{\text{task}}}{C_{CV_i}^{\text{cpu}}}, \quad (5)$$

$$E_i^{\text{loc}}(t) = \kappa_{cv} D_i^{\text{loc}}(t) (C_{CV_i}^{\text{cpu}})^3, \quad (6)$$

where  $\ell_i^{\text{loc}}(t)$  represents the ratio of tasks assigned to local.  $\kappa_{cv}$  is the energy consumption parameter for vehicle computing, which is a hardware-dependent constant.

2) *Edge Processing at RSU*: When the task is executed at RSU, the CV needs to consider whether to select DataT mode or InstrT mode. Additionally, the data volume of the task or computing instructions will be transmitted to the RSU. After the RSU completes the computation task, the calculation results will be returned to the CV. According to [25], [33], [34], the size of the output result is much smaller than the input data, so we ignore the feedback delay. Therefore, the RSU’s task processing latency will further consist of uploading time and processing time, which are formalized below.

- *Uploading Time*: If the CV has selected the DataT mode, the uploading time can be calculated by

$$D_{k,DataT}^{\text{rsu}}(t) = \frac{\ell_{i,k}^{\text{rsu}}(t) G_i^{\text{TasD}}(t)}{R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)}, \quad (7)$$

where  $\ell_{i,k}^{\text{rsu}}(t)$  is the ratio of tasks assigned from CV  $C_i$  to RSU  $\mathcal{R}_k$  at time slot  $t$ .

Then,  $E_{i,k}^{\text{DataT-up}}$  is the energy consumption during task uploading of DataT mode, which is given by

$$E_{i,k}^{\text{DataT-up}}(t) = D_{k,DataT}^{\text{rsu}}(t) P_i(t). \quad (8)$$

The uploading time for InstrT mode is calculated by

$$D_{k,InstrT}^{\text{rsu}}(t) = \frac{G_{i,k}^{\text{InsD}}(t)}{R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)}, \quad (9)$$

where  $G_{i,k}^{\text{InsD}}(t)$  represents the size of computation instruction allocated to RSU  $\mathcal{R}_k$  at time slot  $t$ .

The uploading time for the InstrT model primarily depends on the coordinate transformation time and  $E_{i,k}^{\text{co-tra}}(t)$  represents the energy consumption of the RSU executed coordinate transformation. They are given by

$$D_{k,InstrT}^{\text{co-tra}}(t) = \frac{\ell_{i,k}^{\text{rsu}}(t) G_i^{\text{TasD}}(t) I_c^{\text{co-tra}}}{F_{RSU_k}^{\text{cpu}}(t)}, \quad (10)$$

$$E_{i,k}^{\text{co-tra}}(t) = \kappa_{rsu} D_{k,InstrT}^{\text{co-tra}}(t) (F_{RSU_k}^{\text{cpu}}(t))^3, \quad (11)$$

where  $\kappa_{rsu}$  is the energy consumption parameter for RSU computing.  $I_c^{\text{co-tra}}$  (in cycles/bit) is the computation intensity of the coordinate transformation.  $F_{RSU_k}^{\text{cpu}}(t)$  (in cycles/s) denotes the computation resources of the RSU  $\mathcal{R}_k$  allocated to the offloaded task at time slot  $t$ .

The DT space will choose the one with smaller uploading time between DataT mode and InstrT model. Hence, the uploading time can be defined by

$$D_{i,k}^{\text{up}}(t) = \min\{D_{k,DataT}^{\text{rsu}}(t), D_{k,InstrT}^{\text{co-tra}}(t)\}. \quad (12)$$

- *Computing Time*: After uploading, tasks offloaded to the RSU  $\mathcal{R}_k$  will be processed in parallel on the RSU  $\mathcal{R}_k$ . The RSU processing latency  $D_{i,k}^{\text{compt}}(t)$  and energy consumption of RSU executing offloading task  $E_{i,k}^{\text{compt,rsu}}(t)$  are described as

$$D_{i,k}^{\text{compt}}(t) = \frac{\ell_{i,k}^{\text{rsu}}(t) G_i^{\text{TasD}}(t) I_c^{\text{task}}}{F_{RSU_k}^{\text{cpu}}(t)}, \quad (13)$$

$$E_{i,k}^{\text{compt,rsu}}(t) = \kappa_{rsu} D_{i,k}^{\text{compt}}(t) (F_{RSU_k}^{\text{cpu}}(t))^3. \quad (14)$$

To summarize, the total latency of edge processing on the RSU  $\mathcal{R}_k$  is given by

$$D_{i,k}^{\text{rsu}}(t) = D_{i,k}^{\text{up}}(t) + D_{i,k}^{\text{compt}}(t). \quad (15)$$

#### E. Queuing Model

In our system, to accurately capture the computational load dynamics across heterogeneous nodes. We establish independent task queues for CVs and RSUs. The queue evolution for each type of node is governed by the discrete-time dynamics

based on task arrivals and the available computation resource within time slot duration  $\tau$ .

- *Local Queue*

For each CV  $\mathcal{C}_i$ , the local task queue is maintained privately and only accounts for the portion of tasks processed locally. The queue is updated as:

$$Q_i^{\text{loc}}(t+1) = \max\{Q_i^{\text{loc}}(t) - \underbrace{C_{CV_i}^{\text{cpu}}\tau}_{\text{(I)}} + \underbrace{\lambda_i^{\text{loc}}(t)}_{\text{(II)}}, 0\}, \quad (16)$$

where  $Q_i^{\text{loc}}(t)$  is the backlog of computations in the local task queue of CV  $\mathcal{C}_i$  at time slot  $t$ . The term (I) indicates the amount of computation locally executable within time slot duration  $\tau$ , and the term (II) represents the newly arrived computations of tasks assigned to the local, therein  $\lambda_i^{\text{loc}}(t) = \ell_i^{\text{loc}}(t)G_i^{\text{TasD}}(t)I_c^{\text{task}}$ .

- *RSU Queue*

In contrast, RSUs as shared edge computing nodes and may simultaneously receive offloaded task segments from multiple CV  $\mathcal{C}_i$ . The computation queues maintained by RSU  $\mathcal{R}_k$  must aggregate all arriving task arrivals from different CVs:

$$Q_k^{\text{rsu}}(t+1) = \max\{Q_k^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau + \underbrace{\sum_{i=1}^I \lambda_i^{\text{rsu}}(t)}_{\text{(I)}}, 0\}, \quad (17)$$

where  $Q_k^{\text{rsu}}(t)$  denotes the backlogged computations at the task queue of the RSU  $\mathcal{R}_k$  and the term (I) represents the newly arrived computations from all CV  $\mathcal{C}_i$  that offload their task to the RSU  $\mathcal{R}_k$  at time slot  $t$ . Here  $\sum_{i=1}^I \lambda_i^{\text{rsu}}(t) = \ell_{i,k}^{\text{rsu}}(t)G_i^{\text{TasD}}(t)[\eta_i^{\text{rsu}}(t)I_c^{\text{co-tra}} + I_c^{\text{task}}]$ , therein,  $\eta_i^{\text{rsu}}(t) \in \{0, 1\}$  is a mode selector. According to (12), if InstrT mode has lower total uploading time,  $\eta_i^{\text{rsu}}(t)$  will be equal to 1, and otherwise  $\eta_i^{\text{rsu}}(t) = 0$ .

After modeling the evolution of task queues at CVs and RSUs, we proceed to estimate the corresponding queueing delay encountered by tasks assigned to each node. In order to capture the average waiting time before execution, we apply Little's Law, a widely adopted principle in queueing theory, which states that the expected delay in a stable system is proportional to the average number of tasks in the queue divided by the average arrival rate [35].

Specifically, we define the queueing delay at a given node  $n \in \{\text{loc}, \text{rsu}\}$  for a task originated by CV  $\mathcal{C}_i$  as the ratio between the time-averaged queue length and the time-averaged arrival computation load.

Let  $Q_i^{\text{n}}(t)$  and  $\lambda_i^{\text{n}}(t)$  denote the queue length and task arrival load at time slot  $t$ , the queueing delay is expressed as:

$$D_i^{\text{queue,n}}(t) = \frac{\widetilde{Q}_i^{\text{n}}(t)}{\widetilde{\lambda}_i^{\text{n}}(t)}, \quad (18)$$

where  $\widetilde{Q}_i^{\text{n}}(t) = \frac{1}{m} \sum_{l=t-m+1}^t Q_i^{\text{n}}(l)$  and  $\widetilde{\lambda}_i^{\text{n}}(t) = \frac{1}{m} \sum_{l=t-m+1}^t \lambda_i^{\text{n}}(l)$  are the moving average queue length and arrival rate computed over a sliding time window of size  $m$ , respectively.

Consequently,  $T_i^{\text{task}}(t)$  (i.e., the maximum of local partial latency and RSU partial latency) is taken as the overall latency of all tasks executed in parallel.  $E_i^{\text{task}}(t)$  denotes the total energy consumed by all nodes (for both CVs and RSUs) involved in executing the task initiated by  $\mathcal{C}_i$  at time slot  $t$ . They can be defined as follows:

$$T_i^{\text{task}}(t) = \max\{T_i^{\text{loc}}(t), T_{i,k}^{\text{rsu}}(t)\}, \quad (19)$$

$$E_i^{\text{task}}(t) = E_i^{\text{loc}}(t) + \sum_{k=1}^K [E_{i,k}^{\text{up,rsu}}(t) + E_{i,k}^{\text{compt,rsu}}(t)], \quad (20)$$

where  $T_i^{\text{loc}}(t) = D_i^{\text{loc}}(t) + D_i^{\text{queue,loc}}(t)$  is the local part total latency.  $T_{i,k}^{\text{rsu}}(t) = D_{i,k}^{\text{rsu}}(t) + D_i^{\text{queue,rsu}}(t)$ ,  $\mathcal{R}_k \in \mathbb{R}$  denotes the RSU part total latency.

Combining (5), (15), and (18), the overall latency is updated by

$$\begin{aligned} T_i^{\text{task}}(t) &= \max\{T_i^{\text{loc}}(t), T_{i,k}^{\text{rsu}}(t)\}, \\ T_i^{\text{loc}}(t) &= \frac{\ell_i^{\text{loc}}(t)G_i^{\text{TasD}}(t)I_c^{\text{task}}}{C_{CV_i}^{\text{cpu}}} + \frac{\widetilde{Q}_i^{\text{loc}}(t)}{\widetilde{\lambda}_i^{\text{loc}}(t)}, \\ T_{i,k}^{\text{rsu}}(t) &= \min\left\{\frac{\ell_{i,k}^{\text{rsu}}(t)G_i^{\text{TasD}}(t)}{R_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t)}, \frac{\ell_{i,k}^{\text{rsu}}(t)G_i^{\text{TasD}}(t)I_c^{\text{co-tra}}}{F_{RSU_k}^{\text{cpu}}(t)}\right\} \\ &\quad + \frac{\ell_{i,k}^{\text{rsu}}(t)G_i^{\text{TasD}}(t)I_c^{\text{task}}}{F_{RSU_k}^{\text{cpu}}(t)} + \frac{\widetilde{Q}_i^{\text{rsu}}(t)}{\widetilde{\lambda}_i^{\text{rsu}}(t)} (1 \leq k \leq K). \end{aligned} \quad (21)$$

The energy consumption for uploading tasks depends on the selected transmission mode. To ensure consistency, we refine the mode selector  $\eta_i^{\text{n}'}(t)$  mentioned in Sec. III-E and combine (12), defined as

$$\eta_i^{\text{rsu}}(t) = \begin{cases} 1, & D_{i,\text{DataT}}^{\text{rsu}}(t) < D_{i,\text{InstrT}}^{\text{co-tra}}(t) \\ 0, & \text{otherwise} \end{cases}. \quad (22)$$

Hence, the total energy consumption can be updated by

$$\begin{aligned} E_i^{\text{task}}(t) &= E_i^{\text{loc}}(t) \\ &\quad + \sum_k [(1 - \eta_i^{\text{rsu}}(t))E_{i,k}^{\text{DataT-up}}(t) \\ &\quad + \eta_i^{\text{rsu}}(t)E_{i,k}^{\text{co-tra}}(t) + E_{i,k}^{\text{compt,rsu}}(t)]. \end{aligned} \quad (23)$$

#### IV. PROBLEM FORMULATION

To achieve a practical and interpretable system objective, we formulate the optimization problem in terms of total cost, which jointly considers the task latency and the system-wide energy consumption incurred to support each task. Specifically, for each client vehicle  $\mathcal{C}_i$ , we define its service-related cost as:

$$\mathfrak{C}_i(t) = \alpha T_i^{\text{task}}(t) + (1 - \alpha)E_i^{\text{task}}(t), \quad (24)$$

where  $\alpha \in [0, 1]$  is a configurable weight to flexibly balance latency and energy consumption.

Based on this, the system-level cost is defined as the sum over all CVs as follows:

$$\mathfrak{C}(t) = \sum_{i \in \mathcal{C}} \mathfrak{C}_i(t). \quad (25)$$

The system-wide objective is then to minimize the cumulative cost over all CVs, which can be formulated as

$$\begin{aligned}
\mathcal{P}_1 : \quad & \min_{\{\mathbf{b}(t), \ell(t), F(t)\}} \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[\mathbb{C}(t)], \quad (26) \\
\text{s.t.} \quad & \text{C1 : } \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[Q_i^{\text{loc}}(t)] < \infty, \quad \forall \mathcal{C}_i \in \mathbb{C}, \\
& \text{C2 : } \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[Q_k^{\text{rsu}}(t)] < \infty, \quad \forall \mathcal{R}_k \in \mathbb{R}, \\
& \text{C3 : } \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \mathbb{E}[E^{\text{total}}(t)] \leq E^{\text{max}}, \\
& E^{\text{total}}(t) = \sum_{\mathcal{C}_i \in \mathbb{C}} E_i^{\text{task}}(t), \\
& \text{C4 : } 0 \leq b_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \leq 1, \quad \forall \mathcal{R}_k \in \mathbb{R}, \\
& \sum_{\mathcal{R}_k \in \mathbb{R}} b_{i,k}^{\text{veh} \rightarrow \text{rsu}}(t) \leq 1, \quad \forall \mathcal{C}_i \in \mathbb{C}, \forall t \in \mathcal{T}, \\
& \text{C5 : } 0 \leq \ell_i^{\text{loc}}(t), \ell_{i,k}^{\text{rsu}}(t) \leq 1, \quad \forall \mathcal{R}_k, \\
& \ell_i^{\text{loc}}(t) + \sum_{\mathcal{R}_k \in \mathbb{R}} \ell_{i,k}^{\text{rsu}}(t) = 1, \quad \forall \mathcal{C}_i \in \mathbb{C}, \forall t \in \mathcal{T}, \\
& \text{C6 : } 0 \leq F_{RSU_k}^{\text{cpu}}(t) \leq F_k^{\text{rsu}, \text{max}} - C_k^{\text{twin}}(t), \\
& \forall \mathcal{R}_k \in \mathbb{R}, \forall t \in \mathcal{T}, \\
& \text{C7 : } \eta_i(t) \in \{0, 1\}, \quad \forall \mathcal{C}_i \in \mathbb{C}, \forall t \in \mathcal{T}, \\
& (0: \text{DataT mode}, 1: \text{InstrT mode}), \\
& \text{C8 : } T_i(t) \leq T_i^{\text{max}}, \quad \forall \mathcal{C}_i \in \mathbb{C}, \forall t \in \mathcal{T}.
\end{aligned}$$

where the optimization variables are (i)  $\mathbf{b}(t) = \{b_{i,k}(t)\}_{\mathcal{C}_i \in \mathbb{C}, \mathcal{R}_k \in \mathbb{R}}$  is the bandwidth allocation ratios between CV  $\mathcal{C}_i$  and RSU  $\mathcal{R}_k$ , at time slot  $t$ ; (ii)  $\ell(t) = \{\ell_i^{\text{loc}}(t), \ell_{i,k}^{\text{rsu}}(t)\}_{\mathcal{C}_i \in \mathbb{C}, \mathcal{R}_k \in \mathbb{R}}$  is the task assignment ratio, indicating how the task of  $\mathcal{C}_i$  is distributed among local processing and RSU nodes; (iii)  $F(t) = \{F_{RSU_k}^{\text{cpu}}(t)\}_{\mathcal{R}_k \in \mathbb{R}}$  is the RSU computing resource allocation for subtasks at time  $t$ .

In  $(\mathcal{P}_1)$ , constraints C1 and C2 ensure the long-term stability of task queues at the CVs and RSUs, respectively. Constraint C3 is the long-term average total system energy consumption to be within a predefined system-wide energy budget  $E^{\text{max}}$ . Therein,  $E_i^{\text{task}}(t)$  represents the energy consumption caused by a task initiated by CV  $\mathcal{C}_i$  in the entire system, and  $E^{\text{total}}(t)$  denotes the total system energy consumption caused by all tasks initiated by CVs in time slot  $t$ . Constraint C4 [29] ensures that the bandwidth allocation ratios confined within valid bounds and that the total allocated bandwidth for each CV  $\mathcal{C}_i$  at any given time slot does not exceed the normalized unit bandwidth. Constraint C5 guarantees that the task assignment proportions across all available computing nodes sum to one, with each individual ratio constrained to the interval  $[0, 1]$ . Constraint C6 limits the computing resources allocated by RSUs to ensure that the assigned resources do not exceed the remaining available capacity after reserving a portion  $C_k^{\text{twin}}(t)$  for constructing the DT space. Constraint C7 represents the binary mode selector  $\eta_i(t) \in \{0, 1\}$ , where 0 denotes DataT mode and 1 denotes InstrT mode for CV  $\mathcal{C}_i$ . Constraint C8

ensures that the task execution delay perceived by the CV does not exceed its maximum tolerable latency  $T_i^{\text{max}}$ .

**Remark 1.** The joint optimization problem  $(\mathcal{P}_1)$  involves long-term constraints on queue stability C1 and C2, which inherently couple the task offloading and resource allocation decisions across different time slots. This temporal coupling arises from the dynamic evolution of task queues over time, making the decision space sequentially dependent. Additionally, the coexistence of continuous variables (e.g., task and bandwidth allocation ratios in constraints C4 and C5, and discrete decision variables (e.g., computation mode selector in C7 results in an MINLP structure. Such problems are generally NP-hard and difficult to solve efficiently in real-time, especially under dynamic vehicle mobility and limited prior knowledge of future states.

## V. LYAPUNOV-BASED DYNAMIC LONG-TERM PROBLEM DECOUPLING

To tackle this challenge mentioned above, we employ a Lyapunov optimization approach to reformulate the original long-term problem into a series of tractable per-slot decisions. This enables real-time decision-making without requiring complete knowledge of future system dynamics, while still providing long-term guarantees on the stability and feasibility constraints.

In Lyapunov optimization, physical task queues (C1 and C2) are directly incorporated into the Lyapunov function and drift expressions to ensure long-term queue stability. In contrast, for system-level time-average constraints such as energy consumption C3, which is not related to explicit queueing structures, virtual queues must be introduced to transform these average constraints into queue stability conditions. To handle the long-term system-wide energy constraint C3, we introduce a virtual energy queue  $V_i(t)$  that tracks the cumulative deviation of actual energy usage from the permissible average. Although the time slot index set is defined as  $\mathcal{T} = \{1, \dots, T\}$ , for convenience of Lyapunov analysis, we adopt  $t = 0$  as the initial slot and set the virtual queue state  $V_i(0) = 0$ .

The virtual queue evolves according to

$$V_i(t+1) = \max\{V_i(t) - E^{\text{max}} + E^{\text{total}}(t), 0\}, \quad (27)$$

therein, stabilizing  $V_i(t)$  guarantees that the time-average energy consumption constraint is satisfied in the long run.

To examine the long-term behavior of  $V_i(t)$ , we first relax the max operator for tractability and analyze the inequality:

$$V_i(t+1) - V_i(t) \geq E^{\text{total}}(t) - E^{\text{max}}. \quad (28)$$

Summing both sides of (28) from  $t = 0$  to  $T - 1$ , we obtain:

$$V_i(T) - V_i(0) \geq \sum_{t=0}^{T-1} (E^{\text{total}}(t) - E^{\text{max}}). \quad (29)$$

Dividing both sides by  $T$  and rearranging gives:

$$\frac{1}{T} \sum_{t=0}^{T-1} E^{\text{total}}(t) \leq E^{\text{max}} + \frac{V_i(T) - V_i(0)}{T}. \quad (30)$$

Taking expectations and noting  $V_i(0) = 0$ , we derive:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E^{\text{total}}(t)] \leq E^{\text{max}} + \frac{\mathbb{E}[V_i(T)]}{T}. \quad (31)$$

Hence, if the virtual queue  $V_i(t)$  is mean-rate stable, i.e.,

$$\lim_{T \rightarrow \infty} \frac{\mathbb{E}[V_i(T)]}{T} = 0. \quad (32)$$

Then, the original energy consumption constraint C4 is satisfied [29]:

$$\lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[E^{\text{total}}(t)] \leq E^{\text{max}}. \quad (33)$$

Therefore, stabilizing the virtual queue  $V_i(t)$  ensures the satisfaction of the long-term average energy consumption constraint C3.

We define  $\mathbf{Z}(t)$  as a compact representation of the backlog state at time slot  $t$ , which includes the task queues of the CVs and the RSUs, as well as the virtual queues  $V_i(t)$ :

$$\mathbf{Z}(t) = \{\{Q_i^{\text{loc}}(t)\}_{c_i \in \mathbb{C}}, \{Q_k^{\text{rsu}}(t)\}_{\mathcal{R}_k \in \mathbb{R}}, V_i(t)\}. \quad (34)$$

We define the Lyapunov function as the scalar quadratic measure of queue backlog:

$$L(\mathbf{Z}(t)) = \frac{1}{2} \left[ \sum_{i \in \mathbb{C}} (Q_i^{\text{loc}}(t))^2 + \sum_{k \in \mathbb{R}} (Q_k^{\text{rsu}}(t))^2 + \sum_{i \in \mathbb{C}} V_i(t)^2 \right]. \quad (35)$$

The Lyapunov drift, characterizing the changes in queue states over successive time slots, is derived by applying the Lyapunov function in (35) across two consecutive slots:

$$\Delta(\mathbf{Z}(t)) = \mathbb{E}[L(\mathbf{Z}(t+1)) - L(\mathbf{Z}(t)) \mid \mathbf{Z}(t)]. \quad (36)$$

Leveraging the Lyapunov drift-plus-penalty framework, we formulate a drift-penalty function that jointly incorporates the Lyapunov drift and the objective function of problem  $(\mathcal{P}_1)$ , aiming to minimize the following upper bound of the drift-plus-penalty expression at each time slot as follows:

$$\Lambda(\mathbf{Z}(t)) = \Delta(\mathbf{Z}(t)) + V \cdot \mathbb{E}[\mathbb{C}(t) \mid \mathbf{Z}(t)], \quad (37)$$

where  $V > 0$  is the weight parameter that balances the trade-off between minimizing control cost and ensuring queue stability.

In the following, we obtain an upper bound of  $\Lambda(\mathbf{Z}(t))$  on (37).

*Lemma 1:* For any feasible set of  $\{\mathbf{b}(t), \ell(t), F(t)\}_{t \in \mathcal{T}}$ , which satisfies constraints C1 ~ C8, the Lyapunov drift-plus-penalty expression  $\Lambda(\mathbf{Z}(t))$  can be upper bounded as follows:

$$\begin{aligned} \Lambda(\mathbf{Z}(t)) &= \Delta(\mathbf{Z}(t)) + V \cdot \mathbb{E}[\mathbb{C}(t) \mid \mathbf{Z}(t)] \\ &\leq \mathbb{E} \left[ \sum_{c_i \in \mathbb{C}} Q_i^{\text{loc}}(t) (\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}} \tau) \right. \\ &\quad + \sum_{\mathcal{R}_k \in \mathbb{R}} Q_k^{\text{rsu}}(t) \left( \sum_{c_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t) \tau \right) \\ &\quad + \sum_{c_i \in \mathbb{C}} V_i(t) (E^{\text{total}}(t) - E^{\text{max}}) \\ &\quad \left. + V \cdot \mathbb{C}(t) \mid \mathbf{Z}(t) \right] + B, \end{aligned} \quad (38)$$

where  $B$  is a constant upper bound on the second-order moments of queue variations, determined by the per-time slot decisions  $\{\mathbf{b}(t), \ell(t), F(t)\}_{t \in \mathcal{T}}$ .

The detailed proof is attached to Appendix A, available online.

Based on the upper bound in Lemma 1, the drift-plus-penalty expression  $\Lambda(\mathbf{Z}(t))$  can be minimized by solving a deterministic optimization problem at each time slot.

Since the constant term  $B$  does not influence the decision-making process, it can be safely omitted from the objective function. Accordingly, we reformulate problem  $(\mathcal{P}_1)$  as a per-slot problem  $(\mathcal{P}_2)$ , which can be solved online with current observations, ensuring long-term constraint satisfaction and queue stability.

$$\begin{aligned} \mathcal{P}_2 : \quad &\min_{\{\mathbf{b}(t), \ell(t), F(t)\}} V \cdot \mathbb{C}(t) \\ &+ \sum_{c_i \in \mathbb{C}} Q_i^{\text{loc}}(t) (\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}} \tau) \\ &+ \sum_{\mathcal{R}_k \in \mathbb{R}} Q_k^{\text{rsu}}(t) \left( \sum_{c_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t) \tau \right) \\ &+ \sum_{c_i \in \mathbb{C}} V_i(t) (E^{\text{total}}(t) - E^{\text{max}}) \\ \text{s.t.} \quad &\text{Constraints C4} \sim \text{C8 in } (\mathcal{P}_1). \end{aligned} \quad (39)$$

## VI. LYAPUNOV-BASED DRL FOR RESOURCE ALLOCATION

In this section, we cast the per-slot problem into a Markov decision process (MDP). We then develop the CTDE actor-critic solution, detailing its MAPPO-based architecture and training procedure, followed by a computational complexity analysis that quantifies the cost of trajectory collection and actor-critic updates.

### A. Markov Decision Process

To solve the derived per-slot optimization problem  $(\mathcal{P}_2)$  in an online manner under dynamic vehicular environments and uncertain network conditions, we further formulate it as an MDP. This MDP formulation enables the application of DRL to learn optimal offloading and resource allocation strategies through interaction with the environment. While recent studies have incorporated diffusion models into deep deterministic policy gradient frameworks to enhance exploration efficiency

in high-dimensional continuous spaces [36], our approach differs by embedding Lyapunov drift-plus-penalty theory into the reward design, explicitly guaranteeing queue stability and long-term energy efficiency in dynamic vehicular environments. Specifically, the MDP is defined by the following components: state space, action space, and reward function.

1) *State Space*: In our system, each CV  $\mathcal{C}_i$  is modeled as an individual agent. At each time slot  $t$ , we adopt a DT-enhanced state description incorporating vehicle and RSU dynamics, as well as additional queueing states introduced in our Lyapunov-based model. The local state  $s_i(t)$  is given by

$$s_i(t) = \{\mathcal{U}_{info}^{CV}, \mathcal{U}_{info}^{RSU}, Q_i^{\text{loc}}(t), Q_k^{\text{rsu}}(t), V_i(t), G_{rsu}(t)\}, \quad (40)$$

therein, the DT of CV  $\mathcal{C}_i$  can be defined as  $\mathcal{U}_{info}^{CV} = \{G_i^{\text{TasD}}(t), G_i^{\text{InsD}}(t), C_{CV_i}^{\text{cpu}}, L_i^{\text{cv}}(t), v_i^{\text{cv}}, T_i^{\text{max}}\}$ . The DT of RSU  $\mathcal{R}_k$  can be defined as  $\mathcal{U}_{RSU} = \{L_k^{\text{rsu}}, F_k^{\text{rsu,max}}\}$ .

Then, the joint state of all agents is defined as follows:

$$S(t) = \{s_1(t), \dots, s_i(t), \dots, s_I(t)\}. \quad (41)$$

2) *Action Space*: The agent  $i$ , corresponding to the CV  $\mathcal{C}_i$ , is responsible for making task offloading and bandwidth allocation decisions. Specifically, after a task is generated at the beginning of time slot  $t$ , the DT space provides real-time environment and resource information to assist the decision-making process. Based on this, the agent determines: (i) the partition ratios of the task to be processed locally, and offloaded to RSU nodes; and (ii) the bandwidth allocation ratios for each offloading link. Therefore, the action space of agent  $i$  at time slot  $t$  is defined as:

$$a_i(t) = \{\mathbf{b}(t), \ell(t), F(t)\}, \quad (42)$$

where  $\mathbf{b}(t)$  satisfies constraint C4 and  $\ell(t)$  satisfies constraint C5.

It is important to note that the transmission mode selection variable  $\eta_i(t)$ , is not directly decided by the agent. Instead, it is automatically determined by the DT space.  $\eta_i(t)$  satisfies constraint C7.

Then, the joint action of all agents is defined as below:

$$a(t) = \{a_1(t), \dots, a_i(t), \dots, a_I(t)\}. \quad (43)$$

3) *Reward Function*: After executing action  $a_i(t)$  at state  $s_i(t)$ , agent  $i$  observes the transition of the environment to a new state  $s_i(t+1)$ , and receives a reward that reflects the quality of the decision. In our system, the reward function is derived from the drift-plus-penalty formulation obtained via Lyapunov optimization. Specifically, it is designed as the negative of the transformed per-slot objective function in Problem  $(\mathcal{P}_2)$ , which jointly captures queue stability, task completion delay, and energy consumption [29]. This design ensures that maximizing the expected cumulative reward is equivalent to minimizing the long-term system cost while maintaining queue stability.

The reward function is therefore defined as:

$$R(t) = - \left\{ V \cdot \mathcal{C}(t) + \sum_{\mathcal{C}_i \in \mathcal{C}} Q_i^{\text{loc}}(t)(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau) \right. \\ \left. + \sum_{\mathcal{R}_k \in \mathcal{R}} Q_k^{\text{rsu}}(t) \left( \sum_{\mathcal{C}_i \in \mathcal{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau \right) \right. \\ \left. + \sum_{\mathcal{C}_i \in \mathcal{C}} V_i(t)(E^{\text{total}}(t) - E^{\text{max}}) \right\}. \quad (44)$$

### B. DT-Enhanced DRL Algorithm

To solve the Lyapunov-optimized per-slot decision problem derived in  $(\mathcal{P}_2)$ , this problem is modeled as a cooperative multi-agent reinforcement learning (MARL) problem with shared resource constraints. Although all agents aim to cooperatively minimize a system-wide Lyapunov drift-plus-penalty objective, the agents implicitly compete for limited RSU computation and bandwidth resources. Therefore, this scenario constitutes a soft-coupled cooperative MARL problem, where agents are cooperative in policy optimization but interdependent due to constrained and shared resource allocation.

1) *The Motivation of Adopting MAPPO* [16], [37]: Although Lyapunov optimization effectively decomposes the long-term coupled optimization problem into tractable per-slot subproblems, solving the resulting mixed-integer nonlinear programming problem in real time remains challenging due to the high-dimensional decision space and rapidly changing vehicular environments. Traditional optimization and heuristic approaches are often insufficient to capture the complex dynamics of joint resource allocation and task offloading under mobility, heterogeneity of tasks, and stringent latency constraints. To overcome these challenges, we adopt the MAPPO framework, which naturally fits the cooperative yet resource-competitive characteristics of vehicular networks. Within the CTDE paradigm, MAPPO utilizes a centralized critic to leverage global information during training, while allowing decentralized actors to execute policies independently in real-time. Moreover, by embedding the Lyapunov drift-plus-penalty formulation into the reward function, the algorithm explicitly incorporates queue stability, latency, and energy trade-offs into the learning process. This integration ensures not only adaptability to dynamic network conditions but also long-term system stability.

2) *Network Architecture of Ly-DTMPO*: As shown in Fig. 3, the proposed Ly-DTMPO framework adopts an actor-critic architecture under the CTDE paradigm. Each agent is equipped with an actor network for decentralized decision making, while a centralized critic network is employed during training to guide policy improvement using global information.

- **Actor Network**: For each agent  $i \in \mathcal{I}$ , the actor network takes its local observation  $s_i(t)$ , which includes the DT-enhanced vehicular state (e.g., task size, CPU capability, position, and velocity) and queue-related information and the RSU-level queue aggregation  $G_{rsu}(t) = [\sum_{k=1}^K Q_k^{\text{rsu}}(t), \frac{1}{K} \sum_{k=1}^K Q_k^{\text{rsu}}(t), \max_k Q_k^{\text{rsu}}(t)]$ . When DT information is unavailable, this component is replaced with a zero vector of the same dimension to keep the state

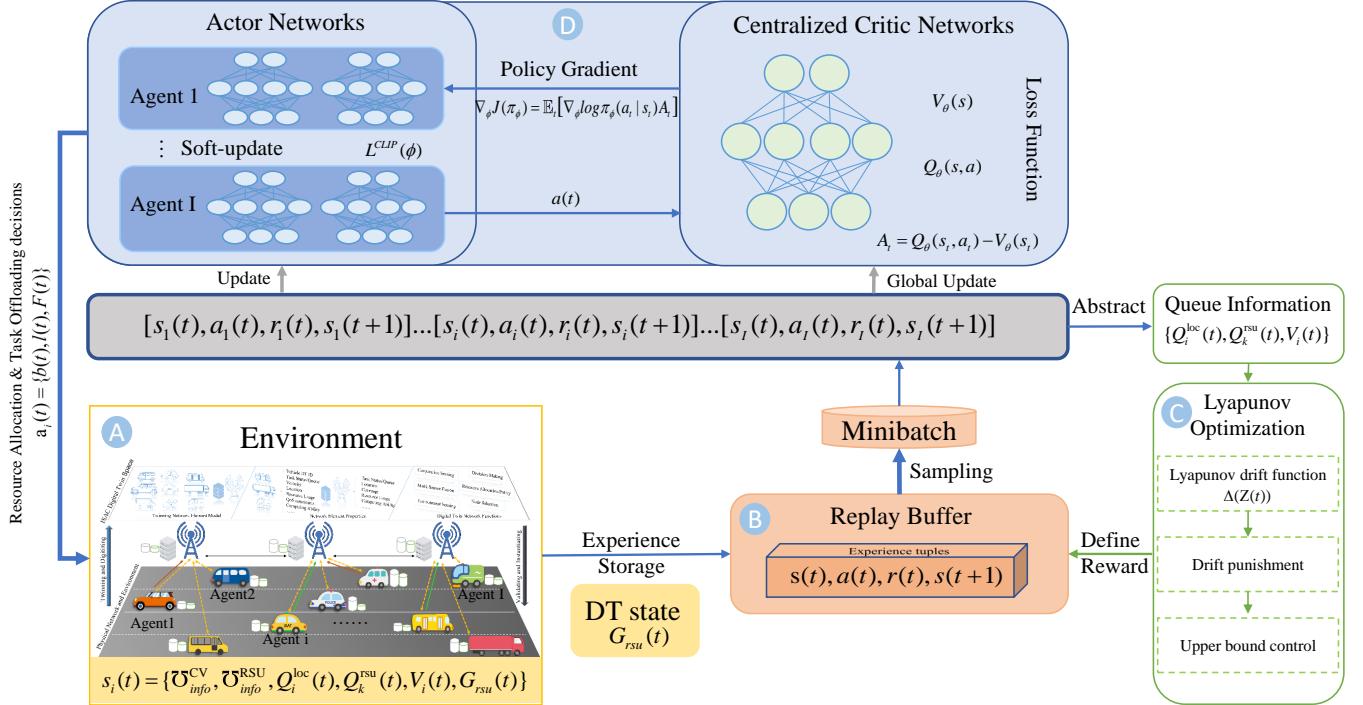


Fig. 3. Framework of the proposed Ly-DTMPO algorithm. Part A describes the DT-assisted vehicular environment where agents interact with the system. Part B stores and samples experiences for policy updates. Part C employs Lyapunov optimization to stabilize queue dynamics and balance delay–energy trade-offs. Part D adopts a CTDE structure with centralized critics and decentralized actors for cooperative policy learning.

representation consistent. The actor consists of two fully connected hidden layers with ReLU activations, followed by an output layer that parameterizes a Gaussian policy. Specifically, the actor outputs the mean and variance of the action distribution, from which the offloading and resource allocation decision  $a_i(t)$  is sampled. To stabilize the policy optimization, the PPO objective is used, where the clipped surrogate loss  $L^{CLIP}(\phi)$  ensures conservative policy updates. During execution, each agent selects its action in a distributed manner based solely on its local state.

**Critic Network:** The critic network is centralized and shared during training. It takes as input the global state  $s(t)$ , which aggregates the observations of all agents and system-level queue information, and the RSU queue aggregation  $G_{rsu}(t)$ . It outputs the state-value estimate  $V_\theta(s(t))$ . The critic consists of two fully connected hidden layers with ReLU activations and one linear output layer. By minimizing the mean squared error between predicted and empirical returns, the critic provides stable value function approximation. The advantage function is then derived as

$$A_t = Q_\theta(s_t, a_t) - V_\theta(s_t), \quad (45)$$

which is used to guide the actor updates via policy gradient.

**3) Proposed Ly-DTMPO algorithm description:** We concretely introduce the proposed Ly-DTMPO for each CV  $\mathcal{C}_i$  in Algorithm 1. The overall training and execution procedure is summarized as follows.

- **Trajectory Collection:** At each decision slot  $t$ , every agent  $i \in \mathcal{I}$  observes its local state  $s_i(t)$  and generates an action  $a_i(t)$  according to its stochastic policy  $\pi_\phi(a_i | s_i)$ . The joint action  $a(t) = \{a_1(t), \dots, a_I(t)\}$  is executed in the environment, which then returns the next state  $s(t+1)$  and reward vector  $r(t) = \{r_1(t), \dots, r_I(t)\}$ . The reward is defined by the Lyapunov drift-plus-penalty formulation to simultaneously capture delay, energy, and queue stability constraints. The experience tuple  $(s(t), a(t), r(t), s(t+1))$  is stored in the replay buffer.

- **Advantage Estimation:** For each agent, the critic network approximates the value function  $V_\theta(s_t)$ . To compute the temporal-difference error and stabilize training, we adopt generalized advantage estimation (GAE). The advantage for step  $t$  is calculated as

$$A_t = \delta_t + (\gamma \lambda) \delta_{t+1} + \dots + (\gamma \lambda)^{T-t+1} \delta_{T-1}, \quad (46)$$

where  $\delta_t = r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)$ , and  $\gamma$  and  $\lambda$  denote the discount and trace-decay factors, respectively.

- **Actor Update:** The actor parameters  $\phi$  are optimized using the PPO clipped surrogate objective to prevent excessively large policy updates:

$$L^{CLIP}(\phi) = \hat{\mathbb{E}}_t [\min(r_t(\phi) A_t, \text{clip}(r_t(\phi), 1-\epsilon, 1+\epsilon) A_t)], \quad (47)$$

where  $r_t(\phi) = \frac{\pi_\phi(a_t | s_t)}{\pi_{\phi^{old}}(a_t | s_t)}$  is the probability ratio between the new and old policies, and  $\epsilon$  is the clipping parameter. The gradient of the actor objective is then computed as

$$\nabla_\phi J(\pi_\phi) \approx \mathbb{E}_t [\nabla_\phi \log \pi_\phi(a_t | s_t) A_t]. \quad (48)$$

- **Critic Update:** The critic network parameters  $\theta$  are updated by minimizing the mean-squared error (MSE) between the predicted value and the empirical return:

$$L_V(\theta) = \mathbb{E}_t[(V_\theta(s_t) - R_t)^2], \quad (49)$$

where  $R_t = A_t + V_\theta(s_t)$  denotes the estimated return at time  $t$ . The critic thus provides a stable baseline for advantage computation.

- **Training Loop:** The actor and critic are updated for multiple epochs using minibatches of trajectories sampled from the replay buffer. To prevent gradient explosion, gradient clipping is applied. An entropy bonus is also added to the actor loss to encourage sufficient exploration in the action space.
- **Execution Phase:** After centralized training, each vehicle executes the learned policy independently based on its own local observation  $s_i(t)$ . This decentralized execution ensures scalability in real vehicular networks, while the Lyapunov-shaped reward guarantees queue stability and energy-latency trade-offs over the long run.

4) *Complexity Analysis:* The computational complexity of the proposed Ly-DTMPO algorithm mainly arises from four components: trajectory collection, advantage estimation, actor-critic updates, and centralized training [22], [38].

In the **trajectory collection** phase, each of the  $\mathcal{I}$  agents executes its policy network once per time slot, resulting in a complexity of  $\mathcal{O}(\mathcal{I} \times d_\pi)$ , where  $d_\pi$  denotes the number of actor parameters. **Advantage estimation** introduces a complexity of  $\mathcal{O}(\mathcal{I} \times T)$  over  $T$  decision steps. For **actor-critic updates**, mini-batch training with batch size  $Bs$  and  $E$  epochs yields a total complexity of  $\mathcal{O}(E \times Bs \times (d_\pi + d_V))$ , where  $d_V$  is the number of critic parameters. Under the CTDE paradigm, the communication overhead during centralized training is  $\mathcal{O}(I \times d_\pi)$  per synchronization, while decentralized execution incurs only  $\mathcal{O}(d_\pi)$  inference cost per agent. Overall, the per-iteration complexity of Ly-DTMPO can be expressed as

$$\mathcal{O}(\mathcal{I} \times (T + d_\pi) + E \times Bs \times (d_\pi + d_V)), \quad (50)$$

which grows linearly with the number of agents and decision steps. Benefiting from decentralized execution and parallel inference, Ly-DTMPO achieves scalable and real-time applicability in large-scale vehicular networks.

## VII. PERFORMANCE EVALUATION

In this section, we first describe the simulation parameter settings and then comprehensively evaluate the performance of our proposed Ly-DTMPO algorithm by comparing it against several representative benchmark schemes, and finally discuss the evaluation results.

### A. Parameters Setting

We adopt the environment of pytorch 1.11 and Python 3.8 with NVIDIA GeForce RTX 4070 SUPER 12GB, 13th Gen Intel(R) Core(TM) i5-13600KF 3.50 GHz, and 32 GB of RAM. The detailed settings are as follows:

1) *Simulation Scenario:* We consider an east-west oriented four-lane road, where each lane is 3.75 m ( $L_0$ ) wide [22] and

---

**Algorithm 1** Proposed Ly-DTMPO algorithm for agent  $i$  in DT-enabled V2X networks

**Input:** State space  $s_i(t)$  defined in (40); actor  $\pi_{\varphi_i}$ ; critic  $V_\theta$ ; buffer  $\mathcal{D}$   
**Output:** Trained actor  $\pi_{\varphi_i}$  and critic  $V_\theta$

```

1: for each episode do
2:   Initialize environment  $\mathcal{E}$ , obtain initial state  $s_i(0)$ , clear
      buffer  $\mathcal{D}$ 
3:   for  $t = 1, 2, \dots, T$  do
4:     Observe current state  $s_i(t)$  as defined in (40);
5:     Each agent  $i$  selects action  $a_i(t) \sim \pi_{\varphi_i}(a_i|s_i(t))$ ;
6:     Execute joint action  $a(t) = \{a_1(t), \dots, a_I(t)\}$  in  $\mathcal{E}$ ;
7:     Observe next state  $s_i(t+1)$  and reward  $r_i(t)$  defined
      by (44);
8:     Store  $(s_i(t), a_i(t), r_i(t), s_i(t+1))$  into  $\mathcal{D}$ ;
9:     Update  $s_i(t) \leftarrow s_i(t+1)$ ;
10:   end for
11:   for epoch = 1, 2, ...,  $E$  do
12:     Sample a mini-batch from  $\mathcal{D}$ ;
13:     Compute advantage using GAE by (46);
14:     Compute actor clipped surrogate loss by (47);
15:     Update actor network with policy gradient by (48);
16:     Compute critic loss by (49);
17:     Update critic network with gradient  $\nabla_\theta L^V$ ;
18:   end for
19: end for

```

---

700 m long. Three RSUs are deployed along the north side of the center road at a distance of 9.375 m ( $l_k$ ), with 150 m spacing, 10 m height ( $H_k$ ) [25], and 200 m coverage radius. The system bandwidth  $\mathcal{B}$  is set to 20 MHz [22], the noise power  $\sigma_0^2$  is  $10^{-13}$  W [25], and the system operates in time slots with a duration of 1 s ( $\tau$ ).

2) *Parameters of CVs and RSUs:* The vehicle speed  $v_i^{cv}$  follows a uniform distribution in [12, 16] m/s. The computing capability of each vehicle  $C_{CV_i}^{cpu}$  is uniformly distributed in [2, 3] GHz [38], while each RSU is equipped with a fixed 20 GHz computing capacity ( $F_k^{rsu,max}$ ). The task data volume  $G_i^{TasD}$  is uniformly distributed in [1, 3] Mb [38], and the computation intensity of task  $I_c^{task}$  is initialized in the range of [1500, 2000] cycles/bit.

3) *Parameters of training :* For the proposed Ly-DTMPO algorithm, the learning rate is set to  $8 \times 10^{-5}$ , with a discount factor  $\gamma = 0.99$ , GAE parameter  $\lambda = 0.95$ , clipping ratio  $\epsilon = 0.2$ , and 10 update epochs. Each episode consists of 30 time slots, and the maximum number of episodes is 1800. The Adam optimizer is employed for parameter updates to ensure stable convergence.

The values of the remaining parameters are summarized in Table II.

### B. Baselines

To demonstrate the effectiveness of the proposed Ly-DTMPO algorithm, we compare it with several representative MARL baselines. For fairness, all algorithms are implemented under the same simulation environment and adopt identical parameter settings for the vehicular network scenario described

TABLE II  
SIMULATION PARAMETERS

PARAMETERS	VALUES
Maximum number of episodes:	1800
Steps per episode:	30
Number of Vehicles: $N$	[2, 6]
Lane width (m): $L_0$	3.75
Height of RSU (m): $H_k$	10
Coverage radius of RSU (m): $Range_k^{rsu}$	200
Distance between neighboring RSUs (m):	150
Distance from RSU to center of road (m): $l_k$	9.75
Bandwidth (MHz): $\mathcal{B}$	20
Duration of time slots (s): $\tau$	1 [39]
Additive white Gaussian noise (W): $\sigma_0^2$	$10^{-13}$ [25]
Transmission power (W): $P_i$	1
Vehicles speed (m/s): $v_i^{cv}$	[12, 16]
CV computation capacity (GHz): $C_{CV_i}^{cpu}$	[2, 3] [29]
RSU computing resources (GHz): $F_{rsu,max}^{k}$	20
Task data volume (Mb): $G_i^{TasD}$	[1, 3]
The computational intensity of a task (cycles/bit): $I_c^{task}$	[1500, 2000] [25]
The computational intensity of the coordinate transformation (cycles/bit): $I_c^{co-tra}$	[100, 500] [25]
The vehicle energy coefficient: $\kappa_{cv}$	$10^{-26}$ [35]
The RSU energy coefficient: $\kappa_{rsu}$	$10^{-28}$ [40]
Communication gain factor: $\Theta$	1.5 [29]
The Lyapunov Control Parameter: $V$	5
Batch size: $B$	512
Discount factor: $\gamma$	0.99
GAE parameter: $\lambda$	0.95
Clipping ratio: $\epsilon$	0.2

in Sec. VII-A. The main differences lie in the decision-making modules for task offloading and resource allocation.

1) *Ly-MADDPG* [13], [41]: This baseline adopts the multi-agent deep deterministic policy gradient (MADDPG) as the backbone, combined with the Lyapunov-based queue stability formulation for resource allocation and task offloading.

2) *Ly-MATD3*: This baseline replaces the PPO core of the proposed scheme with multi-agent twin delayed deep deterministic policy gradient (MATD3) [42]. The Lyapunov optimization framework remains identical, ensuring fair comparison in terms of queue stability.

3) *Ly-MASAC*: This baseline adopts the multi-agent soft actor-critic (MASAC) algorithm, which is an off-policy algorithm leveraging entropy regularization for improved exploration. It extends the LySAC framework in [29] to the multi-agent setting, where the Lyapunov optimization mechanism is incorporated to guarantee queue stability.

4) *Ly-MAPPO* [16]: This baseline applies the MAPPO algorithm under the same Lyapunov optimization framework, but removes the DT-enhanced global visibility and predictive modeling. The comparison with our proposed Ly-DTMPO highlights the unique contribution of DT integration in improving decision-making efficiency under high-mobility vehicular environments.

### C. Simulation Results

In this section, we first evaluate the overall system performance of the proposed Ly-DTMPO algorithm under the

default parameter settings. Subsequently, we conduct a comparative analysis with the benchmark schemes to highlight the effectiveness of our approach. Finally, we investigate the influence of key system parameters on the performance of Ly-DTMPO and the baseline algorithms to provide deeper insights into their adaptability and robustness in dynamic vehicular environments.

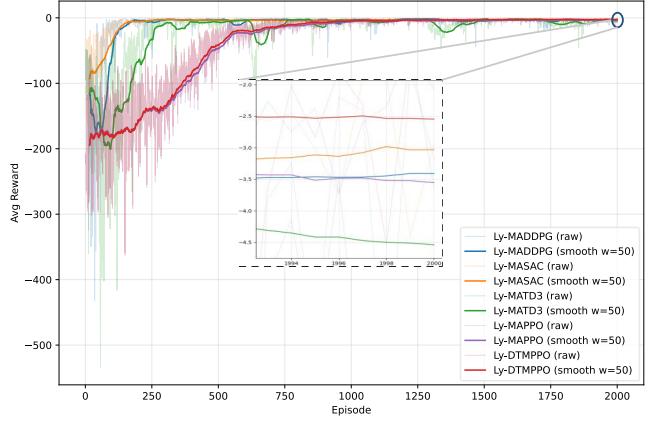


Fig. 4. Average rewards under different algorithms.

1) *Convergence Performance*: Fig. 4 depicts the convergence behaviors of all schemes. All methods eventually converge. This experiment was conducted with 5 CVs, and 3 RSUs. In comparison, Ly-DTMPO (red curve) attains the highest steady-state reward and maintains relatively smooth convergence. Using the average reward over the last 50 episodes as the steady-state metric, Ly-DTMPO achieves relative gains of 25.3%, 43.9%, 15.9%, and 28.3% over Ly-MADDPG, Ly-MATD3, Ly-MASAC, and Ly-MAPPO, respectively. Moreover, Ly-DTMPO exhibits a coefficient of variation (CV) of 0.411, which is significantly smaller than Ly-MAPPO (0.672) and Ly-MADDPG (0.530), and comparable to MASAC (0.366) and MATD3 (0.333), thereby demonstrating superior stability against most baselines. In terms of convergence speed, Ly-DTMPO reaches 95% of its steady-state reward at 1711 episodes, which is slower than the off-policy schemes (e.g., Ly-MADDPG at 185 and Ly-MASAC at 278). Nevertheless, the substantially higher reward underscores the advantage of the on-policy PPO backbone combined with DT-enabled global visibility and Lyapunov-based queue stabilization in achieving both optimality and robustness in highly dynamic vehicular networks. The detailed numerical results are further summarized in Table III.

2) *The Queue Stability Performance*: Fig. 5a evaluates the queue-stability achieved by integrating Lyapunov optimization into the decision loop. We consider a setting with 5 CVs, 3 RSUs, and Lyapunov control parameter  $V=5$ . In Fig. 7(a), the per-vehicle (CV1–CV5) task-queue trajectories rapidly decay from their initial transients to a small steady window and remain bounded thereafter, evidencing that the Lyapunov drift terms effectively suppress backlog growth even under heterogeneous arrivals. To make the trend visually clear, the curves are shown both in raw form and with a moving-average smoothing over the last 50 episodes; both views consistently

TABLE III  
CONVERGENCE STABILITY AND SPEED OF DIFFERENT SCHEMES

Algorithm	Steady Reward <sup>1</sup>	CV <sup>2</sup>	Conv. Ep. (95%) <sup>3</sup>
Ly-MADDPG	-3.412	0.530	185
Ly-MASAC	-3.031	0.366	278
Ly-MATD3	-4.541	0.333	579
Ly-MAPPO	-3.555	0.672	1398
Ly-DTMPO	-2.549	0.411	1711

<sup>1</sup> Steady reward is averaged over the last 50 episodes.

<sup>2</sup> CV denotes the coefficient of variation.

<sup>3</sup> “Conv. Ep.” represents the first episode achieving 95% of the steady reward.

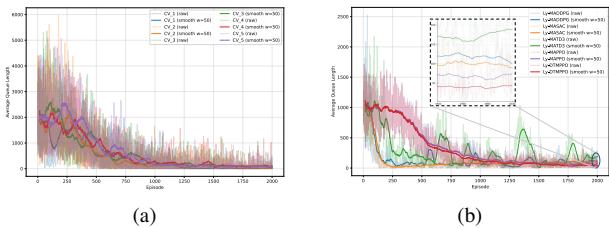


Fig. 5. Task queue length comparison of different CVs and algorithms: (a) Task queue length of different CVs; (b) Task queue length of different algorithms.

indicate fast convergence and low residual oscillation across all agents, which implies stable task admission/offloading and robust queue regulation at the vehicle side.

Fig. 5b compares the average system queue length (averaged over all CVs per episode) across algorithms. Compared with other algorithms, our Ly-DTMPO curve exhibits (i) a steeper descent phase (faster backlog clearance) and (ii) a lower steady-state plateau with reduced fluctuation. This behavior aligns with the design goals: the drift-plus-penalty mechanism enforces queue stability, while the DT-enhanced global visibility improves prediction of workload and channel dynamics, enabling more queue-aware resource allocation. Overall, the convergence of the time-evolving queues demonstrates that the proposed scheme leads to a stable task-assignment process and bounded queues, thereby guaranteeing stable system operation in the considered DT-enabled ISAC-aided IoV scenario.

3) *Effect of the Number of CVs*: Fig. 6a shows the effect of the number of CVs on the system average cost. This comparison is conducted under the condition where the configurable weight  $\alpha$  is set to 0.6, indicating the system prioritizes system delay more than energy consumption. We can observe that the system average cost consistently increases with the number of CVs for all compared algorithms, due to the higher computational and communication load brought by more CVs. However, across all traffic loads (2–6 CVs), the proposed Ly-DTMPO consistently achieves the lowest cost. Taking two representative loads as examples, with 3 CVs Ly-DTMPO reduces the cost by 10.57%, 3.41%, 2.68%, and 20.09% compared with Ly-MAPPO, Ly-MASAC, Ly-MATD3, Ly-MADDPG, respectively. At 4 CVs, the reductions are 0.24%, 1.36%, 2.86%, and 22.62%, respectively. Averaged over all loads (2–6 CVs), Ly-DTMPO yields 2.81%, 3.85%, 4.02%, and 10.03% lower cost than Ly-MAPPO, Ly-

MASAC, Ly-MATD3, and Ly-MADDPG, respectively. This demonstrates that incorporating Lyapunov-guided optimization and DT decision-making effectively balances delay and energy overheads, leading to superior overall performance.

Fig. 6b presents the average delay performance under different numbers of CVs. As expected, the delay consistently increases with the number of vehicles, due to the heavier traffic load and competition for limited communication and computing resources. Among all schemes, Ly-DTMPO maintains the lowest average delay across all scenarios. For example, at 3 CVs, Ly-DTMPO reduces the delay by 10.6%, 20.2%, 2.6%, and 12.8% compared with Ly-MAPPO, Ly-MADDPG, Ly-MASAC, and Ly-MATD3, respectively. On average (2–6 CVs), Ly-DTMPO achieves delay reductions of approximately 2.8%, 9.8%, 3.0%, and 3.7% relative to these baselines. These results demonstrate that the combination of PPO’s on-policy exploration with DT-enhanced global visibility and Lyapunov stabilization yields more efficient resource allocation, thereby alleviating queue accumulation and lowering the overall task processing latency.

Fig. 6c presents the average system energy consumption under different numbers of CVs. It can be observed that Ly-DTMPO and Ly-MAPPO exhibit a monotonic decrease from 2→6 CVs, indicating progressively improved coordination between local computing and RSU offloading. In contrast, Ly-MASAC shows a decrease up to 4 CVs but rebounds at 5–6 CVs; Ly-MATD3 increases at 4 CVs and then drops; Ly-MADDPG fluctuates notably. These non-monotonic behaviors reveal that off-policy baselines may induce unbalanced RSU utilization and redundant offloading under high mobility, leading to oscillatory energy consumption.

For example, at 6 CVs, Ly-DTMPO reduces energy consumption by 2.6%, 79.4%, 62.5%, and 71.1% compared to Ly-MAPPO, Ly-MASAC, Ly-MATD3, and Ly-MADDPG, respectively. Overall, Ly-DTMPO achieves an average energy consumption improvement of approximately 2–3% compared to Ly-MAPPO across the entire load range (2–6 CVs), while the average improvements compared to other schemes are approximately 46%, 36%, and 37%, respectively. This indicates that, under the combined effect of Lyapunov optimization and DT global visibility, Ly-DTMPO can effectively avoid invalid local computations and RSU overload, thereby significantly reducing the total system energy consumption while ensuring quality of service.

4) *Effect of the Lyapunov Control Parameter V*: Fig. 7a and Fig. 7b illustrate the impact of the Lyapunov control parameter  $V$  on the system average cost. As shown in Fig. 7a, all settings of  $V$  eventually lead to convergence. However, the steady-state levels differ significantly. For small  $V$  (e.g.,  $V = 5$  or 10), the system achieves relatively low average costs (around 0.42), since the drift-penalty tradeoff emphasizes queue stability, thereby suppressing excessive task backlog. As  $V$  increases, the average cost gradually rises (e.g., reaching 0.552 at  $V = 50$  and 0.781 at  $V = 100$ ), due to larger average backlog accumulation across task queues, which amplifies the delay component in the overall cost.

This trend indicates that an excessively large  $V$  prioritizes immediate energy minimization but neglects delay control,

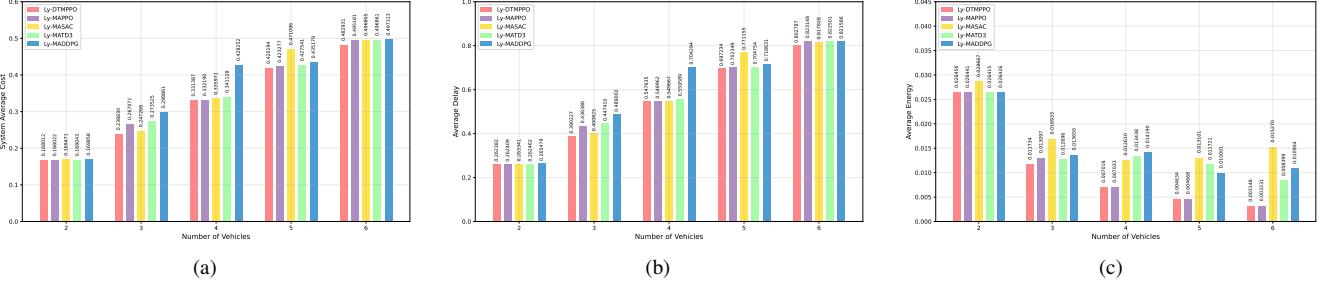


Fig. 6. System performance versus the number of CVs under different algorithms: (a) System average cost versus the number of CVs under different algorithms; (b) Average delay versus the number of CVs under different algorithms; (c) Average energy consumption versus the number of CVs under different algorithms.

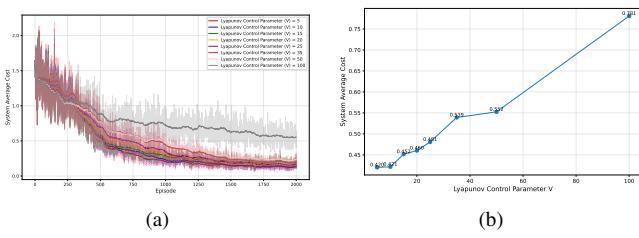


Fig. 7. System performance comparison under different Lyapunov control parameters: (a) System average cost versus Episode under different Lyapunov control parameter  $V$ ; (b) System average cost versus Lyapunov control parameter  $V$ .

while an overly small  $V$  may lead to aggressive queue draining at the expense of energy efficiency. Therefore, selecting a moderate  $V$  achieves the most balanced tradeoff, ensuring minimal system average cost while preserving Lyapunov stability. These results highlight the importance of properly tuning the Lyapunov control parameter to fully exploit the drift-penalty framework in dynamic vehicular environments.

### VIII. CONCLUSION

In this paper, we have studied DT-assisted task offloading and resource allocation in ISAC-enabled IoV. By leveraging Lyapunov optimization, we have transformed the long-term stochastic control problem into per-slot problems that guarantee queue stability while balancing latency and energy consumption. To further exploit the global visibility of DT, we have proposed the Ly-DTMPPPO algorithm, which embeds Lyapunov-based stability modeling into a CTDE multi-agent reinforcement learning framework. This design enables each vehicle to make adaptive and stable decisions using global context information provided by the DT space. Simulation results have validated that Ly-DTMPPPO achieves faster convergence, lower delay, and improved energy efficiency compared with existing Lyapunov-driven and multi-agent baselines, resulting in the lowest overall system cost. These findings demonstrate that integrating Lyapunov optimization with DT-enhanced policy learning provides a robust and scalable framework for intelligent resource management in ISAC-enabled IoV. Future work will extend the proposed framework to heterogeneous vehicular edge networks and explore GenAI-based robust optimization techniques [43] to enhance system adaptability under dynamic vehicular and spectrum conditions.

### PROOF OF LEMMA 1

To begin with, by squaring both sides of (16), (17), and (27), we have

$$Q_i^{\text{loc}}(t+1)^2 = [\max\{Q_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau + \lambda_i^{\text{loc}}(t), 0\}]^2, \quad \forall C_i \in \mathbb{C}, \quad (51)$$

$$Q_k^{\text{rsu}}(t+1)^2 = [\max\{Q_k^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau + \sum_{C_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t), 0\}]^2, \quad \forall R_k \in \mathbb{R}, \quad (52)$$

$$V_i(t+1)^2 = [\max\{V_i(t) - E^{\max} + E^{\text{total}}(t), 0\}]^2. \quad \forall C_i \in \mathbb{C}. \quad (53)$$

Then, we consider inequality  $[\max\{x, 0\}]^2 \leq x^2$ , (51), (52), and (53) can be transformed into the following equations, respectively.

$$Q_i^{\text{loc}}(t+1)^2 - Q_i^{\text{loc}}(t)^2 \leq 2Q_i^{\text{loc}}(t)(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau) + (\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau)^2, \quad (54)$$

$$Q_k^{\text{rsu}}(t+1)^2 - Q_k^{\text{rsu}}(t)^2 \leq 2Q_k^{\text{rsu}}(t)(\sum_{C_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau) + (\sum_{C_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau)^2, \quad (55)$$

$$V_i(t+1)^2 - V_i(t)^2 \leq 2V_i(t)(E^{\text{total}}(t) - E^{\max}) + (E^{\text{total}}(t) - E^{\max})^2. \quad (56)$$

Next, summing (54) over all CVs. Similarly, (55), and (56) have the same operator. They hold:

$$\frac{1}{2} \sum_{C_i \in \mathbb{C}} [Q_i^{\text{loc}}(t+1)^2 - Q_i^{\text{loc}}(t)^2] \leq \frac{1}{2} \sum_{C_i \in \mathbb{C}} \left[ 2Q_i^{\text{loc}}(t)(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau) + (\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau)^2 \right], \quad (57)$$

$$\frac{1}{2} \sum_{R_k \in \mathbb{R}} [Q_k^{\text{rsu}}(t+1)^2 - Q_k^{\text{rsu}}(t)^2] \leq \frac{1}{2} \sum_{R_k \in \mathbb{R}} \left[ 2Q_k^{\text{rsu}}(t)(\sum_{C_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau) + (\sum_{C_i \in \mathbb{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau)^2 \right], \quad (58)$$

$$\frac{1}{2} \sum_{C_i \in \mathbb{C}} [V_i(t+1)^2 - V_i(t)^2] \leq \frac{1}{2} \sum_{C_i \in \mathbb{C}} \left[ 2V_i(t)(E^{\text{total}}(t) - E^{\max}) + (E^{\text{total}}(t) - E^{\max})^2 \right]. \quad (59)$$

We define

$$L(V_i(t)) = \frac{1}{2} \sum_{c_i \in \mathcal{C}} V_i(t)^2, \quad (60)$$

$$\Delta(V_i(t)) = \mathbb{E}[L(V_i(t+1)) - L(V_i(t)) \mid \mathbf{Z}(t)]. \quad (61)$$

And by taking the conditional expectation on both sides of (57), (58), and (59), we have

$$\Delta(Q_i^{\text{loc}}(t)) \leq B_1(t) + \sum_{c_i \in \mathcal{C}} \mathbb{E}[Q_i^{\text{loc}}(t)(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau) \mid \mathbf{Z}(t)],$$

$$B_1(t) = \frac{1}{2} \sum_{c_i \in \mathcal{C}} \mathbb{E}[(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau)^2 \mid \mathbf{Z}(t)], \quad (62)$$

$$\Delta(Q_k^{\text{rsu}}(t)) \leq B_2(t)$$

$$+ \sum_{\mathcal{R}_k \in \mathbb{R}} \mathbb{E} \left[ Q_k^{\text{rsu}}(t) \left( \sum_{c_i \in \mathcal{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau \right) \mid \mathbf{Z}(t) \right],$$

$$B_2(t) = \frac{1}{2} \sum_{\mathcal{R}_k \in \mathbb{R}} \mathbb{E} \left[ \left( \sum_{c_i \in \mathcal{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau \right)^2 \mid \mathbf{Z}(t) \right], \quad (63)$$

$$\Delta(V_i(t)) \leq B_3(t) + \sum_{c_i \in \mathcal{C}} \mathbb{E}[V_i(t)(E^{\text{total}}(t) - E^{\text{max}}) \mid \mathbf{Z}(t)],$$

$$B_3(t) = \frac{1}{2} \sum_{c_i \in \mathcal{C}} \mathbb{E}[(E^{\text{total}}(t) - E^{\text{max}})^2 \mid \mathbf{Z}(t)]. \quad (64)$$

Summing over the four inequalities in (62), (63), and (64):

$$\begin{aligned} \Delta(\mathbf{Z}(t)) &= \mathbb{E} \left[ \frac{1}{2} \sum_{c_i \in \mathcal{C}} (Q_i^{\text{loc}}(t+1)^2 - Q_i^{\text{loc}}(t)^2) \right. \\ &\quad + \frac{1}{2} \sum_{\mathcal{R}_k \in \mathbb{R}} (Q_k^{\text{rsu}}(t+1)^2 - Q_k^{\text{rsu}}(t)^2) \\ &\quad \left. + \frac{1}{2} \sum_{c_i \in \mathcal{C}} (V_i(t+1)^2 - V_i(t)^2) \mid \mathbf{Z}(t) \right] \\ &\leq B(t) + \mathbb{E} \left[ \sum_{c_i \in \mathcal{C}} Q_i^{\text{loc}}(t)(\lambda_i^{\text{loc}}(t) - C_{CV_i}^{\text{cpu}}\tau) \right. \\ &\quad + \sum_{\mathcal{R}_k \in \mathbb{R}} Q_k^{\text{rsu}}(t) \left( \sum_{c_i \in \mathcal{C}} \lambda_i^{\text{rsu}}(t) - F_{RSU_k}^{\text{cpu}}(t)\tau \right) \\ &\quad \left. + \sum_{c_i \in \mathcal{C}} V_i(t)(E^{\text{total}}(t) - E^{\text{max}}) \mid \mathbf{Z}(t) \right], \quad (65) \end{aligned}$$

where  $B(t) = B_1(t) + B_2(t) + B_3(t)$ .

Finally, by grouping the elements in the upper bound that are independent of the queue states into a constant term  $B$ , and by determining the offloading decisions  $\ell(t)$ , the bandwidth allocation  $\mathbf{b}(t)$ , the computation resource allocation  $F(t)$ , and the energy consumption term  $E^{\text{total}}(t)$ , all  $B_1(t)$ ,  $B_2(t)$ ,  $B_3(t)$  become deterministic constants at each time slot. Thus, the overall upper bound  $B(t)$  reduces to a constant  $B$ , which completes the proof of Lemma 1.

## REFERENCES

- [1] L. Wang and S. Yang, "The network selection strategy for connected vehicles based on mobile edge computing," in *Proc. 2022 14th Int. Conf. Commun. Softw. Networks*, 2022, pp. 56–62.
- [2] P. Hou, Y. Huang, H. Zhu, Z. Lu, S.-C. Huang, and H. Chai, "Intelligent decision-based edge server sleep for green computing in MEC-enabled IoV networks," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 2, pp. 3687–3703, 2024.
- [3] Z. Liu, H. Du, J. Lin, Z. Gao, L. Huang, S. Hosseinalipour, and D. Niyato, "DNN partitioning, task offloading, and resource allocation in dynamic vehicular networks: A Lyapunov-guided diffusion-based reinforcement learning approach," *IEEE Trans. Mobile Comput.*, vol. 24, no. 3, pp. 1945–1962, 2025.
- [4] C. Li, Y. Zhang, J. Wu, Y. Luo, and S. Yu, "Smart contract-based decentralized data sharing and content delivery for intelligent connected vehicles in edge computing," *IEEE Trans. Intell. Transport. Sys.*, vol. 25, no. 10, p. 14535–14545, Oct. 2024.
- [5] L. Yin, J. Luo, C. Qiu, C. Wang, and Y. Qiao, "Joint task offloading and resources allocation for hybrid vehicle edge computing systems," *IEEE Trans. Intell. Transport. Sys.*, vol. 25, no. 8, p. 10355–10368, Aug. 2024.
- [6] 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 Trans. Intell. Transport. Sys.*, vol. 25, no. 7, p. 7712–7725, Jul. 2024.
- [7] C. Li, M. Dong, Y. Fu, F. Richard Yu, and N. Cheng, "Integrated sensing, communication, and computation for IoV: Challenges and opportunities," *IEEE Commun. Surv. Tut.*, pp. 1–1, 2025.
- [8] F. Liu, Y. Cui, C. Masouros, J. Xu, T. X. Han, Y. C. Eldar, and S. Buzzi, "Integrated sensing and communications: Toward dual-functional wireless networks for 6G and beyond," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 6, pp. 1728–1767, 2022.
- [9] D. Wen, Y. Zhou, X. Li, Y. Shi, K. Huang, and K. B. Letaief, "A survey on integrated sensing, communication, and computation," *IEEE Commun. Surv. Tut.*, pp. 1–1, 2024.
- [10] X. Tan, M. Wang, T. Wang, Q. Zheng, J. Wu, and J. Yang, "Adaptive task scheduling in digital twin empowered cloud-native vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 8973–8987, 2024.
- [11] C. Xu, Z. Tang, H. Yu, P. Zeng, and L. Kong, "Digital twin-driven collaborative scheduling for heterogeneous task and edge-end resource via multi-agent deep reinforcement learning," *IEEE Commun. Surv. Tut.*, vol. 41, no. 10, pp. 3056–3069, 2023.
- [12] B. Huang, X. Fan, S. Zheng, N. Chen, Y. Zhao, L. Huang, Z. Gao, and H.-C. Chao, "Collaborative sensing-aware task offloading and resource allocation for integrated sensing-communication-and computation-enabled internet of vehicles (IoV)," *Sensors*, vol. 25, no. 3, 2025.
- [13] A. S. Kumar, L. Zhao, and X. Fernando, "Task offloading and resource allocation in vehicular networks: A Lyapunov-based deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 13 360–13 373, 2023.
- [14] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and Appl.*, vol. 26, no. 3, pp. 1145–1168, 2021.
- [15] J. M. Gimenez-Guzman, I. Leyva-Mayorga, A. Azarbahram, O. A. López, and P. Popovski, "Energy-autonomous roadside nodes in V2I using RF energy harvesting," *IEEE Open J. of the Commun. Soc.*, vol. 5, pp. 4024–4035, 2024.
- [16] Y. Jia, C. Zhang, Y. Huang, and W. Zhang, "Lyapunov optimization based mobile edge computing for internet of vehicles systems," *IEEE Trans. Commun.*, vol. 70, no. 11, pp. 7418–7433, 2022.
- [17] 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 Trans. Mobile Comput.*, vol. 23, no. 8, pp. 8158–8174, 2024.
- [18] B. Hu, W. Zhang, Y. Gao, J. Du, and X. Chu, "Multiagent deep deterministic policy gradient-based computation offloading and resource allocation for ISAC-aided 6G V2X networks," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33 890–33 902, 2024.
- [19] P. Liu, Z. Fei, X. Wang, J. Huang, J. Hu, and J. Andrew Zhang, "Joint offloading and beamforming design in integrating sensing, communication, and computing systems: A distributed approach," *IEEE Trans. Commun.*, vol. 73, no. 7, pp. 4697–4712, 2025.
- [20] Q. Liu, R. Luo, H. Liang, and Q. Liu, "Energy-efficient joint computation offloading and resource allocation strategy for ISAC-aided 6G V2X networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 413–423, 2023.
- [21] J. Wang, L. Bai, J. Chen, and J. Wang, "Starling flocks-inspired resource allocation for ISAC-aided green ad hoc networks," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 1, pp. 444–454, 2023.
- [22] Y. Xie, Q. Wu, P. Fan, N. Cheng, W. Chen, J. Wang, and K. B. Letaief, "Resource allocation for twin maintenance and task processing in vehicular edge computing network," *IEEE Internet Things J.*, vol. 12, no. 15, pp. 32 008–32 021, 2025.
- [23] K. Sun, J. Wu, Q. Pan, X. Zheng, J. Li, and S. Yu, "Leveraging digital twin and DRL for collaborative context offloading in C-V2X

- autonomous driving,” *IEEE Trans. Veh. Technol.*, vol. 73, no. 4, pp. 5020–5035, 2024.
- [24] B. Li, W. Liu, W. Xie, N. Zhang, and Y. Zhang, “Adaptive digital twin for UAV-assisted integrated sensing, communication, and computation networks,” *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 4, pp. 1996–2009, 2023.
- [25] Y. Gong, Y. Wei, Z. Feng, F. R. Yu, and Y. Zhang, “Resource allocation for integrated sensing and communication in digital twin enabled internet of vehicles,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 4510–4524, 2023.
- [26] W. Xu, Y. Zhang, J. Wang, L. Zhang, and Q. Li, “Dynamic resource allocation for ISAC enabled internet of vehicles,” in *Proc. 30th Annu. Int. Conf. Mobile Comput. Netw.*, ser. ACM MobiCom ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2216–2221.
- [27] N. Chen, Z. Cheng, X. Fan, Z. Liu, B. Huang, J. Yang, Y. Zhao, and L. Huang, “Integrated sensing, communication, and computing: An information-oriented resource transaction mechanism,” *arXiv preprint arXiv:2401.11759*, 2024.
- [28] N. Ren, Q. Zhang, Z. Jiang, S. Liu, and J. Liu, “Integrated sensing and communication resource allocation for latency sensitive services of connected automated vehicles,” in *Proc. 2023 IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2023, pp. 482–487.
- [29] Y. Liang, H. Tang, H. Wu, Y. Wang, and P. Jiao, “Lyapunov-guided offloading optimization based on soft actor-critic for ISAC-aided internet of vehicles,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14 708–14 721, 2024.
- [30] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, “Sdn/nfv-empowered future iov with enhanced communication, computing, and caching,” *Proc. of the IEEE*, vol. 108, no. 2, pp. 274–291, 2020.
- [31] Y. Qi, Y. Zhou, Y.-F. Liu, L. Liu, and Z. Pan, “Traffic-aware task offloading based on convergence of communication and sensing in vehicular edge computing,” *IEEE Internet of Things J.*, vol. 8, no. 24, pp. 17 762–17 777, 2021.
- [32] X. Zhang, M. Peng, S. Yan, and Y. Sun, “Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications,” *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6380–6391, 2020.
- [33] Y. Deng, Z. Chen, X. Yao, S. Hassan, and A. M. A. Ibrahim, “Parallel offloading in green and sustainable mobile edge computing for delay-constrained IoT system,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 12 202–12 214, 2019.
- [34] M. Huang, Q. Zhai, Y. Chen, S. Feng, and F. Shu, “Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing,” *Sensors*, vol. 21, no. 8, 2021.
- [35] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, “Lyapunov-guided deep reinforcement learning for stable online computation offloading in mobile-edge computing networks,” *IEEE Trans. Wireless Commun.*, vol. 20, no. 11, pp. 7519–7537, 2021.
- [36] Z. Liu, H. Du, X. Hou, L. Huang, S. Hosseinalipour, D. Niyato, and K. B. Letaief, “Two-timescale model caching and resource allocation for edge-enabled AI-generated content services,” *IEEE Trans. Mobile Comput.*, pp. 1–17, 2025.
- [37] X. Liu, Y. Yi, W. Zhang, and G. Zhang, “Lyapunov-based madrl policy in wireless powered MEC assisted monitoring systems,” in *Proc. IEEE INFOCOM 2024 - IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2024, pp. 01–06.
- [38] Y. Liu, D. Li, H. Wu, Z. Sun, W. Qin, J. Li, H. Du, and G. Sun, “Task offloading and resource allocation for MEC-assisted consumer internet of vehicle systems,” *IEEE Trans. on Consum. Electronics*, pp. 1–1, 2025.
- [39] Z. Liu, L. Huang, Z. Gao, X. Wang, D. Niyato, Xuemin, and Shen, “A Lyapunov-guided diffusion-based reinforcement learning approach for UAV-assisted vehicular networks with delayed CSI feedback,” *arXiv preprint arXiv:2507.20524*, 2025.
- [40] Y. Jang, J. Na, S. Jeong, and J. Kang, “Energy-efficient task offloading for vehicular edge computing: Joint optimization of offloading and bit allocation,” in *Proc. 2020 IEEE 91st Veh. Technol. Conf. (VTC2020-Spring)*, 2020, pp. 1–5.
- [41] M. Z. Alam, K. S. Khan, and A. Jamalipour, “Multiagent best routing in high-mobility digital-twin-driven internet of vehicles (IoV),” *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13 708–13 721, 2024.
- [42] L. Wu, C. Zhang, B. Zhang, J. Du, and J. Qu, “Toward energy-efficiency: Integrating MATD3 reinforcement learning method for computational offloading in RIS-aided UAV-MEC environments,” *IEEE Internet Things J.*, vol. 12, no. 14, pp. 26 582–26 595, 2025.
- [43] C. Zhao, J. Wang, R. Zhang, D. Niyato, G. Sun, H. Du, D. I. Kim, and A. Jamalipour, “Generative AI-enabled wireless communi- cations for robust low-altitude economy networking,” *arXiv preprint arXiv:2502.18118*, 2025.