Multi-Agent Reinforcement Learning-Based Trading Decision-Making in Platooning-Assisted Vehicular Networks

Tingting Xiao, Chen Chen, *Senior Member, IEEE*, Mianxiong Dong, *Member, IEEE*, Kaoru Ota, *Member, IEEE*, Lei Liu, *Member, IEEE*, and Schahram Dustdar, *Fellow, IEEE*

***Abstract*— Utilizing the stable underlying and cloud-native functions of vehicle platoons allows for flexible resource provisioning in environments with limited infrastructure, particularly for dynamic and compute-intensive applications. To maximize this potential, we propose the creation of a trading market to encourage interactions between service supporters (vehicle platoons) and requesters (task vehicles). Current trading decisions based on game and negotiations can lead to unpredicted handover costs and increased communication overhead in dynamic environments. Moreover, existing research tends to overlook a mutually beneficial trading philosophy by focusing on either the service supporters' profitability or the user experience of resource-restrained requesters. Addressing these issues, we introduce a multi-objective optimization problem to model environmental dynamics and uncertainty, aiming to maximize both platoons' and task vehicles' long-term utilities while maintaining a satisfactory service access ratio. To tackle the problem within acceptable time frames, we develop a global-local training architecture, incorporating a hybrid action space and prioritized sampling into a multi-agent reinforcement learning algorithm that utilizes a twin delayed deep deterministic gradient (GL-HPMATD3). This approach facilitates consensus in the trading market on key issues, including service request selection, resource allocation, and trading pricing. Through extensive experimentation and comparison, we demonstrate our mechanism's superior performance in convergence, service access ratio, player utility, execution latency, and trading pricing relative to several state-of-the-art and baseline methods.**

***Index Terms*— Platooning-assisted vehicular networks, trading market, multi-agent reinforcement learning (MARL).**

Manuscript received 26 November 2022; revised 27 July 2023 and 9 October 2023; accepted 28 November 2023; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor O. Yagan. Date of publication 25 December 2023; date of current version 18 June 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1807500; in part by the National Natural Science Foundation of China under Grant 62072360, Grant 62001357, Grant 62172438, and Grant 61901367; in part by the Key Research and Development Plan of Shaanxi Province under Grant 2021ZDLGY02-09, Grant 2023-GHZD-44, and Grant 2023-ZDLGY-54; in part by the Natural Science Foundation of Guangdong Province of China under Grant 2022A1515010988; in part by the Key Project on Artificial Intelligence of Xi'an Science and Technology Plan under Grant 23ZDCYJSGG0021-2022, Grant 23ZDCYYYCJ0008, and Grant 23ZDCYJSGG0002-2023; in part by the Xi'an Science and Technology Plan under Grant 20RGZN0005; in part by the Proof-of-Concept Fund from the Hangzhou Research Institute of Xidian University under Grant GNYZ2023QC0201; in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP20H04174 and Grant JP22K11989; in part by the Leading Initiative for Excellent Young Researchers (LEADER), the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Japan; and in part by the Japan Science and Technology Agency (JST), Precursory Research for Embryonic Science and Technology (PRESTO), Japan, under Grant JPMJPR21P3. *(Corresponding authors: Chen Chen; Mianxiong Dong.)*

Tingting Xiao, Chen Chen, and Lei Liu are with the School of Telecommunications Engineering, Xidian University, Xi'an 710071, China (e-mail: ttxiao_1@stu.xidian.edu.cn; cc2000@mail.xidian.edu.cn; leiliu@xidian.edu.cn).

Mianxiong Dong and Kaoru Ota are with the Department of Information and Electronic Engineering, Muroran Institute of Technology, Muroran 050-8585, Japan (e-mail: mx.dong@csse.muroran-it.ac.jp; ota@csse.muroran-it.ac.jp).

Schahram Dustdar is with the Distributed Systems Group, TU Wien, 1040 Vienna, Austria (e-mail: dustdar@dsg.tuwien.ac.at).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNET.2023.3342020, provided by the authors.

Digital Object Identifier 10.1109/TNET.2023.3342020

I. INTRODUCTION

A. Background

IN RECENT years, Vehicular Edge Computing (VEC) has emerged as a pivotal approach in the Intelligent Transportation Systems (ITS) domain, offering and managing resources near vehicles [1]. Exploring potential computing resources could alleviate the workload of central Base Stations (BSs). Beyond Roadside Units (RSUs), any entities boasting computing, caching, or networking capabilities can serve as VEC nodes [2]. However, VEC nodes' limited resources pose challenges to the execution and viability of compute-intensive applications, signaling an urgent need for traditional VEC to evolve. Such progression must better support high scalability, real-time data delivery, and mobility within vehicular networks. Enhancing road BSs deployment or upgrading vehicular software and hardware will undoubtedly incur substantial costs, thus clashing with the economic efficiency and energy-saving objectives of edge computing [3].

Pooling underutilized resources from various organizations could create services akin to Computation-as-a-Service (CaaS), thereby enhancing traditional VEC [4]. Vehicle platoons, a common driving paradigm on motorways, have the potential to support emerging applications by harnessing intra-platoon resources. This cloud-native function allows vehicle platoons to serve as Mobile-Edge-Platooning-Clouds (MEPCs) [5]. Moreover, deploying MEPCs typically outpaces and offers a more flexible configuration than traditional clouds. Therefore, to stimulate vehicle platoons' active participation in resource sharing, it is critical to establish a trading market adaptable to dynamic topologies and diverse Quality-of-Service (QoS) requirements. We illustrate the potential trading market scenario in Fig.1. This market can foster resource trading between MEPCs and task vehicles, thereby ensuring the sustainability of resource sharing in platooning-assisted vehicular networks.

However, decision-making strategies based on games or negotiation often precipitate unpredictable handover costs[1]

[1]The handover process involves disconnecting a vehicle from its current task and establishing a new connection with another task. This transition leads to a 'handover cost' that includes time delays, potential deterioration in service quality, and additional resource consumption.

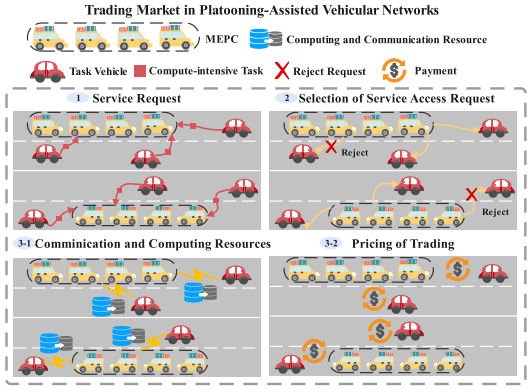


Fig. 1. The trading market in a platooning-assisted vehicular network.

and communication overhead in dynamic, stochastic environments [6]. Furthermore, a significant proportion of existing research either prioritizes the profitability of service supporters or the user experience of the resource-constrained requesters, neglecting the concept of a mutually beneficial trading philosophy. This necessitates designing appropriate economic incentives to expedite trading decisions and foster mutually beneficial trading. Therefore, the relatively uncharted territory of resource trading in platooning-assisted vehicular networks has been a major impetus for our research.

Several pertinent questions warrant exploration: 1) How can we construct a resource trading market that capitalizes on the potential of vehicle platoons? 2) What are the guidelines for formulating a trading policy between supporters and requesters? 3) What is the strategy for swiftly attaining a trading consensus rooted in mutually beneficial relationships? To address these research gaps and explore the challenge of resource trading in platooning-assisted vehicular networks, we conduct thorough investigations in the subsequent sections.

B. Challenge and Motivation

In this part, we summarize the main challenges as follows:

1) *Resource Trading Market Building*: To address the conflict between compute-intensive applications and resource-constrained vehicles, resource sharing is an effective strategy for alleviating computational pressure and enhancing resource utilization. However, the effectiveness of resource-sharing mechanisms based on incentives or contracts depends on thoughtful design, as poorly conceived strategies can harm motivation and performance, leading to suboptimal outcomes [6]. Existing models often prioritize requester service satisfaction, overlooking the equilibrium between supply and demand. The pace of advanced processing unit installation lags behind the explosive growth of emerging applications, resulting in long-term resource demand. Vehicle platoons, especially truck platoons on highways, hold the potential for facilitating resource sharing in the Internet of Vehicles (IoVs) scenario, but research in this area is limited. Therefore, an adaptive trading market is crucial to enhance the participation of service supporters and requesters, considering diverse optimization objectives, dynamic market participants, and individual service demands.

2) *Real-time Trading Decision-making*: The excessive latency caused by multi-round games and negotiation poses a significant challenge to trading decision-making. Furthermore, the dynamic and stochastic conditions of the network affect the timeliness of decisions as well. The long latency necessary to achieve consensus cripples service performance. Factors, such as the latency requirement, varying network conditions, and dynamically available resources can substantially hinder trading decision-making among players.

3) *Optimization to Multi-objectives*: In light of the complex nature of our multi-stakeholder trading market, we adopt a multi-objective optimization to shape our scenario. The complexity primarily stems from the diverse objective requirements among task vehicles, MEPCs, and the overall trading market. Some of these objectives might even be conflicting under certain circumstances. Moreover, service competition between MEPCs, potential communication interference, the changeable resource of MEPCs, and the mobility characteristics of vehicles, coupled with the constraints of trading costs, collectively amplify the intricacy of the trading scenarios. Lastly, the dynamic nature of the trading market itself, along with the continuously evolving environmental conditions, necessitates the optimization of multiple variables to achieve a balance among multi-objectives. This approach fosters a comprehensive balance among diverse factors, bypassing the shortcomings associated with single-objective optimization. Nonetheless, navigating the nuances between objectives and addressing the inherent trade-offs continue to be challenging dimensions of this methodology.

4) *Reinforcement Learning Training Architecture Designing*: In platooning-assisted vehicular networks, designing a decentralized Multi-agent Reinforcement Learning (MARL) algorithm must address challenges like non-stationarity, scalability, and partial observability. A training architecture adaptable to the trading market, incorporating a hybrid action space, and ensuring mutual benefits is essential.

To tackle these challenges, we propose a tailored trading market for platooning-assisted vehicular networks, supported by a training architecture for timely trading decisions. Our approach involves multi-MEPC as service supporters and resource-constrained task vehicles as requesters, serving as the key players in resource trading.

II. Related Work and Contribution

Table I provides a systematic overview of related works, facilitating a thorough comparison while emphasizing the prevailing gaps, thereby underscoring the significance of this study's contributions.

A. Resource Allocation in Vehicular Networks

Driven by advanced 5G/6G wireless communication technologies, existing studies have focused on resource allocation in vehicular networks. Strategies like collaborative MEC and cloud computing for service offloading [7], treating vehicles with idle resources as fog computing nodes for computation offloading in fog-based vehicular networks [8], mobility-aware task offloading and resource allocation schemes [9] have been proposed. However, few of these works emphasize the importance of a trading manner in resource allocation. Moreover, integrating edge servers with individual vehicular clouds faces challenges due to the irregular distribution of edge servers and resource constraints of vehicular clouds.

B. Resource Trading

Resource trading involves the dynamic exchange of computational power, storage, or other digital resources in

TABLE I
SUMMARY OF LITERATURE

Theme	Attributes		Our Contribution
Resource Allocation	Static Scenarios	No Trading Manner	Dynamic scenario with trading manner
	[7], [8]	[7], [8], [9]	
Resource Trading	Costly Handovers	Immediate Benefits	Long-term utility and mutual benefit based on MARL algorithm
	[3], [10], [11], [12]	[10], [13]	
Optimization Models	Single-Objective	High Time Complexity	Multi-objective optimization through real-time decision making
	[14]	[15], [16]	
MARL Algorithm & Training Framework	Local/Global Reward Gap	Hybrid Action Space Issue	Integrated rewards and latent action representation
	[17], [18], [19], [20]	[20], [21], [22]	

a marketplace, guided by algorithms to maximize efficiency and profitability. Several works explore different aspects: optimizing power producers' profits [13], offloading strategy-based consortium blockchain in MEPC [3], market models and resource trading in computing-power network [11], futures-based resource trading approach [10], and two-tiered Stackelberg game-based computing resource trading [12]. However, trading decision-making-based games can lead to unpredictable handover costs and communication overhead in a dynamic environment. Additionally, existing research often neglects a mutually advantageous trading philosophy, focusing excessively on either the profitability of service supporters or the user experience of resource-constrained requesters, failing to balance both parties' interests.

C. Optimization Model

Resource trading and allocation aim to optimize overall system performance and profitability by efficiently distributing resources among entities. In IoV, integrated frameworks like edge cloud are essential for resource management across vehicles, RSUs, and BSs, handling resource-intensive applications [15]. Innovative methodologies such as blockchain-based data trading [16] and fog computing for local energy trading [14] improve efficiency and stimulate participation. However, the optimization problem remains challenging. Single-objective optimization can disadvantage certain stakeholders, while multi-objective optimization requires intricate understanding due to the complex interplay between objectives, potentially imposing a substantial computational burden. Addressing multi-objective challenges often involves trade-offs, adding analytical complexity to achieve objective equilibrium.

D. MARL Algorithm and Training Framework

MARL offers a framework for agents to optimize resource distribution and trading in a shared environment. In [17], MARL algorithms overcome non-stationarity among multiple devices. In [18], a comprehensive framework combines hierarchical reinforcement learning and meta-learning for adaptive resource allocation in dynamic vehicular environments. Vehicle-to-multi-Edges (V2Es) communication framework proposes dynamic communications status learning between vehicles and edge nodes for task offloading or edge caching decisions [19]. A novel MARL algorithm is introduced for dynamic resource allocation in multi-UAV-aided wireless networks [20]. Authors propose an intelligent resource trading framework integrating MADRL, blockchain, and game theory for dynamic resource trading [21]. However, for mutually beneficial trading, local and global rewards should be considered, often overlooked in the existing literature. Additionally, the complexity of hybrid trading actions is frequently oversimplified, hindering actual model training and decision-making processes [20], [21], [22].

Compared with the existing works, our work has made the following differences:

Trading Model: We introduce a novel trading scenario in platooning-assisted vehicular networks, where high-traffic vehicle platoons with resource-sharing capabilities act as mobile resource servers and resource-constrained vehicles play as requesters. This unique scenario enables the construction of a dynamic trading market that enhances resource utilization and interaction frequency. In this trading market, we set comprehensive objectives that not only maximize the utility of both trading parties but also ensure the sustainability of the overall market, illustrated by the service access ratio indicator. Our approach is dynamic and flexible to reflect the constantly changing nature of vehicular networks. Key decision-making aspects include the selection of service requests, assignment of communication resources, allocation of computing resources, and setting of trading pricing. These decisions are not made in isolation but are instead interrelated, jointly influencing the performance and sustainability of the trading market.

Trading Training Architecture: Furthermore, we have built a global-local training architecture to achieve a mutually beneficial philosophy. Moreover, the action space of existing Deep Reinforcement Learning (DRL) frameworks is homogeneous, i.e., only continuous or discrete action spaces are considered [23]. Nevertheless, much of the current research in DRL has focused on improving the performance of a single machine within the computing budget, with relatively little exploration of how to leverage more of the available resources.

E. Contributions

Generally speaking, this work focuses on the intelligent, mutually beneficial trading approach based on multi-agent training architecture. The main contributions are as follows:

Trading Market Building: Considering a specific application scenario wherein MEPC could enable VEC, we have constructed a trading market along the motorway. Here, vehicle platoons act as service supporters, striving not only for their gains from requesters but also for the overall profitability of the trading market. Moreover, this trading model has long-term value in enhancing user experience, thus providing a sustainable approach to resource exchange.

TABLE II
SUMMARY OF KEY NOTATIONS

System Model			
Symbol	**Definition**	**Symbol**	**Definition**
$\mathcal{M}, \mathcal{N}$	Set of MEPCs and task vehicles	$\mathcal{X}_t^n$	Task properties
v_t^n	Data size of task in units of bits	d_t^n	Number of CPU cycles required per bit
τ_t^n	Maximum tolerance delay at time slot t	$\mathcal{K}_t^m$	Covered set by MEPC m
$\mathbb{K}_t^{m,\cap}$	Overlap set of MEPC m	$d_t^{n,m}$	Relative distance between task vehicle and MEPC
R^m	Communication range of MEPC m	$\mathbf{N}_t^m$	Service capability of MEPC m
$\nu_t^{n,m} \in \{0,1\}$	Binary variable of serve decision	$\mathcal{W}_t^m$	Service access ratio
$\mathcal{B}_t = \{b_1, ..., b_K\}$	Communication resources pool	$b_t^{n,m}$	Decision of spectrum resource assignment
$\mathbb{K}_t^m$	Intersection set of MEPC m	$p_t^{n,m}$	Transmission power
$G_t^{n,m}$	Channel gain	$\Gamma_t^{n,m}$	Signal-to-interference-plus-noise ratio
N_0	Additive white Gaussian noise	$c_t^{n,m}$	Decision of computing resource allocation
f	Processing capacity of each platoon member	μ	Effective coefficient related to chip structure
$P_t^{n,m}$	Decision of trading pricing	$\omega_t^{m,p}$	Weight of trading income
$\omega_t^{m,e}$	Weight of energy consumption	$\omega_t^{m,t}$	Weight of the saved completion time
MARL Environment			
$\mathcal{S}$	Set of state	$\mathcal{O}$	Finite set of observation
$\mathcal{H}$	Set of hybrid action space	$\mathcal{R}$	Reward function
γ	Discounted factor	α	Weight of MEPC's utility
$\pi(\cdot\|\theta_m)$	Policy of MEPC m	(x, y)	Position of sites
Θ	Parameter set of policy π	Θ'	Parameter set of target policy π'
Φ_l	Parameter set of action-value function Q_l	Φ_l'	Parameter set of target action-value function Q_l'
Ψ_g^i	Parameter set of global action-value function Q_g^i	$\Psi_g'^{,i}$	Parameter set of target global action-value function $Q_g'^{,i}$
E_ζ	Embedding table with learnable parameter ζ	$q_\sigma(\cdot)$	Encoder parameterized by σ
$p_\varrho(\cdot)$	Decoder parameterized by ϱ	β	Weight of dynamic predictive representation loss
$\widetilde{C}_b, \widetilde{P}_b$	Reconstructed continuous parameters	D_{KL}	Kullback-Leribler divergence

MOOP Formulation: To optimize both the service access ratio and the utility of trading players, we present a Multi-Objective Optimization Problem (MOOP). This aims to determine optimal decisions concerning the selection of service access requests, assignment of communication resources, allocation of computing resources, and setting of trading prices.

Algorithm Construction: To more intuitively simulate the interaction and cooperation among trading players, we built a new MEPC scenario to enrich the use case environment for reinforcement learning. To effectively address the above MOOP, we introduce a Global-Local training architecture that blends a Hybrid action space and Prioritized sampling with a Multi-agent reinforcement learning algorithm, employing a Twin Delayed Deep Deterministic gradient (GL-HPMATD3).

The rest of this paper is as follows: Section III introduces the system model and formulates the MOOP. Section IV presents the preliminary review of the MARL environment. Section V develops a MARL environment for resource trading. Section VI details the proposed GL-HPMATD3 algorithm. Simulation results are demonstrated in Section VII and concluded in Section VIII.

III. SYSTEM MODEL AND PROBLEM FORMATION

A. Platooning-Assisted Vehicular Network

The high mobility and diverse applications in the vehicular network lead to frequent changes in network topology and service requests. To address this, we propose a service-based trading model in a platooning-assisted vehicular network, aiming to satisfy QoS demands and promote service trading. In Table II, we summarize the notations used in this work.

We consider a four-lane road on a motorway, with M MEPCs driving on the road, denoted as $\mathcal{M} = \{1, \ldots, m, \ldots, M\}$. Each MEPC covers different road segments and drives at a time-varying speed, with some overlapping among adjacent MEPCs. Task vehicles, denoted by $\mathcal{N} = \{1, \ldots, n, \ldots, N\}$, generate tasks at the beginning of each time slot t, defined as $\mathcal{X}_t^n = \{v_t^n, d_t^n, \tau_t^n\}$, where v_t^n, d_t^n, and τ_t^n represent task size, CPU cycles required per bit, and maximum tolerance delay at time slot t for vehicle n, respectively. Task size follows a continuous uniform distribution $v_t^n \backsim U(v_{\min}^s, v_{\max}^s)$. For delay-sensitive applications, we set the delay constraint not to exceed the length of a time slot, $\tau_t^n \leq \tau$, and assume no buffer for queuing computation requests. When a task vehicle cannot process a task, it sends a service access request, containing task details and the vehicle's driving state (position, velocity), to an appropriate MEPC. Each MEPC then makes quick trading decisions, including the selection of service access requests, allocation of communication and computing resources, and setting of trading pricing.

B. System Model

We mainly discuss the models (selection of service access request, assignment of communication resources, allocation of computing resources, setting of trading pricing) to support the analysis of the following contents.

1) Selection of Service Access Request: To ensure service satisfaction and trading benefits, the selection of the appropriate service audience is vital. We define the task vehicle set covered by MEPC m at time slot t as $\mathcal{K}_t^m$. Here, we consider that each task vehicle n can only be served by one MEPC, while MEPC m can simultaneously serve multiple task vehicles. It is important to note that some task vehicles may not be covered by any MEPC. Thus, $\mathbb{K}_t^{\cup} \triangleq \{\mathcal{K}_t^1 \cup \ldots \mathcal{K}_t^m \cup \ldots \mathcal{K}_t^M\}_{m \in \mathcal{M}} \subseteq \mathcal{N}$ is satisfied. The overlap set for MEPC m can be denoted as

$$\mathbb{K}_t^{m,\cap} \triangleq \{(m', \mathcal{K}_t^{m,m'}) | \mathcal{K}_t^m \cap \mathcal{K}_t^{m'} = \mathcal{K}_t^{m,m'}\}_{m' \in \mathcal{M}/m}. \quad (1)$$

The relative distance between task vehicle $n \in \mathcal{K}_t^m$ and MEPC m at time slot t is denoted as $d_t^{n,m} \in \mathbb{R}$. MEPC m

has a communication range of R^m, so $d_t^{n,m} \leq R^m$ determines whether MEPC m can serve task vehicle n. The processing capability of MEPC is powerful but not unlimited. We define $\mathbf{N}_t^m$ as the maximum number of task vehicles MEPC m can serve concurrently at time slot t. To make the selection decision, we introduce the binary variable $\nu_t^{n,m} \in \{0,1\}$, indicating whether MEPC m serves task vehicle n at time slot t. The set $\mathbb{V}_t^m = \{n|\nu_t^{n,m} = 1, n \in \mathcal{K}_t^m\}$ represents the selected task vehicles. There are several preconditions for task vehicle selection as follows:

Case 1: $0 \leq |\mathcal{K}_t^m| \leq \mathbf{N}_t^m$, and $\mathbb{K}_t^{m,\cap} = \emptyset$. Namely, when the number of request vehicles is within the processing power of the MEPC m and there is no overlap with other MEPCs, the task vehicle selection process becomes relatively straightforward. MEPC m can satisfy the computing needs of all vehicles within its coverage area. This situation is ideal as it minimizes the potential for resource contention and reduces the complexity of task assignments.

Case 2: $0 \leq |\mathcal{K}_t^m| \leq \mathbf{N}_t^m$, and $\mathbb{K}_t^{m,\cap} \neq \emptyset$. Although the MEPC m can handle all vehicles within its coverage, it overlaps with other MEPCs. In this case, the task vehicle selection process needs to account for the presence of other MEPCs, which could also provide services to the vehicles. There may need to be a process to decide which MEPC a vehicle should choose, perhaps based on factors like signal strength, latency, or the availability of computing resources.

Case 3: $|\mathcal{K}_t^m| > \mathbf{N}_t^m$, and $\mathbb{K}_t^{m,\cap} = \emptyset$. Here, the number of request vehicles exceeds the MEPC's processing power, but there's no overlap with other MEPCs. Task vehicle selection in this scenario will need to incorporate a mechanism to prioritize vehicles, possibly based on the urgency or importance of their tasks. Some vehicles might have to be denied service, leading to the necessity for an efficient resource allocation strategy.

Case 4: $|\mathcal{K}_t^m| > \mathbf{N}_t^m$, and $\mathbb{K}_t^{m,\cap} \neq \emptyset$. This is the most complex scenario, where the MEPC cannot meet the computing needs of all task vehicles, and there's an overlap with other MEPCs. The task vehicle selection process here is more challenging and might need to incorporate elements from both Case 2 and Case 3. A sophisticated mechanism for vehicle prioritization and MEPC selection could be required.

These preconditions indeed play a pivotal role in shaping the task vehicle selection process. They inform the rules and mechanisms for how vehicles choose an MEPC and how an MEPC allocates its resources so that the selected vehicles can effectively fulfill the tasks, minimize risks, and optimize the overall operation of the system.

To better make a selection decision, the service access ratio for MEPC m is taken as one of the evaluation indicators, which can be described as:

$$\mathcal{W}_t^m = \frac{|\mathbb{V}_t^m|}{|\mathcal{K}_t^m|}, m \in \mathcal{M}, t \in \mathcal{T}. \tag{2}$$

2) Assignment of Communication Resources: MEPC m autonomously assigns appropriate spectrum bands to task vehicles within the set of selection decisions. The communication resources pool $\mathcal{B}_t = \{b_1, b_2, \ldots, b_K\}$ consists of orthogonal bands with a time-varying size $|\mathcal{B}_t|$. The assignment of communication resources is represented by $\mathbb{B}_t^m = \{b_t^{n,m}|b_t^{n,m} \in \mathcal{B}_t, n \in \mathbb{V}_t^m\}$, allowing different vehicles to use the same spectrum band. To describe communication interference, we denote the intersection set $\mathbb{K}_t^m$ as:

$$\mathbb{K}_t^m \triangleq \{m'|\mathcal{K}_t^{m,m'} \neq \emptyset\}_{m' \in \mathcal{M}/m}. \tag{3}$$

There exists other communication interference sources for the task vehicle $n \in \mathbb{V}_t^m$ apart from the Additive White Gaussian Noise (AWGN) N_0:

(i) $n \in \mathcal{K}_t^{m,m'}$, interference from $\mathbb{V}_t^m$ and $\mathbb{V}_t^{m'}$, $b_t^{n,m} = b_t^{n',m}|_{n' \in \mathbb{V}_t^m}$, $b_t^{n,m} = b_t^{n',m'}|_{n' \in \mathbb{V}_t^{m'}}$, $m' \in \mathbb{K}_t^m$.

(ii) $n \notin \mathcal{K}_t^{m,m'}$, interference from $\mathbb{V}_t^m$ and $\mathcal{K}_t^{m,m'}$, $b_t^{n,m} = b_t^{n',m}|_{n' \in \mathbb{V}_t^m}$, $b_t^{n,m} = b_t^{n',m'}|_{n' \in \mathcal{K}_t^{m,m'}}$, $m' \in \mathbb{K}_t^m$

(iii) $n \notin \mathcal{K}_t^{m,m'}$, interference from $\mathbb{V}_t^m$, $b_t^{n,m} = b_t^{n',m}|_{n' \in \mathbb{V}_t^m}$.

To quantify the impact of interference on communication resource assignment, we define the communication interference caused by using the same spectrum band as $I_t^{n,m}$:

$$\begin{aligned} I_t^{n,m} = &\sum_{n' \in \mathbb{V}_t^m} \mathbf{1}(b_t^{n,m} = b_t^{n',m}) p_t^{n',m} G_t^{n',m} \\ &+ \sum_{m' \in \mathbb{K}_t^m} \sum_{n' \in \mathcal{K}_t^{m,m'}} \mathbf{1}(b_t^{n,m} = b_t^{n',m'}) p_t^{n',m'} G_t^{n',m'} \\ &+ \sum_{m' \in \mathbb{K}_t^m} \sum_{n' \in \mathbb{V}_t^{m'}} \mathbf{1}(b_t^{n,m} = b_t^{n',m'}) p_t^{n',m'} G_t^{n',m'}, \end{aligned} \tag{4}$$

where $\mathbf{1}(\cdot)$ is the indicator function with $\mathbf{1}(\cdot) = 1$ if event $\cdot$ is true and $\mathbf{1}(\cdot) = 0$ otherwise. Moreover, $p_t^{n',m}$ and $p_t^{n',m'}$ is the transmission power from task vehicle n' to MEPC m and m'. $G_t^{n',m}$ and $G_t^{n',m'}$ are channel gain.

Then the achievable data rate from task vehicle n to MEPC m in time slot t is:

$$R_t^{n,m} = |b_t^{n,m}| \log_2(1 + \Gamma_t^{n,m}), \tag{5}$$

where $b_t^{n,m}$ indicates the selection of spectrum band between task vehicle n and MEPC m. $\Gamma_t^{n,m}$ is the Signal-to-Interference-plus-Noise Ratio (SINR), which is described as:

$$\Gamma_t^{n,m} = \frac{p_t^{n,m} G_t^{n,m}}{N_0 + I_t^{n,m}}, \tag{6}$$

where $p_t^{n,m}$ is the transmission power from task vehicle n to MEPC m, $G_t^{n,m}$ is the channel gain, which is shown as:

$$G_t^{n,m} = \rho_t^{n,m} G_0 (d_0/d_t^{n,m})^{\aleph}, \tag{7}$$

where $\rho_t^{n,m}$ denotes the small-scale fading channel gains between the nth task vehicle and mth MEPC within the time slot t. Here, G_0 signifies the path-loss constant, while d_0 and $d_t^{n,m}$ represent the reference distance and the actual distance between task vehicle n and MEPC m, respectively. Given the mobility of task vehicles and platoons, the actual relative distance is subject to variation over time. The term $\aleph$ stands for the path-loss exponent. While channel conditions are assumed to remain consistent throughout a single time slot, they can exhibit variations across different slots.

Thus, the transmission latency from task vehicle n to MEPC m can be expressed as:

$$\mathcal{T}_t^{tr,n \to m} = v_t^n / R_t^{n,m}. \tag{8}$$

The energy consumption $\mathcal{E}_t^{tr,n \to m}$ is considered as the task vehicle's extra overhead when transmitting a certain amount of data to MEPC m via wireless access, which is given as:

$$\mathcal{E}_t^{tr,n \to m} = p_t^{n,m} \mathcal{T}_t^{tr,n \to m}. \tag{9}$$

3) Allocation of Computing Resources: Since the result of the task is relatively small, the time consumption of the back-haul link can be neglected.[2] Make $\mathbb{C}_t^m = \{c_t^{n,m} | \sum_{n\in\mathbb{V}_t^m} c_t^{n,m} \le C_t^{m,co}\}$ be mth MEPC's decision on the allocation of computing resources at time slot t. We use $C_t^{m,co} = f\mathbf{N}_t^m$ to denote the total processing power of MEPC m. f is the processing capacity of each Platoon Member (PM). When a task completes, we calculate the execution latency as:

$$\mathcal{T}_t^{co,n\to m} = v_t^n d_t^n / c_t^{n,m}. \tag{10}$$

The corresponding energy consumption for nth task is:

$$\mathcal{E}_t^{co,n\to m} = \mu d_t^n v_t^n \left(c_t^{n,m}\right)^2, \tag{11}$$

where μ is the effective coefficient related to chip structure.

Overall, the completion time of a task can be defined as the period from task generation to the return of the results to the task vehicle. We express the completion time as follows:

$$T_t^{n,m} = \mathcal{T}_t^{co,n\to m} + \mathcal{T}_t^{tr,n\to m}, n \in \mathbb{V}_t^m. \tag{12}$$

Correspondingly, the whole energy consumption of MEPC m can be written as:

$$E_t^m = \sum_{n\in\mathbb{V}_t^m} \mathcal{E}_t^{co,n\to m} = \sum_{n\in\mathbb{V}_t^m} \mu d_t^n v_t^n \left(c_t^{n,m}\right)^2. \tag{13}$$

4) Setting of Trading Pricing: To boost the trading economy, we build the utility model of MEPC and task vehicle, respectively. We define $\mathbb{P}_t^m = \{P_t^{n,m} | n \in \mathbb{V}_t^m\}$ as the decision on the setting of trading pricing.

MEPC's Utility $\mathcal{U}_t^{p,m}$: It is defined as the profit made through the provision of resource services, which consists of two components: the income G_t^m obtained from trading with the task vehicles; the cost E_t^m incurred by service trading. We formulate MEPC's utility as follows:

$$\begin{aligned}\mathcal{U}_t^{p,m} &= \omega_t^{m,p} G_t^m - \omega_t^{m,e} E_t^m \\ &= \sum_{n\in\mathbb{V}_t^m} (\omega_t^{m,p} P_t^{n,m} \mathbf{1}(T_t^n < \tau_t^n) c_t^{n,m} - \omega_t^{m,e} \mathcal{E}_t^{co,n}),\end{aligned} \tag{14}$$

where $P_t^{n,m} \in [P_{\min}, P_{\max}]$ is the relevant unit price of MEPC m for compute resources. $0 < \omega_t^{m,p} < 1$ (in units 1/Dollar) and $0 < \omega_t^{m,e} < 1$ (in units 1/Joule) are weights of the trading income and energy consumption.

Task Vehicle's Utility $\mathcal{U}_t^{v,n}$: It is defined as the benefit obtained from service trading, which mainly involves three features: the task completion time saved $T_t^{s,n}$ from the MEPC services; the payment H_t^n for the required resources and service; and the energy consumption $\mathcal{E}_t^{tr,n\to m}$ incurred by offloading tasks to MEPC via Vehicle-to-Vehicle (V2V) communication. Correspondingly, the $\mathcal{U}_t^{v,n}$ is given by:

$$\begin{aligned}\mathcal{U}_t^{v,n} &= \omega_t^{m,t} T_t^{s,n} - \omega_t^{m,p} H_t^n - \omega_t^{m,e} \mathcal{E}_t^{tr,n\to m} \\ &= \omega_t^{m,t} (\tau_t^n - T_t^n) - \omega_t^{m,p} P_t^{n,m} \mathbf{1}(T_t^n < \tau_t^n) c_t^{n,m} \\ &\quad - \omega_t^{m,e} \mathcal{E}_t^{tr,n\to m},\end{aligned} \tag{15}$$

where $0 < \omega_t^{m,t} < 1$ (in units 1/Second) is the weight of the saved completion time.

[2]Drawing from the current literature [24], [25], we assume that the transmission power for both the task vehicle and the MEPC is equivalent.

C. Problem Formulation

Trading mode: To avoid time consumption caused by the frequent information exchange between the central controller and MEPC, we consider distributed trading strategies in this trading market; MOOP: Make optimal decisions of service selection, spectrum band assignment, computing resources allocation, trading pricing setting, and achieve good performance in service access ratio and players' utility. Accordingly, we can express the MOOP for MEPC m as:

$$\begin{aligned}\mathbf{P1}: &\max_{\mathbb{V}_t^m, \mathbb{B}_t^m, \mathbb{C}_t^m, \mathbb{P}_t^m} \lim_{T\to\infty} \frac{1}{T} \sum\nolimits_{t\in\mathcal{T}} \mathcal{W}_t^m \\ &\lim_{T\to\infty} \frac{1}{T} \sum\nolimits_{t\in\mathcal{T}} \mathcal{U}_t^{p,m} \\ &\lim_{T\to\infty} \frac{1}{T} \sum\nolimits_{t\in\mathcal{T}} \sum\nolimits_{n\in\mathbb{V}_t^m} \mathcal{U}_t^{v,n} \\ \text{s.t. } &C1: \mathbb{V}_t^m = \{n | v_t^{n,m} = 1, n \in \mathcal{K}_t^m\}, v_t^{n,m} \in \{0,1\}, \\ &C2: \mathbb{B}_t^m = \{b_t^{n,m} | b_t^{n,m} \in \mathcal{B}, n \in \mathbb{V}_t^m\}, \\ &C3: \mathbb{C}_t^m = \{c_t^{n,m} | \sum\nolimits_{n\in\mathbb{V}_t^m} c_t^{n,m} \le C_m^{co}\}, \\ &C4: \mathbb{P}_t^m = \{P_t^{n,m} | P_t^{n,m} \in [P_{\min}, P_{\max}], n \in \mathbb{V}_t^m\}.\end{aligned} \tag{16}$$

where $C1$ is the constraint of service selection decision, $C2$ enforces the spectrum band to be selected in an inherent resource pool, $C3$ ensures that, in each time slot, the total allocated computing resources cannot exceed the processing capability of MEPC m, $C4$ imposes the maximum and minimum trading pricing.

It is a hybrid control (continuous and discrete) problem for which conventional dynamic programming methods have no way of solving directly for the optimal solution. Therefore, we choose DRL methods to find the sub-optimal solution to this hybrid problem. However, because our multi-agent environment is fully distributed, the reward of one MEPC is affected by other MEPCs' actions. Besides, existing DRL methods, such as Deep Deterministic Policy Gradient (DDPG), require that the agents' reward be influenced only by their actions and are not suitable for solving this multi-agent scenario in this work. Hence, we propose our MARL algorithm and build a training architecture to tackle the above problem. In the following section, we then introduce the preliminary study for our MARL environment.

IV. Preliminaries Study for MARL Environment

A. Policy Gradients

Given policy $\pi : \mathcal{S} \to \mathcal{A}$ parameterized by θ, the objective function is the expected discounted cumulative reward $J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Policy gradient aims to optimize θ by maximizing this objective through gradient ascent, expressed as $\nabla_\theta J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \pi(s|\theta) \nabla_a Q(s,a|\phi) | a = \pi(s|\theta)\right], \tag{17}$$

where $Q(s,a|\phi)$ is a state-action value function parameterized by ϕ, approximated by a neural network known as the Q-function. DDPG combines ideas from Deterministic Policy Gradient (DPG) and Deep Q-Network (DQN) to concurrently learn $Q(s,a|\phi)$ and policy $\pi(s|\theta)$. However, DDPG suffers from overestimated value estimates and sub-optimal policies [26].

B. Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic policy gradient (TD3) addresses DDPG's overestimation issue through three key tricks: Clipped Double-Learning, Delayed Policy Update, and Target Policy Smoothing Regularization.

Clipped Double-Q Learning: Two Q-functions are learned and the smaller Q-value is used to be the target value in the Bellman error loss function, which helps fend off overestimation in the Q-function.

$$y = r + \gamma \min_{i=1,2} Q'(s', a'|\psi_i'), \tag{18}$$

where ψ_i' is the parameter of the target critic network i, and the arget action is $a' = \pi'(s'|\theta')$.

Both Q-functions are learned by regressing to this target value y, with the loss function below for $i = 1, 2$:

$$L(\psi_i) = \mathbb{E}_{s,a,r,s' \sim D}\big[(Q(s, a|\psi_i) - y)^2\big]. \tag{19}$$

The policy $\pi(s|\theta)$ can be learned by maximizing Q as follows:

$$\max_{\theta} \mathbb{E}_{s \sim D}\big[Q(s, \pi(s|\theta)|\psi_1)\big]. \tag{20}$$

Delayed Policy Update: The policy (including target policy) usually updates after twice updates of Q-functions.

Target Policy Smoothing Regularization: Noise is added to the target policy output to avert the exploitation of Q-function errors.

$$a' = \text{clip}(\pi'(s'|\theta') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}), \tag{21}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$ is the clipped Gaussian noise.

C. Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3)

Multi-agent Twin Delayed Deep Deterministic policy gradient (MATD3) is an extension of TD3 under a multi-agent setting. Each agent has its own policy $\pi(o_m|\theta_m), m \in \mathcal{M}$, and the gradient of each policy is written as:

$$\begin{aligned}&\nabla_{\theta_m} J(\theta_m)\\ &\quad= \mathbb{E}_{s,a \sim D}\big[\nabla_{\theta_m} \pi(o_m|\theta_m)\\ &\qquad \nabla_{a_m} Q(s, a_1, \ldots, a_M|\psi_{m,1})|a_m = \pi(o_m|\theta_m)\big],\end{aligned} \tag{22}$$

where $Q(s, a_1, \ldots, a_m, \ldots, a_M|\psi_{m,1})$ is a centralized action-value function representing the Q-value estimation for an agent with all agents' actions and the overall environment state s. Each agent has its own action space and rewards function to derive its value function. MATD3 ensures environment stationarity by obtaining the actions of all other agents and accommodating changes in their policies. It is a versatile algorithm in MARL, applicable to hybrid cooperative and competitive environments [27].

D. Prioritized Experience Replay

Reinforcement learning benefits from experience replay, reusing past experiences to address correlated updates and scarce but valuable experiences. Prioritized experience replay focuses on replaying significant transitions more frequently to address this limitation [28].

Greedy TD-error Prioritizing: Transitions with higher absolute TD errors are replayed with greater probability. The weights are updated proportionally to TD error in a Q-learning manner. New transitions without a known TD error receive maximal priority to ensure they are replayed at least once.

Stochastic Prioritizing: To address the limitations,[3] a stochastic sampling method is introduced, striking a balance between pure greedy and random sampling. It allows non-zero probability for sampling low-priority transitions. The sampling probability is monotonic concerning the transition's priority and shown as:

$$P(i) = p_i^{\alpha} / \sum_k p_k^{\alpha}, \tag{23}$$

where $p_i > 0$ is transition's priority. α denotes prioritization usage, especially $\alpha = 0$ for uniform sampling.

Annealing the Bias: The bias introduced by prioritized replay are corrected by using importance-sampling weights.

V. MARL Environment for Trading Market

In this dynamic and complex multi-MEPC environment, the decision process does not conform to the stationary property required by the MDP. Hence, we introduce Partially Observable Markov Games (POMDPs) to represent the interactions among MEPCs [29]. POMDPs provide a realistic portrayal of the multi-MEPC scenario. This POMDPs $\mathcal{M} =< \mathcal{S}, \mathcal{H}, \mathcal{R}, \mathcal{O}, \gamma >$ includes the state set $\mathcal{S}$, a hybrid action space $\mathcal{H}$, an observation space $\mathcal{O}$, a reward function $\mathcal{R}$, and a discounted factor γ.

A. State Space

$\mathcal{S} \triangleq \{s_t = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)\}$ is the state space consisting of four components as follows:

- The first channel of state $\mathcal{S}_1$ includes the covering conditions of all MEPCs, $\mathcal{S}_1 = \{\mathcal{K}_t^1, \ldots, \mathcal{K}_t^m, \ldots, \mathcal{K}_t^M\}_{m \in \mathcal{M}, t \in \mathcal{T}}$.
- The second channel of sate $\mathcal{S}_2$ is the task information of the covered task vehicles, $\mathcal{S}_2 = \{\mathcal{X}_t^1, \ldots, \mathcal{X}_t^n, \ldots, \mathcal{X}_t^N\}_{n \in \mathcal{N}, t \in \mathcal{T}}$.
- The third channel of state $\mathcal{S}_3$ is the number of PMs $\mathbf{N}_t^m$, $\mathcal{S}_3 = \{\mathbf{N}_t^1, \ldots, \mathbf{N}_t^m, \ldots, \mathbf{N}_t^M\}_{m \in \mathcal{M}, t \in \mathcal{T}}$.
- The forth channel of state $\mathcal{S}_4$ is the available spectrum bands, $\mathcal{S}_4 = \{|\mathcal{B}_t|\}_{t \in \mathcal{T}}$.

B. Observation Space

$\mathcal{O} \triangleq \{o_t^m = (\mathcal{O}_1^m, \mathcal{O}_2^m, \mathcal{O}_3^m, \mathcal{O}_4^m)\}$ is a finite observation space, describing MEPC's experience. leftmargin=*

- The first channel of observation $\mathcal{O}_1^m$ is the covering condition $\mathcal{K}_t^m$ of MEPC m. Namely, $\mathcal{O}_1^m = \mathcal{K}_t^m, t \in \mathcal{T}$.
- The second channel of observation $\mathcal{O}_2^m$ is the information of vehicles covered by MEPC m. Accordingly, $\mathcal{O}_2^m = \{\mathcal{X}_t^n\}_{n \in \mathcal{K}_t^m, t \in \mathcal{T}}$.
- The third channel of observation $\mathcal{O}_3^m$ is the number of PMs of MEPC m. Namely, $\mathcal{O}_3^m = \mathbf{N}_t^m, t \in \mathcal{T}$.
- The fourth channel of observation $\mathcal{O}_4^m$ is the available spectrum bands, which is global information for all MEPCs. Namely, $\mathcal{O}_4^m = |\mathcal{B}_t|, t \in \mathcal{T}$.

Overall, the observation of MEPC m at time slot t is:

$$o_t^m = (\mathcal{K}_t^m, \{\mathcal{X}_t^n\}_{n \in \mathcal{K}_t^m}, \mathbf{N}_t^m, |\mathcal{B}_t|), m \in \mathcal{M}, t \in \mathcal{T}. \tag{24}$$

[3]Issues include: 1) infrequent replay of transitions with low TD error; 2) significant impact of noise spikes on approximation errors; 3) susceptibility to overfitting due to lack of diversity.

C. Hybrid (Discrete-Continuous) Action Space

Different from the action space in conventional RL (only considering discrete or continuous action space), the hybrid action space is considered in this work, which can be defined as: $\mathcal{H} \triangleq \{h_t^m = (\mathbb{V}_t^m, \mathbb{B}_t^m, \mathbb{C}_t^m, \mathbb{P}_t^m) | \mathbb{V}_t^m \subseteq \mathcal{N}\}_{m\in\mathcal{M}, t\in\mathcal{T}}$, where the action space of $\mathbb{V}_t^m$ and $\mathbb{B}_t^m$ are discrete, and the action space of $\mathbb{C}_t^m$ and $\mathbb{P}_t^m$ are continuous.

Traditional RL algorithms prove insufficient when applied to the hybrid action space $\mathcal{H}$. This arises from the standard policy representations, typically characterized by polynomial or Gaussian distributions, which fail to account for the heterogeneity of discrete and continuous actions. Furthermore, implicit policies derived from action value functions in value-based algorithms are not capable of maximizing over an infinite hybrid action space. A connection exists between discrete actions and continuous parameters, wherein discrete actions generally define the space of their associated continuous parameters. From a theoretical perspective, before continuous parameters can be determined, the optimal hybrid strategy necessitates the prior selection of discrete actions.

D. Reward Function

To ensure mutually beneficial outcomes in this trading market, we define the local reward function of MEPC to incorporate three objectives: service access ratio, service quality, and trading profit, in a "utility-efficient" manner. Additionally, a global trading reward is defined to evaluate the overall trading performance.

- *Global Trading Reward*: The global reward evaluates the MEPCs' cooperation, which defined as:

$$r_g^t = \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{N}} \lg\{1 + \frac{\mathcal{W}_t^m}{1 + I_t^{n,m}}\}. \tag{25}$$

- *Local Trading Reward*: The local reward measures each MEPC's performance (service access ratio, service quality, and trading profit), which is designed as:

$$r_{l,m}^t = \mathcal{W}_t^m \big(\alpha \mathcal{U}_t^{p,m} + (1-\alpha) \sum_{n\in\mathbb{V}_t^m} \mathcal{U}_t^{v,n}\big), m \in \mathcal{M}, \tag{26}$$

where α is the weight of the MEPC's utility.

E. MARL Goal

For the M MEPCs, the optimization problems can be expresses as:

$$\max_{\theta_m} J(\theta_m), m \in \mathcal{M}, \pi(\cdot|\theta_m) \in \Pi, \tag{27}$$

where $\pi(\cdot|\theta_m)$ is the policy of MEPC m. Π is the set of all feasible policies for MEPC m. Each MEPC interacts with the trading market and takes action according to its policy π, aiming at solving the optimization problem (16), or in other words, maximizing its total expected reward (27).

Rewards are functions of states and actions. During the training stage, a corresponding reward will be returned to the MEPC at time slot t once the chosen action is taken by this MEPC at the previous time slot. Then according to the received reward, each MEPC updates its policy π to direct to an optimal one. In a single time slot t, each observation o_t^m only represents the current observation of a platoon, and each reward r_t^m is just an indicator to measure gains or losses at this moment.

VI. Solution: GL-HPMATD3-Based Trading Decision

A. Parameter for Policy and Value-Function of MEPC

To better present our algorithm, we first define the parameter of each policy and action-value function.

1) *Policy* π parameterized by

$$\Theta = \{\theta_1, \ldots, \theta_m, \ldots, \theta_M\}, m \in \mathcal{M}$$

where θ_m represents the learnable variables that define the behavior of the evaluation network of the local actor.

2) *Target Policy* π' parameterized by

$$\Theta' = \{\theta'_1, \ldots, \theta'_m, \ldots, \theta'_M\}, m \in \mathcal{M}$$

where θ'_m represents the learnable variables that define the behavior of the mth MEPC's target network of the local actor.

3) *Action-Value Function* Q_l parameterized by

$$\Phi_l = \{\phi_{l,1}, \ldots, \phi_{l,m}, \ldots, \phi_{l,M}\}, m \in \mathcal{M}$$

where parameters $\phi_{l,m}$ are determined through the local critic's interaction with the environment and are indispensable for evaluating the quality or value of actions.

4) *Target Action-Value Function* Q'_l parameterized by

$$\Phi'_l = \{\phi'_{l,1}, \ldots, \phi'_{l,m}, \ldots, \phi'_{l,M}\}, m \in \mathcal{M}$$

where parameters $\phi'_{l,m}$ are peculiar to the target action-value function Q'_l, connected with the mth MEPC.

5) *Global Action-Value Function* Q_g^i parameterized by

$$\Psi_g^i = \{\psi_{g,1}^i, \ldots, \psi_{g,m}^i, \ldots, \psi_{g,M}^i\}, m \in \mathcal{M}, i = 1, 2$$

where parameters $\psi_{g,m}^i$ are unique to the global action-value function Q_g^i, corresponding to the mth MEPC.

6) *Target Global Action-Value Function* $Q_g'^i$ parameterized by

$$\Psi_g'^{,i} = \{\psi_{g,1}'^{,i}, \ldots, \psi_{g,m}'^{,i}, \ldots, \psi_{g,M}'^{,i}\}, m \in \mathcal{M}, i = 1, 2$$

where parameters $\psi_{g,m}'^{,i}$ are specific to the target global action-value function $Q_g'^i$ associated with the mth MEPC.

B. Hybrid Action Space Representation

The action space has a hierarchical structure with discrete actions parameterized by real-valued vectors. The hybrid action space serves as a latent representation space, enabling the agent to learn the latent policy. To interact with the real environment, the agent selects a latent action based on the latent policy, which is then decoded into the hybrid action space. The selection of service access is an integral part of the hybrid action space, but it is not involved in the training for hybrid action representation. This decision is a prerequisite for the other three hybrid decisions, as the set of task vehicles requiring service must be established before proceeding with the assignment of communication resources, allocation of computation resources, and setting of trading pricing.

To determine the service access decision, we employ four pre-set screening scenarios that consider factors such as distance, task priority, communication conditions, and potential service capabilities. However, due to environmental dynamics and complexity, deterministic screening alone is insufficient. Hence, we utilize a training model to maximize player utility and service access ratio. This model continuously optimizes service access decisions, learning from extracted features of

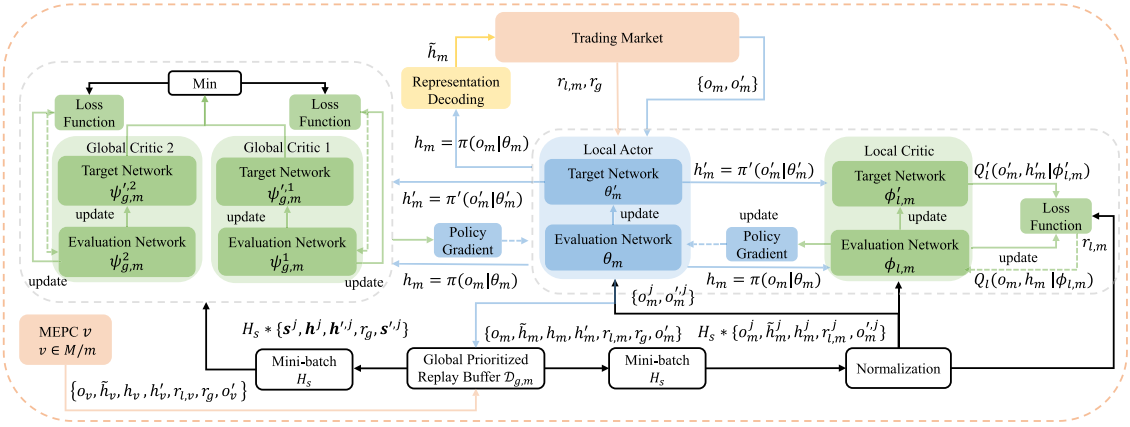


Fig. 2. The global-local training framework.

screening data. This enables each MEPC to accurately identify task vehicles needing service, laying a robust foundation for subsequent hybrid action training and determination.

1) Parameterized Action Space: For each task vehicle $n \in \mathbb{V}_t^m$, an applicable parameterized action space is defined as:

$$\mathcal{H}_n = \{(b, C_b, P_b) | C_b \in \mathbb{C}_t^m, P_b \in \mathbb{P}_t^m \text{ for all } b \in \mathcal{B}_t\}, \quad (28)$$

where $\mathcal{B}_t = \{b_1, b_2, \ldots, b_K\}$ is the discrete action set, $\mathbb{C}_t^m$ and $\mathbb{P}_t^m$ are the corresponding continues parameter sets for each $b \in \mathcal{B}_t$. We call any pair of b, C_b, P_b as a hybrid action and also call $\mathcal{H}_n$ as hybrid action space of task vehicle n for short in this paper. In this work, different discrete actions can share the same continuous parameters.

2) Encoding and Decoding: **Encoding**: $e_{\zeta,b} = E_\zeta(b)$, $z_c, z_p \sim q_\sigma(\cdot|C_b, P_b, s, e_{\zeta,b})$ where E_ζ is an embedding table with learnable parameter ζ to represent the discrete actions, $q_\sigma(\cdot)$ is a encoder parameterized by σ to map continuous action C_b, P_b into the latent variable z_c, z_p.

Decoding: $b = g_E(e) = \arg\min_{b' \in \mathcal{B}_t} \|e_{\zeta,b'} - e\|_2$ $C_b, P_b = p_\varrho(z_c, z_p, s, e_{\zeta,b})$, where $p_\varrho(\cdot)$ is the decoder parameterized by ϱ to reconstruct the continues parameter C_b, P_b from z_c, z_p.

3) Dynamic Predictive Representation: The prediction $\widetilde{\delta}_{s,s'}$ is produced as follows:

$$\widetilde{\delta}_{s,s'} = p_\varrho(z_c, z_p, s, e_{\zeta,b}), \text{for } s, e, z_c, z_p. \quad (29)$$

We minimize the L_2-norm square prediction error:

$$L_{\text{Dyn}}(\sigma, \varrho, \zeta) = \mathbb{E}_{s,b,C_b,P_b,s'}\left[\|\widetilde{\delta}_{s,s'} - \delta_{s,s'}\|_2^2\right], \quad (30)$$

where $\delta_{s,s'} = s' - s$ is denoted as state residual.

To better represent the hybrid action space, the ultimate training loss is derived as follows:

$$L_{\text{HyAR}}(\sigma, \varrho, \zeta) = L_{\text{VAE}}(\sigma, \varrho, \zeta) + \beta L_{\text{Dyn}}(\sigma, \varrho, \zeta), \quad (31)$$

where β is a hyper-parameter to weight the representation loss for dynamic prediction. $L_{\text{VAE}}(\sigma, \varrho, \zeta)$ can be described as:

$$L_{\text{VAE}}(\sigma, \varrho, \zeta) = \mathbb{E}_{s,b,C_b,P_b \sim \mathcal{D}, z \sim q_\sigma}\left[\|C_b - \widetilde{C}_b\|_2^2 + \|P_b - \widetilde{P}_b\|_2^2 + D_{\text{KL}}(q_\sigma(\cdot|C_b, P_b, s, e_{\zeta,b}) \| \mathcal{N}(0, I))\right], \quad (32)$$

where $\widetilde{C}_b$, $\widetilde{P}_b$, are the reconstructed continuous parameters from z_c, z_p, respectively. D_{KL} is the Kullback-Leribler divergence. The embedding table E_ζ and the conditional VAE q_σ, q_ϱ are trained together by minimizing the loss function L_{VAE}.

C. Trading Decision-Based GL-HPMATD3

The optimal discrete action is expected to find by our proposed algorithm to avoid the exhaustive search for continuous parameters that match it. The global-local training framework is described in Fig.2.

1) Local DDPG: The local DDPG consists of two main components: the critic (target critic) and the actor (target actor). The actor generates an action based on the policy π using the input observation. The critic, represented by the action-value function Q_l, evaluates the selected action. The critic network updates its parameters by minimizing the loss, using a random mini-batch of H_s transitions $(\mathbf{o}_m, \mathbf{r}_m^l, \mathbf{h}_m, \mathbf{o}'_m)$ to update $\phi_{l,m}$.

$$L(\phi_{l,m}) = \mathbb{E}_{o_m, r_{l,m}, h_m, o'_m \sim \mathcal{D}_{g,m}}\left[(y_l - Q_l(o_m, h_m|\phi_{l,m}))^2\right], \quad (33)$$

where y_l is defined as:

$$y_l = r_{l,m} + \gamma Q'_l(o'_m, h'_m|\phi'_{l,m})|_{h'_m = \pi'(o'_m|\theta'_m)}. \quad (34)$$

Update critic parameter

$$\phi_{l,m} \leftarrow \phi_{l,m} - \alpha^Q \cdot \nabla_{\phi_{l,m}} L, \quad (35)$$

where α^Q is the learning rate.

We can adjust the $\phi_{l,m}$ with the gradient of the loss function $L(\phi_{l,m})$, if $L(\phi_{l,m})$ is continuously differentiable. Meanwhile, the actor function is deterministic and approximated by a DNN with parameter θ_m. Thus, we define $\pi(o_m|\theta_m) = h_m$. The parameters of the actor can update by maximizing the policy objective function, which can be described as:

$$J(\theta_m) = \mathbb{E}_{o_m, h_m \sim \mathcal{D}_{g,m}}\left[Q_l(o_m, h_m|\phi_{l,m})|_{h_m = \pi(o_m|\theta_m)}\right]. \quad (36)$$

For each MEPC, the policy gradient of the objective function can be shown as follows:

$$\nabla_{\theta_m} J(\theta_m) = \mathbb{E}_{o_m, h_m \sim \mathcal{D}_{g,m}} \left[\nabla_{h_m} Q_l(o_m, h_m|\phi_{l,m}) \nabla_{\theta_m} \pi(o_m|\theta_m)\right], \quad (37)$$

where π represents the deterministic policy $\pi : o_m \mapsto h_m$, which is continuous. θ_m can be adjusted in the direction of $\nabla_{\theta_m} J(\theta_m)$. The actor network can be updated by

$$\theta_m \leftarrow \theta_m - \alpha^\pi \cdot \nabla_{\theta_m} J(\theta_m), \quad (38)$$

where α^π is the learning rate for MEPC's policy.

A copy of the actor and critic, Q_l' and π', are used for calculating the target values. With real-time updated θ_m and $\phi_{l,m}$, the parameters of the target networks, θ_m', $\phi_{l,m}'$, then can be softly updated as follows,

$$\begin{aligned}\theta_m' &= \kappa^{\pi}\theta_m + (1-\kappa^{\pi})\theta_m' \\ \phi_{l,m}' &= \kappa^{Q}\phi_{l,m} + (1-\kappa^{Q})\phi_{l,m}'\end{aligned} \tag{39}$$

where $\kappa^{\pi} \ll 1$ and $\kappa^{Q} \ll 1$ are the update rate. In this way, the target network values change slowly, different from the design in DQN in that the target network stays frozen for some time. Batch normalization is applied to fix it by normalizing every dimension across samples in one mini-batch. Modification to DPG, inspired by the success of DQN, allows it to use neural network function approximators to learn in large state and action spaces online.

2) Global TD3: The MATD3 for MEPC m can be written as:

$$\nabla_{\theta_m} J_m = \mathbb{E}_{\mathbf{s},\mathbf{h}\sim\mathcal{D}_{g,m}}\left[\nabla_{h_m} Q_g^1(\mathbf{s},\mathbf{h}|\psi_{g,m}^1)\nabla_{\theta_m}\pi(o_m|\theta_m)\right], \tag{40}$$

where $\mathbf{s} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ and $\mathbf{h} = (h_1, \ldots, h_m, \ldots, h_M)$ are the total state and hybrid action space. For the trading system, we consider the modified policy gradient for each MEPC m.

The twin global critics are updated as:

$$L(\psi_{g,m}^i) = \mathbb{E}_{s,h,r_g,s'\sim\mathcal{D}_{g,m}}\left[(Q_g^i(s,h_1,\ldots,h_M|\psi_{g,m}^i) - y_g)^2\right], \tag{41}$$

where y_g is defined as:

$$y_g = r_g + \gamma \min_{i=1,2} Q_g'^{,i}(s', h_1', \ldots, h_M'|\psi_m'^{,i})|_{h_m'=\pi'(o_m'|\theta_m')}. \tag{42}$$

Combine the local and global rewards, we describe the modified policy gradient of each MEPC as follows:

$$\begin{aligned}\nabla_{\theta_m} J_m(\theta_m) = &\overbrace{\mathbb{E}_{\mathbf{s},\mathbf{h}\sim\mathcal{D}_{g,m}}[\nabla_{h_m} Q_g^1(\mathbf{s},\mathbf{h}|\psi_{g,m}^1)\nabla_{\theta_m}\pi(o_m|\theta_m)]}^{\text{MATD3}} \\ &+\underbrace{\mathbb{E}_{o_m,h_m\sim\mathcal{D}_{g,m}}[\nabla_{h_m} Q(o_m,h_m|\phi_{l,m})\nabla_{\theta_m}\pi(o_m|\theta_m)]}_{\text{DDPG}}.\end{aligned} \tag{43}$$

3) GL-HPMATD3 Algorithm: Overall, there are three stages of GL-HPMATD3, **initialization**, **hybrid representation**, and **training**, the pseudo-code of which are described in **Algorithm 1** [Please refer to Appendix A, See the Supplememtary Material].

Initialization: In the initialization stage, we initialize the parameters of the neural network (global and local network of each MEPC) and training processing (lines 2-7).

Hybrid representation: In the hybrid representation stage, the first thing is to prepare the replay buffer $D_{g,m}$ of each MEPC. For simplicity, the agent interacts with the environment and generates data with a random policy. The hybrid action representation can be pre-trained with each MEPC's replay buffer. The parameters of the embedding table and conditional VAE can be updated by minimizing the loss of dynamic prediction until reaching the maximum training time of hybrid representation (lines 11-12).

Training: In the training stage, we first reset simulation parameters and randomly generate an initial state $\mathbf{s}$ (line 16). Then, select the latent action according to the current policy and exploration noise (line 19). To interact with the environment, decode the latent action into the original action (line 20). After calculating the global and local reward, transitions are stored in each prioritized replay buffer $\mathcal{D}_{g,m}$ (line 25). By using **Algorithm 4** [Please refer to Appendix B, See the Supplememtary Material], sample a prioritized mini-batch of H_s transitions from the replay buffer (line 28). Through **Algorithm 2** [Please refer to Appendix B, See the Supplememtary Material], the global critic networks are trained (lines 27-35). Meanwhile, the local critic networks are trained through **Algorithm 3** [Please refer to Appendix C, See the Supplememtary Material] (lines 32-33). Finally, the latent policy is also continuously updated as the global and local critic networks are trained (lines 37-39).

The implementation of an advanced MARL algorithm greatly improves the accuracy of service access selection, leading to significant reductions in unnecessary handovers. This predictive modeling approach integrates user mobility patterns, network conditions, and service requirements, refining decision-making and minimizing handover costs. The innovative strategy demonstrates its profound impact on the overall system performance.

D. Modeling MARL Environment

In this part, we present our "Trading on Motorway" environment, which implements the following four API functions.

1) Application Programming Interface Functions: There are four inherent functions for environment modeling. The interactive data can be obtained through this environment.

- `env.init()`: This function can initialize the environment, including the number, position, velocity of MEPCs and vehicles, the number of PMs, task information, available spectrum bands, etc.
- `env.reset()`: The reset function works the same as `env. init()` but is executed at least once after the environment has initialized.
- `env.step(obs_n, action_n) = [obs_next_n, reward_n, done, info]()`: This function output the observation information of the next time slot, reward value, Boolean value of whether to end the episode and other optional information according to the current observation information and selected action.
- `env.render()`: This function visualizes the scenario of multiple MEPCs interactions.

2) MARL Environment Features: In our framework, some pre-defined environments can be used directly, but of course, it is also possible to create specific environments by modifying the following parameters:

- *Number of MEPCs and Task vehicles*: M, N
- *Number of PMs*: $\mathbf{N}_t^m$
- *Position of Sites*: (x, y) for each MEPC and task vehicle
- *Task Information*: $\mathcal{X}_t^n = \{v_t^n, d_t^n, \tau_t^n\}$

VII. Numerical Results

A. Parameter Setting

In this section, we present numerical results to demonstrate the efficiency of our proposed GL-HPMATD3 algorithm. The specific parameter setting of the neural network and training parameters can be found in Table III.

B. Training Performance

1) Change of Loss Curve: The loss curves of local and global training are depicted in Fig. 3, showcasing a decaying

TABLE III
THE SPECIFIC PARAMETER SETTING OF NEURAL NETWORK AND TRAINING PARAMETERS

The Neural Network and Training Parameters			
Parameter	Value	Parameter	Value
Layers of Critic Network	5	Layer Type of Critic Network	Fully Connected
Neurons of First Hidden Layer for Critic Network	1024	Neurons of Second Hidden Layer for Critic Network	512
Neurons of Third Hidden Layer for Critic Network	300	Learning Rate of Critic Network	0.0001
Layers of Actor Network	4	Layer Type of Actor Network	Fully Connected
Neurons of First Hidden Layer for Actor Network	500	Neurons of Second Hidden Layer for Critic Network	128
Learning Rate of Actor Network	0.0001	Activation Function	Relu
Mini-batch	128	Buffer Size	30000
Summary of Simulation Parameters			
Total channel bandwidth B	1MHz	Noise power spectrum density N_0	-174 dBm/Hz
Channel gain $G_{n,t}, G_{t,n}$	uniform in [-50,-30]dBm	Computing power f_t and f_n	$\{1, \cdots, 1.5\}$G cycles/s
CPU cycles for 1 bit ω	100 cycles/bit	Transmission power $P_{n,t}, P_{t,n}$	0.1W [30]
Number of PMs	[5,20]	Input task data R	uniform in [100, 150]Mb
The Hyper-parameters of the GL-HPMATD3			
Steps T	1000	Episode	10000
Memory Capacity	2000	γ	0.95
Learning Frame	Pytorch 1.12.1	Python Version	3.6

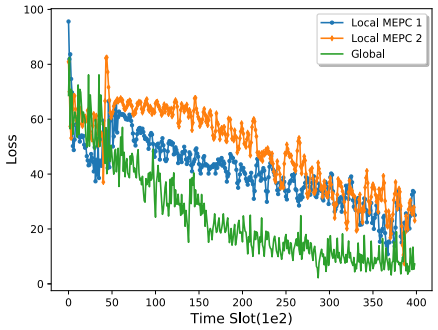


Fig. 3. Training curves on local loss and global loss.

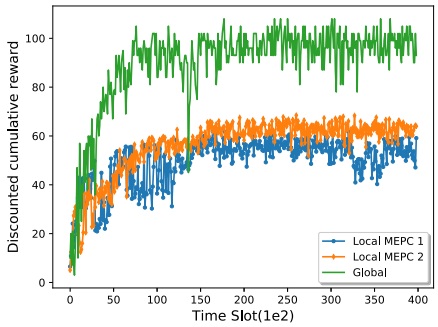


Fig. 4. Training curves on discounted cumulative reward.

and converging trend, despite occasional spikes. These spikes are normal and occur due to extreme situations in the trading process, such as fluctuations in service requests. Notably, the global loss curve demonstrates superior convergence performance compared to the local loss curve. This advantage primarily stems from the global network's training data, which incorporates information from all MEPCs, effectively addressing the non-stationarity that arises from the simultaneous learning of multiple MEPCs. On the other hand, the local networks of individual MEPCs utilize only observable information for policy training, resulting in a relatively slower decline and convergence of the loss curve.

2) Learning Curves of Cumulative Rewards: In Fig.4, we also accumulated the immediate local and global reward for each MEPC in a service trading. We note that the reward value is always positive since the immediate rewards act as encouragement, which is always positive according to our definition. Specifically, each point on the local reward curve is a cumulative result of the immediate rewards that each MEPC obtains at each time slot during its trading service process. In addition, we can see in this figure that the cumulative value of both global and local rewards increases over time. It is mainly because the built model has learned a good policy to guarantee the incremental service access ratio, service quality, and trading profit. When reaching around 15000 and 20000 timeslots, accumulated and global local rewards start to saturate with slight fluctuations. This is because the optimal policy has been learned and the corresponding losses have stabilized gradually.

C. Comparing With Baseline and State-of-the-Art Methods

The methods compared in this section are as follows:

State-of-Art Methods:

DE-MADDPG: A cooperative MARL framework that can maximize the global and local rewards simultaneously [31].

DRLRM: A DRL algorithm based Resource Management scheme, which is decoupled into a pricing policy based on the DDPG algorithm and an offloading policy based on MADDPG [32].

PS-DDPG: An improved DRL algorithm with the PER and stochastic weight averaging mechanisms based on DDPG [33].

Baseline Methods:

Baseline-MADDPG: An learning approach based on MADDPG to learn task scheduling and power control policy [34].

Baseline-DDPG: A model-free and off-policy deterministic RL algorithm based on actor-critic which continuously learns the Q-function to learn the policy and is well adaptive in continuous action spaces [35].

Random Policy (RP): For each MEPC, it selects the service access request from nearby task vehicles randomly. The task vehicle in the overlap set is selected according to the relative shortest distance. The decisions on the assignment of communication resources, allocation of computing resources, and pricing of trading obey uniform distribution.

Utility Greedy Policy (UGP): To increase each MEPC's utility, the overall trading strategy is based on the decisions of selection, assignment, allocation, and pricing.

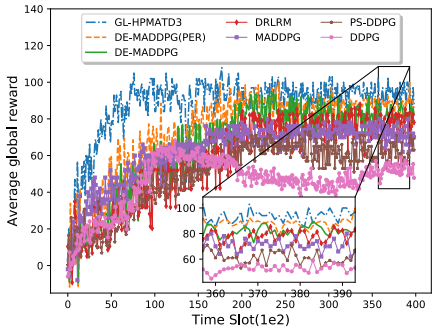


Fig. 5. Convergence of different algorithms.

1) Convergence Performance of the Proposed GL-HPMATD3 Algorithm: In Fig.5, we compare the convergence performance of the proposed GL-HPMATD3 algorithm with state-of-the-art methodologies (DE-MADDPG(PER), DE-MADDPG, DRLRM, PS-DDPG) and baseline methods (MADDPG, DDPG) respectively. The changing trend of each method curve is roughly the same. As the time slot increases, the average global reward raises go up well and eventually tend to a steady value. It can be seen that the GL-HPMATD3 approach outperforms all other methods by a significant margin in convergence speed and average global reward. Compared with DE-MADDPG(PER), GL-HPMATD3 adopts the TD3 training network to alleviate the overestimation DDPG caused. Additionally, the representation of hybrid action space is fully exploited for system training, which is not considered by other methods. Compared with DRLRM, GL-HPMATD3 makes the decisions of request selection, resource allocation, and pricing trading based on DDPG and MATD3, which is more suitable for the overall trading market. The DDPG does not meet the needs of multi-agent training, so the convergence performance is worse than MADDPG and its variants. The variance in convergence time slots holds crucial significance in vehicular networks, particularly for computation-intensive, delay-sensitive tasks. Even minor disparities in decision-making can have a profound impact on real-time performance and user experience, leading to substantial improvements. The distinction lies in computational complexity and real-time decision-making capability.

2) Comparison of MEPC's Utility: In Fig.6 (a)-(c), we show the utility values of each algorithm of the MEPC based on different numbers of PMs. At the same time, the MEPC utility values based on a different number of task vehicles are shown in Fig.6 (d)-(f). Except for the UGP algorithm, which is only used to optimal the MEPC utility, our proposed algorithm GL-HPMATD3 algorithm can ensure that the utility value of each MEPC remains at a high level. This is because the utility function of MEPC is formulated as one objective in the multi-objective function. Our global and local reward functions are designed based on the objective function and constraints, which require MEPC to make near-optimal trading decisions for each accessed task vehicle to maximize the objective function. Therefore, the MEPC utility values based on the GL-HPMATD3 algorithm can always be higher than the MEPC utility values of other algorithms. With the increase in task vehicles, the MEPC utility gradually increases. When the number of task vehicles exceeds the MEPC's capability, the utility of the MEPC decreases to ensure the service access ratio. Obviously, with the increase in the number of PMs, the service capacity of the MEPC becomes powerful, the number of served task vehicles increases and the utility function increases accordingly.

3) Comparison of Task Vehicle's Utility: In Fig. 7, we present a comparison of the utility values of task vehicles at different driving speeds and MEPCs. Our proposed GL-HPMATD3 algorithm significantly enhances the vehicle's utility compared to DE-MADDPG(PER), DRLRM, MADDPG, PS-DDPG, DDPG, RP, and UGP algorithms. The vehicle's utility function is based on energy consumption, execution latency, and service revenue. Through task offloading to the MEPC, energy consumption, and execution latency remain low. Task vehicles driving at speeds within the range of $[30, 80]$ Km/h achieve the highest utility values. This is because high-speed task vehicles can offload tasks to slower-moving MEPCs via V2V communication, leveraging the MEPC's more relaxed task delay constraints. Consequently, the frequency of task offloading between the task vehicle and the MEPC increases, resulting in improved user experience and increased processing revenue, leading to higher utility values for the task vehicle. Furthermore, as the number of PMs increases, the MEPC's processing capacity becomes more powerful, resulting in more efficient task processing. As a result, the utility value of the task vehicle increases with the number of PMs.

4) Comparison of Service Access Ratio: In Fig. 8, the service access ratio based on different numbers of task vehicles is compared for various algorithms. Our proposed GL-HPMATD3 algorithm ensures a higher service access ratio for MEPCs compared to DE-MADDPG(PER), DRLRM, MADDPG, PS-DDPG, DDPG, RP, and UGP algorithms. GL-HPMATD3 achieves this by making quick decisions on selecting requesters and rearranging service requests, ensuring that each task vehicle finds an appropriate service supporter. Consequently, the service access ratio of each MEPC is guaranteed to be high. The service access ratio naturally increases with the number of task vehicles. When the number of task vehicles is 50, the service access ratio of the first MEPC is lower than that of other service supporters due to an excess supply of services relative to the demand. This results in a lower actual service access ratio for some task vehicles. However, as the number of requesters increases, supply and demand become balanced, leading to larger service access ratios for each service supporter.

In Fig.9, with the increase in the number of task vehicles, the service access ratio gradually increases, and finally trends to a stable value. Because the UGP algorithm is designed to increase the MEPC's utility without considering the service access ratio, there is no doubt that UGP has poor performance on the service access ratio. We can see that the proposed GL-HPMATD3 algorithm is ideal in terms of service access ratio. For DE-MADDPG(PER), DRLRM, MADDPG, PS-DDPG, DDPG, RP, and UGP, the average deviation rates between them and GL-HPMATD3 are 11.19%, 13.77%, 18.91%, 37.07%, 43.29%, 58.12%, 63.58%, respectively. In addition, the gap between our proposed GL-HPMATD3 algorithm and other compared algorithms becomes wider, which is because the proposed GL-HPMATD3 algorithm can reasonably distribute global service requests to each MEPC based on the number of task vehicles, overlap range, and relative motion.

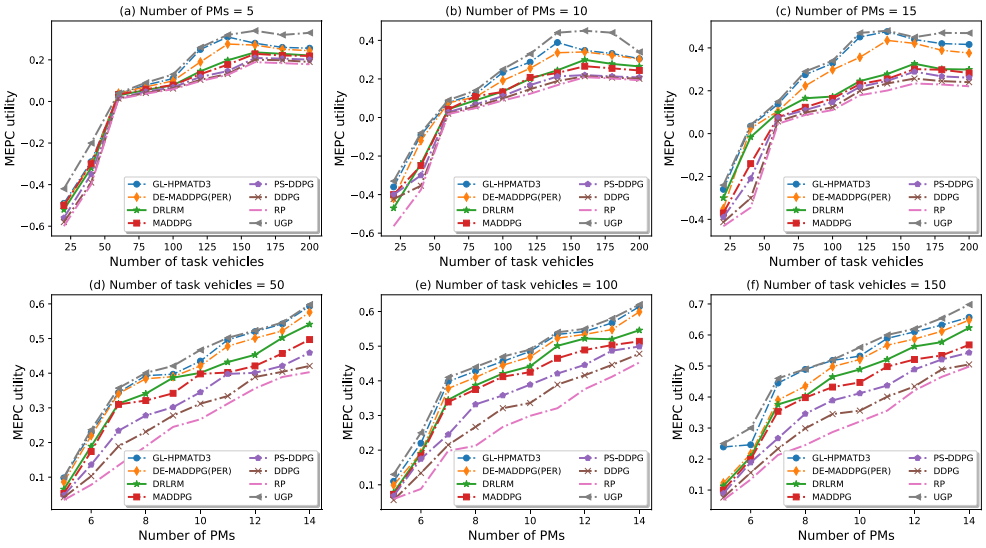


Fig. 6. MEPC utility of different algorithms: (a) Number of PMs is 5. (b) Number of PMs is 10. (c) Number of PMs is 15. (d) Number of task vehicles is 50.(e) Number of task vehicles is 100. (f) Number of task vehicles is 150.

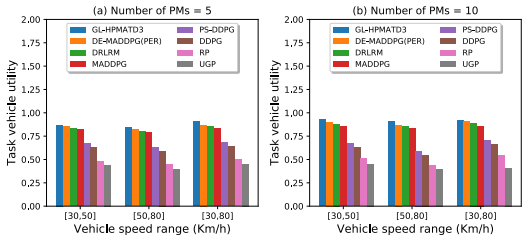


Fig. 7. Task vehicle utility of different algorithms: (a) Vehicle speed range is [30,50]Km/h. (b) Vehicle speed range is [50,80]Km/h. (c) Number of PMs is 5. (d) Number of PMs is 10.

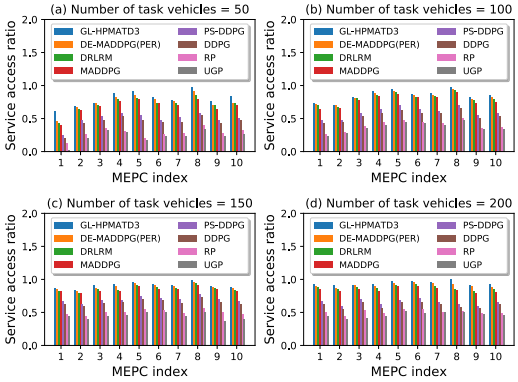


Fig. 8. Service access ratio of different algorithms: (a) Number of task vehicle is 50. (b) Number of task vehicle is 100. (c) Number of task vehicle is 150. (d) Number of task vehicle is 200.

5) Comparison of Execution Latency: In Fig.10 (a) and (b), we show the comparison of the execution latency with different algorithms as the number of task vehicles increases. As the number of task vehicles increases, the execution latency to each algorithm increases. For an MEPC, the service requests will increase with the increase of task vehicles. When the service request exceeds the capacity of MEPC, some task vehicles will be rejected, and the execution latency will increase accordingly. The MEPC with 10 PMs has better performance in execution latency than the MEPC with 5 PMs.

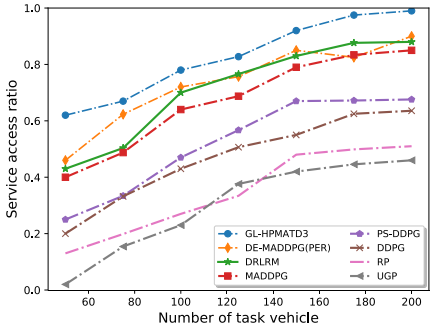


Fig. 9. Service access ratio of per MEPC versus the different task vehicle number.

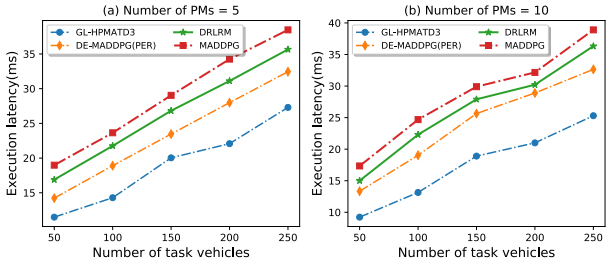


Fig. 10. Execution latency vs. the number of task vehicles: a) the execution latency when PMs is 5. b) the execution latency when PMs is 10.

In Fig. 11, we present a comprehensive comparison of execution latency across various speed ranges. In contrast to algorithms such as DE-MADDPG(PER), DRLRM, and MADDPG, our proposed GL-HPMATD3 algorithm demonstrates superior performance in terms of execution latency. This edge is rooted in our optimization goal, which balances the utility value of task vehicles against task execution latency constraints.[4] Importantly, the driving speed has a

[4]If the majority of vehicles are concentrated in the lower speed range, there is a heightened overall processing latency due to the high delay tolerance. Conversely, when vehicles predominantly operate at higher speeds, resource contention becomes a significant challenge, likely causing task processing disruptions and subsequently prolonging the processing delay.

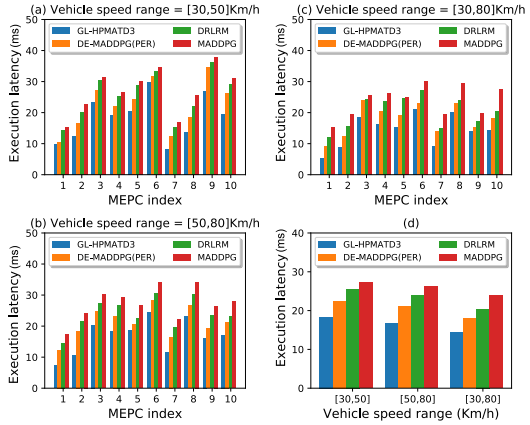


Fig. 11. Execution latency of different algorithms under different speed range: a) the speed range of vehicle is [30,50]Km/h. b) the speed range of vehicle is [50,80]Km/h. c) the speed range of vehicle is [30,80]Km/h. d) average execution latency of different driving speed ranges.

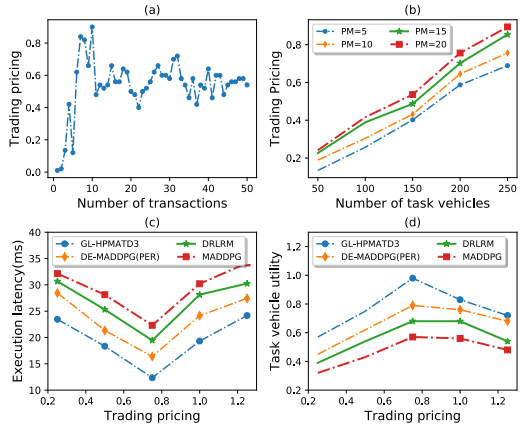


Fig. 12. Performance of trading pricing: a) change curve of trading pricing. b) trading pricing vs. different capability. c) average execution latency of different trading pricing. d) average vehicle utility of different trading pricing.

pronounced effect on the execution latency of our platoon cloud. The data in the figure underscores that execution latency is optimal within the 30 to 80 km/h speed bracket. This trend emerges from the shifting task execution latency constraints due to varied driving speeds. Therefore, our platoon cloud astutely channels more communication resources to compute-intensive, latency-sensitive tasks, refining the overall performance.

6) Trading Pricing: In Fig. 12, we observe the performance of trading pricing. Fig. 12(a) shows the fluctuation of trading pricing with the increase in transaction times, reflecting the equilibrium of the trading market. Fig. 12(b) illustrates the changes in trading pricing as the number of task vehicles increases, where a demand-supply imbalance results in a market premium. In Fig. 12(c) and (d), we present the average execution latency and average utility under different trading pricing. Our proposed GL-HPMATD3 algorithm consistently maintains a lower average execution latency and higher utility values for varying trading pricing. This superiority stems from GL-HPMATD3's ability to adjust trading decisions for each service access vehicle, leading to optimized outcomes in terms of execution latency and utility.

VIII. Conclusion

In conclusion, this study introduces an innovative trading market in platooning-assisted vehicular networks, where vehicle platoons act as service providers and task vehicles as service requesters. To achieve mutually beneficial trading and overall performance, we formulate an MOOP for optimal trading decisions, encompassing service access requests, resource allocation, and trade pricing. To efficiently solve the MOOP, we devise a global-local training architecture using the GL-HPMATD3 algorithm. Simulation results demonstrate the superior performance of GL-HPMATD3, showing improved convergence, service access ratio, and player utility, along with reduced execution latency.

Looking ahead, we recognize the importance of validating our model under complex, real-world conditions. We plan to extend our validation using NS3 for network dynamics and SUMO for mobility patterns to simulate more realistic and complex environments in the future [36], [37].

References

[1] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: Architecture, resource management, security, and challenges," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–46, 2023.

[2] Z. Ning, P. Dong, J. J. P. C. Rodrigues, F. Xia, and X. Wang, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.

[3] T. Xiao, C. Chen, Q. Pei, and H. H. Song, "Consortium blockchain-based computation offloading using mobile edge platoon cloud in Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17769–17783, 2022.

[4] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15513–15526, 2023.

[5] T. Xiao, C. Chen, T. Qiu, C. He, Q. Pei, and H. Cao, "Joint computation resource allocation using mobile-edge-platooning-cloud in the Internet of Vehicles," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, 2021, pp. 1–6.

[6] M. Liwang, Z. Gao, and X. Wang, "Let's trade in the future! A futures-enabled fast resource trading mechanism in edge computing-assisted UAV networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3252–3270, 2021.

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

[8] X. Zhang, M. Peng, S. Yan, and Y. Sun, "Joint communication and computation resource allocation in fog-based vehicular networks," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13195–13208, 2022.

[9] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, "Task offloading and resource allocation for IoV using 5G NR-V2X communication," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10397–10410, 2022.

[10] M. Liwang, R. Chen, and X. Wang, "Resource trading in edge computing-enabled IoV: An efficient futures-based approach," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2994–3007, 2022.

[11] Q. Bao, X. Ren, C. Liu, X. Wang, X. Wang, and C. Qiu, "Resource trading with hierarchical game for computing-power network market," in *Proc. Asia–Pacific Web (APWeb), Web-Age Inf. Manage. (WAIM), Joint Int. Conf. Web Big Data*. Guangzhou, China: Springer, 2021, pp. 94–109.

[12] Z. Yang, K. Liu, Y. Chen, W. Chen, and M. Tang, "Two-level Stackelberg game for IoT computational resource trading mechanism: A smart contract approach," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 1883–1895, 2022.

[13] H. Algarvio and F. Lopes, "Agent-based retail competition and portfolio optimization in liberalized electricity markets: A study involving real-world consumers," *Int. J. Electr. Power Energy Syst.*, vol. 137, 2022.

[14] G. Sun, F. Zhang, D. Liao, H. Yu, X. Du, and M. Guizani, "Optimal energy trading for plug-in hybrid electric vehicles based on fog computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2309–2324, 2019.

[15] U. Awada, J. Zhang, S. Chen, S. Li, and S. Yang, "Resource-aware multi-task offloading and dependency-aware scheduling for integrated edge-enabled IoV," *J. Syst. Archit.*, vol. 141, 2023.

[16] C. Chen, J. Wu, H. Lin, W. Chen, and Z. Zheng, "A secure and efficient blockchain-based data trading approach for Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9110–9121, 2019.

[17] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, 2021.

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

[19] J. Wu et al., "Resource allocation for delay-sensitive vehicle-to-multi-edges (V2Es) communications in vehicular networks: A multi-agent deep reinforcement learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1873–1886, 2021.

[20] X. Liu, Y. Liu, and Y. Chen, "Machine learning empowered trajectory and passive beamforming design in UAV-RIS wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2042–2055, 2021.

[21] M. S. Abegaz, H. N. Abishu, Y. H. Yacob, T. A. Ayall, A. Erbad, and M. Guizani, "Blockchain-based resource trading in multi-UAV-assisted industrial IoT networks: A multi-agent DRL approach," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 1, pp. 166–181, 2023.

[22] H. U. Sheikh and L. Bölöni, "Multi-agent reinforcement learning for problems with combined individual and team reward," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Glasgow, U.K., 2020, pp. 1–8.

[23] J. Xiong et al., "Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," 2018, *arXiv:1810.06394*.

[24] P. Wang, B. Di, H. Zhang, K. Bian, and L. Song, "Platoon cooperation in cellular V2X networks for 5G and beyond," *IEEE Trans. Wireless Commun.*, vol. 18, no. 8, pp. 3919–3932, 2019.

[25] C. Yang, W. Lou, Y. Liu, and S. Xie, "Resource allocation for edge computing-based vehicle platoon on freeway: A contract-optimization approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15988–16000, 2020.

[26] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[27] F. Zhang, J. Li, and Z. Li, "A TD3-based multi-agent deep reinforcement learning method in mixed cooperation-competition environment," *Neurocomputing*, vol. 411, pp. 206–215, 2020.

[28] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.

[29] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Mach. Learn.*, 1994, pp. 157–163.

[30] X. Dai et al., "A learning-based approach for vehicle-to-vehicle computation offloading," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7244–7258, 2023.

[31] H. U. Sheikh and L. Bölöni, "Multi-agent reinforcement learning for problems with combined individual and team reward," 2020, *arXiv:2003.10598*.

[32] X. Zhu, Y. Luo, A. Liu, N. N. Xiong, M. Dong, and S. Zhang, "A deep reinforcement learning-based resource management game in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2422–2433, 2022.

[33] X. He, H. Lu, M. Du, Y. Mao, and K. Wang, "QoE-based task offloading with deep reinforcement learning in edge-enabled Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2252–2261, 2021.

[34] Z. Cheng, M. Min, Z. Gao, and L. Huang, "Joint task offloading and resource allocation for mobile edge computing in ultra-dense network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taipei, Taiwan, 2020, pp. 1–6.

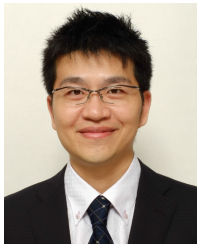
[35] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8027–8038, 2022.

[36] A. Soua, O. Shagdar, and J.-M. Lasgouttes, "Toward efficient simulation platform for platoon communication in large scale C-ITS scenarios," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, Rome, Italy, 2018, pp. 1–6.

[37] C. M. R. Carletti, C. Casetti, J. Härri, and F. Risso, "Platoon-local dynamic map: Micro cloud support for platooning cooperative perception," in *Proc. 19th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Montreal, QC, Canada, 2023, pp. 405–410.

Tingting Xiao received the B.Eng. degree in communication engineering from the Xi'an University of Technology, Xi'an, China, in 2017. She is currently pursuing the Ph.D. degree with Xidian University, Xi'an. Her research interests include wireless communication, the Internet of Vehicle, blockchain, and mobile edge computing.

Chen Chen (Senior Member, IEEE) received the B.Eng., M.Sc., and Ph.D. degrees in telecommunication from Xidian University, Xi'an, China, in 2000, 2006, and 2008, respectively. He was a Visiting Professor with the Department of Electrical Engineering and Computer Science, The University of Tennessee, and the Department of Computer Science, University of California. He is currently a Professor with the School of Telecommunications Engineering, Xidian University. He is also the Director of the Xi'an Key Laboratory of Mobile Edge Computing and Security and the Director of the Intelligent Transportation Research Laboratory, Xidian University. He has authored/coauthored four books and more than 150 scientific papers in international journals and conference proceedings. He has contributed to the development of 11 copyrighted software systems and invented more than 100 patents. He serves as the general chair, the PC chair, the workshop chair, or a TPC member for a number of conferences. He is also a Distinguished Member of the China Computer Federation (CCF) and a Senior Member of the China Institute of Communications (CIC).



Mianxiong Dong (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from The University of Aizu, Aizuwakamatsu, Japan. He was a JSPS Research Fellow with the School of Computer Science and Engineering, The University of Aizu, and a Visiting Scholar with the Broadband Communications Research (BBCR) Group, University of Waterloo, Waterloo, ON, Canada, supported by the JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. He is currently the Vice President and a Professor with the Muroran Institute of Technology, Muroran, Japan. He was selected as a Foreigner Research Fellow (a total of three recipients all over Japan) by Nippon Electric Company Computer and Communication (NEC C&C) Foundation in 2011. He was a recipient of the 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017, the Funai Research Award 2018, the NISTEP Researcher 2018 (one of only 11 people in Japan) in recognition of significant contributions in science and technology, the Young Scientists' Award from MEXT in 2021, the SUEMATSU-Yasuharu Award from IEIEC in 2021, and the IEEE TCSC Middle Career Award in 2021. He is Clarivate Analytics in 2019 and 2021 Highly Cited Researcher (Web of Science) and a Foreign Fellow of the Engineering Academy of Japan (EAJ).

Kaoru Ota (Member, IEEE) was born in Aizuwakamatsu, Japan. She received the B.S. degree in computer science and engineering from The University of Aizu, Aizuwakamatsu, in 2006, the M.S. degree in computer science from Oklahoma State University, Stillwater, OK, USA, in 2008, and the Ph.D. degree in computer science and engineering from The University of Aizu, in 2012. From March 2010 to March 2011, she was a Visiting Scholar at the University of Waterloo, Waterloo, ON, Canada. She was also a Japan Society of the Promotion of Science (JSPS) Research Fellow at Tohoku University, Sendai, Japan, from April 2012 to April 2013. She is currently an Associate Professor and a Ministry of Education, Culture, Sports, Science and Technology (MEXT) Excellent Young Researcher with the Department of Sciences and Informatics, Muroran Institute of Technology, Muroran, Japan. She was a recipient of the IEEE TCSC Early Career Award in 2017 and The 13th IEEE ComSoc Asia-Pacific Young Researcher Award in 2018. She is Clarivate Analytics 2019 Highly Cited Researcher (Web of Science).

Lei Liu (Member, IEEE) received the B.Eng. degree in communication engineering from Zhengzhou University, Zhengzhou, China, in 2010, and the M.Sc. and Ph.D. degrees in communication engineering from Xidian University, Xi'an, China, in 2013 and 2019, respectively. From 2013 to 2015, he was with the Technology Company. From 2018 to 2019, he was a Visiting Ph.D. Student with the University of Oslo, Norway, supported by the China Scholarship Council (CSC). He is currently an Associate Professor with the School of Telecommunications Engineering, Xidian University. His research interests include vehicular ad hoc networks, intelligent transportation, mobile edge computing, and the Internet of Things.

Schahram Dustdar (Fellow, IEEE) was an Honorary Professor of information systems with the University of Groningen, The Netherlands, from 2004 to 2010, a Visiting Professor with the University of Seville, Spain, from 2016 to 2017, and a Visiting Professor with the University of California at Berkeley, Berkeley, CA, USA, in 2017. He is currently a Professor of computer science with the Distributed Systems Group, TU Vienna, Austria. He is elected as a member of the Academia Europaea, where he is the Chairperson of the Informatics Section. He was a recipient of the ACM Distinguished Scientist Award in 2009, the IBM Faculty Award in 2012, and the IEEE TCSVC Outstanding Leadership Award for out-standing leadership in services computing in 2018. He is also an Associate Editor of IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*. He serves on the editorial board for IEEE INTERNET COMPUTING and *IEEE Computer Magazine*. He is the Co-Editor-in-Chief of the *ACM Transactions on Internet of Things* and the Editor-in-Chief of *Computing* (Springer).