Personalized LLM Service Updating in Collaborative Edge-Enabled Vehicle Autonomous System

Dongkun Huo, Jiajie Yin, Hongbo Liu, Yixue Hao, Rui Wang, Long Hu, Yijun Mo

***Abstract*—Personalized services are increasingly critical for autonomous systems such as smart vehicles, where Large Language Models (LLMs) enhance the driving experience. However, efficiently deploying and continuously updating these models is challenging due to resource constraints and dynamic environments. Existing approaches often neglect personalization and effective cross-edge collaboration, leading to inconsistent service quality and inefficient resource utilization. To address these gaps, we propose a vehicle-edge-cloud collaborative framework centered on our Intent-Driven Multi-Agent Communication (IDMAC) algorithm. IDMAC intelligently coordinates the LLM update lifecycle. It utilizes an intent encoder to model the short-term goals of each edge server from its trajectory. This explicit intent then guides attention mechanisms to facilitate efficient, goal-aligned communication, reducing network overhead and focusing on relevant information. This allows for the dynamic scheduling and routing of parameter-efficient model patches. In our architecture, lightweight models are deployed on vehicles, edge servers handle specialized fine-tuning, and the cloud periodically aggregates global knowledge to create unified updates. Experimental results show that the proposed framework significantly improves service quality and adaptability. IDMAC also strengthens inter-edge collaboration while reducing latency and energy consumption.**

***Index Terms*—Edge Network, LLM, Multi-Agent Reinforcement Learning, Communication**

I. Introduction

With the rapid development of Intelligent Transportation Systems (ITS), autonomous driving, and smart cockpits, in-vehicle artificial intelligence systems are evolving from task-specific models to general-purpose large models. Large Language Models (LLMs) [1] demonstrate strong reasoning and generalization capabilities. By integrating multimodal data such as speech, vision, and navigation, these models enable natural human–machine interaction, complex scenario reasoning, and dynamic decision-making. Such advances provide new support for safe autonomous driving and significantly improve the intelligence of vehicle services [2].

Despite these advantages, deploying LLMs in vehicles remains highly challenging [3], [4]. Vehicle hardware platforms face strict limitations in computation, memory, and energy consumption, which makes it infeasible to run billion-scale or trillion-scale models directly on vehicles. Meanwhile, many in-vehicle applications, such as driving assistance, voice interaction, and risk prediction, require real-time responses and personalized adaptation. Traditional cloud-based paradigms are insufficient, as they introduce excessive latency, high bandwidth consumption, and privacy risks. Consequently, efficient deployment and personalized updating of LLMs in vehicular edge networks has become a key research problem.

Recent works have mainly explored two research directions: lightweight large models with incremental updates and cloud–edge–end collaborative optimization. Regarding the former, existing works adapt the pruning, quantization, and knowledge distillation to reduce model complexity. Parameter-Efficient Fine-Tuning (PEFT) [5] methods, such as LoRA, allow updating only small adapter modules instead of the entire model, making incremental updates feasible and significantly reducing computation and communication costs. However, PEFT-based models deployed at different edge nodes may exhibit inconsistent performance. Expert models trained with long-term local data often surpass general foundation models with LoRA adapters, which creates a trade-off between reduced latency and slight accuracy degradation.

For edge collaboration, Vehicular Edge Computing (VEC) is regarded as a promising solution to the constraints of vehicle hardware. Research in this area has focused on inference offloading, federated learning and edge caching [6], which enable data aggregation while preserving privacy, and offloading reduces inference latency. However, most of these approaches target one-time inference tasks rather than continuous lifecycle updating of LLMs. Existing Over-the-Air or caching solutions often ignore the structural properties of LLMs and rarely support user-level personalization [7]. In addition, they cannot effectively adapt to dynamic conditions such as vehicle mobility, unstable networks, and heterogeneous edge resources.

More recently, Multi-Agent Reinforcement Learning (MARL) has been introduced to optimize model updating and resource allocation in cloud–edge networks [8]. In these approaches, edge nodes are modeled as agents that make local decisions based on network states, user demands, and system efficiency. This paradigm enables distributed optimization, yet still encounters several limitations. Communication overhead between agents is high, which reduces scalability. Personalization for diverse vehicles remains insufficient, as most methods focus on global aggregation. Furthermore, inconsistencies such as model drift and version mismatch may emerge across cloud and edge nodes. These issues highlight that although progress has been achieved in lightweighting and edge collaboration, personalized updating and systematic optimization within the cloud–edge–end architecture remain unresolved.

Dongkun Huo, Jiajie Yin, Hongbo Liu, Yixue Hao, and Yijun Mo are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. Long Hu and Rui Wang are with School of Computer Science and Technology, Huazhong University of Science and Technology and also with Guangdong HUST Industrial Technology Research Institute. (e-mail: {dongkunhuo, jiajieyin, hongboliu, yixuehao, ruiwang2020, hulong, moyj}@hust.edu.cn)

Yixue Hao is the corresponding author.

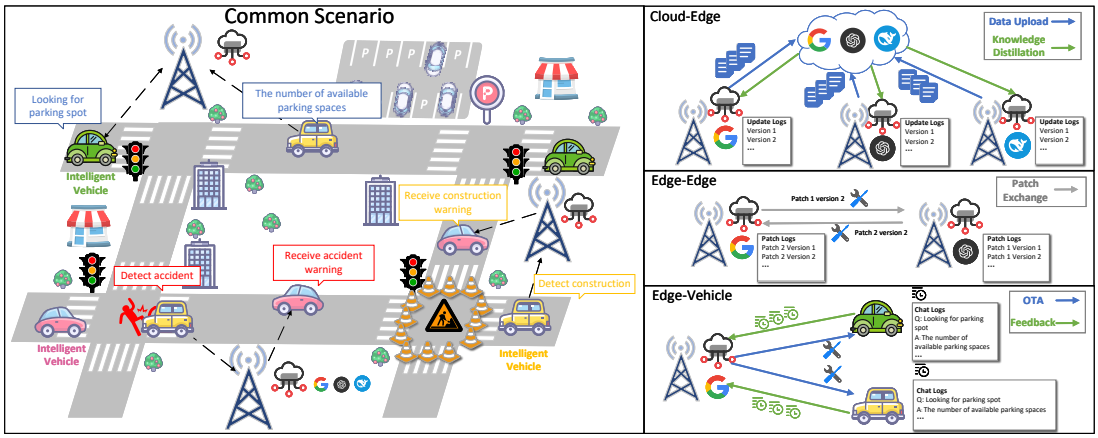

Fig. 1. LLM-based edge-enabled autonomous vehicle intelligent Systems.

Our research is motivated by the key question: how to realize personalized LLM updates in resource-constrained and dynamic vehicular edge networks, while ensuring efficient collaborative strategies across cloud, edge, and end devices? The key challenges can be summarized as follows. (i) How to design a unified vehicle-edge-cloud architecture that holistically manages the LLM service lifecycle under the constraints of a dynamic vehicular environment. (ii) How to deliver personalized, incremental model updates to individual vehicles efficiently, balancing the need for local adaptation with global consistency while minimizing resource consumption. (iii) How to enable intelligent collaboration among distributed edge servers to make coordinated decisions on resource allocation and update scheduling in real-time.

To address these problems, we propose a collaborative optimization framework for vehicular LLMs that integrates personalized updating, edge collaboration, and cloud coordination. The framework is built upon MARL and intent-based communication, which together enable distributed yet coordinated decision-making. Our main contributions are summarized as follows:

- **Cloud-Edge-Vehicle Collaborative Framework**: We propose a comprehensive framework for LLM updates in intelligent vehicular networks, integrating personalized updating and edge collaboration to address resource constraints and dynamic environments.
- **Personalized Incremental LLM Updating Scheme**: We formulate an efficient update mechanism based on PEFT and LoRA injection, coupled with task capability abstraction, to enable personalized model updates while minimizing computation and communication costs.
- **Intent-Driven Multi-Agent Communication (IDMAC) algorithm**: We propose IDMAC, which acts as both scheduling and update-control, selecting and routing PEFT adapters, triggering OTA to vehicles, and coordinating edge–cloud synchronization, thereby coupling intent-driven decisions with parameter-efficient updating.
- **Extensive Experimental Validation**: We conduct comprehensive experiments demonstrating that our proposed framework, IDMAC, significantly improves service quality, adaptability, and achieves superior convergence, reduced latency, and lower energy consumption in dynamic vehicular networks compared to existing baselines.

The remainder of this paper is organized as follows. Section II reviews related work on LLM updating and collaborative edge intelligence. Section III presents the overall system model and formulates the optimization problem for vehicle–edge–cloud collaborative updates. Section IV introduces the proposed Intent-Driven Multi-Agent Communication (IDMAC) framework. Section V provides simulation settings and performance evaluation results. Finally, Section VI concludes the paper and discusses potential directions for future work.

II. Related Work

A. System Foundations and Scheduling for LLM on Edge

Research on deploying large language models at the edge has centered on the goals of high throughput and predictable latency. Advances in runtime systems have refined memory and batching strategies such as paging key–value caches to reduce fragmentation and increase batch efficiency [9], or reducing overhead with control flow and constrained decoding [10]. On the scheduling side, novel systems address workload heterogeneity by disaggregating prefill from decoding [11], adopting chunked-prefill schedules [12], and enabling cross-instance rescheduling [13]. More recently, the demand for personalization has driven serving stacks toward a shared base model with many lightweight adapters. The unification of paging for adapter parameters [14] and cross-adapter batching [15] demonstrate that thousands of LoRA-style modules can be co-hosted efficiently. These efforts have demonstrated strong results within single clusters. Although initial attempts at cloud–edge–device collaboration are beginning to emerge, existing approaches do not yet provide a unified framework for coordinating personalization and scheduling under bandwidth and service-level constraints.

B. Personalization in the Internet of Vehicles with Semantic Intent

In vehicular networks, personalization is essential but remains challenging due to highly dynamic environments and non-IID [16] data distributions. Conventional personalized federated learning techniques address this issue by implicitly tailoring models to individual distributions [17], but they rarely provide explicit representations of user preferences or goals. To overcome this limitation, recent work has integrated language and vision–language models into the driving stack. Drivelm [18] captures the dependencies among perception, prediction, and planning for reasoning-oriented evaluation. Lingoqa [19] exposes gaps between current models and human-level performance. Field experiments demonstrate that natural-language preferences can be mapped into executable driving commands, with memory mechanisms maintaining user-specific consistency [20]. Collectively, these studies highlight the potential of semantic and intent-based approaches to enhance personalization in the internet of vehicles (IoV), though how such representations can be translated into modular updates and coordinated orchestration across edge resources is still unresolved.

C. Collaboration across Cloud, Edge, and Devices

When model update has been completed, the main challenge often shifts to how data and intermediate representations are exchanged among cloud, edge, and device nodes. VSSN-DTDE [21] emphasizes offloading and caching, where edge servers determine when to serve from cache or forward queries to the cloud, thus improving both quality of service (QoS) and cost efficiency. VELO [22] introduces a databased cloud–edge framework that caches LLM responses at the edge and uses Multi-Agent Reinforcement Learning (MARL) to decide between edge retrieval and cloud querying, improving QoS under cost and latency constraints. Other work adopts adaptive split [23], allowing weaker devices to process shallow network segments while offloading deeper layers, with reinforcement learning (RL) adjusting split points and bandwidth allocations in real time. U-SFL [24] further reduce communication overhead while safeguarding privacy. Looking further ahead, foundation-model–native perspectives argue that cross-tier collaboration and intent-aware control should be embedded in system design from the outset, especially in the context of 6G, where distributed, multimodal training and serving are expected to be intrinsic [25]. Taken together, these contributions supply valuable building blocks—caching and offloading, adaptive split strategies, and FM-native architectures. But these methods remain fragmented. A unified and lightweight intent layer that converts semantic user or task descriptions into coordinated adapter-level updates, cross-node scheduling, and globally consistent optimization is still missing.

III. System Model and Problem Formulation

In this section, we present the cloud-edge-end vehicular LLM service system and then formulate the optimization problem. The system design to support personalized and versioned LLM-based services under practical constraints of computation, communication, and energy. We first describe the system overview and then detail the service, communication, latency, and energy models. Finally, we provide the problem formulation that captures the trade-off among service quality, latency, and energy consumption.

TABLE I
Summary of Notations in System Model

Notation	Meaning
N	Set of vehicles
M	Set of edge servers (edge clouds)
K	Set of service types (demands)
$a_{n,m}(t)$	Association of vehicle n with edge m at time t (binary)
$r_{n,k}(t)$	Indicator if vehicle n requests service k at t
$v_{m,k}(t)$	Version index of service k deployed at edge m
$\theta_{k,v}$	Capability parameter of service k version v
$\varphi_{n,k}(t)$	Demand intensity of vehicle n for service k
$\bar{q}_k$	Maximum QoS score for service k
α_k	Growth rate parameter in QoS function
$q_{n,k}(t)$	Perceived QoS of vehicle n for service k at t
$S_{n,k}(t)$	User task score of vehicle n for service k at t
$d_{m,n}(t)$	Distance between vehicle n and edge m
$h_{m,n}(t)$	Channel gain between vehicle n and edge m
$B^{ul}_{m,n}(t)$	Uplink bandwidth allocated to vehicle n
$B^{dl}_{m,n}(t)$	Downlink bandwidth allocated to vehicle n
$P^{ul}_{n}(t)$	Uplink transmit power of vehicle n
$P^{dl}_{m}(t)$	Downlink transmit power of edge m
$\gamma^{ul}_{m,n}(t)$	Uplink SNR of vehicle n–edge m link
$\gamma^{dl}_{m,n}(t)$	Downlink SNR of edge m–vehicle n link
$R^{ul}_{m,n}(t)$	Achievable uplink rate
$R^{dl}_{m,n}(t)$	Achievable downlink rate
$R^{bh}_{m,m'}(t)$	Backhaul rate between edges m and m'
$S^{ul}_{n,k}(t)$	Uplink request size of vehicle n for service k
$S^{dl}_{n,k}(t)$	Downlink response size for service k
$S^{bh}_{m,m',k}(t)$	Module transfer size between edges m and m'
$C_{k,v}$	Compute cycles required by service k version v
$f_{m,n}(t)$	CPU/GPU frequency share at edge m for vehicle n
$T^{comm}_{n,m,k}(t)$	Communication latency
$T^{bh}_{m,m',k}(t)$	Backhaul latency
$T^{proc}_{m,n,k,v}(t)$	Processing latency for service k version v
$E^{ul}_{n,m,k}(t)$	Uplink transmission energy
$E^{dl}_{m,n,k}(t)$	Downlink transmission energy
$E^{bh}_{m,m',k}(t)$	Backhaul transmission energy
$E^{proc}_{m,n,k,v}(t)$	Processing energy at edge m
κ_m	Hardware energy coefficient of edge m
λ_T, λ_E	Weights for latency and energy in objective
U	System utility (QoS–latency–energy trade-off)

A. System Overview

We consider a cloud–edge–end vehicular network over discrete time slots $t = 0, 1, 2, \cdots, T$. Let $\mathcal{N}$ be the set of vehicles and $\mathcal{M}$ be the set of edge clouds. $\mathcal{K}$ denotes the set of one-to-one mapping between demand and service types, each corresponding to a user demand (e.g., traffic compliance, navigation, auto-parking, entertainment). Each vehicle can be associated with at most one edge server in a given time slot. A binary variable $a_{n,m}(t)$ captures the association of vehicle n to edge m at time slot t. Each demand type $k \in \mathcal{K}$ corresponds to a unique service offered by the vehicular or edge LLM, where services can exist in multiple versions (e.g., different LoRA modules or incremental model updates).

B. Service Model

We define the service model in the vehicle-edge-cloud architecture. The model describes how vehicle demands are met by different service versions deployed at edge servers, and how the service quality (QoS) evolves with model improvement.

1) Model Improvement: Model improvement refers to the incremental updates applied to service models, improving their capability and performance. Each service version v is associated with a capability parameter $\theta_{k,v}$ that increases with the version, leading to enhanced service quality. When a new version $v+1$ is cached, it offers better performance compared to version v, contributing to higher Quality of Service (QoS) for the vehicle.

$$\theta_{k,v+1} > \theta_{k,v}, \quad \forall v \in V_k \tag{1}$$

The improvement in service quality due to the model version update is captured by the difference between the new and old service quality values:

$$\Delta q_{n,k}(t) = q_{n,k,\text{new}}(t) - q_{n,k,\text{old}}(t) \tag{2}$$

This relationship ensures that each new model version provides an improvement in service quality. The QoS increases as the model's capability increases, but it follows a diminishing return curve. The non-linear relationship between model improvement and QoS is given by:

$$q_{n,k}(t) = \bar{q}_k \left(1 - \exp\left(-\alpha_k [\theta_{k,v} - \varphi_{n,k}(t)]^+\right)\right) \tag{3}$$

where $\theta_{k,v}$ represents the model's capability, and $\varphi_{n,k}(t)$ is the demand level for the service. This formula indicates that while model improvement leads to higher QoS, the improvement is most significant at lower version levels and saturates as the version number increases.

2) User Task Score: The user task score, $S_{n,k}(t)$, quantifies the satisfaction of the vehicle user based on the service provided. It combines the perceived service quality (QoS) with the latency and energy consumption during the task execution.

Given the QoS model, the user task score is defined as:

$$S_{n,k}(t) = w_q \cdot q_{n,k}(t) - w_t \cdot T_{n,k}(t) - w_e \cdot E_{n,k}(t) \tag{4}$$

where w_q, w_t, w_e are the weights for QoS, latency, and energy, respectively. $T_{n,k}(t)$ is the latency experienced by the vehicle user when accessing service k. $E_{n,k}(t)$ is the energy consumption for processing and communication during the service execution.

The user task score provides a balanced measure of how well the service satisfies the vehicle's requirements, considering both the improvement in the model and the associated time and energy costs.

3) QoS and Latency: The Quality of Service (QoS) is a function of the model's version and the vehicle's demand for a particular service. As shown in the previous sections, when a vehicle requests a service, the QoS depends on the model's capability and the level of demand. However, latency also plays a crucial role in determining the perceived QoS.

The latency $T_{n,k}(t)$ for a vehicle to receive a service depends on multiple factors, including the communication delay between the vehicle and the edge server, the processing time at the edge, and any backhaul delays. The end-to-end latency is given by:

$$T_{n,k}(t) = T_{\text{comm}}(t) + T_{\text{proc}}(t) + T_{\text{bh}}(t) \tag{5}$$

where $T_{\text{comm}}(t)$ is the communication latency, which depends on bandwidth and signal strength. $T_{\text{proc}}(t)$ is the processing latency at the edge server. $T_{\text{bh}}(t)$ is the backhaul latency if modules are exchanged between edge servers.

Reducing latency improves the user's task score as it directly impacts the service delivery speed, making it crucial to balance the QoS improvement with the delay constraints.

4) Model Improvement and Task Score Trade-off: While improving the model version enhances QoS, it often comes with an increased computational and communication cost, which can affect latency and energy consumption. Hence, optimizing the task score requires balancing model improvement with the associated delays and energy consumption.

In practice, a trade-off must be found between improving the model and minimizing latency and energy costs. This balance is essential for achieving the best possible user experience while maintaining efficient use of resources. The vehicle-edge-cloud architecture dynamically adjusts this trade-off based on the specific demands of the task, ensuring that users receive the highest quality service within the constraints of the system.

C. Communication Model

Vehicles communicate with edge servers via wireless links. The large-scale path loss for vehicle n to edge m is $L(d_{m,n}(t)) = G_0 d_{m,n}(t)^{-\eta}$, with distance $d_{m,n}(t)$, exponent $\eta > 2$, and constant G_0. Including fading $g_{m,n}(t)$ with $\mathbb{E}[|g_{m,n}(t)|^2] = 1$, the effective channel gain is $h_{m,n}(t) = L(d_{m,n}(t))|g_{m,n}(t)|^2$. Given uplink/downlink bandwidths $B^{\text{ul}}_{m,n}(t)$, $B^{\text{dl}}_{m,n}(t)$ and transmit powers $P^{\text{ul}}_n(t)$, $P^{\text{dl}}_m(t)$, the SNRs are:

$$\gamma^{\text{ul}}_{m,n}(t) = \frac{P^{\text{ul}}_n(t)\, h_{m,n}(t)}{N_0 B^{\text{ul}}_{m,n}(t) + I^{\text{ul}}_{m,n}(t)}, \tag{6}$$

$$\gamma^{\text{dl}}_{m,n}(t) = \frac{P^{\text{dl}}_m(t)\, h_{m,n}(t)}{N_0 B^{\text{dl}}_{m,n}(t) + I^{\text{dl}}_{m,n}(t)}, \tag{7}$$

where N_0 is the noise spectral density and $I^{\text{ul/dl}}_{m,n}(t)$ is interference. According to [26], the achievable uplink and downlink rates are:

$$R^{\text{ul}}_{m,n}(t) = B^{\text{ul}}_{m,n}(t) \log_2\big(1 + \gamma^{\text{ul}}_{m,n}(t)\big), \tag{8}$$

$$R^{\text{dl}}_{m,n}(t) = B^{\text{dl}}_{m,n}(t) \log_2\big(1 + \gamma^{\text{dl}}_{m,n}(t)\big). \tag{9}$$

Edges may also exchange modules or updates via backhaul links. The achievable rate between edges m and m' is

$$R^{\text{bh}}_{m,m'}(t) = B^{\text{bh}}_{m,m'}(t) \log_2\big(1 + \gamma^{\text{bh}}_{m,m'}(t)\big). \tag{10}$$

D. Latency Model

For a request of size $S^{\text{ul}}_{n,k}(t)$ and response of size $S^{\text{dl}}_{n,k}(t)$, the communication latency is:

$$T^{\text{comm}}_{n,m,k}(t) = \frac{S^{\text{ul}}_{n,k}(t)}{R^{\text{ul}}_{m,n}(t)} + \frac{S^{\text{dl}}_{n,k}(t)}{R^{\text{dl}}_{m,n}(t)}. \tag{11}$$

If an edge fetches a module of size $S^{\text{bh}}_{m,m',k}(t)$ from peer m', the backhaul latency is

$$T^{\text{bh}}_{m,m',k}(t) = \frac{S^{\text{bh}}_{m,m',k}(t)}{R^{\text{bh}}_{m,m'}(t)}. \tag{12}$$

For version v, let $C_{k,v}$ denote required computation cycles. Edge m allocates compute frequency $f_{m,n}(t)$ to vehicle n. The processing latency is

$$T^{\text{proc}}_{m,n,k,v}(t) = \frac{C_{k,v}}{f_{m,n}(t)}. \tag{13}$$

Thus, the end-to-end latency is the sum of communication, optional backhaul, and processing latencies. This reflects the joint influence of wireless resources and computational allocation.

E. Energy Consumption Model

Both communication and computation consume energy. The uplink and downlink transmission energies are:

$$E^{\text{ul}}_{n,m,k}(t) = P^{\text{ul}}_{n}(t)\,\frac{S^{\text{ul}}_{n,k}(t)}{R^{\text{ul}}_{m,n}(t)}, \tag{14}$$

$$E^{\text{dl}}_{m,n,k}(t) = P^{\text{dl}}_{m}(t)\,\frac{S^{\text{dl}}_{n,k}(t)}{R^{\text{dl}}_{m,n}(t)}. \tag{15}$$

The backhaul energy can be calculated as follow:

$$E^{\text{bh}}_{m,m',k}(t) = P^{\text{bh}}_{m}(t)\,\frac{S^{\text{bh}}_{m,m',k}(t)}{R^{\text{bh}}_{m,m'}(t)}. \tag{16}$$

The computation energy at edge m follows a standard CMOS model with coefficient $\kappa_m > 0$:

$$E^{\text{proc}}_{m,n,k,v}(t) = \kappa_m f_{m,n}(t) C_{k,v}. \tag{17}$$

This highlights the fundamental trade-off: higher frequencies reduce latency but increase energy consumption.

F. Problem Formulation

We formulate a joint optimization problem to maximize the overall **system utility**, which is designed to reflect the end-user experience by balancing three critical and often competing objectives. This utility is expressed as the minimization of a weighted cost function comprising: 1) **Service Quality**, by maximizing the perceived Quality of Service (QoS) to ensure accurate and relevant LLM responses; 2) **Service Responsiveness**, by minimizing end-to-end latency, which is crucial for real-time interactions; and 3) **System Efficiency**, by reducing energy consumption for sustainable operation. Let $\pi_{n,m}(t) \in [0,1]$ denote the soft association between vehicle n and edge m at time t, and $v_{m,k}(t)$ the version index of service k at edge m. The decision variables also include bandwidth allocations $B^{\text{ul}}_{m,n}(t), B^{\text{dl}}_{m,n}(t)$, frequency allocations $f_{m,n}(t)$, and transmission powers $P^{\text{ul}}_{n}(t), P^{\text{dl}}_{m}(t)$. The optimization problem is expressed as:

$$\begin{aligned}\mathcal{P}_{\mathbf{s}}: \quad & \min_{\{\pi,v,B,f,P\}} \ \alpha \sum_{t,n,k} \big(- r_{n,k}(t)\, q_{n,k}(t)\big) \\ & +\beta \sum_{t,n,k} T^{\text{e2e}}_{n,m,k}(t) \\ & +\gamma \sum_{t,n,k} \big(E^{\text{ul}}_{n,m,k}(t) + E^{\text{dl}}_{m,n,k}(t) + E^{\text{proc}}_{m,n,k,v}(t)\big)\end{aligned} \tag{18}$$

subject to:

$$C1: \ \sum_{m\in\mathcal{M}} \pi_{n,m}(t) = 1,\ 0 \le \pi_{n,m}(t) \le 1,\ \forall n,t \tag{19}$$

$$\begin{aligned}C2: \ & \sum_{n\in\mathcal{N}} \Big(B^{\text{ul}}_{m,n}(t) + B^{\text{dl}}_{m,n}(t)\Big) \le B^{\max}_{m}, \\ & \sum_{n\in\mathcal{N}} f_{m,n}(t) \le f^{\max}_{m},\ \forall m,t\end{aligned} \tag{20}$$

$$C3: \ 0 \le P^{\text{ul}}_{n}(t) \le P^{\max}_{n},\ 0 \le P^{\text{dl}}_{m}(t) \le P^{\max}_{m},\ \forall n,m,t \tag{21}$$

where α, β, and γ are nonnegative weights controlling the trade-off between service quality, latency, and energy consumption. This simplified formulation preserves the key system trade-offs while ensuring tractability for reinforcement learning-based optimization.

IV. Intent-Driven Multi-Agent Collaboration Framework

The proposed method operates in a cloud-edge-end collaborative architecture where the IDMAC framework serves as the intelligent decision-making core to orchestrate the LLM service update lifecycle. Each vehicle generates a service request $d = (k, s, I, r)$ based on mobility traces. Edge servers, acting as MARL agents, make decisions $\mathbf{a}_m(t)$ based on local observations and messages. Crucially, these decisions do not perform the model training itself but rather manage the logistics of the updating process. The agents' actions in task migration, resource allocation, and patch routing directly control *how* local data is collected for fine-tuning, *where* model updates are generated, and *when* personalized service patches are disseminated to vehicles. For instance, if a requested function is available locally, the edge processes it; otherwise, it forwards the request. For primary functions, the edge also delivers a patch to the vehicle via OTA. Interaction logs are periodically uploaded to the cloud for global fine-tuning and subsequent update distribution. The reward function integrates task score, system cost, and user capability gain, guiding the policy to learn an optimal strategy for balancing service quality with the costs of updating. The pseudo-code for this collaborative process is detailed in Algorithm 1.

A. Dec-POMDP Formulation for Cloud–Edge–Vehicle LLM Services

We cast the cooperative multi-edge scheduling, task migration, and model updating problem into a decentralized partially observable Markov decision process (Dec-POMDP). The tuple is $\langle \mathcal{I}, \mathcal{S}, \{\mathcal{O}_i\}_{i\in\mathcal{I}}, \{\mathcal{A}_i\}_{i\in\mathcal{I}}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \gamma\rangle$, where agents are

Algorithm 1 Collaborative Cloud–Edge–End Procedure for Versioned LLM Services

Input: Region mobility traces $\{\mathrm{GPS}(n,t)\}$; service set $\mathcal{K}$; edges $\mathcal{M}$; initial capability vectors $\{\mathbf{c}_m(0)\}$; request generator $p(k)$; return-flag rule $r(k,\text{patch})$; resource budgets $\{B^{\mathrm{ul/dl}}, f^{\max}, P^{\max}\}$; update periods T_{edge}, T_{cloud}.

Output: Association, resource and version decisions π; updated edge/vehicle model versions; buffered logs.

Initialize vehicle models (base adapters), edge models (primary functions), cloud full model; empty buffers.
for $t = 0, 1, \ldots, T$ **do**
  Task generation: Each vehicle n samples a request type $k \sim p(k)$ and dialogue tuple $d = (k, s, I, r)$, where $r{=}1$ if no local patch of k is installed, else 0.
  Association: Create/refresh link between vehicle n and a serving edge m (policy π_{assoc}).
  Local observation at edge m**:** $\mathcal{O}_m(t) = \{\mathbf{c}_m(t),\ \text{fixed-size } \{(\mathrm{id}, d)\}\}$.
  Inter-edge messaging: Edge m encodes message $u_m(t)$ from $\{\mathcal{O}_m, \text{history}\}$ and broadcasts/receives $\{u_{m'}(t)\}$.
  Decision: Produce joint action $\mathbf{a}_m(t) = \{e_1, \ldots, e_{|\mathcal{O}_m|}, a^{\mathrm{comp}}, a^{\mathrm{comm}}\}$ using $\{\mathcal{O}_m, \{u_{m'}\}\}$.
  Resource adaptation: Apply $a^{\mathrm{comp}}, a^{\mathrm{comm}}$ to reallocate f, B, P subject to budgets.
  Patch routing (edge→edge): If needed, send primary-function patch of type k to a peer expert edge; receive patches and update $\mathbf{c}_m(t)$ and answerable list.
  Edge–vehicle interaction:
  if $r{=}1$ **then**
    if k in edge m's answerable list **then**
      Serve on m and return result to n;
    else
      Forward to expert edge, and return result to n;
    end if
  end if
  Compute task score $q_{n,k}(t)$ and latency/energy; if k is m's primary function then OTA a patch to vehicle n.
  Append $(d, \text{result}, \text{metrics})$ to edge buffer.
  Periodic maintenance:
  if $t \bmod T_{\mathrm{edge}} = 0$ **then**
    Upload buffered Q&A to cloud (compressed/filtered); clear buffer.
  end if
  if $t \bmod T_{\mathrm{cloud}} = 0$ **then**
    Receive cloud primary-function updates and refresh local models.
  end if
  Reward: $R(t) = R^{score} - R^{cost} + R^{capability}$ (for learning/ablation).
end for

edge servers $\mathcal{I} \equiv \mathcal{M}$, time is slotted $t = 0, 1, \ldots, T$, and $\gamma \in (0, 1]$ is the discount factor.

The global state $s_t \in \mathcal{S}$ aggregates wireless, compute, traffic, and model-version information:

$$s_t = \big(\{h_{m,n}(t)\},\ \{B_m^{\mathrm{ul/dl}}(t)\},\ \{f_m^{\max}\},\ \{v_{m,k}(t)\},\ \{q_{n,k}^{\mathrm{req}}(t)\},\ \{R_{m,m'}^{\mathrm{bh}}(t)\},\ \text{queues},\ \text{cloud pkg}\big), \tag{22}$$

where $q_{n,k}^{\mathrm{req}}(t)$ denotes the request indicator or rate for vehicle n and service k, $v_{m,k}(t)$ is the installed version at edge m, $h_{m,n}(t)$ and $R_{m,m'}^{\mathrm{bh}}(t)$ are wireless and backhaul conditions, and *queues* capture pending requests or transfers.

Each agent $i \in \mathcal{I}$ receives a local observation $o_i(t) \in \mathcal{O}_i$ that includes only partial state:

$$o_i(t) = \big(\mathbf{c}_i(t),\ \mathcal{U}_i(t),\ \widehat{\mathbf{h}}_i(t),\ \widehat{\mathbf{R}}_i^{\mathrm{bh}}(t),\ \widehat{\mathbf{v}}_i(t),\ \text{local queues},\ \mathbf{m}_i(t)\big), \tag{23}$$

where $\mathbf{c}_i(t)$ is the capability vector of edge i (available bandwidth, compute headroom), $\mathcal{U}_i(t)$ is the set of attached vehicle requests with payload sizes, $\widehat{\cdot}$ are local/neighbor estimates, and $\mathbf{m}_i(t)$ is the received cooperation message under a limited message budget. The observation kernel is $\mathcal{Z}(o_t | s_t)$, possibly influenced by previous actions and message policies.

The joint action $a_t = (a_i(t))_{i \in \mathcal{I}}$ factorizes across agents with feasibility induced by resource constraints. For edge i, we define a compact action composition.

$$a_i(t) = \big(\mathbf{r}_i(t),\ \mathbf{x}_i(t),\ \mathbf{u}_i(t)\big), \tag{24}$$

where $\mathbf{r}_i(t)$ allocates radio/compute to served vehicles (e.g., $\{B_{i,n}^{\mathrm{ul/dl}}(t)\}, \{f_{i,n}(t)\}, P_i^{\mathrm{dl}}(t)$), $\mathbf{x}_i(t)$ routes tasks to peer edges or cloud (migration/forwarding choices), and $\mathbf{u}_i(t)$ triggers model operations (e.g., send/receive patches for a subset of services). The admissible set embeds hard constraints,

$$\mathcal{A}_i(o_i) = \Big\{ a_i :\ \sum_n B_{i,n}^{\mathrm{ul/dl}} \le B_i^{\mathrm{ul/dl}},\ \sum_n f_{i,n} \le f_i^{\max},\ 0 \le P_i^{\mathrm{dl}} \le P_i^{\max} \Big\}. \tag{25}$$

State transitions follow the stochastic dynamics induced by mobility, wireless fading, queue evolution, and update operations:

$$\mathcal{T}(s_{t+1} \mid s_t, a_t) = \Pr\big(s_{t+1} \text{ given } s_t \text{ and } a_t\big), \tag{26}$$

where service versions $\{v_{m,k}\}$ evolve under $\mathbf{u}_i$, queues evolve under $\mathbf{r}_i$ and $\mathbf{x}_i$, and channels/backhaul follow exogenous processes.

The team reward aggregates QoS and costs over all served (n,k) in slot t:

$$\mathcal{R}(s_t, a_t) = \sum_{(n,k) \in \mathcal{U}(t)} \Big[w_q\, q_{n,k}(t) - w_T\, T_{n,k}^{\mathrm{e2e}}(t) - w_E\, E_{n,k}(t) \Big] - w_C\, \Xi_t, \tag{27}$$

where $q_{n,k}(t)$ follows the saturating model, $T_{n,k}^{\mathrm{e2e}}(t)$ sums communication/backhaul/processing delays along the chosen route, $E_{n,k}(t)$ accounts for radio and compute energies, and Ξ_t penalizes constraint violations or excessive migration/updates. We aim to maximize the discounted return

$$J(\boldsymbol{\psi}) = \mathbb{E}_{\pi_{\boldsymbol{\psi}}}\Big[\sum_{t=0}^{T} \gamma^t\, \mathcal{R}(s_t, a_t) \Big], \tag{28}$$

with a joint policy that factorizes into decentralized actors $\pi_{\boldsymbol{\psi}}(a_t|o_t) = \prod_{i\in\mathcal{I}} \pi_{\psi_i}\big(a_i(t) \mid o_i(t)\big)$. In practice we adopt centralized training and decentralized execution (CTDE), where a centralized critic $Q_{\boldsymbol{\omega}}(s_t, a_t)$ or a monotonic mixing network estimates joint value while each edge runs π_{ψ_i} using only $o_i(t)$ and messages $\mathbf{m}_i(t)$. This Dec-POMDP formalization naturally supports cooperative MARL for joint resource scheduling, task migration, and model updating under partial observability and hard system constraints.

B. Intent-Driven Multi-Agent Communication

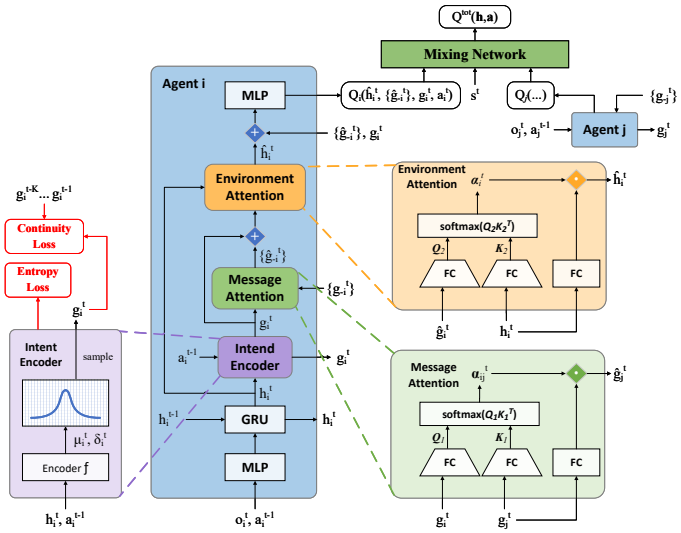

Fig. 2. Intent-driven multi-agent communication network architecture.

Multi-agent reinforcement learning (MARL) greatly benefits from communication mechanisms. However, many existing methods rely on attention structures that learn correlations from instantaneous observations, which may be insufficient for the dynamic and goal-oriented coordination required in vehicular networks. To address this, we propose IDMAC (Intent-Driven Multi-Agent Communication). While built upon the established concepts of latent variables and attention, IDMAC's novelty lies in synthesizing them to explicitly model an agent's short-term "intent". This intent, encoded from the agent's recent trajectory, provides a semantic, forward-looking representation of its objective. By conditioning both message and environment attention mechanisms on this dynamic intent variable, agents can communicate their goals proactively, rather than just their current states. This approach helps overcome the limitations of relying on static correlations in conventional attention, enabling more adaptive and efficient collaboration. The algorithmic details of IDMAC are as shown in algorithm 2.

The IDMAC framework enhances multi-agent communication by combining **short-term intent modeling** with attention mechanisms. The architecture follows the CTDE paradigm, where agents communicate through intent-driven message and environment attention.

1) Intent Encoder: Each agent encodes its trajectory $\tau_i^t = (o_i^1, a_i^1, \ldots, o_i^t)$ using a GRU to obtain h_i^t. Intent encoder maps (h_i^t, a_i^{t-1}) into Gaussian parameters (μ_i^t, δ_i^t), and a latent intent variable is sampled:

$$g_i^t \sim \mathcal{N}(\mu_i^t, (\delta_i^t)^2). \tag{29}$$

Reparameterization $g_i^t = \mu_i^t + \delta_i^t \odot \epsilon,\ \epsilon \sim \mathcal{N}(0, I)$ allows differentiability and controllable stochasticity. Intent encoder overcomes the limitation of using only instantaneous observations, encodes historical context, and provides semantic guidance for subsequent attention modules.

2) Message Attention: Intent g_i^t serves as Query, while teammates' intents $\{g_j^t\}$ act as Key/Value. The attention weights are computed as:

$$\alpha_{ij}^t = \text{softmax}\left(\frac{(W_Q g_i^t)(W_K g_j^t)^\top}{\sqrt{d_k}}\right), \tag{30}$$

$$\hat{g}_i^t = \sum_{j\neq i} \alpha_{ij}^t W_V g_j^t + g_i^t, \tag{31}$$

which selects the most relevant teammates, reduces redundant communication, and enhances efficiency by aligning messages with task goals.

3) Environment Attention: The aggregated intent $\hat{g}_i^t$ is used as Query over local hidden state h_i^t:

$$\beta_i^t = \text{softmax}\left(\frac{(W_Q \hat{g}_i^t)(W_K h_i^t)^\top}{\sqrt{d_k}}\right), \tag{32}$$

$$\hat{h}_i^t = \sum \beta_i^t W_V h_i^t, \tag{33}$$

which suppresses noisy or irrelevant local features, emphasizes goal-related observations, and improves the interpretability of environmental representation.

4) Regularization Losses: Entropy loss encourages distributional diversity:

$$L_{\text{entropy}} = -\tfrac{1}{2}\sum_{i=1}^{n}\Big((\mu_i^t)^2 + (\delta_i^t)^2 - \log((\delta_i^t)^2) - 1\Big), \tag{34}$$

while continuity loss enforces temporal smoothness:

$$L_{\text{continuity}} = -\tfrac{1}{K}\sum_{k=1}^{K}\cos(g_i^t, g_i^{t-k}), \tag{35}$$

which can guarantee discriminability, reduce fluctuations, and enhance the robustness of intent-guided communication.

Let s_t denote the mixing-network input (e.g., global state) and $Q_{\text{tot}}(s_t, \mathbf{a}_t; \theta)$ the joint action-value. With target parameters $\bar{\theta}$ and discount γ,

$$y_t = r_t + \gamma\,(1 - \text{done}_t)\ \max_{\mathbf{a}'} Q_{\text{tot}}(s_{t+1}, \mathbf{a}'; \bar{\theta}), \tag{36}$$

$$L_{\text{TD}} = \mathbb{E}_t\Big[\big(y_t - Q_{\text{tot}}(s_t, \mathbf{a}_t; \theta)\big)^2\Big]. \tag{37}$$

5) Total Objective: The overall training loss is:

$$\mathcal{L}_{\text{total}} = L_{\text{TD}} + \beta_{\text{cont}} L_{\text{Continuity}} + \beta_{\text{ent}} L_{\text{Entropy}}, \tag{38}$$

which balances task performance with intent quality, enabling semantic-rich and temporally stable intent learning.

6) Algorithmic Complexity Analysis: To evaluate the scalability of IDMAC, we analyze its computational complexity. The algorithm is executed in a decentralized manner by each of the M edge servers (agents). For a single agent at each time step, the primary computational costs are associated with the core components of the IDMAC architecture:

- **Intent Encoder:** The GRU-based encoder processes the agent's local trajectory information. Its complexity depends on the size of the hidden state, which is a fixed hyperparameter, not on the number of agents or vehicles. Thus, its complexity is $O(1)$.
- **Message Attention:** This is the most significant component in terms of inter-agent coordination. Each agent uses its intent as a query to compute attention scores against the intents of all other $M-1$ agents. This operation has a complexity of $O(M)$.
- **Environment Attention:** This mechanism operates on the agent's local observations and hidden states, which are of a fixed dimension. Therefore, its complexity is also $O(1)$.

The overall computational complexity for a single agent per decision step is dominated by the message attention mechanism, resulting in a complexity of $O(M)$. Critically, this complexity scales linearly with the number of edge servers M, not the number of vehicles N. In typical large-scale vehicular networks, $M << N$. This ensures that the decision-making overhead of IDMAC remains computationally tractable and scales effectively, even as the number of vehicles in the system grows into the thousands.

In summary, IDMAC establishes a closed-loop pipeline of "intent modeling – communication alignment – environmental filtering – regularization," addressing limitations of static attention. Each module fulfills a distinct role: the intent encoder provides semantic direction, message attention enhances communication efficiency, environment attention improves state relevance, and regularization guarantees robustness. Together, they enable adaptive and goal-aligned communication in dynamic multi-agent tasks.

Algorithm 2 IDMAC: Training and Execution Procedure

Input: Trajectories $\{\tau_i\}$, replay buffer $\mathcal{D}$, episodes E, horizon T
Input: Networks: GRU(ψ), VAE(ϕ), Attn(W_Q, W_K, W_V), $Q_i(\cdot;\theta_i)$, Mixer $Q_{\text{tot}}(\cdot;\theta_m)$
Input: Coefficients $\beta_{\text{ent}}, \beta_{\text{cont}}$, window K, discount γ, lr η
Output: Trained params $\Psi = \{\psi, \phi, W_Q, W_K, W_V, \theta_i, \theta_m\}$

1: **Init** Ψ; target params $\bar{\Psi} \leftarrow \Psi$; buffer $\mathcal{D} \leftarrow \emptyset$
2: **for** episode $= 1$ to E **do**
3: Reset env; for all agents i, reset GRU state
4: **for** $t = 1$ to T **do**
5: Observe o_i^t; form $x_i^t = (o_i^t, a_i^{t-1})$
6: $h_i^t \leftarrow \text{GRU}_\psi(h_i^{t-1}, o_i^t)$
7: $(\mu_i^t, \delta_i^t) \leftarrow \text{VAE}_\phi(h_i^t, a_i^{t-1})$
8: Sample $g_i^t = \mu_i^t + \delta_i^t \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$
9: {Message attention (standard scaled dot-product)}
10: $q_i = W_Q g_i^t$; $k_j = W_K g_j^t$; $v_j = W_V g_j^t$
11: $\alpha_{ij}^t = \text{softmax}_j\left(\frac{q_i k_j^\top}{\sqrt{d_k}}\right)$
12: $\hat{g}_i^t = \sum_{j \neq i} \alpha_{ij}^t v_j + g_i^t$
13: {Environment attention (intent-guided filtering)}
14: $\tilde{q}_i = W_Q \hat{g}_i^t$; $\tilde{k}_i = W_K h_i^t$; $\tilde{v}_i = W_V h_i^t$
15: $\beta_i^t = \text{softmax}\left(\frac{\tilde{q}_i \tilde{k}_i^\top}{\sqrt{d_k}}\right)$
16: $\hat{h}_i^t = \sum \beta_i^t \tilde{v}_i$
17: {Action selection and env step}
18: $a_i^t \sim \pi(a|\,\hat{h}_i^t)$ (e.g., ϵ-greedy over Q_i)
19: Execute $\mathbf{a}^t$, observe r_t, o^{t+1}, done
20: Store $(o^t, \mathbf{a}^t, r_t, o^{t+1}, \{g_i^t\})$ in $\mathcal{D}$
21: **if** done **then**
22: break
23: **end if**
24: **end for**
25: {Training with mini-batches from $\mathcal{D}$}
26: **for** each minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
27: Recompute $h_i^t, g_i^t, \hat{g}_i^t, \hat{h}_i^t$ on $\mathcal{B}$
28: $Q_i^t \leftarrow Q_i(\hat{h}_i^t; \theta_i)$
29: $Q_{\text{tot}}^t \leftarrow Q_{\text{tot}}(s_t, \mathbf{a}^t; \theta_m)$
30: $y_t = r_t + \gamma(1 - \text{done}_t) \max_{\mathbf{a}'} Q_{\text{tot}}(s_{t+1}, \mathbf{a}'; \bar{\theta})$
31: $L_{\text{TD}} = \mathbb{E}_{\mathcal{B}}[(y_t - Q_{\text{tot}}^t)^2]$
32: $L_{\text{Entropy}} = -\frac{1}{2}\sum_d [(\mu_{i,d}^t)^2 + (\delta_{i,d}^t)^2 - \log((\delta_{i,d}^t)^2) - 1]$
33: $L_{\text{Cont}} = -\frac{1}{K}\sum_{k=1}^{K} \cos(g_i^t, g_i^{t-k})$
34: $L_{\text{total}} = L_{\text{TD}} + \beta_{\text{ent}} L_{\text{Entropy}} + \beta_{\text{cont}} L_{\text{Cont}}$
35: $\Psi \leftarrow \Psi - \eta \nabla_\Psi L_{\text{total}}$
36: $\bar{\Psi} \leftarrow \tau\Psi + (1-\tau)\bar{\Psi}$ {target update}
37: **end for**
38: **end for**
39: **Execution (deployed):** freeze Ψ; run lines 5–18; no updates

V. Performance Evaluation

A. Experiment Setting

The simulation environment is based on real taxi trajectories from Xicheng District, Beijing. We set N_v={50, 100, 200} vehicles and $N_e = \{6, 12\}$ edge servers, each with a coverage radius of 800m. Vehicle–edge associations evolve dynamically along trajectories, capturing realistic spatio-temporal traffic patterns. Each episode lasts $T = 60$ steps, corresponding to 60 minutes. Task categories are $K = 6$, with prior distribution from LaMPilot-Bench [27]:{0.2455, 0.2455, 0.0415, 0.3025, 0.0825, 0.0825}. Task size $s \sim \ln\mathcal{N}(\mu = \ln(0.1), \sigma = 1)$ ranges from KB to tens of MB. Task importance $I \sim \text{Beta}(0.5, 2.0)$, mostly low-to-medium importance. If a vehicle has zero capability on task k, a request $r = 1$ is triggered. Capability upper bound is $C_{max} = 1.0$: vehicles start at 0, edge primary functions at 0.6, and cloud at 0.6. The cloud fine-tunes every 20 steps and updates via OTA: $c_k^{cloud} = c_k^{cloud} + \alpha \cdot \left(\sum_{d \in \mathcal{D}_k} I_d\right) \cdot \left(1 - \frac{c_k^{cloud}}{C_{max}}\right)$, $\alpha = 0.01$. Each edge server has $C_{\text{edge}} = 100\,\text{GHz}$. Task demand is 20 Gigacycles/(MB·Importance). The uplink bandwidth is 1000 Mbps; the fiber bandwidth is 10,000 Mbps. Power settings: $P_c = 150$W (compute), $P_t = 50$W (uplink), $P_f = 20$W (fiber). Wireless links follow Shannon's formula with $N_0 = 4 \times 10^{-21}$ W/Hz and path loss exponent $\gamma = 3.0$. The reward includes task score S_t, system cost C_t, model gain M_t, and

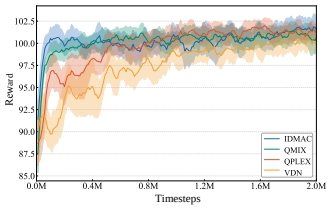

Fig. 3. The return in 50 vehicles and 6 edge servers scenario.

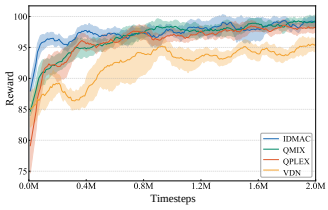

Fig. 4. The return in 100 vehicles and 6 edge servers scenario.

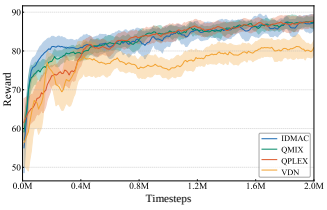

Fig. 5. The return in 200 vehicles and 6 edge servers scenario.

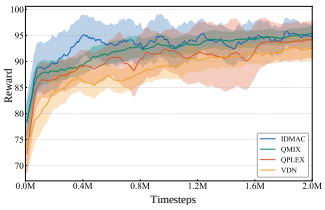

Fig. 6. The return in 200 vehicles and 12 edge servers scenario.

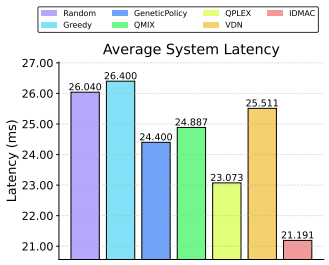

Fig. 7. Average latency in 50 vehicles and 6 edge servers scenario.

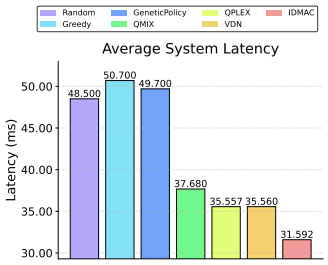

Fig. 8. Average latency in 100 vehicles and 6 edge servers scenario.

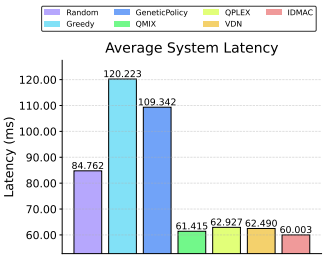

Fig. 9. Average latency in 200 vehicles and 6 edge servers scenario.

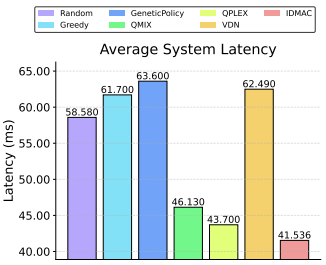

Fig. 10. Average latency in 200 vehicles and 12 edge servers scenario.

latency penalty L_t. Task score is normalized by Sigmoid ($\theta = 0.6$, $\tau = 0.1$). Latency threshold is set to 1s to ensure real-time responsiveness.

B. Comparison Algorithm

Six benchmark algorithms are used to evaluate the proposed IDMAC algorithms: 1) **Random:** Randomly generates the task offloading and resource allocation strategy from $[0, 1]$ [28]; 2) **Greedy:** Always chooses the action with the highest immediate reward, reflecting greedy decision-making [29]. 3) **Genetic:** Explores solutions using evolutionary operators such as selection, crossover, and mutation, widely applied to combinatorial optimization [30]. 4) **VDN:** Decomposes the global Q-value into a sum of individual Q-values, providing a simple and effective CTDE method [31]. 5) **QMIX:** Employs a mixing network to factorize individual Q-values monotonically into a joint Q-value, enabling centralized training with decentralized execution [32]. 6) **QPLEX:** Extends QMIX with duplex advantage decomposition, allowing more flexible joint value modeling [33]. These algorithms are chosen as they represent foundational and effective CTDE methods, providing a solid basis for evaluating the specific performance gains attributable to our proposed IDMAC communication mechanism. While other important research areas like vehicular-edge LLM updating frameworks and federated adaptation address related challenges, our focus here is on the decision-making and coordination aspect within a multi-agent system, for which VDN, QMIX, and QPLEX serve as highly relevant benchmarks.

C. Experiment Performance

1) Convergence Performance: The experimental results demonstrate that IDMAC achieves superior convergence and performance across different vehicular scales. As shown in Fig. 3, the return curve of IDMAC rises rapidly and reaches a stable level with fewer training episodes, surpassing all baseline methods. As the network scale increases from 50 to 200 vehicles, as presented in Fig. 4 to Fig. 5, IDMAC consistently exhibits smooth and stable convergence, while traditional methods such as QMIX, QPLEX, and VDN tend to converge slowly or display noticeable oscillations. These observations demonstrate that IDMAC effectively coordinates the interactions between edge servers and vehicles, maintaining efficient learning dynamics even in mid-scale vehicular systems. To further evaluate scalability, we extend the experiment to a larger configuration involving 12 edge servers and 200 vehicles, as shown in Fig. 6. The results reveal that IDMAC continues to achieve stable convergence and superior final returns, confirming that the proposed algorithm not only performs robustly under moderate workloads but also scales gracefully to more complex and large-scale vehicular networks. This consistent trend across all experimental settings highlights the scalability, robustness, and adaptability of IDMAC in distributed multi-agent coordination scenarios.

The performance superiority of IDMAC stems from its intent-driven modeling and attention mechanisms. The intent encoder abstracts vehicles' personalized demands, while message attention facilitates efficient cross-edge information exchange by suppressing redundant communication. Environment attention enables edge servers to filter irrelevant observations and concentrate on task-related features. Furthermore, entropy and continuity regularizations ensure diversity and stability of intent representations, which helps the system maintain consistency and efficiency even under larger scales and dynamic conditions. Overall, IDMAC consistently outperforms baselines in terms of convergence speed, stability, and final performance.

2) Network Performance Analysis: The experimental results demonstrate the advantages of IDMAC from two perspec-

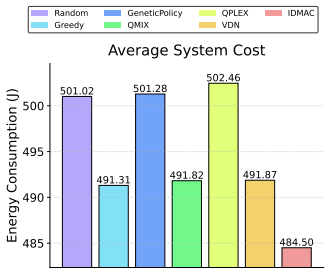

Fig. 11. Average cost in 50 vehicles and 6 edge servers scenario.

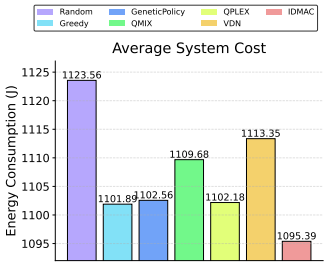

Fig. 12. Average cost in 100 vehicles and 6 edge servers scenario.

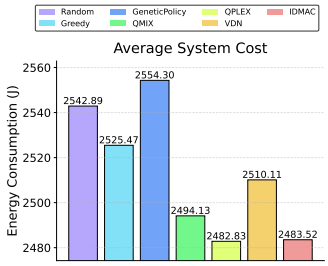

Fig. 13. Average cost in 200 vehicles and 6 edge servers scenario.

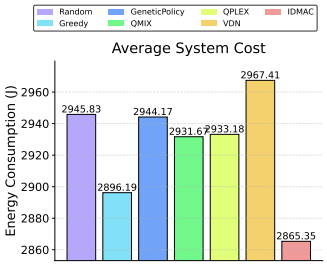

Fig. 14. Average cost in 200 vehicles and 12 edge servers scenario.

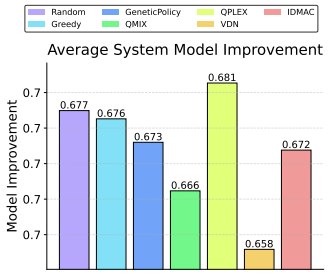

Fig. 15. Average model improvement in 50 vehicles and 6 edge servers scenario.

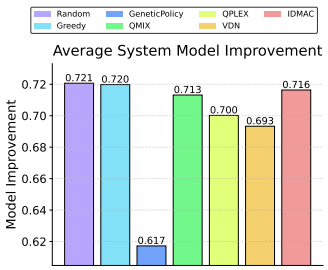

Fig. 16. Average model improvement in 100 vehicles and 6 edge servers scenario.

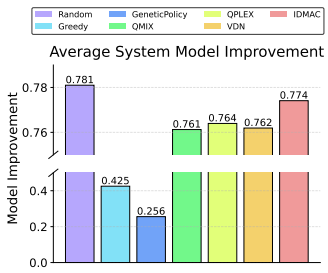

Fig. 17. Average model improvement in 200 vehicles and 6 edge servers scenario.

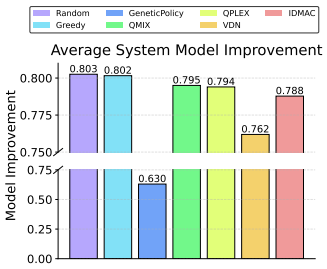

Fig. 18. Average model improvement in 200 vehicles and 12 edge servers scenario.

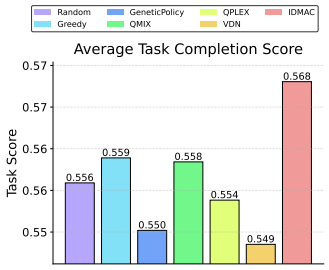

Fig. 19. Average service score in 50 vehicles and 6 edge servers scenario.

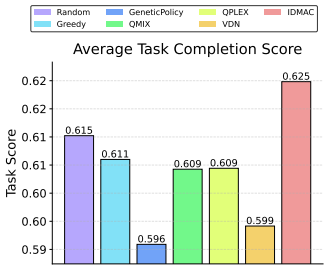

Fig. 20. Average service score in 100 vehicles and 6 edge servers scenario.

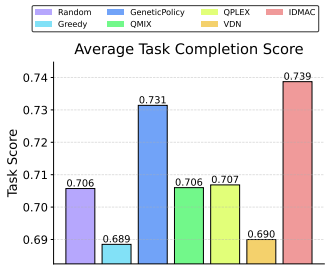

Fig. 21. Average service score in 200 vehicles and 6 edge servers scenario.

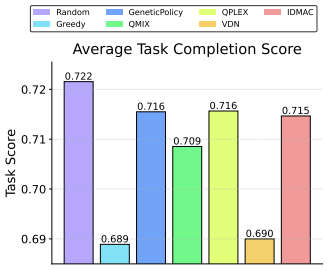

Fig. 22. Average service score in 200 vehicles and 12 edge servers scenario.

tives: network performance metrics and user service quality. In terms of network performance, IDMAC achieves consistently lower latency and reduced energy consumption compared with existing methods. In terms of service quality, it provides significantly higher user scores, especially in larger-scale vehicular environments.

For average latency, Fig. 7, Fig. 8, Fig. 9 and Fig. 10 illustrate the results across different scales. With 50 vehicles, IDMAC reduces latency by approximately 10%–15% compared with decomposition-based methods (QMIX, QPLEX, VDN) and by more than 25% compared with heuristic methods (Random, Greedy). This indicates that even in small-scale systems, IDMAC effectively mitigates congestion. At 100 vehicles, the overall network burden increases, leading to a sharp latency rise for baselines, whereas IDMAC maintains lower levels with a 15%–20% advantage. In the 200-vehicle scenario, IDMAC consistently achieves the best performance, converging stably and yielding around 2% lower latency than the best baseline with 6 edge servers and more than 5% lower latency with 12 edge servers, both nearly 40% lower than heuristic methods. These results reveal that IDMAC's scheduling and optimization mechanisms ensure efficient response under more complex and large-scale vehicular networks.

For average energy consumption, Fig. 11, Fig. 12, Fig. 13 and Fig. 14 show that IDMAC consistently outperforms others. With 50 vehicles, it reduces energy by about 5% compared with mainstream baselines. The advantage grows to 8% at 100 vehicles and nearly 10% at 200 vehicles with 6 edge servers, while in the 200-vehicle scenario with 12 edge servers, the advantage is close to 5%. Unlike traditional methods, whose energy costs grow rapidly with scale, IDMAC remains energy-efficient. The advantage comes from intent-driven communication, which reduces redundant exchanges, and cross-edge collaboration, which avoids repeated model transfers and redundant computations.

User service quality: task completion score. User service quality is measured using model improvement and task completion score.

For model improvement, Fig. 15, Fig. 16 Fig. 17 and Fig. 18 show that IDMAC consistently surpasses baselines.

With 50 vehicles, improvements are about 3%–5% compared to decomposition-based methods, increasing to 6% at 100 vehicles. In the 200-vehicle scenario, IDMAC maintains the top-performing status, achieving 4%–6% advantage with 6 edge servers and remaining the best model even with 12 edge servers. Traditional approaches suffer from resource constraints and model drift, while IDMAC leverages task capability abstraction and parameter-efficient fine-tuning to achieve stable lifecycle updates.

For task completion score, Fig. 19, Fig. 20, Fig. 21, and Fig. 22 illustrate the comparison. With 50 vehicles, IDMAC scores around 3% higher than mainstream baselines and 5% higher than heuristics. At 100 vehicles, the margin grows to 5%, and at 200 vehicles, IDMAC achieves 6%–7% higher scores with 6 edge servers, while remaining among the top-performing models when extended to 12 edge servers. This confirms that IDMAC effectively captures personalized demands and aligns them with model capabilities through intent modeling and attention mechanisms, ensuring stable service quality under dynamic and large-scale conditions.

The observed improvements can be attributed to the design principles of IDMAC. The intent encoder abstracts vehicle task demands into compact representations, capturing personalization while avoiding redundant state transmission. The message attention mechanism selectively aggregates essential cross-edge information, reducing communication overhead and enabling efficient allocation of model patches and task capabilities. The environment attention module allows edge servers to focus on task-relevant features, achieving efficient decision-making under limited computational resources. Finally, entropy and continuity regularizations ensure diversity and temporal smoothness of intent representations, which help the system maintain stability and efficiency under large-scale and dynamic conditions.

Overall, IDMAC achieves continuous advantages in both network performance and service quality. Its reduced latency and energy consumption ensure real-time responsiveness and energy efficiency in vehicular networks, while its higher user scores reflect superior support for personalization and global coordination. These results validate the effectiveness and superiority of IDMAC as a vehicle–edge–cloud collaborative optimization framework in complex environments.

3) Ablation Study: To further examine and validate the effectiveness of our algorithmic design, we conduct ablation experiments under the 200 vehicles and 6 edge servers scenario. Each curve in Fig. 23 represents the averaged results over multiple random seeds to eliminate randomness and ensure statistical reliability. Overall, the experimental results clearly demonstrate that each component of our proposed algorithm contributes effectively to performance improvement. Moreover, we observe that IDMAC-w/o-CL achieves better final convergence performance than IDMAC-w/o-EL, which may indicate that, in this experimental setting, the diversity of intentions plays a more critical role than their short-term continuity.

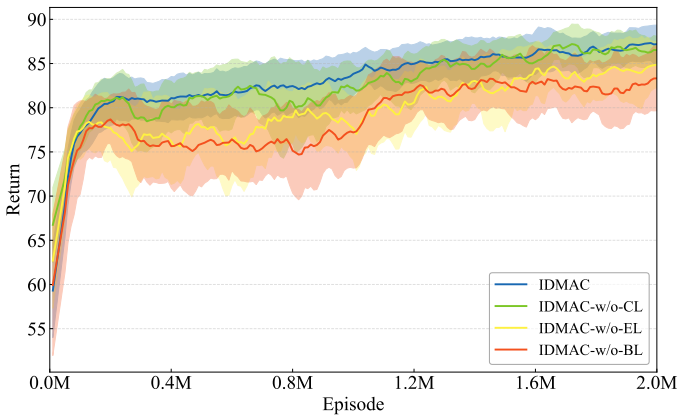

Fig. 23. Ablation studies of proposed algorithm in 200 vehicles and 6 edge servers scenario. CL denotes the intention continuity loss $L_{\text{continuity}}$; EL denotes the intention entropy loss L_{entropy}; and BL denotes the combination of both.

VI. Conclusion

This paper addresses the challenges of deploying and updating vehicular large language models by proposing a vehicle-edge-cloud collaborative framework featuring the Intent-Driven Multi-Agent Communication (IDMAC) mechanism. Our approach uses parameter-efficient fine-tuning and intent-driven scheduling to balance personalization with global consistency. Experimental results demonstrate that IDMAC exhibits strong convergence, reducing average latency and energy consumption by approximately 10%-20% while improving user service quality by 4%-7% over existing methods. Beyond these performance metrics, the framework is designed for practical deployment. The edge-centric architecture inherently enhances user privacy by localizing data processing. Hardware feasibility is achieved by deploying lightweight adapters on vehicles and offloading intensive computations to edge and cloud servers, while standard security protocols are assumed for all communications. These findings validate the effectiveness, superiority, and real-world viability of our proposed framework in complex and dynamic vehicular environments.

VII. Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant 2025ZD1302300, in part by Wuhan Natural Science Foundation Exploratory Program (Chenguang Program) under Grant 2024040801020212, in part by Guangdong Basic and Applied Basic Research Foundation Grant 2024A1515011153, Grant 2024A1515110155, in part by Hubei intelligent edge computing research institute, Hubei science and technology talent service project (2024DJC078), and in part by Postdoctoral Fellowship Program of the China Postdoctoral Science Foundation (CPSF) under Grant GZB20240244, Grant 2024M761016.

References

[1] Y. Ge, W. Hua, K. Mei, J. Tan, S. Xu, Z. Li, Y. Zhang *et al.*, "OpenAGI: When LLM meets domain experts," *Advances in Neural Information Processing Systems*, vol. 36, pp. 5539–5568, 2023.

[2] R. Chen, W. Song, W. Zu, Z. Dong, Z. Guo, F. Sun, Z. Tian, and J. Wang, "An LLM-driven framework for multiple-vehicle dispatching and navigation in smart city landscapes," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2147–2153.

[3] Y. Hu, F. Wang, D. Ye, M. Wu, J. Kang, and R. Yu, "LLM-based misbehavior detection architecture for enhanced traffic safety in connected autonomous vehicles," *IEEE Transactions on Vehicular Technology*, 2025.

[4] X. Chen, X. Lu, Q. Li, D. Li, and F. Zhu, "Integration of LLM and Human-AI coordination for power dispatching with connected electric vehicles under sagvns," *IEEE Transactions on Vehicular Technology*, 2024.

[5] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen *et al.*, "Parameter-efficient fine-tuning of large-scale pre-trained language models," *Nature machine intelligence*, vol. 5, no. 3, pp. 220–235, 2023.

[6] M. K. Hasan, N. Jahan, M. Z. A. Nazri, S. Islam, M. A. Khan, A. I. Alzahrani, N. Alalwan, and Y. Nam, "Federated learning for computational offloading and resource management of vehicular edge computing in 6g-v2x network," *IEEE Transactions on Consumer Electronics*, vol. 70, no. 1, pp. 3827–3847, 2024.

[7] Z. Wang, Y. Zhao, Y. Zhou, Y. Shi, C. Jiang, and K. B. Letaief, "Over-the-air computation for 6g: Foundations, technologies, and applications," *IEEE Internet of Things Journal*, vol. 11, no. 14, pp. 24 634–24 658, 2024.

[8] N. Yang, S. Chen, H. Zhang, and R. Berry, "Beyond the edge: An advanced exploration of reinforcement learning for mobile edge computing, its applications, and future research trajectories," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 1, pp. 546–594, 2024.

[9] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 611–626.

[10] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, C. Barrett, and Y. Sheng, "Sglang: Efficient execution of structured language model programs," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 62 557–62 583.

[11] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. Santa Clara, CA: USENIX Association, Jul. 2024, pp. 193–210.

[12] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. S. Gulavani, A. Tumanov, and R. Ramjee, "Taming throughput-latency tradeoff in llm inference with sarathi-serve," in *Proceedings of the 18th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'24. USA: USENIX Association, 2024.

[13] B. Sun, Z. Huang, H. Zhao, W. Xiao, X. Zhang, Y. Li, and W. Lin, "Llumnix: Dynamic scheduling for large language model serving," in *18th USENIX symposium on operating systems design and implementation (OSDI 24)*, 2024, pp. 173–191.

[14] Y. Sheng, S. Cao, D. Li, C. Hooper, N. Lee, S. Yang, C. Chou, B. Zhu, L. Zheng, K. Keutzer *et al.*, "S-LoRA: Serving thousands of concurrent lora adapters," *arXiv preprint arXiv:2311.03285*, 2023.

[15] L. Chen, Z. Ye, Y. Wu, D. Zhuo, L. Ceze, and A. Krishnamurthy, "Punica: Multi-tenant lora serving," in *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. D. Sa, Eds., vol. 6, 2024, pp. 1–13.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[17] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9587–9603, 2023.

[18] C. Sima, K. Renz, K. Chitta, L. Chen, H. Zhang, C. Xie, J. Beißwenger, P. Luo, A. Geiger, and H. Li, "DriveLM: Driving with graph visual question answering," in *European conference on computer vision*. Springer, 2024, pp. 256–274.

[19] A.-M. Marcu, L. Chen, J. Hünermann, A. Karnsund, B. Hanotte, P. Chidananda, S. Nair, V. Badrinarayanan, A. Kendall, J. Shotton *et al.*, "LingoQA: Visual question answering for autonomous driving," in *European Conference on Computer Vision*. Springer, 2024, pp. 252–269.

[20] C. Cui, Z. Yang, Y. Zhou, Y. Ma, J. Lu, L. Li, Y. Chen, J. Panchal, and Z. Wang, "Personalized autonomous driving with large language models: Field experiments," in *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2024, pp. 20–27.

[21] S. Zhang, Q. Gao, and H. Wu, "Cooperative edge caching with multi-agent reinforcement learning using consensus updates," in *2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 7, 2024, pp. 1804–1809.

[22] Z. Yao, Z. Tang, W. Yang, and W. Jia, "Enhancing llm qos through cloud-edge collaboration: A diffusion-based multi-agent reinforcement learning approach," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1412–1427, 2025.

[23] M. Zhang, J. Cao, Y. Sahni, X. Chen, and S. Jiang, "Resource-efficient parallel split learning in heterogeneous edge computing," *arXiv preprint arXiv:2403.15815*, 2024.

[24] L. Yu, Z. Chang, Y. Jia, and G. Min, "Model partition and resource allocation for split learning in vehicular edge networks," *IEEE Transactions on Intelligent Transportation Systems*, 2025.

[25] X. Chen, Z. Guo, X. Wang, H. H. Yang, C. Feng, J. Su, S. Zheng, and T. Q. Quek, "Foundation model based native ai framework in 6g with cloud-edge-end collaboration," *arXiv preprint arXiv:2310.17471*, 2023.

[26] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[27] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-Drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99–108.

[28] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[29] D. Bertsekas, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

[30] A. Lambora, K. Gupta, and K. Chopra, "Genetic algorithm-a literature review," in *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*. IEEE, 2019, pp. 380–384.

[31] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[32] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[33] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "QPLEX: Duplex dueling multi-agent q-learning," *arXiv preprint arXiv:2008.01062*, 2020.

BIOGRAPHIES

Dongkun Huo (dongkunhuo@hust.edu.cn) is currently a Ph.D. candidate at Embedded and Pervasive Computing (EPIC) Laboratory in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China. He received a B.S. degree in computer science and technology from School of Computer and Information Engineering at Henan University in 2020, China. He received his M.S. degree in computer technology from School of Computer and Information Technology at Beijing Jiaotong University in 2022, China. His research interests include MARL, MANET, and edge computing.

Jiajie Yin (jiajieyin@hust.edu.cn) received his B.Sc. degree in Data Science from Beijing Normal University, China, in 2025. He is currently pursuing the M.Sc. degree in Computer Science at the School of Computer Science and Technology, Huazhong University of Science and Technology, China. His research interests include reinforcement learning, deep learning, and multi-agent systems.

Hongbo Liu (hongboliu@hust.edu.cn) received the M.S. degrees in Control Science and Technology from Northeastern University, China. He is currently a Ph.D. candidate with the Embedded and Pervasive Computing (EPC) Laboratory, School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China. His research interests include reinforcement learning and multi-agent systems, with an emphasis on data-efficient decision making and representation learning for sequential prediction.

Yixue Hao (yixuehao@hust.edu.cn) is an associate professor in the School of Computer Science and Technology at Huazhong University of Science and Technology. He received the Ph.D degree in computer science from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017. His current research interests include 5G network, internet of things, edge computing, edge caching and cognitive computing.

Rui Wang (ruiwang2020@hust.edu.cn) is currently a post-doctoral scholar in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China. She received her bachelor degree in Computer Science and Technology from the Lanzhou University, Lanzhou, China, 2018, and the Ph.D. degree in Computer Science from the Huazhong University of Science and Technology, Wuhan, China, 2024. Her research interests include deep learning, multimodal learning, and emotion recognition.

Long Hu (hulong@hust.edu.cn) is an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), China. He was a visiting student in the Department of Electrical and Computer Engineering, University of British Columbia from August 2015 to April 2017. His research includes the Internet of Things, software defined networking, caching, 5G, body area networks, body sensor networks, and mobile cloud computing.

Yijun Mo (moyj@hust.edu.cn) received the BE, ME, and PhD degrees in communication and information systems from the Huazhong University of Science and Technology, China. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include edge computing, intelligent networks, and artificial intelligence.