Mobility-Aware Multi-Task Decentralized Federated Learning for Vehicular Networks: Modeling, Analysis, and Optimization

Dongyu Chen, Tao Deng, He Huang, *Senior Member, IEEE*, Juncheng Jia, Mianxiong Dong, Di Yuan, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Federated learning (FL) is a promising paradigm that can enable collaborative model training between vehicles while protecting data privacy, thereby significantly improving the performance of intelligent transportation systems (ITSs). In vehicular networks, due to mobility, resource constraints, and the concurrent execution of multiple training tasks, how to allocate limited resources effectively to achieve optimal model training of multiple tasks is an extremely challenging issue. In this paper, we propose a mobility-aware multi-task decentralized federated learning (MMFL) framework for vehicular networks. By this framework, we address task scheduling, subcarrier allocation, and leader selection, as a joint optimization problem, termed TSLP. For the case with a single FL task, we derive the convergence bound of model training. For general cases, we first model TSLP as a resource allocation game, and prove the existence of a Nash equilibrium (NE). Then, based on this proof, we reformulate the game as a decentralized partially observable Markov decision process (DEC-POMDP), and develop an algorithm based on heterogeneous-agent proximal policy optimization (HAPPO) to solve DEC-POMDP. Finally, numerical results are used to demonstrate the effectiveness of the proposed algorithm.**

***Index Terms*—Multi-task federated learning, mobility-aware, heterogeneous-agent proximal policy optimization, vehicular networks.**

Received 16 June 2025; revised 21 August 2025; accepted 4 September 2025. Date of publication 8 September 2025; date of current version 9 January 2026. The work of Dongyu Chen and Tao Deng was supported by Natural Science Foundation of Jiangsu Province, China under Grant BK20230477. The work of He Huang was supported by NSFC under Grant 62332013. The work of Juncheng Jia was supported by Collaborative Innovation Center of Novel Software Technology and Industrialization, Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Suzhou Frontier Science and Technology Program under Grant SYG202310. The work of Mianxiong Dong was supported by JSPS KAKENHI under Grant JP24K14910, Grant JP25K00139. The work of Di Yuan was supported by the Swedish Research Council with under Grant 2022-04123. Recommended for acceptance by V. Q. Pham. *(Corresponding author: Tao Deng.)*

Dongyu Chen, He Huang, and Juncheng Jia are with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China (e-mail: dychan2000@gmail.com; huangh@suda.edu.cn; jiajuncheng@suda.edu.cn).

Tao Deng is with the School of Computer Science and Technology, Soochow University, Suzhou 215006, China, and also with the Key Laboratory of Photonic-Electronic Integration and Communication-Sensing Convergence, Southwest Jiaotong University, Ministry of Education, Chengdu 610031, China (e-mail: dengtao@suda.edu.cn).

Mianxiong Dong is with the Department of Sciences and Informatics, Muroran Institute of Technology, Muroran 050-8585, Japan (e-mail: mx.dong@csse.muroran-it.ac.jp).

Di Yuan is with the Department of Information Technology, Uppsala University, 751 05 Uppsala, Sweden (e-mail: di.yuan@it.uu.se).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TMC.2025.3607496

I. Introduction

With the rapid advancement of intelligent vehicles (IVs), the intelligent transportation system (ITS) significantly enhances travel experience and safety [1]. Deep learning technologies have enabled various neural network models to offer driving assistance features such as traffic flow prediction (TFP), free-space detection (FSD), and driving behavior monitoring (DBM), thus enhancing the overall driving experience [2]. Traditional deep learning methods, which rely on centralized data aggregation and training, face challenges related to communication pressure and data privacy issues. To address these issues, federated learning (FL) has been introduced for ITS [3]. FL involves multiple vehicles and infrastructures, where each participating vehicle has its own local dataset for training. The aggregator, vehicle or infrastructure, collects local model updates from the vehicles. This approach shows great potential in improving both the efficiency and privacy of deep learning in vehicular networks.

In vehicular networks, the proliferation of intelligent transportation services has given rise to an unprecedented demand for model training, and these models exhibit diversity. The diversity requirements for FL model training primarily stem from two dimensions, i.e., inter-task and intra-task diversities. For the inter-task diversity, vehicles execute multiple tasks concurrently (e.g., TFP, FSD and DBM), each necessitating task-specific architectural configurations and training objectives [4]. For the intra-task diversity, even within a single task category (e.g., path planning), spatial-temporal context variations result in diverse model requirements [5]. For instance, urban canyon environments with dynamic occlusions encounter distinct signal propagation patterns compared to highways, necessitating separate model instantiations to handle location-specific environmental dynamics and mobility constraints. Implementing multi-task FL presents a substantial challenge in vehicular networks. First, vehicle mobility causes temporal instability as nodes frequently join or depart, disrupting consistent model aggregation. Second, real-time latency constraints in safety-critical applications demand communication-efficient FL protocols. Finally, limited edge resources demand minimized training iterations and

resource overhead. Therefore, addressing the multi-task FL challenge in vehicular networks is of urgent technical importance.

We explore vehicle mobility in vehicular networks to develop optimization strategies for multi-task FL while efficiently utilizing scarce resources. Our goal is to minimize the model training loss of each task. The main contributions of this paper are as follows.

1) We design a mobility-aware multi-task decentralized federated learning (MMFL) framework for vehicular networks. The MMFL framework adopts a multi-leader-multi-follower approach for multi-task scenarios. Vehicles that are close to each other communicate directly using vehicle-to-vehicle (V2V) communication, while vehicles that are farther apart communicate indirectly using vehicle-to-infrastructure (V2I) communication.
2) To enhance the training efficiency of MMFL, we formulate task scheduling, subcarrier allocation, and leader selection as a joint optimization problem (TSLP), which takes into account vehicle mobility, resource limitations, and latency constraints.
3) For problem solving, we analyze the convergence bound of model training for the single-task scenario under the MMFL framework. For general scenarios, we first model TSLP as a resource allocation game among multiple tasks. We prove the existence of a Nash equilibrium (NE) for the game. Subsequently, we reformulate the game as a decentralized partially observable Markov decision process (DEC-POMDP). Finally, we design an algorithm based on Heterogeneous-Agent Proximal Policy Optimization (HAPPO) to solve DEC-POMDP.
4) We design a series of experiments to validate the effectiveness of the proposed algorithm. Specifically, we use both the urban mobility simulation tool SUMO and the real-world vehicle trajectory dataset next generation simulation (NGSIM) to generate traffic flows and simulate multi-task FL on four benchmark datasets: MNIST, FashionMNIST, SVHN, and CIFAR-10. We evaluate the proposed algorithm against several baseline methods under diverse conditions, including varying initial vehicle energy levels, the number of vehicles, the number of tasks, and indirect transmission costs.

II. Related Work

In vehicular networks, FL can be categorized into two types: centralized FL (CFL) and decentralized FL (DFL). CFL relies on a central server for aggregating client models, requiring extra physical nodes, e.g., road side units (RSUs), as aggregation centers [6], [7], [8], [9], [10], [11], [12]. The works in [6], [7], [8] focus on resource allocation optimization during the vehicles' sojourn period by optimizing round duration, client selection, and compression ratio, thereby accelerating the convergence speed. DFL dynamically assigns model aggregation tasks to clients, eliminating the need for a fixed central server and enabling broader training through vehicular ad-hoc networks (VANETs) without reliance on fixed-location RSUs. This dynamic selection of aggregation nodes also mitigates the risk of single point of failure. DFL is divided into fully DFL and leader-follower DFL. In fully DFL, all clients act as leaders to perform aggregation tasks, and there is no clear hierarchical structure among the clients [13], [14], [15], [16]. In contrast, leader-follower DFL maintains a hierarchical model where the aggregation nodes centrally handle the aggregation tasks [17], [18]. As vehicular networks operate in highly dynamic and adversarial environments, the security and privacy challenges in FL are of great concern. Potential security vulnerabilities such as model poisoning, inference attacks, and adversarial robustness need to be addressed to ensure the reliability and trustworthiness of the FL process. The works in [19], [20], [21], [22] investigate these aspects. For instance, the work in [20] proposes a distributed edge-based architecture for secure and privacy-preserving collaboration among multiple edge nodes, along with a decentralized and secure blockchain-based reputation system to ensure the reliability and trustworthiness of the FL process in vehicular networks.

The works in [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22] mainly focus on FL for a single task. However, in vehicular networks, it is common to encounter scenarios where multiple tasks need to be solved. Recently, the works in [23], [24], [25], [26], [27], [28], [29], [30] conduct research on multi-task learning. The works in [23], [24], [25] investigate using a single model to handle multiple tasks. The work in [23] uses soft parameter sharing to achieve optimal recognition accuracy while considering resource efficiency. The work in [24] employs neural architecture search for projecting task gradients onto orthogonal planes relative to conflicting gradients, thereby mitigating gradient conflicts. The work in [25] first performs joint training, and then splits the model for separate training to alleviate the negative transfer issue between tasks. These works merge multiple models into a single model, enabling existing FL frameworks in vehicular networks to address the challenges of multi-task training. However, this also leads to a sharp increase in the number of model parameters. The works in [26], [27], [28], [29], [30] investigate the training of multiple models, where each model is responsible for a single task. The work in [26] designs a client-sharing mechanism across multiple tasks, achieving a performance improvement of up to $2.03\times$. The work in [27] addresses the scenario where multiple FL servers simultaneously attempt to select clients from the same pool. Based on Bayesian optimization, it jointly considers training time and the importance of preventing long waiting times. The work in [28] delivers superior performance of multi-task FL for edge computing by leveraging the block coordinate descent algorithm. The work in [29] jointly optimizes task scheduling and resource allocation by considering the triple heterogeneity of data, devices, and tasks in a wireless network framework. The work in [30] shows that FL models with adaptive topologies have lower learning costs, and subsequently employs a heuristic algorithm to minimize the total cost. This approach can improve training efficiency through device scheduling while reducing resource consumption.

In summary, existing models are limited in two respects. On one hand, they primarily focus on FL for single-task scenarios in vehicular networks [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], making it hard to be used for multi-task scenarios. They need to address

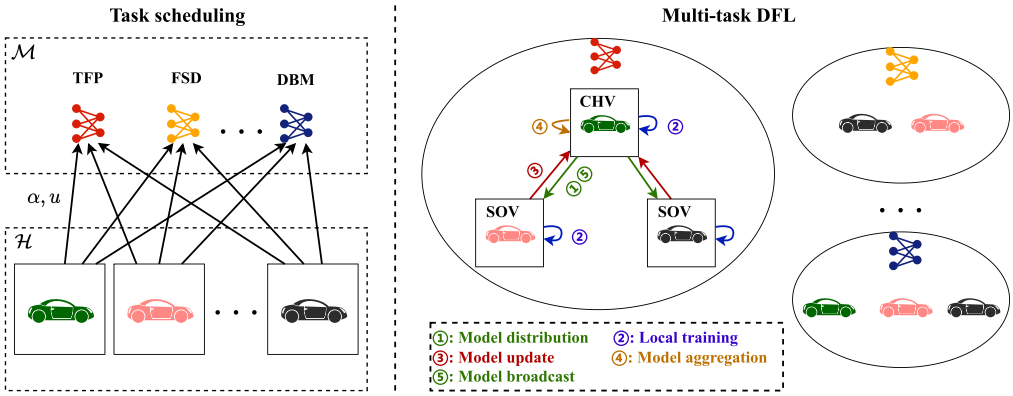


Fig. 1. Workflow of one communication round with task scheduling followed by multi-task DFL.

issues such as single point of failure and overwriting model update [31]. On the other hand, while some models are trained to handle multi-task scenarios [23], [24], [25], [26], [27], [28], [29], [30], these models overlook the impact of client mobility and the resource constraints, and both exist in vehicular networks. Due to vehicle mobility, the unpredictable mobility patterns and fluctuating network conditions caused by vehicle movement often result in unstable model transmission, thereby decreasing training efficiency. Therefore, the existing models are difficult to be directly applied to the multi-task federated learning scenarios in vehicular networks. Compared to the existing models, we investigate multi-task decentralized federated learning for vehicular networks. Our proposed leader-follower DFL framework dynamically selects leaders through vehicular ad hoc networks, effectively mitigating single point of failure while maintaining communication overhead comparable to that of centralized FL.

III. System Model

A. MMFL Framework

Fig. 1 shows the MMFL framework that includes H vehicles and these vehicles collaboratively complete M training tasks, such as TFP, FSD, and DBM [1]. Denote by $\mathcal{H}$ and $\mathcal{M}$ the sets of vehicles and tasks, respectively. Denote by $\mathcal{W}_h = \{\boldsymbol{w}_h^1, \boldsymbol{w}_h^2, \ldots, \boldsymbol{w}_h^M\}$ the set of model parameters for vehicle h, where $\boldsymbol{w}_h^m$ (for $m = 1, \ldots, M$) are in one-to-one correspondence with the elements in the training task set $\mathcal{M}$. Denote by $\mathcal{K}$ the set of communication rounds, $\mathcal{K} = \{1, 2, \ldots, K\}$. All communication rounds are of the same maximum time duration t^{round}. In the k-th communication round, the two-dimensional coordinates of vehicle h are denoted by (x_{kh}, y_{kh}), and its speed is denoted by $s_{kh}, \forall h \in \mathcal{H}, \forall k \in \mathcal{K}$. We assume that each vehicle can participate in at most one task training during a communication round. Specifically, denote by α_{kh}^m a binary variable, $\alpha_{kh}^m \in \{0, 1\}$, with $\sum_{m \in \mathcal{M}} \alpha_{kh}^m \leq 1, \forall h \in \mathcal{H}, \forall k \in \mathcal{K}$, which is one if and only if vehicle h selects to train task m in the k-th communication round. Denote by $\boldsymbol{\alpha}$ the corresponding tensor, where $\boldsymbol{\alpha}_k$ is a matrix containing the task scheduling status for each task of all vehicles in the k-th communication round. The model training process employs a leader-follower DFL strategy [18]. We divide the vehicles into two categories, i.e., cluster head vehicles (CHVs) and source vehicles (SOVs). Each CHV corresponds to a task. The set of SOVs is divided into M subsets, each responding to a CHV. Since we consider a dynamic cluster head strategy, the elements in these two sets will change dynamically in each communication round. Denote by u_{kh}^m a binary variable, $u_{kh}^m \in \{0, 1\}$, with $\sum_{h \in \mathcal{H}} \alpha_{kh}^m u_{kh}^m = 1, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}$, and $u_{kh}^m \leq \alpha_{kh}^m, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}, \forall h \in \mathcal{H}$, which is one if and only if vehicle h is selected as the leader for task m in the k-th communication round, and also serves as the CHV for task m. Denote by $\boldsymbol{u}$ the corresponding tensor, where $\boldsymbol{u}_k$ is a matrix containing the leader selection status for each task of all vehicles in the k-th communication round, and $\boldsymbol{u}_k^m$ is a vector containing the leader selection status for task m of all vehicles in the k-th communication round.

B. MMFL Training Model

In the MMFL framework, each vehicle $h \in \mathcal{H}$ holds M local datasets corresponding to the M tasks. For task m, the training process consists of the following five parts.

1) *Model distribution:* At the beginning of the k-th communication round, denote by r the CHV. This CHV sends its local model, denoted by $\boldsymbol{w}_{k-1,r}^m$, to the SOVs. Furthermore, this model $\boldsymbol{w}_{k-1,r}^m$ also serves as the global model for task m, so we simplify the notation to $\boldsymbol{w}_{k-1}^m$. The local model of each SOV s, defined as $\boldsymbol{w}_{k-1,s}^m$ is initialized to match the current global model $\boldsymbol{w}_{k-1}^m$ through model distribution.
2) *Local training:* After receiving the global model $\boldsymbol{w}_{k-1}^m$, each vehicle $h \in \mathcal{H}$ with $\alpha_{kh}^m = 1$ uses the stochastic gradient descent (SGD) algorithm to update the local model. Denote by $\mathcal{D}_h^m$ the dataset used for training by vehicle h for task m. Denote by $\mathcal{D}_h^{m'}$ the dataset used for testing by vehicle h for task m. For each task m, vehicle h draws training samples from the associated distribution $\mathcal{X}_h^m$ to construct SGD mini-batches, and tests samples from $\mathcal{Y}_h^m$ for evaluation. The local training process is expressed as

$$\boldsymbol{w}_{kh}^m = \boldsymbol{w}_{k-1}^m - \frac{\eta_k}{B_k^m} \sum_{x \in \mathcal{B}_{kh}^m} \nabla f^m(\boldsymbol{w}_{k-1,h}^m; \boldsymbol{x}), \quad (1)$$

where η_k represents the learning rate of the k-th communication round, $\nabla f^m(\boldsymbol{w}^m_{k-1,h};\boldsymbol{x})$ represents the gradient of the local model loss function, and $\mathcal{B}^m_{kh}$ represents the subset of $\mathcal{D}^m_h$ generated by vehicle h based on task m during the k-th communication round. Assuming all vehicles have equal batch sizes, we have $|\mathcal{B}^m_{kh}| = B^m_k$. Denote by I the number of local iterations, which each vehicle performs in every communication round. In each training iteration of the k-th communication round, vehicle h randomly samples a mini-batch $\mathcal{B}^m_{kh} \subseteq \mathcal{D}^m_h$ from the local training dataset for model updating, where each data sample $\boldsymbol{x} \in \mathcal{B}^m_{kh}$. Periodically, a sample $\boldsymbol{y} \in \mathcal{D}^{m'}_h$ from the local test dataset is used for performance evaluation. The loss function of task m in vehicle h, denoted by $f^m_h(\boldsymbol{w}^m)$, is expressed as

$$f^m_h(\boldsymbol{w}^m) = \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathcal{X}^m_h}[f^m(\boldsymbol{w}^m;\boldsymbol{x})], \tag{2}$$

where the loss function $f^m(\boldsymbol{w}^m;\boldsymbol{x})$ is used to measure the fitting performance of the model parameter $\boldsymbol{w}^m$. In (2), $f^m_h(\boldsymbol{w}^m)$ describes the average loss over the distribution $\mathcal{X}^m_h$.

3) *Model update:* Each SOV s sends its local model $\boldsymbol{w}^m_{ks}$ to the CHV.
4) *Model aggregation:* The CHV aggregates all the received models by using a weighted average method,

$$w^m_k = \frac{\sum_{h\in\mathcal{H}} \alpha^m_{kh}|\mathcal{D}^m_h|\boldsymbol{w}^m_{kh}}{\sum_{h\in\mathcal{H}} \alpha^m_{kh}|\mathcal{D}^m_h|}, \tag{3}$$

where $|\mathcal{D}^m_h|$ represents the size of the training dataset for vehicle h in task m. We assume that the vehicles participating in task m are drawn from the given distribution $\mathcal{P}$ [10]. The global loss function of task m, denoted by $F^m(\boldsymbol{w}^m)$, is expressed as

$$F^m(\boldsymbol{w}^m) = \mathop{\mathbb{E}}_{h\sim\mathcal{P}}[f^m_h(\boldsymbol{w}^m)]. \tag{4}$$

5) *Model broadcast:* The CHV sends the aggregated model $\boldsymbol{w}^m_k$ to the SOVs.

The above five training steps end until the loss function of each task converges or the number of communication rounds reaches its upper bound.

C. Mobility Model

Due to mobility, the communication state of vehicles changes rapidly with their positions. For the convenience of analysis, we make a quasi-static assumption where a vehicle's position does not change during the communication and computation process. This assumption allows us to accurately estimate vehicular communication conditions and resource consumption within each time slot. This modeling approach is consistent with existing advanced studies [32], [33]. Denote by $d_{kh_ih_j}$ the distance between vehicle h_i and vehicle h_j for the k-th communication round, which is expressed as

$$d_{kh_ih_j} = \sqrt{(x_{kh_i} - x_{kh_j})^2 + (y_{kh_i} - y_{kh_j})^2}. \tag{5}$$

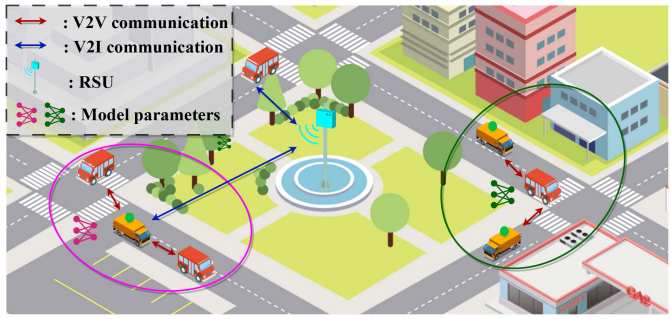


Fig. 2. Network architecture.

As shown in Fig. 2, we adopt a hybrid communication mode for model distribution, update, and broadcast. We assume that the energy and time consumption incurred by the sender and receiver during data transmission are identical for each individual communication link [14], [15]. This assumption applies exclusively to paired nodes with a specific link, and it does not imply homogeneity across all vehicles in the system model. Specifically, denote by d^U the direct communication radius. When $d_{kh_ih_j} \le d^U$, vehicle h_i and vehicle h_j use V2V communication. Conversely, when $d_{kh_ih_j} > d^U$, the communication between vehicle h_i and vehicle h_j involves indirect communication.

D. Communication Model

We consider orthogonal frequency division multiple access (OFDMA) as the transmission scheme [8]. This technology divides the bandwidth into multiple subcarriers, which are then allocated to different users. The subcarrier resources are also orthogonally allocated by each CHV. Denote by l_{ksr} the bandwidth allocation ratio assigned to SOV s by CHV r in the k-th communication round, which is expressed as

$$l_{ksr} = \frac{1}{N^S}\sum_{n=1}^{N^S}\sum_{m\in\mathcal{M}} u^m_{kr}c^m_{ksn}, \tag{6}$$

where N^S represents the number of subcarriers with the condition that $N^S \ge H$, and c^m_{ksn} represents a binary variable, $c^m_{ksn} \in \{0,1\}$, with $\sum_{s\in\mathcal{H}} c^m_{ksn} \le 1, \forall n \le N^S, \forall m \in \mathcal{M}, \forall k \in \mathcal{K}$, which is one if and only if the n-th subcarrier is allocated to SOV s in the k-th communication round for the m-th task. Denote by $\boldsymbol{c}$ a tensor, where $\boldsymbol{c}_k$ is a tensor containing the bandwidth allocation status in the k-th communication round, and $\boldsymbol{c}^m_k$ is a matrix containing the bandwidth allocation status of task m in the k-th communication round. Denote by R_{ksr} the transmission rate from SOV s to CHV r during the k-th communication round. Based on the definition in [8], R_{ksr} is expressed as

$$R_{ksr} = l_{ksr}W\log_2\left(1 + \frac{p_{ks}h_{ksr}d^{-\nu}_{ksr}}{\sigma^2}\right), \tag{7}$$

where W represents the total uplink bandwidth, p_{ks} represents the uplink transmission power of SOV s in the k-th communication round, h_{ksr} represents the power gain of the channel at

a reference distance of 1m in the k-th communication round, ν represents the path loss exponent, and σ^2 represents the variance of the complex white Gaussian channel noise. For indirect communication, d takes a large fixed value $\xi \cdot d^U$, where ξ is the distance scaling factor, as a suboptimal alternative for cases where direct communication is not possible.

Denote by T_{ks}^U the time duration taken for communication of SOV s in the k-th communication round, which is expressed as

$$T_{ks}^U = \sum_{m\in\mathcal{M}} \sum_{r\in\mathcal{H}\setminus s} \alpha_{ks}^m u_{kr}^m \frac{Z^m}{R_{ksr}}, \tag{8}$$

where Z^m represents the size of the model parameter $\boldsymbol{w}^m$. Please note the definitions of α_{ks}^m and u_{kr}^m, with $\alpha_{ks}^m u_{kr}^m \in \{0,1\}$. The product $\alpha_{ks}^m u_{kr}^m$ represents the communication between the CHV and the corresponding SOVs. Denote by E_{ks}^U the energy consumption due to data transmission of SOV s in the k-th communication round, which is expressed as

$$E_{ks}^U = \sum_{m\in\mathcal{M}} \sum_{r\in\mathcal{H}\setminus s} \alpha_{ks}^m u_{kr}^m p_{ks} \frac{Z^m}{R_{ksr}}. \tag{9}$$

The CHV and the corresponding SOVs communicate in parallel on different subcarriers. Denote by T_{kr}^U the communication time duration required by CHV r in the k-th communication round, which is expressed as

$$T_{kr}^U = \sum_{m\in\mathcal{M}} u_{kr}^m \max_{s\in\mathcal{H}} \{(\alpha_{ks}^m - u_{ks}^m) T_{ks}^U\}. \tag{10}$$

Denote by E_{kr}^U the communication energy consumption of CHV r in the k-th communication round, which is expressed as

$$E_{kr}^U = \sum_{m\in\mathcal{M}} u_{kr}^m \left(\sum_{s\in\mathcal{H}} (\alpha_{ks}^m - u_{ks}^m) E_{ks}^U \right). \tag{11}$$

E. Computation Model

Denote by p_{kh}^C the local computing power of vehicle h in the k-th communication round. Based on the definition in [7], p_{kh}^C is expressed as

$$p_{kh}^C = \lambda (f_{kh}^{\text{CPU}})^3, \tag{12}$$

where λ represents the effective switching capacitance of the CPU, and f_{kh}^{CPU} represents the CPU frequency of vehicle h in the k-th communication round. In the k-th communication round, vehicle $h \in \mathcal{H}$ needs to train the model for the assigned task m. Denote by T_{kh}^C the computation time required by vehicle h, which is expressed as

$$T_{kh}^C = \sum_{m\in\mathcal{M}} \alpha_{kh}^m \frac{I|\mathcal{D}_h^m| q}{f_{kh}^{\text{CPU}}}, \tag{13}$$

where q represents the CPU frequency required to process 1 b. The computation energy consumption of vehicle h in the k-th communication round is expressed as

$$E_{kh}^C = p_{kh}^C T_{kh}^C = \sum_{m\in\mathcal{M}} \alpha_{kh}^m I\lambda |\mathcal{D}_h^m| q (f_{kh}^{\text{CPU}})^2. \tag{14}$$

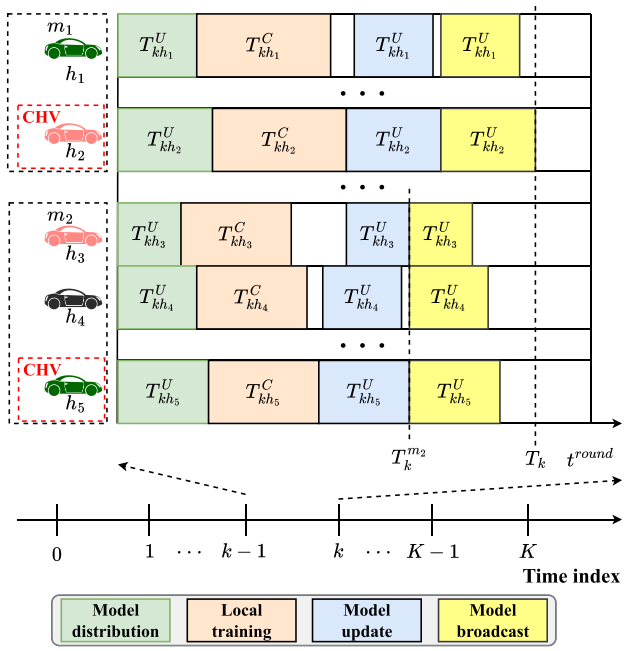


Fig. 3. Timeline.

F. TSLP Model

The timeline of MMFL is shown in Fig. 3. The k-th communication round corresponds to the interval between the $(k-1)$-th and k-th discrete time indices. In each communication round, there is a synchronization point after the model aggregation process. Before the synchronization point, vehicle h undergoes model distribution, model update, and local training. The communication time duration for both model distribution and model update is T_{kh}^U. The computing time duration of model update is T_{kh}^C. In addition, the aggregation process is not incorporated into our timeline illustration due to its comparatively short duration. Thus, denote by T_k^m the synchronization point of task m in the k-th communication round, which is expressed as

$$T_k^m = \max_h \{\alpha_{kh}^m (T_{kh}^C + 2T_{kh}^U)\}. \tag{15}$$

After the synchronization point, vehicle h undergoes model broadcast. The communication time duration of model broadcast is T_{kh}^C. Thus, denote by T_k the total time consumption in the k-th communication round, which is expressed as

$$T_k = \max_h \left\{ \sum_{m\in\mathcal{M}} \alpha_{kh}^m T_k^m + T_{kh}^U \right\}. \tag{16}$$

In a single communication round, vehicle h needs to go through three communication processes, namely model distribution, model update, and model broadcast. The communication energy consumption of each process is E_{kh}^U. In addition, the vehicle also needs to undergo a local training process, where the computing energy consumption is E_{kh}^C. Thus, denote by E_{kh} the total energy consumption of vehicle h in the k-th communication round, which is expressed as

$$E_{kh} = E_{kh}^C + 3E_{kh}^U. \tag{17}$$

Denote by E_{kh}^{res} the residual energy of vehicle h in the k-th communication round, which is expressed as

$$E_{kh}^{res} = E_h - \sum_{j=1}^{k-1} E_{jh}, \tag{18}$$

where E_h represents the initial energy of vehicle h.

Denote by F^{loss} the average loss of each task over the K-th (i.e., the final) communication round, which is expressed as

$$F^{loss} = \frac{1}{M} \sum_{m \in \mathcal{M}} \left(\frac{1}{\sum_{h \in \mathcal{H}} \alpha_{Kh}^m} \sum_{h \in \mathcal{H}} \alpha_{Kh}^m f_h^m(\boldsymbol{w}_{Kh}^m) \right), \tag{19}$$

where α_{Kh}^m represents the task selection status of vehicle h in the final communication round, and $\boldsymbol{w}_{Kh}^m$ represents the local model of vehicle h in the final communication round. In this paper, we aim to minimize the loss function for each training task subject to latency and energy consumption. Denote by ρ_{kh}^m the communication round index that corresponds to the most recent participation of vehicle h for task m prior to the k-th communication round. Therefore, TSLP is given in (20).

$$\mathcal{P}0: \min_{\boldsymbol{\alpha}, \boldsymbol{u}, \boldsymbol{c}} F^{loss} \tag{20a}$$

$$\text{s.t.} \quad \max_k (\alpha_{kh}^m T_k) \leq t^{round}, h \in \mathcal{H}, m \in \mathcal{M}, k \in \mathcal{K}, \tag{20b}$$

$$\alpha_{kh}^m E_{kh} \leq E_{kh}^{res}, h \in \mathcal{H}, m \in \mathcal{M}, k \in \mathcal{K}, \tag{20c}$$

$$\sum_{m \in \mathcal{M}} \alpha_{kh}^m \leq 1, h \in \mathcal{H}, k \in \mathcal{K}, \tag{20d}$$

$$u_{kh}^m \leq \alpha_{kh}^m, m \in \mathcal{M}, k \in \mathcal{K}, h \in \mathcal{H}, \tag{20e}$$

$$\sum_{h \in \mathcal{H}} \alpha_{kh}^m u_{kh}^m = 1, m \in \mathcal{M}, k \in \mathcal{K}, \tag{20f}$$

$$\sum_{h \in \mathcal{H}} c_{khn}^m \leq 1, n \leq N^S, m \in \mathcal{M}, k \in \mathcal{K}, \tag{20g}$$

$$u_{kr}^m = \arg\max_{\boldsymbol{u}_k^m} \rho_{kh}^m, m \in \mathcal{M}, k \in \mathcal{K}, \tag{20h}$$

$$\alpha_{kh}^m \in \{0,1\}, u_{kh}^m \in \{0,1\}, c_{khn}^m \in \{0,1\}. \tag{20i}$$

In (20), constraint (20b) ensures that the time spent by each vehicle in any communication round is strictly bounded by a time duration of t^{round}. The constraint is influenced by $\boldsymbol{\alpha}$, $\boldsymbol{u}$, and $\boldsymbol{c}$, as seen in (6), (7), (8), (10), (15), and (16). This implies that the variables are coupled. Constraint (20c) ensures that the energy consumption of each participating vehicle in any communication round does not exceed its current energy capacity. Constraints (20d) and (20e) state that a vehicle participates in at most one task. Constraint (20f) represents that each task is allocated a cluster head. Constraint (20g) indicates orthogonal allocation of subcarriers. Constraint (20h) is used to mitigate the issue that a model update becomes overwritten. Constraint (20i) sets the range of values that the optimization variables can take.

MMFL achieves fairness in three aspects. Firstly, it considers the trade-off between vehicle model training and resource consumption. Given that each vehicle has limited energy, those that frequently participate in training need to consume more energy, which naturally limits their participation frequency. This mechanism ensures that all vehicles can participate fairly within their resource constraints, preventing over-participation that could lead to unfairness. Secondly, MMFL shares model updates only among vehicles involved in the task. Vehicles that do not participate in training cannot obtain the aggregated model. This approach prevents free-riding and ensures that only participating vehicles benefit from model updates, thereby maintaining fairness among the participants. Finally, MMFL dynamically designates leaders and allocates communication resources based on a combination of task participation and communication conditions. This strategy helps reduce the communication burden on leaders while ensuring that all participating vehicles can communicate and share data under fair conditions. By implementing these mechanisms, MMFL not only enhances the overall system efficiency but also ensures equitable resource allocation and participation.

IV. Problem Analysis

In this section, we first discuss the convergence bounds for a single task under the MMFL framework. For multi-task scenarios, we model TSLP as a resource allocation game, and then prove the existence of NE for the game.

A. Convergence Analysis

For each task m, we make the following assumptions [10], [34]:

Assumption 1: The local loss function $f_h^m(\boldsymbol{w}^m)$ is L-smooth for each vehicle $h \in \mathcal{H}$ in each communication round $k \in \mathcal{K}$, i.e,

$$f_h^m(\boldsymbol{w}_k^m) - f_h^m(\boldsymbol{w}_{k-1}^m) \leq \left\langle \nabla f_h^m(\boldsymbol{w}_{k-1}^m), \boldsymbol{w}_k^m - \boldsymbol{w}_{k-1}^m \right\rangle + \frac{L}{2} \left\| \boldsymbol{w}_k^m - \boldsymbol{w}_{k-1}^m \right\|^2.$$

Assumption 2: The local loss function $f_h^m(\boldsymbol{w}^m)$ is μ-strongly convex for each vehicle $h \in \mathcal{H}$ in each communication round $k \in \mathcal{K}$, i.e,

$$f_h^m(\boldsymbol{w}_k^m) - f_h^m(\boldsymbol{w}_{k-1}^m) \geq \left\langle \nabla f_h^m(\boldsymbol{w}_{k-1}^m), \boldsymbol{w}_k^m - \boldsymbol{w}_{k-1}^m \right\rangle + \frac{\mu}{2} \left\| \boldsymbol{w}_k^m - \boldsymbol{w}_{k-1}^m \right\|^2.$$

Assumption 3: The stochastic gradient is unbiased, i.e.,

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{X}_h^m} [\nabla f^m(\boldsymbol{w}^m; \boldsymbol{x})] = \mathop{\mathbb{E}}_{h \sim \mathcal{P}_h} [\nabla f_h^m(\boldsymbol{w}^m)] = \nabla F^m(\boldsymbol{w}^m).$$

Assumption 4: The stochastic gradient is variance-bounded, i.e.,

$$\mathop{\mathbb{E}}_{x \sim \mathcal{X}_h^m} \left[\left\| \nabla f^m(\boldsymbol{w}^m; \boldsymbol{x}) - \nabla F^m(\boldsymbol{w}^m) \right\|^2 \right] \leq G^2.$$

Assumption 5: In the k-th communication round beyond the initial one, the CHV of task m is selected from the cluster of task m in the $(k-1)$-th communication round.

Assumption 5 prevents outdated models from overwriting new updates. Based on the above assumptions, we obtain the following lemma:

Lemma 1: With Assumptions 1-5 and the aggregation rule (3), the expected decrease of loss after one round is upper

bounded by

$$\mathbb{E}\left[F^m(\boldsymbol{w}_k^m)\right]-\mathbb{E}\left[F^m(\boldsymbol{w}_{k-1}^m)\right] \leq \eta_k\left(\frac{L\eta_k}{2}-1\right)\left\|\nabla F^m(\boldsymbol{w}_{k-1}^m)\right\|^2+\frac{L\eta_k^2}{2}\frac{G^2}{B_k^m\sum_{h\in\mathcal{H}}\alpha_{kh}^m}, \tag{21}$$

where the expectation is taken over the randomness of SGD.

Proof: See Appendix A. ■

Based on Lemma 1, we obtain the convergence performance after K communication rounds, presented in Theorem 1.

Theorem 1: After K communication rounds of training, the difference between $F^m(\boldsymbol{w}_K^m)$ and the optimal global loss function denoted by $F^m(\boldsymbol{w}^{m*})$ is upper bounded by

$$\begin{aligned}&\mathbb{E}\left[F^m(\boldsymbol{w}_K^m)\right]-F^m(\boldsymbol{w}^{m*})\\ &\leq\left(\mathbb{E}[F^m(\boldsymbol{w}_0^m)]-F^m(\boldsymbol{w}^{m*})\right)\prod_{k=1}^{K}(1-\mu\eta_k)\\ &\quad+\sum_{k=1}^{K-1}\frac{\eta_k}{2}\frac{G^2}{B_k^m\sum_{h\in\mathcal{H}}\alpha_{kh}^m}\prod_{j=k+1}^{K}(1-\mu\eta_j)\\ &\quad+\frac{\eta_K}{2}\frac{G^2}{B_K^m\sum_{h\in\mathcal{H}}\alpha_{Kh}^m}.\end{aligned} \tag{22}$$

Proof: See Appendix B. ■

The convergence analysis highlights the impact of three optimization variables on FL, i.e., the number of participants $\boldsymbol{\alpha}$, the update status $\boldsymbol{u}$, and the bandwidth allocation $\boldsymbol{c}$. First, the convergence bound of the global loss function for task m decreases with the increase of the number of participants. This implies that the decision of $\boldsymbol{\alpha}$ directly affects the convergence of an individual FL task. More participants lead to a tighter convergence bound, enhancing the overall training performance. Second, variable $\boldsymbol{u}$ affects the update status of an individual FL task, i.e., the extent to which model updates are overwritten. It also influences the resource consumption of each vehicle due to mobility. Optimizing $\boldsymbol{u}$ ensures that the updates are effectively utilized, balancing the trade-off between convergence and resource allocation. Furthermore, optimizing the bandwidth allocation variable $\boldsymbol{c}$ helps reduce the operational latency of the framework. Given that $\mu>0$ and $\eta_k>0$, we have $1-\mu\eta_k<1$, which indicates that as the number of communication rounds increases, the convergence bound of the global loss function for task m becomes stronger. Therefore, the decisions of $\boldsymbol{\alpha}$, $\boldsymbol{u}$, and $\boldsymbol{c}$ collectively affect the training performance of each task.

B. Resource Allocation Game Formulation

For multiple task scenarios, the convergence rate of the loss within the framework's operation is influenced by $\boldsymbol{\alpha}$, $\boldsymbol{u}$, and $\boldsymbol{c}$, for which the relations are not analytically available. By Theorem 1, we find that the convergence bound of a single FL task is directly related to the number of participants. To address the interdependencies between tasks in multi-task scenarios, we model the inter-task relationship as a game. Specifically, we prove the existence of a Nash equilibrium, which ensures that each task can achieve optimal performance given the resource allocation decisions of other tasks. This game-theoretic formulation allows us to analyze the overall system dynamics beyond the single-task assumption and provides a comprehensive understanding of the multi-task FL framework. Note that the loss function represents the distance between the predicted labels and the true labels, which is used to assess the model's fitness during training. This value decreases as the training progresses. Accuracy represents the agreement between the predicted labels and the true labels, and is used to measure the model's performance on the test set. The closer this value is to one, the better the model's performance. The work in [35] models the relationship between accuracy and training efficiency in multi-task FL. The transition between minimizing the loss function and maximizing training efficiency does not affect the objective of TSLP. Consequently, we adopt the aggregate training efficiency as a measure. A higher training efficiency implies that the number of communication rounds K required for convergence is smaller, thereby enhancing energy efficiency. The training efficiency is defined in accordance with the work in [35].

Definition 1: The training efficiency of task m in the k-th communication round increases with the number of assigned vehicles, but the increase in training efficiency is diminishing, roughly following a logarithmic relationship, which is expressed as

$$\psi_k^m=\beta^m\log\left(\sum_{h\in\mathcal{H}}\alpha_{kh}^m\right)+\theta^m, \tag{23}$$

where β^m and θ^m represent the scaling coefficient and bias term of task m, respectively, which govern how training efficiency scales with participant count in task m.

We model the problem as a non-cooperative game among the tasks, where each task is regarded as a player and it independently decides on its vehicle allocation strategy, with the goal of faithfully improving its own training efficiency. The TSLP in (20) is transformed into

$$\begin{aligned}&\mathcal{P}1:\max_{\boldsymbol{\alpha},\boldsymbol{u},\boldsymbol{c}}\sum_{k\in\mathcal{K}}\sum_{m\in\mathcal{M}}\psi_k^m\\ &\text{s.t.}\quad (120b)-(20i).\end{aligned} \tag{24a}$$

Based on the conclusions of Theorem 1, we observe that the tasks in the framework are interrelated. Therefore, it is necessary to model these inter-task relationships. Drawing on the conclusions in [35], model loss, accuracy, and training efficiency all measure the effectiveness of the model's training. In addition, as derived from Theorem 1, the conclusion that more participants lead to a smaller loss aligns with the setting that more participants result in higher training efficiency. The insight motivates the game-theoretic formulation of problem (24a).

To describe the competitive relationship between tasks in the k-th communication round, we define resource allocation game $\mathcal{G}_k=\{\mathcal{M},\mathbb{S}_k,\{U_k^m\}_{m\in\mathcal{M}}\}$, where $\mathbb{S}_k$ represents the strategy

space of the game, defined as the Cartesian product of all individual strategy sets of the tasks: $\mathbb{S}_k = \mathbf{S}_k^1 \times \cdots \times \mathbf{S}_k^m \times \cdots \times \mathbf{S}_k^M$, where $\mathbf{S}_k^m$ represents the set of all strategies for task m in the k-th communication round. Each $\mathscr{S}_k \in \mathbb{S}_k$ is a strategy profile. For task m, the profile $\mathscr{S}_k = (\mathscr{S}_k^1, \ldots, \mathscr{S}_k^m, \ldots, \mathscr{S}_k^M)$ can be rewritten as $\mathscr{S}_k = (\mathscr{S}_k^m, \mathscr{S}_k^{-m})$, where $\mathscr{S}_k^m$ denotes the strategy of task m in the k-th communication round, represented as $\mathscr{S}_k^m = \{h \in \mathcal{H} \mid \alpha_{kh}^m = 1\}$, and $\mathscr{S}_k^{-m}$ represents the joint strategy adopted by the tasks other than task m. Denote by $U_k^m(\mathscr{S}_k)$ the utility function of task m in the k-th communication round, which is defined as follows.

Definition 2: The utility function $U_k^m(\mathscr{S}_k) : \mathbb{S}_k \mapsto \mathbb{R}$ is defined as the training efficiency of task m under the strategy configuration $\mathscr{S}_k$, where $\mathbb{R}$ represents the set of real numbers. Specifically, $U_k^m(\mathscr{S}_k)$ is expressed as

$$U_k^m(\mathscr{S}_k) = \begin{cases} \psi_k^m, & \text{if (120b) – (20i) are satisfied,} \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Next, we demonstrate that the game model $\mathcal{G}_k$ is a potential game (PG) with at least one NE by providing the potential function. We first present the definition of a potential game.

Definition 3: If there exists a potential function $\Omega_k^m(\mathscr{S}_k)$ that satisfies (26), then the game is a potential game.

$$\begin{aligned} &U_k^m(\mathscr{S}_k^{m\prime}, \mathscr{S}_k^{-m}) > U_k^m(\mathscr{S}_k^m, \mathscr{S}_k^{-m}) \\ &\mapsto \Omega_k^m(\mathscr{S}_k^{m\prime}, \mathscr{S}_k^{-m}) > \Omega_k^m(\mathscr{S}_k^m, \mathscr{S}_k^{-m}). \end{aligned} \tag{26}$$

Theorem 2: For the potential function defined below for task m, game $\mathcal{G}_k$ is a potential game.

$$\Omega_k^m(\mathscr{S}_k) = \sum_{i \in \mathcal{M}} \left[U_k^i(\mathscr{S}_k^m, \mathscr{S}_k^{-m}) - U_k^i(-\mathscr{S}_k^m, \mathscr{S}_k^{-m}) \right], \tag{27}$$

where $U_k^i(-\mathscr{S}_k^m, \mathscr{S}_k^{-m})$ represents the utility that can be achieved when $\mathscr{S}_k^m$ is ineffective, meaning that no valid set of $\boldsymbol{\alpha}$, $\boldsymbol{u}$ and $\boldsymbol{c}$ can satisfy this strategy.

Proof: See Appendix C. ■

In the game model $\mathcal{G}_k$, all tasks attempt to achieve a NE by maximizing their utility in the presence of conflicting interests. Since Theorem 2 proves that the resource allocation game is a potential game, it has the finite improvement property (FIP), thus a NE allocation strategy can be obtained through a finite number of iterations [36]. Solving the NE in the resource allocation game can enhance the training efficiency of all tasks in the framework, improving the convergence rate.

V. Algorithm Design

Based on the proof of the NE existence, we first reformulate the problem in (24) as a DEC-POMDP, and then propose an HAPPO-based algorithm to solve DEC-POMDP.

A. DEC-POMDP Reformulation

As shown in Fig. 4, we design a centralized training and decentralized execution (CTDE) multi-agent reinforcement learning (MARL) algorithm to solve the resource allocation game. The

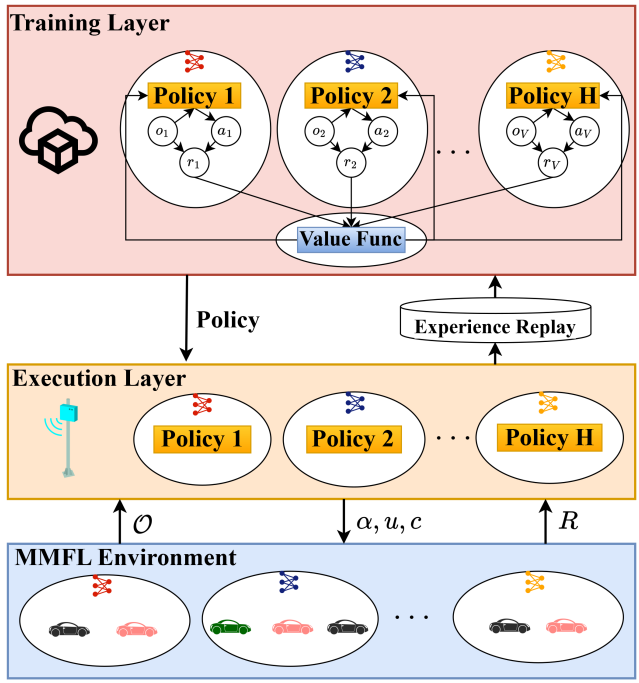


Fig. 4. Diagram of centralized training and distributed execution framework.

training layer, deployed on a cloud server, is responsible for optimizing policy and value networks using offline data collected from vehicles. Once trained, the optimal policy networks are distributed to individual agents in the execution layer, where each agent corresponds to a single vehicle. During execution, agents rely solely on local observations, feeding them into their policy networks to generate actions and interact with the MMFL environment in a fully decentralized manner. Within each task group, vehicles dynamically elect leaders and allocate subcarriers through a consensus-based mechanism, ensuring efficient coordination without centralized control. After executing for a predefined number of steps, vehicles upload their historical experience data to the cloud server, where the training layer performs further policy updates and refinements until the changes in the agents' policies or value functions fall below a specified threshold. This closed-loop process enables continuous improvement while maintaining scalability and adaptability in real-world deployment. Compared to fully distributed systems, CTDE leverages centralized training to make full use of global information, leading to more stable and optimal policies. Compared to fully centralized systems, CTDE reduces communication and computational demands during execution, making it more suitable for resource-constrained application scenarios. Specifically, utilizing global information during training, HAPPO learns policies that are more robust and optimal. In addition, during execution, agents operate independently in HAPPO, minimizing the need for real-time communication and reducing latency. Finally, decentralized execution allows the system to scale more effectively, accommodating a larger number of agents without significant increases in computational or communication costs.

Denote by $\zeta = \left\{\{\zeta_h\}_{h=1}^{H}\right\}$ the agent set of task distributors, where each agent corresponds to a vehicle. The resource allocation game (24) is reformulated to a DEC-POMDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, R, P \rangle$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{O}$, R, P represent the global state, action set, local observation set, reward function, and state transition function, respectively [37]. The local observation space, denoted by o_{kh}, consists of the most recent communication round index of the vehicle's participation in any task, remaining energy, transmission rate, position, and speed of vehicle h in the k-th communication round, which is expressed as

$$o_{kh} = \left\{ \{\rho_{kh}^m\}_{m\in\mathcal{M}}, E_{kh}^{res}, \sum_{r\in\mathcal{H}} \alpha_{kh}^m u_{kr}^m R_{khr}, x_{kh}, y_{kh}, s_{kh} \right\}. \tag{28}$$

The global state $\mathcal{S}$ contains the local observations of all agents across all K communication rounds, which is expressed as

$$\mathcal{S} = \left\{ o_{kh} | h \in \mathcal{H}, k \in \mathcal{K} \right\}. \tag{29}$$

For the decision of $\boldsymbol{\alpha}_k$, denote by a_v the action space of the agent corresponding to vehicle h. In each communication round, vehicle h can choose any task, corresponding to the task indexes in the action space; it can also choose not to participate in any task, corresponding to value zero. The action space is expressed as

$$a_v = \{0, 1, 2, \ldots, M\}. \tag{30}$$

For the decision of $\boldsymbol{u}_k$ and $\boldsymbol{c}_k$, we need to ensure a balance between the efficient execution of FL and resource consumption. Specifically, we propose a joint optimization for leader selection and subcarrier allocation, as shown in Algorithm 1. We score all vehicles within the task group m. In Line (19), the first term considers communication efficiency and the second term considers training efficiency, where ε represents the weighting coefficient. Denote by h_k^{m*} the vehicle with the highest score in the group, which is then selected as the leader for task m in the k-th communication round. Note that $\boldsymbol{\alpha}_k$ is fixed at this point. The complexity of Algorithm 1 is $O(M \cdot N \log H)$. By applying this algorithm, the dimensionality of the action space is effectively reduced, leading to improved search efficiency.

In Section IV-A, the convergence analysis demonstrates the achievable convergence bound when Assumption 5 holds. However, with presence of vehicle mobility, the conditions for Assumption 5 are relatively stringent. To address this challenge, the proposed algorithm incorporates a reward mechanism that incentivizes decisions satisfying Assumption 5 with a positive reward. This mechanism ensures that the algorithm prioritizes decisions that lead to better training outcomes, even in the presence of vehicle mobility. Denote by r_{kh} the local reward of the agent corresponding to vehicle h in the k-th communication round; the reward includes the utility of the corresponding task in game (24). Additionally, we introduce ρ_{kh}^m with the aim of encouraging the agent to satisfy Assumption 5 as much as possible. The local reward is expressed as

Algorithm 1: Joint Optimization of Leader Selection and Subcarrier Allocation.

Input: $\boldsymbol{\alpha}_k$, $\{\rho_{kh}^m | m \in \mathcal{M}, h \in \mathcal{H}\}$, and $\{(x_{kh}, y_{kh}) | h \in \mathcal{H}\}$;
Output: $\boldsymbol{u}_k$ and $\boldsymbol{c}_k$;
1: // At the beginning of the k-th communication round;
2: **Initialize:** $\boldsymbol{u}_k \leftarrow \mathbf{0}$, $\boldsymbol{c}_k \leftarrow \mathbf{0}$;
3: **for all** $m \in \mathcal{M}$ **do**
4: **for all** $r \in \mathcal{H}$ **with** $\alpha_{kr}^m = 1$ **do**
5: **Initialize:** Max-Heap $Q = \emptyset$;
6: **for all** $h \in \mathcal{H} \setminus r$ **with** $\alpha_{kh}^m = 1$ **do**
7: Assign a subcarrier to vehicle h;
8: Calculate T_{kh}^U based on (8) and (10);
9: Insert vehicle h with key T_{kh}^U into Q;
10: **end for**
11: **while** Remaining subcarriers > 0 **do**
12: $h \leftarrow$ Pop the top element from Q;
13: Assign a subcarrier to vehicle h;
14: Calculate T_{kh}^U based on (8) and (10);
15: Insert vehicle h with key T_{kh}^U into Q;
16: **end while**
17: $\hat{h} \leftarrow$ Pop the top element from Q;
18: Calculate $T_{k\hat{h}}^U$ based on (8) and (10);
19: Calculate the score for vehicle r:

$$score_r = \frac{\rho_{kr}^m}{k} - \varepsilon \frac{T_{k\hat{h}}^U}{t^{round}}; \tag{31}$$

20: **end for**
21: $h_k^{m*} = \arg\max_{h\in\mathcal{H}} score_v$;
22: $u_{k,h_k^{m*}}^m = 1$;
23: Generate $\boldsymbol{c}_k^m$ for h_k^{m*} according to the above process;
24: **end for**

as

$$r_{kh} = \sum_{m\in\mathcal{M}} \alpha_{kh}^m \psi_k^m \rho_{kh}^m. \tag{32}$$

Furthermore, for action outputs that do not satisfy the constraints of game (24), we apply a penalty to all agents.

The global reward is defined as the sum of the local rewards (32) over all K communication rounds and all agents in the framework. The global reward is expressed as

$$R = \sum_{k\in\mathcal{K}} \sum_{h\in\mathcal{H}} r_{kh}. \tag{33}$$

For the transition of local observations P from o_{kh} to $o_{k+1,h}$, the remaining energy is updated by (18), the position and speed of the vehicle are acquired in real time, and the transmission rate is estimated based on the current conditions and (7).

B. HAPPO-Based Optimization Algorithm

Denote by π_k the joint decision made by ζ in the k-th communication round, and $(\pi_k)_{k=0}^{\infty}$ the sequence of joint decisions. Sequence $(\pi_k)_{k=0}^{\infty}$ obtained through the HAPPO algorithm possesses four properties. First, the expected return of the joint policy is monotonically increasing. Second, as the number of

iterations increases, the generated value functions converge to the value function corresponding to a NE. Third, the expected return of the joint policy converges to the expected return of a NE. Finally, the ω-limit set of the policy sequence contains NE strategies [38]. Furthermore, using HAPPO allows for a clearer description of the interactions between agents, and it converges faster compared to traditional reinforcement learning (RL).

To solve DEC-POMDP in Section V-A and obtain a NE solution, we propose an HAPPO-based optimization algorithm. The goal of the proposed algorithm is to train H local policy networks and one global value network.

Denote by π_{θ^h} the policy of agent ζ_h, which is formulated by a multilayer perceptron (MLP) neural network with parameter θ^h. Denote by V_ϕ the global V-value function, which is formulated by an MLP neural network with parameter ϕ. Here, π_{θ^h} is a multinomial distribution used for action selection, i.e.,

$$a^h \sim \text{Categorical}(\pi_{\theta^h}(a^h \mid o^h)), \tag{34}$$

where a^h represents the action taken by agent ζ_h in the local observation o^h. We use V_ϕ to represent the expected reward for the entire system when all agents act according to their respective policies π_{θ^h}, given the global state s, i.e.,

$$V_\phi(s) = \mathbb{E}_{\pi_{\theta^1}, \pi_{\theta^2}, \ldots, \pi_{\theta^H}} \left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s \right],$$

where R_t represents the joint reward of all agents at time step t, and γ represents the discount factor. Denote by T_s the time steps for trajectory collection. At time step t, the expectation reward $\hat{R}_t$ is expressed as

$$\hat{R}_t = \sum_{m=0}^{T_s - t - 1} \gamma^m R_m. \tag{35}$$

Note that one time step in the algorithm refers to executing one FL communication round in the MMFL environment. When we reset the MMFL environment, the FL communication round is set to zero, but the time step remains unchanged.

The advantage function in reinforcement learning is used to estimate the advantage of a particular action relative to the average policy. Denote by $\hat{A}_t(s, \boldsymbol{a})$ the advantage function at time step t. It is calculated by generalized advantage estimation (GAE), which is expressed as

$$\hat{A}_t(s, \boldsymbol{a}) = \sum_{m=0}^{T_s - t - 1} (\gamma\beta)^m \delta_{t+m}, \tag{36}$$

where $\boldsymbol{a}$ represents the joint action, β represents the smoothing factor, and δ_t represents the temporal difference error, which is expressed as

$$\delta_t = R_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t).$$

Denoting by ϵ the discount ratio, the training of the proposed HAPPO-based optimization algorithm is shown in Algorithm 2. In HAPPO, the policy networks of individual agents are updated sequentially, with no shared policy parameters between agents [39]. This approach mitigates conflicts in policy updates among agents, thereby improving the training efficiency.

Algorithm 2: Training of HAPPO-Based Optimization Algorithm

Input: Batch size Θ, number of agents H, training episodes K_s, steps per training episode T_s;
Output: Actor networks $\{\theta^h_{K_s} | \zeta_h \in \zeta\}$, global V-value network ϕ_{K_s};

Initialize: Actor networks $\{\theta^h_0 | \zeta_h \in \zeta\}$, global V-value network ϕ_0, replay buffer $\mathcal{U}$;
for $k_s = 0, 1, \ldots, K_s - 1$ **do**
  Collect a set of trajectories by running the joint policy $\boldsymbol{\pi}_{\boldsymbol{\theta}_{k_s}} = (\pi^1_{\theta^1_{k_s}}, \ldots, \pi^H_{\theta^H_{k_s}})$;
  for $t = 0, 1, \ldots, T_s - 1$ **do**
    Obtain $\boldsymbol{\alpha}_t$ based on sampling (34);
    Obtain $\boldsymbol{u}_t, \boldsymbol{c}_t$ based on Algorithm 1;
    Execute a communication round t of the MMFL environment using $\boldsymbol{\alpha}_t, \boldsymbol{u}_t, \boldsymbol{c}_t$;
    if Constraints (20b)-(20i) are not all satisfied **then**
      Reset the MMFL environment;
    end if
    Obtain $\{(o_{th}, r_{th}) \mid \forall \zeta_h \in \zeta\}$;
  end for
  Push transitions $\{(o_{th}, a_{th}, o_{t+1,h}, r_{th}) \mid \forall \zeta_h \in \zeta, t \le T_s\}$ into $\mathcal{U}$;
  Sample a random minibatch of U transitions from $\mathcal{U}$;
  Compute advantage function $\hat{A}(s, \boldsymbol{a})$ based on (36);
  Draw a random permutation of agents $\zeta_{1:H}$;
  Set $M^{\zeta_1}(s, \boldsymbol{a}) = \hat{A}(s, \boldsymbol{a})$;
  for all $\zeta_h \in \zeta$ **do**
    Update actor network $\theta^{\zeta_h}_{k+1}$, the argmax of the PPO-Clip objective [40]:

$$\frac{1}{\Theta T} \sum_{i=1}^{\Theta} \sum_{t=0}^{T} \min \left(\frac{\pi^{\zeta_h}_{\theta^{\zeta_h}} \left(a^{\zeta_h}_t | o^{\zeta_h}_t \right)}{\pi^{\zeta_h}_{\theta^{\zeta_h}_k} \left(a^{\zeta_h}_t | o^{\zeta_h}_t \right)} M^{\zeta_{1:h}}(s_t, \boldsymbol{a}_t), \right.$$
$$\left. \text{clip} \left(\frac{\pi^{\zeta_h}_{\theta^{\zeta_h}} \left(a^{\zeta_h}_t | o^{\zeta_h}_t \right)}{\pi^{\zeta_h}_{\theta^{\zeta_h}_k} \left(a^{\zeta_h}_t | o^{\zeta_h}_t \right)}, 1 \pm \epsilon \right) M^{\zeta_{1:h}}(s_t, \boldsymbol{a}_t) \right); \tag{37}$$

    Compute unless $\zeta_h = \zeta_V$:

$$M^{\zeta_{1:h+1}}(s, \boldsymbol{a}) = \frac{\pi^{\zeta_h}_{\theta^{\zeta_h}} (a^{\zeta_h}_t \mid o^{\zeta_h}_t)}{\pi^{\zeta_h}_{\theta^{\zeta_h}_k} (a^{\zeta_h}_t \mid o^{\zeta_h}_t)} M^{\zeta_{1:h}}(s, \boldsymbol{a}); \tag{38}$$

  end for
  Update the V-value network by the following formula:

$$\pi_{k+1} = \arg\min_{\pi} \frac{1}{\Theta T} \sum_{u=1}^{U} \sum_{t=0}^{T_s} \left(V_{\pi_k}(s_t) - \hat{R}_t \right)^2; \tag{39}$$

end for

C. Complexity Analysis

The computational complexity of the HAPPO algorithm primarily arises from forward propagation for action prediction and backward propagation during network training. We employ a

multi-layer perceptron (MLP) with Γ layers to construct both the actor network θ and the V-value network ϕ, which share the same network depth. For the i-th layer ($i \in \{1, 2, \ldots, \Gamma\}$) in both θ and ϕ, the computational complexity is given by $\mathcal{O}(N_{i-1}^E N_i^E + N_i^E N_{i+1}^E)$, where N_i^E represents the number of neurons in the i-th layer [41]. All agents perform parallel computation within a single time step. For K_s episodes, each episode consists of T_s steps and the update time per training step is T_p. Thus, the overall computational complexity of the proposed algorithm can be expressed as $\mathcal{O}(K_s((T_s + T_p)\sum_{i=2}^{\Gamma-1}(N_{i-1}^E N_i^E + N_i^E N_{i+1}^E)))$.

D. Communication Cost Analysis

In the MMFL framework, the communication cost of SOV s is the same as the client in centralized FL. In the model distribution and broadcast phases, SOV s receives $\sum_{m\in\mathcal{M}} \alpha_{ks}^m Z^m$ model parameters. It sends the same amount during the model update phase. The communication consumption of CHV r is the same as the server in centralized FL. During the model distribution and broadcast phases, CHV r sends $\sum_{s\in\mathcal{H}\setminus r}\sum_{m\in\mathcal{M}} \alpha_{ks}^m u_{kr}^m Z^m$ model parameters. It receives the same amount during the model update phase. Denote by E^{edge} the cost (e.g., energy consumption) for directly transmitting a single model parameter via V2V communication, E^{cloud} the cost for indirectly transmitting a single model parameter via V2I communication, and p^{edge} the probability of adopting V2V communication. Denote by $Cost_k^M$ and $Cost_k^C$ the total cost of all vehicles in the k-th communication round under the MMFL framework and the centralized FL framework, respectively. These can be expressed in (40), where the numeral 3 reflects the three communication phases, i.e., model distribution, model update, and model broadcast. Compared to centralized FL, the communication bottleneck in MMFL arises from the data exchange between the CHV and all SOVs within each task group. In addition, the proportion of different communication modes also affects the communication overhead of MMFL.

$$\begin{cases} Cost_k^M = 3\Big[\sum_{r\in\mathcal{H}}\sum_{s\in\mathcal{H}\setminus r}\sum_{m\in\mathcal{M}} \alpha_{ks}^m u_{kr}^m Z^m \\ \qquad + \sum_{s\in\mathcal{H}}\sum_{m\in\mathcal{M}} \alpha_{ks}^m (1-u_{ks}^m) Z^m\Big] \\ \qquad \times \left[p^{edge}E^{edge} + (1-p^{edge})E^{cloud}\right], \\ Cost_k^C = 3\sum_{s\in\mathcal{H}}\sum_{m\in\mathcal{M}} \alpha_{ks}^m Z^m E^{cloud}. \end{cases} \tag{40}$$

VI. Experiments

A. Experiment Setting

Numerical experiments are used to validate the effectiveness of the proposed algorithm. Specifically, we use the SUMO software to generate traffic flow and simulate multi-task FL, as shown in Fig. 5. We assume that H vehicles are always driving within the road network and collaboratively complete a total of M tasks. Each CHV divides the bandwidth into N^S subcarriers using OFDMA technology. We consider four classification tasks over datasets MNIST with LeNet [42], Fashion-MNIST [43] (referred to as FMNIST) with LeNet, SVHN [44] with LeNet, and CIFAR-10 [45] with ResNet-18 [46]. Based on the analysis in Section IV-B, we use the test accuracy to evaluate the model's

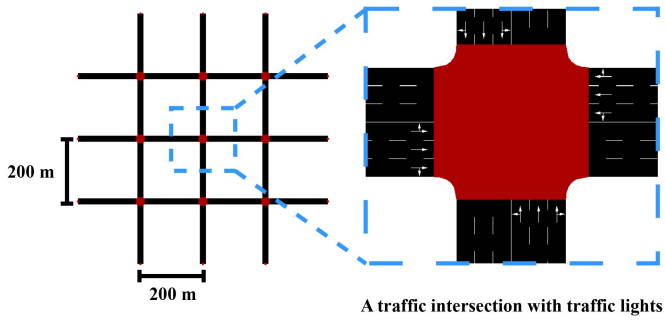


Fig. 5. The SUMO road network.

TABLE I
Simulation Parameters [8], [37]

Parameters	Values
Local iterations (I)	5
Total uplink bandwidth (W)	20 MHz
Noice power (σ^2)	-104 dBm
Channel power gain at 1 m reference distance (h)	-34 dBm
Uplink power (p)	30 dBm
Path loss exponent (ν)	2
CPU-cycle frequency (f^{CPU})	6 GHz
CPU frequency for processing 1 bit of data (q)	10^3 cycles/bit
The effective switching capacitance (λ)	10^{-27}
Communication radius (d^U)	100 m
The number of vehicles (H)	30
Maximum time duration (t^{round})	30 s
The number of subcarriers (N^S)	60
Initial vehicle energy	3,000 J
Distance scaling factor (ξ)	1
Weighting coefficient (ε)	1
Training episodes (K_s)	100
Steps per training episode (T_s)	4,000
Clip range (ϵ)	0.2
Discount factor (γ)	0.99

performance. Note that under the same experimental setup, different algorithms yield different strategies. Therefore, the value of K is not fixed across different scenarios or algorithms. All experiments are conducted using TensorFlow and PyTorch [47] with Ubuntu 22.04. The parameter settings are shown in Table I.

We compare the proposed algorithm with the following algorithms.

- Equal resource allocation (ERA): ERA distributes vehicles evenly across the tasks, randomly designates a leader within the vehicle group of each task, and then evenly allocates subcarriers to the vehicles within the group.
- Bayesian optimization-based device scheduling (BODS) [27]: BODS applies Bayesian optimization for assigning vehicles to the tasks, while the leader selection and bandwidth allocation strategies are the same as in ERA.
- Distributed proximal policy optimization (DPPO): DPPO uses deep reinforcement learning to jointly optimize task scheduling, leader selection, and bandwidth allocation; this is a modified version of [18].
- Joint client selection and bandwidth allocation optimization (CSBWA) [48]: CSBWA employs a reinforce-based DRL algorithm for vehicle assignment and allocates bandwidth according to the policy network's probability

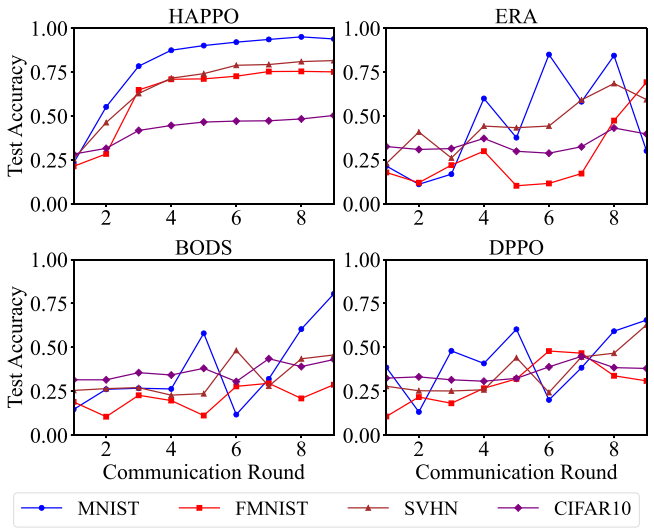


Fig. 6. Accuracy curves for HAPPO, ERA, BODS, and DPPO.

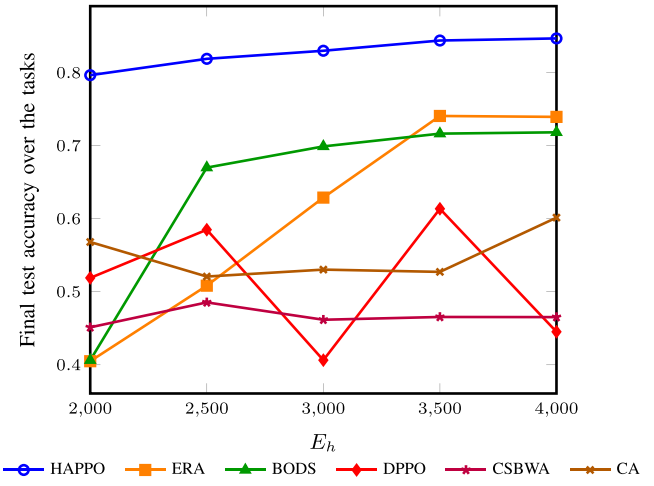


Fig. 7. Comparison of final test accuracy with respect to initial vehicle energy level.

distribution. Furthermore, we align its leader selection strategy with ERA.

- Clustering algorithm (CA) [16]: The CA algorithm employs neighborhood density and cosine similarity for cluster partitioning and leader selection. Furthermore, the bandwidth allocation strategy is aligned with ERA.

B. Comparison With Respect to Various Settings

Fig. 6 shows the accuracy curves for each task when the algorithms are training on all four tasks simultaneously. We gain the following insights. First, the accuracy curve of HAPPO is relatively smooth, indicating that HAPPO can effectively alleviate the issue of model updates being overwritten. Secondly, the final test accuracy of HAPPO is higher, surpassing the other algorithms by at least 51%. This suggests that overwriting model updates indeed can lead to inefficient resource utilization, resulting in a decrease in the final accuracy. Lastly, the final test accuracy of the four tasks in HAPPO is the highest. This indicates that HAPPO can utilize resources more efficiently for training.

Fig. 7 and the first section of Table II show the impact of the initial energy level E_h on the algorithms. We set up three tasks: MNIST with LeNet, FMNIST with LeNet, and SVHN with LeNet. We gain the following insights. First, HAPPO consistently achieves higher final test accuracy across all initial vehicle energy levels, outperforming other algorithms by at least 13%. In particular, HAPPO outperforms CSBWA by 76% when $E_h = 2000\,\text{J}$, and exceeds CA by 40% when $E_h = 4000\,\text{J}$. This is because the issue of overwriting continues to affect the entire training process, while HAPPO effectively alleviates this problem, resulting in better performance. Fig. 8 provides a case illustration of the model update overwriting issue. For task 1, the SOV that participated in training during the $(k-1)$-th communication round is selected as the CHV for the k-th round,

TABLE II
COMPARISON OF MAXIMUM TIME DURATION OVER ALL COMMUNICATION ROUNDS WITH RESPECT TO VARIOUS SCENARIOS (FAILURES IN ITALICS)

Parameter	$\max_{k\in\mathcal{K}} T_k$ (s)			
	HAPPO	ERA	BODS	DPPO
$E_h = 2,000$	23.8807	24.1694	**20.9445**	*69.3173*
$E_h = 2,500$	23.8807	24.1694	**20.9445**	*69.3173*
$E_h = 3,000$	**19.3659**	24.1694	20.9445	*35.4563*
$E_h = 3,500$	23.3918	24.1694	**20.9445**	28.6842
$E_h = 4,000$	23.3918	24.1694	**20.9445**	*31.4053*
$H = 20$	15.2720	13.6801	**11.4227**	24.9670
$H = 25$	21.0747	18.8449	**15.4588**	29.0032
$H = 30$	**19.3659**	24.1694	20.9445	*69.3173*
$H = 35$	**23.8046**	23.9414	23.9414	*31.2186*
$H = 40$	**23.7705**	28.2853	28.2853	*68.9185*
Task group 1	**27.1197**	*46.4546*	*46.4546*	*91.6026*
Task group 2	**19.3659**	23.8807	23.8807	*46.4546*
Task group 3	**19.3659**	24.1694	20.9445	*46.7433*
Task group 4	19.3659	15.1398	**12.8824**	28.5110
Task group 5	20.7714	12.8824	**10.6250**	*69.1442*
$\xi = 1.0$	**19.3659**	24.1694	20.9445	*31.4053*
$\xi = 1.5$	23.9227	27.1662	**23.5132**	27.1662
$\xi = 2.0$	**25.7886**	29.8208	**25.7886**	*62.9792*
$\xi = 2.5$	**25.8602**	*32.2873*	27.9027	*42.2292*
$\xi = 3.0$	**27.7431**	*34.6409*	29.9201	*100.7320*

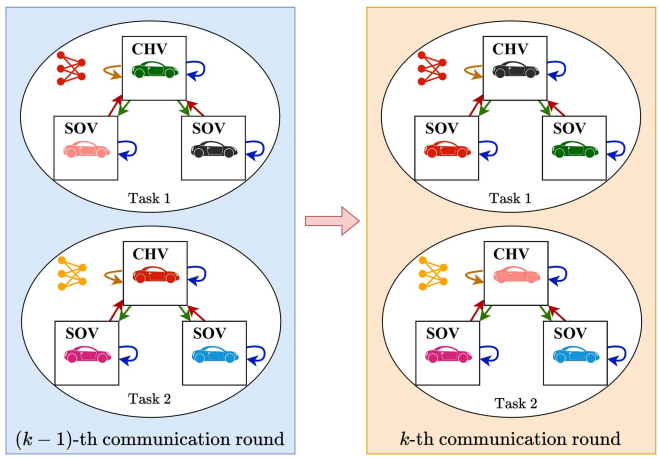


Fig. 8. Case illustration of the overwriting issue.

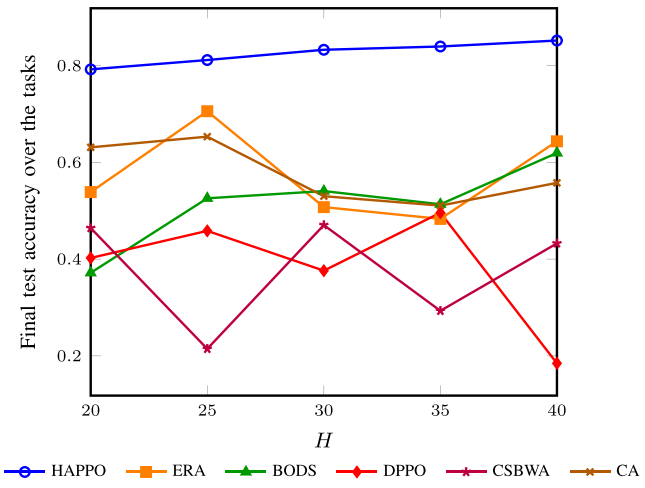


Fig. 9. Comparison of final test accuracy with respect to the number of vehicles.

and the most recently updated model is correctly distributed within the task group. In contrast, the CHV for task 2 in the k-th communication round did not participate in training during the $(k-1)$-th communication round. The CHV distributed an outdated model, causing the most recent update to be overwritten and resulting in a decrease in the overall training performance. Secondly, as the initial energy of the vehicles increases, the final test accuracy of ERA and BODS also improves. This is because, as the training process progresses, more vehicles in the framework receive the updated models, which helps to alleviate the issue of overwriting to some extent. Lastly, DPPO encounters timeout issues frequently. This is because the large solution space of DPPO leads to incorrect resource allocation strategies, resulting in excessively long transmission times.

Fig. 9 and the second section of Table II show the impact of the number of vehicles H on the algorithms. We set up three tasks: MNIST with LeNet, FMNIST with LeNet, and SVHN with LeNet. We gain the following insights. First, the final test accuracy of HAPPO is consistently higher across all numbers of vehicles, outperforming other algorithms by at least 14%. In particular, when $H = 20$, HAPPO surpasses CSBWA by 70%, and when $H = 25$, it exceeds CA by 24%. The increase in the number of vehicles implies an expansion of the solution space, and HAPPO's multi-agent sequential updates can effectively utilize this situation. Second, the final test accuracy of ERA, BODS, and DPPO exhibits significant fluctuations when varying the number of vehicles. This is due to the expansion of the solution space, which exacerbates the interference of model updates. Notably, when $H \geq 30$, DPPO experiences timeout. Lastly, HAPPO demonstrates superior performance in managing the time limit, even as the number of vehicles increases, with at least a 16% reduction compared to ERA and BODS when $H = 40$. This is because, as the number of vehicles grows, the strategy of uniformly allocating subcarriers becomes less effective. Efficient subcarrier allocation based on communication status is needed. The proposed algorithm efficiently allocates subcarriers, enabling the vehicles to participate in training while avoiding timeout.

TABLE III
COMPARISON OF FINAL TEST ACCURACY WITH RESPECT TO DIFFERENT TASK GROUPS

Task group	Dataset	Final test accuracy			
		HAPPO	ERA	BODS	DPPO
1	MNIST	**0.9548**	0.3024	0.3298	0.4773
2	MNIST	**0.9624**	0.7728	0.8705	0.6920
	FMNIST	**0.7369**	0.6133	0.7209	0.1784
3	MNIST	**0.9524**	0.3176	0.8591	0.8956
	FMNIST	**0.7707**	0.6100	0.1227	0.1213
	SVHN	**0.7682**	0.6085	0.6176	0.6022
4	MNIST	**0.9388**	0.3015	0.8047	0.6553
	FMNIST	**0.7514**	0.6925	0.2873	0.3093
	SVHN	**0.8158**	0.5934	0.4569	0.6279
	CIFAR-10	**0.5032**	0.3977	0.4307	0.3800
5	MNIST	**0.9336**	0.1333	0.8792	0.2813
	FMNIST	**0.7220**	0.2293	0.3016	0.4800
	SVHN	**0.7865**	0.4690	0.2323	0.7385
	CIFAR-10	**0.5180**	0.4620	0.4040	0.3927
	CIFAR-10	**0.5107**	0.2926	0.4168	0.3793

Table III and the third section of Table II show the impact of the task group on the algorithms. Please note that in Task group 5, there are two tasks involving CIFAR-10 with ResNet-18, and these two tasks are trained independently. We gain the following insights. Firstly, HAPPO outperforms the other algorithms in terms of accuracy across all task combinations. By dynamically adjusting the action space, HAPPO is particularly well-suited for multi-task FL, effectively handling different task combinations. Secondly, ERA, BODS, and DPPO fail when $M = 1$, while HAPPO exhibits strong performance. This is because when $M = 1$, the task scheduling problem reduces to a client selection problem. ERA and BODS are unable to effectively determine which clients should participate, resulting in excessive bandwidth pressure. DPPO continues to experience timeout issues due to the large solution space. Lastly, when $M \leq 3$, HAPPO requires less time within a round. As the number of tasks increases, this time required by ERA and BODS also decreases, but their final test accuracy does not improve steadily. When the number of tasks is small, each task experiences higher bandwidth pressure. The proposed algorithm effectively allocates subcarriers to mitigate this pressure. As the number of tasks increases, bandwidth pressure decreases, making leader selection management important for mitigating the issue of overwriting.

Fig. 10 and the fourth section of Table II show the impact of the distance scaling factor ξ. We set up three tasks: MNIST with LeNet, FMNIST with LeNet, and SVHN with LeNet. We gain the following insights. The final test accuracy of HAPPO is highest among all four algorithms, with a margin of at least 53%, though a downward trend is observed. This is because the indirect communication cost increases, leading to a reduction in the number of vehicles participating in the training. In addition, the increase in communication cost is also reflected in T_k. The T_k of HAPPO, ERA, and BODS increase as the communication cost rises. ERA experiences a timeout when $\xi \geq 2.5$.

Fig. 11 shows the results of accuracy on the NGSIM dataset [49]. Here, we select trajectories of 40 vehicles moving along Peachtree Street, marking those without complete trajectory data during the specified time period as unavailable. Each vehicle was randomly assigned a CPU frequency ratio

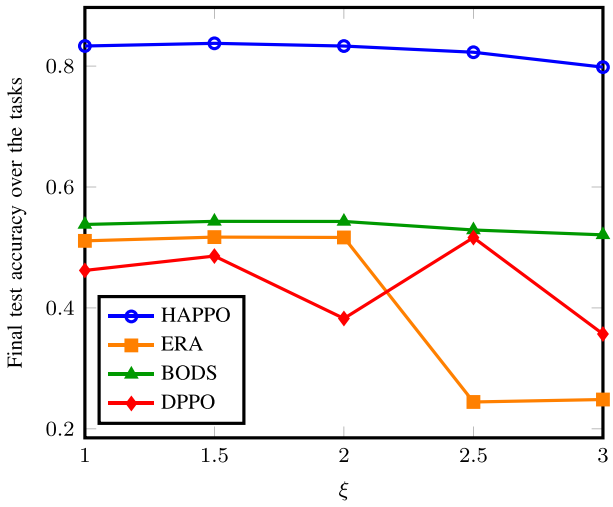


Fig. 10. Comparison of final test accuracy with respect to the distance scaling factor.

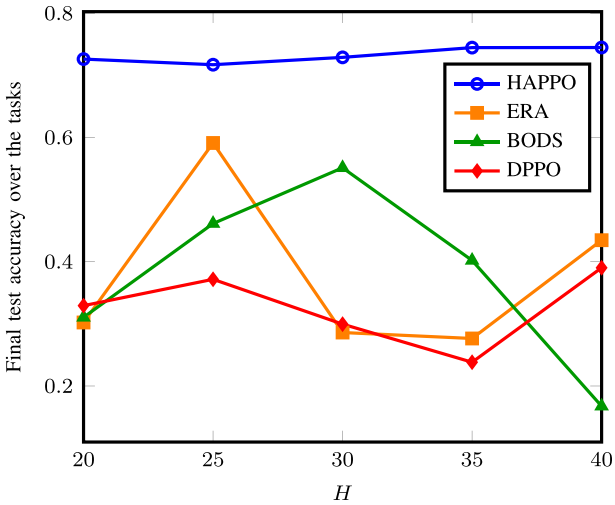


Fig. 11. Comparison of final test accuracy with respect to the number of vehicles on the NGSIM Dataset.

uniformly distributed within $[0.8, 1.2]$ to simulate heterogeneous computing capabilities. If the communication round is fewer than 3, the task set includes MNIST and FMNIST. Starting from round 3, SVHN is added to form a combination of MNIST, FMNIST, and SVHN. It can be observed that HAPPO outperforms the other algorithms, achieving at least 23% higher final test accuracy, demonstrating its efficacy on real-world vehicle trajectory datasets. In addition, the decision-making of HAPPO at $H \geq 30$ leads to improved performance. This suggests that as the number of vehicles increases, the solution space of the optimization problem becomes more complex, while also providing more options for vehicle assignment. Vehicular networks involve heterogeneous devices, unpredictable mobility patterns, and varying network conditions, all of which can significantly affect model performance in real-world scenarios. Our proposed HAPPO algorithm addresses these challenges through the following strategies. Firstly, HAPPO performs effective task

TABLE IV
COMPARISON OF OVERALL COMMUNICATION ENERGY CONSUMPTION WITH RESPECT TO VARIOUS SCENARIOS (FAILURES IN ITALICS)

Parameter	Overall communication energy consumption (J)			
	HAPPO	ERA	BODS	DPPO
$E_h = 2,000$	86653.29	79494.31	88374.03	*85936.55*
$E_h = 2,500$	88257.83	79257.90	93022.25	*88250.00*
$E_h = 3,000$	82907.13	84096.31	82679.77	*69461.99*
$E_h = 3,500$	81587.83	78998.06	87240.34	76887.47
$E_h = 4,000$	81587.83	78998.06	87240.34	*48518.74*
$H = 20$	56391.49	56027.46	65250.44	62349.46
$H = 25$	73439.24	76769.49	75834.04	74222.47
$H = 30$	82907.13	79257.90	88204.82	*81649.85*
$H = 35$	95898.57	91697.52	100765.54	*45637.66*
$H = 40$	99756.81	95429.31	93526.64	*25928.24*
Task group 1	71715.66	*10823.60*	*10846.67*	*16664.91*
Task group 2	76263.22	86383.57	75949.84	*38972.93*
Task group 3	82915.66	79268.42	88214.29	*72922.88*
Task group 4	84768.06	87013.49	85974.11	83319.79
Task group 5	86537.71	86419.30	94889.82	*93856.69*
$\xi = 1.0$	82915.66	79268.42	88214.29	*83723.56*
$\xi = 1.5$	85438.58	79990.51	88973.56	73396.31
$\xi = 2.0$	83283.65	80630.14	89646.12	*46312.04*
$\xi = 2.5$	81405.18	*10123.14*	80212.36	*42744.83*
$\xi = 3.0$	82378.02	*10189.27*	80736.12	*51977.01*

allocation and efficiently utilizes heterogeneous computing resources, thereby improving the overall performance. In addition, the decentralized nature of HAPPO enables each vehicle to operate independently based on its local observations, reducing the impact of mobility on system performance and ensuring the algorithm remains effective even when vehicles move in and out of communication range. Finally, HAPPO achieves efficient allocation of leaders and bandwidth resources, ensuring robust performance under dynamic network conditions.

Table IV presents the communication overhead of the four algorithms with respect to initial vehicle energy level E_h, number of vehicles H, task group, and distance scaling factor ξ. We gain the following insights. Firstly, the communication overhead of HAPPO is comparable to the baseline algorithms across various scenarios. This indicates that HAPPO achieves efficient FL with minimal communication cost by appropriately adjusting the optimization variables. Secondly, as E_h increases, the overall communication energy consumption of HAPPO decreases. This indicates that HAPPO is capable of making decisions in respect of vehicle resource limitations. Finally, when the baseline algorithms fail, training is prematurely halted, resulting in lower overall communication consumption. This suggests that the solution space of the optimization problem is sparse, and only through reasonable decision-making can vehicle resources be fully utilized for training.

C. Generalization Performance of HAPPO

To validate the generalization performance of the proposed algorithm in the presence of gradient update bias, we conduct experiments under the non-independent and identically distributed (Non-IID) settings. Following the setup in [50], the data of each class is distributed across 30 clients according to a Dirichlet distribution, with the concentration parameter set to 0.3. Fig. 12 shows the experimental results. We gain the following insights. First, HAPPO demonstrates superior performance, indicating

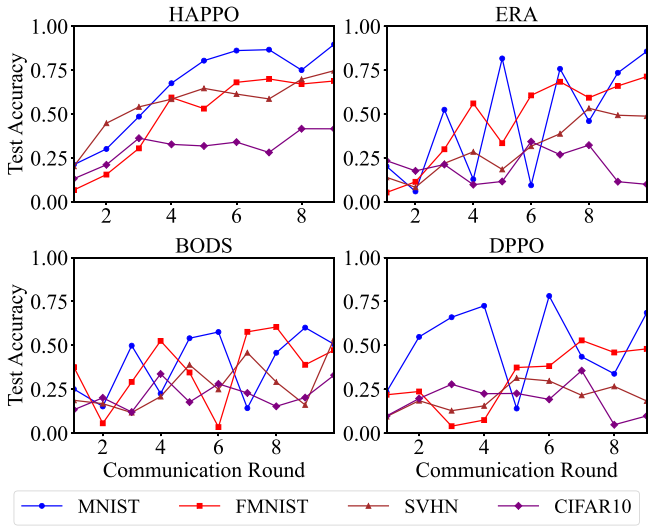


Fig. 12. Accuracy curves under Non-IID for HAPPO, ERA, BODS, and DPPO.

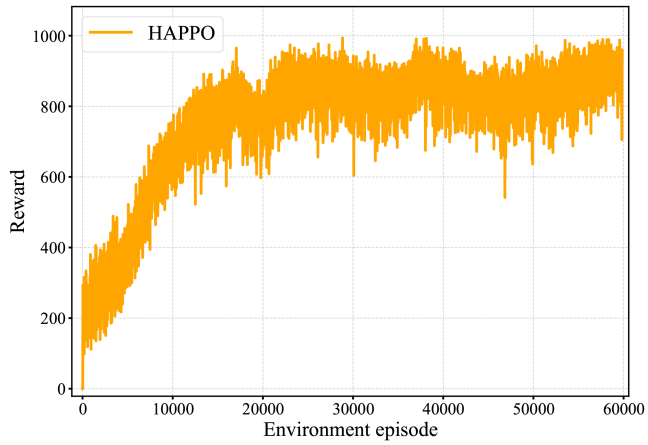


Fig. 13. Cumulative reward per environment episode across full training episodes. The experiments are conducted on three tasks: MNIST with LeNet, FMNIST with LeNet, and SVHN with LeNet.

that the proposed algorithm does possess generalization ability in scenarios with data heterogeneity. In addition, under data heterogeneity scenarios, the final test accuracy of HAPPO decreases by 9% and there are fluctuations in accuracy during the training process. This indicates that the MMFL framework is affected by inconsistent data distribution like other FL frameworks.

D. Convergence Performance of HAPPO

Fig. 13 shows the reward over environment episodes during the training process of HAPPO. We define the environment episode reward as the cumulative undiscounted reward obtained from environment reset to termination. As can be observed, the reward gradually increases during the initial 2×10^4 environment episodes, which clearly indicates that our model is effectively optimizing the FL strategy. Following this period of improvement, the reward stabilizes, suggesting that the algorithm is nearing convergence.

VII. Conclusion

In this paper, we have proposed a mobility-aware multi-task decentralized federated learning framework for vehicular networks. Considering the impact of mobility and resource constraints, we have presented a joint optimization problem for task scheduling, subcarrier allocation, and leader selection. For problem solving, we analyze the convergence bound of a single FL task, and then model multiple FL tasks as a resource allocation game. The game is further reformulated as a DEC-POMDP, and we propose an HAPPO-based algorithm to solve it. We compare the proposed algorithm with baselines under varying initial vehicle energy, number of vehicles, number of tasks, and indirect communication cost. We obtain the following three main insights. First, the experimental results show that the proposed algorithm improves the final average accuracy by at least 13%. Second, for all the experiments, the proposed algorithm efficiently completes the training, while ERA and BODS both encounter time-out in some cases. These experimental results demonstrate that the proposed algorithm can stably solve the TSLP problem. Finally, the proposed algorithm exhibits more stable convergence compared to DPPO. These experimental results indicate that the proposed algorithm can search for the NE solution more efficiently.

References

[1] G. Yan, K. Liu, C. Liu, and J. Zhang, "Edge intelligence for Internet of Vehicles: A survey," *IEEE Trans. Consum. Electron.*, vol. 70, no. 2, pp. 4858–4877, May 2024.

[2] S. Zhang et al., "Federated learning in intelligent transportation systems: Recent applications and open problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 5, pp. 3259–3285, May 2024.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Proc. Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[4] X. Deng et al., "A review of 6G autonomous intelligent transportation systems: Mechanisms, applications and challenges," *J. Syst. Archit*, vol. 142, 2023, Art. no. 102929.

[5] T. Zeng et al., "Multi-task federated learning for traffic prediction and its application to route planning," in *Proc. IEEE Intell. Veh. Symp.*, 2021, pp. 451–457.

[6] B. Xie et al., "MOB-FL: Mobility-aware federated learning for intelligent connected vehicles," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 3951–3957.

[7] X. Zhang, Z. Chang, T. Hu, W. Chen, X. Zhang, and G. Min, "Vehicle selection and resource allocation for federated learning-assisted vehicular network," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 3817–3829, May 2024.

[8] X. Zhang, W. Chen, H. Zhao, Z. Chang, and Z. Han, "Joint accuracy and latency optimization for quantized federated learning in vehicular networks," *IEEE Internet Things J.*, vol. 11, no. 17, pp. 28876–28890, Sep. 2024.

[9] J. Zhang, S. Chen, X. Zhou, X. Wang, and Y.-B. Lin, "Joint scheduling of participants, local iterations, and radio resources for fair federated learning over mobile edge networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 7, pp. 3985–3999, Jul. 2023.

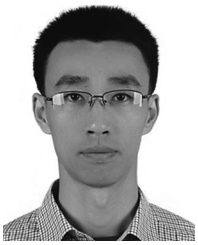
[10] J. Yan, T. Chen, Y. Sun, Z. Nan, S. Zhou, and Z. Niu, "Dynamic scheduling for vehicle-to-vehicle communications enhanced federated learning," *IEEE Trans. Wirel. Commun.*, early access, Jun. 02, 2025, doi: 10.1109/TWC.2025.3573048.

[11] T. Chen, J. Yan, Y. Sun, S. Zhou, D. Gündüz, and Z. Niu, "Mobility accelerates learning: Convergence analysis on hierarchical federated learning in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 74, no. 1, pp. 1657–1673, Jun. 2025.
[12] T. Xiang et al., "Federated learning with dynamic epoch adjustment and collaborative training in mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 4092–4106, May 2024.
[13] L. Zhao, M. Valero, S. Pouriyeh, F. Li, L. Guo, and Z. Han, "A decentralized communication-efficient federated analytics framework for connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 10856–10861, Jul. 2024.
[14] H. He, J. Du, C. Jiang, J. Wang, J. Song, and Z. Han, "Mobility-aware decentralized federated learning for autonomous underwater vehicles," *IEEE Trans. Wirel. Commun.*, vol. 24, no. 8, pp. 7046–7061, Aug. 2025, doi: 10.1109/TWC.2025.3557803.
[15] N. Masmoudi and W. Jaafar, "OCD-FL: A novel communication-efficient peer selection-based decentralized federated learning," *IEEE Trans. Veh. Technol.*, vol. 74, no. 4, pp. 6856–6861, Apr. 2025.
[16] G. Tan, H. Yuan, H. Hu, S. Zhou, and Z. Zhang, "A framework of decentralized federated learning with soft clustering and 1-bit compressed sensing for vehicular networks," *IEEE Internet Things J.*, vol. 11, no. 13, pp. 23617–23629, Jul. 2024.
[17] H. Yang et al., "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nat. Commun.*, vol. 13, no. 1, 2022, Art. no. 4269.
[18] H. Zhang, H. Zhao, R. Liu, X. Gao, and S. Xu, "Leader federated learning optimization using deep reinforcement learning for distributed satellite edge intelligence," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2544–2557, Sep./Oct. 2024.
[19] Z. A. E. Houda, H. Moudoud, and B. Brik, "Federated deep reinforcement learning for efficient jamming attack mitigation in O-RAN," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 9334–9343, Jul. 2024.
[20] Z. Abou El Houda, H. Moudoud, B. Brik, and L. Khoukhi, "Blockchain-enabled federated learning for enhanced collaborative intrusion detection in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 7, pp. 7661–7672, Jul. 2024.
[21] Z. A. E. Houda, D. Naboulsi, and G. Kaddoum, "A privacy-preserving collaborative jamming attacks detection framework using federated learning," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12153–12164, Apr. 2024.
[22] Z. A. El Houda, H. Moudoud, B. Brik, and L. Khoukhi, "Securing federated learning through blockchain and explainable ai for robust intrusion detection in IoT networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2023, pp. 1–6.
[23] X. Sun, R. Panda, R. Feris, and K. Saenko, "AdaShare: Learning what to share for efficient deep multi-task learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 8728–8740.
[24] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 5824–5836.
[25] W. Zhuang, Y. Wen, L. Lyu, and S. Zhang, "MAS: Towards resource-efficient federated multiple-task learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 23414–23 424.
[26] B. Fu, F. Chen, P. Li, and Z. Su, "Efficient scheduling for multi-job federated learning systems with client sharing," in *Proc. IEEE DASC/PiCom/CBDCom/CyberSciTech*, 2023, pp. 891–898.
[27] C. Zhou et al., "Efficient device scheduling with multi-job federated learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 9971–9979.
[28] X. Cao, T. Ouyang, K. Zhao, Y. Li, and X. Chen, "Efficient multi-task asynchronous federated learning in edge computing," in *Proc. IEEE/ACM 32nd Int. Symp. Qual. Service*, 2024, pp. 1–10.
[29] T. Li, Z. Wei, H. Liu, Z. Lin, and T.-T. Chan, "Joint job assignment and resource allocation for multi-job wireless federated learning," in *Proc. IEEE 21st Int. Conf. Mobile Ad-Hoc Smart Syst.*, 2024, pp. 419–427.
[30] X. Wei, K. Ye, X. Shi, C.-Z. Xu, and Y. Wang, "Joint participant and learning topology selection for federated learning in edge clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 8, pp. 1456–1468, Aug. 2024.
[31] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020.
[32] H. Zhou, H. Wang, Z. Yu, G. Bin, M. Xiao, and J. Wu, "Federated distributed deep reinforcement learning for recommendation-enabled edge caching," *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3640–3656, Nov./Dec. 2024.
[33] M. F. Pervej, R. Jin, and H. Dai, "Resource constrained vehicular edge federated learning with highly mobile connected vehicles," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 6, pp. 1825–1844, Jun. 2023.
[34] X. Chang, M. S. Obaidat, J. Ma, X. Xue, Y. Yu, and X. Wu, "Efficient federated learning via adaptive model pruning for internet of vehicles with a constrained latency," *IEEE Trans. Sustain. Comput.*, vol. 10, no. 2, pp. 300–316, Mar./Apr. 2025.
[35] C. Li, C. Li, Y. Zhao, B. Zhang, and C. Li, "An efficient multi-model training algorithm for federated learning," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.
[36] Y. Chen, J. Zhao, Y. Wu, J. Huang, and X. Shen, "QoE-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach," *IEEE Trans. Mob. Comput.*, vol. 23, no. 1, pp. 769–784, Jan. 2024.
[37] T. Yu, X. Wang, J. Hu, and J. Yang, "Multi-agent proximal policy optimization-based dynamic client selection for federated AI in 6G-oriented Internet of Vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 9, pp. 13611–13624, Sep. 2024.
[38] Y. Zhong, J. G. Kuba, X. Feng, S. Hu, J. Ji, and Y. Yang, "Heterogeneous-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 25, no. 1-67, 2024, Art. no. 1.
[39] J. G. Kuba et al., "Trust region policy optimisation in multi-agent reinforcement learning," 2021, *arXiv:2109.11251*.
[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv: 1707.06347*.
[41] W. Liu, Y. Fu, Y. Guo, F. Lee Wang, W. Sun, and Y. Zhang, "Two-timescale synchronization and migration for digital twin networks: A multi-agent deep reinforcement learning approach," *IEEE Trans. Wirel. Commun.*, vol. 23, no. 11, pp. 17294–17309, Nov. 2024.
[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[43] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.
[44] Y. Netzer et al., "Reading digits in natural images with unsupervised feature learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2011, Art. no. 4.
[45] A. Krizhevsky et al., "Learning multiple layers of features from tiny images," Master's Thesiss, Univ. Tront, Toronto, ON, USA, 2009.
[46] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.
[47] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.
[48] W. Mao, X. Lu, Y. Jiang, and H. Zheng, "Joint client selection and bandwidth allocation of wireless federated learning by deep reinforcement learning," *IEEE Trans. Serv. Comput.*, vol. 17, no. 1, pp. 336–348, Jan./Feb. 2024.
[49] E. Zlotchenko, "Next generation simulation (NGSIM) open data," Mar. 2025 [Online]. Available: https://catalog.data.gov/dataset/next-generation-simulation-ngsim-open-data
[50] J. Kim, G. Kim, and B. Han, "Multi-level branched regularization for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 11058–110 73.

Dongyu Chen received the BE degree in software engineering from Jiangsu University, Zhenjiang, China, in 2023. He is currently working toward the MS degree with the School of Computer Science and Technology, Soochow University, Suzhou, China. His research interests include federated learning, vehicular networks, mobile computing, and reinforcement learning.

Tao Deng received the PhD degree from Southwest Jiaotong University, Chengdu, China, in 2019. He is currently a lecturer with the School of Computer Science and Technology, Soochow University, Suzhou, China. From 2019 to 2022, he worked as a research engineer with Huawei Technologies Company Ltd., Shanghai, China. From 2016 to 2018, he was a visiting Ph.D. student with the Department of Information Technology, Uppsala University, Uppsala, Sweden. His research interests include machine learning, edge intelligence, and large-scale optimization.

He Huang (Senior Member, IEEE) received the PhD degree from the School of Computer Science and Technology, University of Science and Technology of China (USTC), China, in 2011. He is currently a distinguished professor with the School of Computer Science and Technology, Soochow University, P. R. China. From 2019 to 2020, he was a visiting research scholar with Florida University, Gainesville, USA. He has authored more than 160 papers in related international conference proceedings and journals. His current research interests include traffic measurement, computer networks, and algorithmic game theory. He is the member of the Association for Computing Machinery (ACM). He received the best paper awards from ISPA 2023, Bigcom 2016, and IEEE MSN 2018. He has served as the Technical Program Committee Member of several conferences, including IEEE INFOCOM, IEEE MASS, IEEE ICC, and IEEE Globecom.



Juncheng Jia received the BEng degree in computer science and technology from Zhejiang University, in 2005, and the PhD degree in computer science from the Hong Kong University of Science and Technology, in 2009. He worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is currently working as an associate professor with the School of Computer Science and Technology, Soochow University, China. His research interests include edge intelligence and federated learning.

Mianxiong Dong received BS, MS, and PhD degrees in computer science and engineering from the University of Aizu, Japan. He is the vice president and professor of Muroran Institute of Technology, Japan. He is the recipient of The 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017, Funai Research Award 2018, 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, SUEMATSU-Yasuharu Award from IEICE in 2021, IEEE TCSC Middle Career Award in 2021. He is Clarivate Analytics 2019, 2021, 2022, 2023 Highly Cited Researcher (Web of Science) and Foreign fellow of EAJ.

Di Yuan (Senior Member, IEEE) received the MSc degree in computer science and engineering, and the PhD degree in optimization from the Linköping Institute of Technology, Linköping, Sweden, in 1996 and 2001, respectively. He was an associate professor and then a full professor with the Department of Science and Technology, Linköping University. In 2016, he joined Uppsala University, as a chair professor. He has been a guest professor with the Technical University of Milan (Politecnico di Milano), Milan, Italy, since 2008, and a senior visiting scientist with Ranplan Wireless Network Design Ltd., U.K., in 2009 and 2012. From 2011 to 2013, he was part time with Ericsson Research, Sweden. From 2014 to 2015, he was a visiting Professor with the University of Maryland, College Park, Maryland. He was with the management committee of four European Cooperation in field of scientific and technical research (COST) actions, an Invited Lecturer of the European Network of Excellence EuroNF, and the Principal Investigator of several European FP7 and Horizon 2020 projects. His research interests include network optimization of 4 G and 5 G systems, and capacity optimization of wireless networks. He was the co-recipient of the IEEE ICC'12 Best Paper Award, and the supervisor of the Best Student Journal Paper Award by the IEEE Sweden Joint VT-COM-IT Chapter, in 2014.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985 and the PhD degree in computer science from the University of Houston, in 1990. He is a SUNY Distinguished professor with the State University of New York and a National Distinguished Professor at Hunan University (China). He has authored or co-authored more than 1130 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. Since 2020, he has been among the world's top few most influential scientists in parallel and distributed computing regarding single-year impact (ranked #2) and career-long impact (ranked #4) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023-2024) and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications. He is listed in ScholarGPS Highly Ranked Scholars (2022-2024) and is among the top 0.002% out of over 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic) in 2023. He was a recipient of the 2022-2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a Member of the SUNY Distinguished Academy. He is an AAAS fellow, an AAIA fellow, an ACIS fellow, and an AIIA fellow. He is a member of the European Academy of Sciences and Arts. He is a member of Academia Europaea (Academician of the Academy of Europe).