MobiTask: A Federated Learning-Based Task Migration Strategy for Mobile Crowdsensing

Siyuan Yin, Haifeng Jiang, Chaogang Tang, *Member, IEEE*, Shuo Xiao, and Huaming Wu, *Senior Member, IEEE*

***Abstract*—In mobile crowdsensing (MCS), the quality of sensing is often enhanced through the rational allocation of tasks in the initial phase. However, during task execution, some participants may fail to complete their tasks on time, adversely affecting the overall completion rate. Although existing studies have introduced the concept of task migration, they lack a comprehensive migration strategy. To address this issue, we propose MobiTask, a federated learning (FL)-based task migration strategy for MCS. The proposed strategy consists of two stages: task execution progress prediction and task successor selection. In the task execution progress prediction stage, various factors leading to task delay are considered. A long short-term memory (LSTM) model with an attention mechanism is designed to predict participants' task execution progress. Based on the predictions, participants who are unlikely to complete tasks on time are identified and removed. In the task successor selection stage, a multiagent reinforcement learning (MARL) algorithm incorporating a graph attention network (GAT) is proposed to construct a candidate pool and select appropriate task successors. Additionally, FL is employed to train the models, mitigating the impact of device heterogeneity while ensuring strategy security and stability. Simulation results demonstrate that MobiTask outperforms existing baseline strategies in terms of task completion rate and migration cost, validating its effectiveness and feasibility.**

***Index Terms*—Federated learning (FL), graph attention network (GAN), mobile crowdsensing (MCS), multiagent reinforcement learning (MARL), task migration.**

I. Introduction

With the widespread adoption of mobile devices, such as smartphones and smart wearable devices, along with the rapid development of wireless networks, mobile crowdsensing (MCS) has emerged as a novel distributed sensing and computing paradigm, gradually demonstrating its significant value [1]. MCS leverages the extensive distribution and powerful sensing capabilities of mobile devices such as smartphones to collect and process environmental information, providing convenient services for societal development. Currently, MCS has broad applications in areas such as social cognition [2], environmental monitoring [3], [4], and transportation [5]. Compared with traditional sensing technologies, MCS offers more significant advantages in terms of cost control, coverage, and flexibility.

MCS systems complete the sensing tasks by recruiting a large number of participants who carry mobile devices to collect data. To improve the quality of sensing services, it is necessary to allocate tasks to participants in a reasonable way in advance. Therefore, task allocation has become a key research topic in MCS [6], which includes multitask allocation [7], online task allocation [8], two-stage task allocation [9], group collaborative task allocation [10], and cross-platform task allocation [11], among others. While effective task allocation can improve sensing quality, it often overlooks subjective awareness and interests of participants. As a result, recent studies have focused on task recommendation [12], [13], [14], where personalized task lists are provided to participants based on their preferences and characteristics, allowing them to select tasks to execute. Compared with traditional task allocation, task recommendation further enhances the quality of task completion.

Although task allocation and recommendation can improve the quality of sensing through rational planning in the initial phase, they are based on offline data of participants for matching or recommendation. Various subjective and objective factors during task execution can lead to the actual task completion rate of MCS falling short of the expected effects of task allocation. For example, several effective task allocation frameworks exhibit a failure rate of approximately 20% [11], [15], [16], which may increase due to unexpected circumstances during task execution. Various unexpected situations, such as participants being passive in task execution or lacking the ability to perform tasks leading to task completion timeouts, can affect the actual effectiveness of task allocation strategy. Additionally, unexpected issues such as insufficient resources on mobile devices, device damage, rugged roads, or traffic congestion can hinder participants' task execution progress. These factors may result in participants failing to complete tasks on time, and task allocation or recommendation strategies cannot anticipate these unforeseen influencing factors in advance.

Currently, there is limited research on issues related to the task execution phase. Jiang et al. [17] proposed a task rescheduling scheme following the initial task allocation to address the

Received 25 March 2025; revised 16 June 2025 and 3 August 2025; accepted 22 August 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62476276 and Grant 62271486. *(Corresponding author: Haifeng Jiang.)*

Siyuan Yin, Haifeng Jiang, Chaogang Tang, and Shuo Xiao are with the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China (e-mail: ts24170055a31@cumt.edu.cn; jhfeng@cumt.edu.cn; cgtang@cumt.edu.cn; sxiao@cumt.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Digital Object Identifier 10.1109/TCSS.2025.3603299

problem where participants are unable to access computing services provided by edge servers during task execution. During task execution, there are also cases where participants fail to provide services properly (e.g., failing to complete tasks on time). To address this issue, Li et al. [18] introduced the concept of task migration for the first time. However, their study only designed conditions for task migration and data inheritance, aiming to enhance the robustness of initial task allocation algorithms. It did not account for the various reasons why participants may fail to complete tasks on time, nor did it provide a comprehensive solution for the task migration process. In contrast to the numerous established strategies available for task allocation [15], [19], [20], no studies have yet conducted an in-depth analysis of the various scenarios where participants fail to complete tasks on time during execution or proposed a complete solution to address these issues.

To address the aforementioned challenges, this article considers various scenarios in which participants fail to complete tasks on time during execution and proposes a comprehensive task migration strategy for MCS systems, named MobiTask. MobiTask introduces a task migration strategy that includes task execution progress prediction and task successor selection. First, by predicting the task execution progress of numerous participants and considering task deadlines, the participant set is divided into two subsets: those who can complete the task on time and those who cannot. Participants who cannot complete the task on time are eliminated and their tasks are migrated. Several successors are then selected from the pool of candidates who can complete the task on time, ensuring timely completion. In the task execution progress prediction stage, this article proposes an LSTM-attention model, enhanced by personalized federated learning (FL) to mitigate the impact of device heterogeneity on prediction accuracy, ensuring high precision in progress prediction. For the successor selection stage, we introduce GAT-QMIX, a multiagent reinforcement learning (MARL) algorithm integrated with GAT. Trained through FL, GAT-QMIX not only ensures a high task completion rate, but also reduces the computational load on cloud servers. Furthermore, an enhancement mechanism is incorporated into FL to improve the security and reliability of the MobiTask strategy. The simulation results demonstrate that the MobiTask strategy can effectively predict participants who are unlikely to complete tasks on time and perform timely task migration, thus improving the task completion rate in MCS systems. Based on the above discussion, the contributions of this article include.

1) By analyzing various situations in which participants fail to complete tasks on time during the task execution stage in MCS, we propose an FL-based task migration strategy, named MobiTask. Compared with existing work, this strategy is the first to design a complete task migration process. It enables tasks that would otherwise fail to be completed on time to be successfully finished through two stages: task execution progress prediction and task successor selection.
2) We introduce a task execution progress prediction model based on LSTM with an attention layer, as well as a MARL method integrated with GAT, to implement the two key stages of task execution progress prediction and task successor selection. FL is employed to train these two models, ensuring the security of the strategy and the stability of the system.
3) The MobiTask is evaluated using datasets such as Geolife and Random dataset. Experimental results demonstrate that the proposed strategy is more effective and stable compared with other baseline methods, particularly in improving task completion rate and reducing migration cost.

The remainder of this article is organized as follows. Section II reviews related work. Section III provides the system model. In Section IV, the design concept and algorithm flow of the MobiTask strategy are presented in detail. Section V conducts experiments to evaluate the performance of the proposed strategy. Finally, Section VI summarizes the whole article.

II. Related Work

The work in this article primarily focuses on the design of the task migration mechanism and the use of FL. Task migration addresses the limitations of task allocation. Consequently, we review the development of task allocation mechanisms and the application of FL in MCS.

A. Task Allocation

A substantial body of research has emerged on task allocation strategies aimed at improving the quality of sensing services in MCS. Yang et al. [21] proposed an improved genetic algorithm (GA) to address the online task allocation problem, where participants arrive dynamically, enabling real-time assignment of an appropriate set of tasks to each incoming participant. Additionally, some studies have combined both offline and online task allocation models and introduced a two-stage task allocation strategy that jointly optimizes task allocation under a total incentive budget constraint [15]. Hu et al. [22] considered the quality of service (QoS)-sensitive nature of MCS and employed a utility function-based greedy algorithm to address the task allocation problem.

These studies primarily utilize traditional mathematical methods and heuristic algorithms for task allocation. However, with the widespread adoption of artificial intelligence techniques, such as deep learning, an increasing number of studies have utilized deep learning-based approaches for task allocation. For instance, Zhao et al. [23] utilized a spatiotemporal attention network to capture the relationships between factors such as workers' location preferences and capabilities, thereby improving the accuracy of task allocation.

The above studies focus on improving sensing quality in the initial phase of MCS by assigning appropriate tasks to participants. However, it lacks strategies to address situations in which participants are unable to complete tasks on time during the task execution phase.

B. FL

FL, as an emerging distributed machine learning paradigm, has shown significant effectiveness in protecting data privacy

TABLE I
SYMBOLS AND DEFINITIONS

Symbol	Definition
N, M	Set of tasks and set of participants in the platform
p_t^m	Task execution progress of participant m at time t
N', M'	Set of tasks waiting to be inherited and set of candidate participants
$\|N'_r\|$	Number of remaining uninherited tasks in set N'
n'	Initial total number of tasks in N'
T_p^j	Time spent by the original participant on task j before migration
T_r^i, $T_e^{i,j}$	The time for participant i to complete its original task and the time to complete its inherited task
$T_m^{i,j}$	Travel time for participant i to reach the location of task j
T_d^j	Deadline of task j
$CostAdd_i$	Incentive cost for participant i
N_M	Number of participants selected as task successors
K', $Cost'$	Number of newly recruited participants and the cost per new participant
B	Remaining budget of the MCS system
T_{ij}	Task-participant type compatibility matrix
H_i, $T_c^{i,j}$	Historical task completion rate of participant i and estimated time to complete task j
d_{ij}	Distance between participant i and task j

and reducing communication costs [24]. Due to their distributed nature, FL and MCS are naturally well-suited for integration. For example, Dongare et al. [25] proposed a federated reinforcement learning algorithm to address the mobile user selection problem in MCS, while Chen et al. [26] used horizontal FL to enhance the security of MCS, effectively preventing server congestion caused by malicious task injection. Other studies, such as Zhang et al. [27], have applied FL to the participant recruitment problem in MCS, significantly improving sensing quality.

These studies demonstrate that the application of FL in ensuring the security of MCS, reducing cloud server load, and minimizing communication costs has become well-established. Therefore, this article adopts FL to make the task migration strategy more secure and efficient.

III. SYSTEM MODEL

In this article, the MCS model is defined as a participatory crowdsensing paradigm, where users accept tasks assigned by the platform and actively collect and upload data. Let the platform have a task set $N = \{N_1, N_2, \ldots, N_n\}$, where each task can be divided into multiple task slices. The execution progress of the task N_i is denoted as δ_i, $\delta_i = C_f^i / C_t^i$. C_t^i represents the total number of slices that task N_i can be divided into. C_f^i represents the number of slices that have been completed for task N_i. All tasks in the set N have been assigned to participants for execution. The platform has a participant set $M = \{M_1, M_2, \ldots, M_m\}$, and each participant can only execute one task at a time. Therefore, the task execution progress for participant M_i is calculated in the same manner as δ_i. We summarize the key symbols used in this article in Table I.

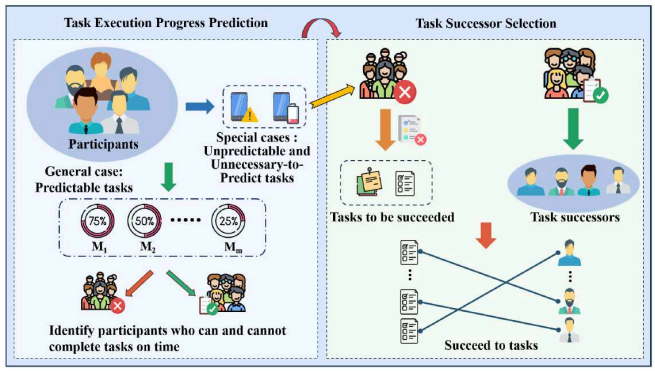

Fig. 1. MobiTask strategy design.

IV. STRATEGY DESIGN

In this section, we will provide a detailed description of the design of the MobiTask strategy. First, in Section IV-A, we will give an overview of the overall process of the MobiTask strategy. Then, in Sections IV-B and IV-C, we will delve into the two key stages of the strategy: task execution progress prediction and task successor selection. Finally, in Section IV-D, we design an FL enhancement mechanism to improve the security and reliability of MobiTask.

A. Strategy Overview

As shown in Fig. 1, MobiTask consists of two stages: task execution progress prediction and task successor selection. During the task execution progress prediction stage, the determination of whether tasks can be completed on time can be categorized into two cases: model-based prediction (predictable tasks) and nonmodel-based prediction (unpredictable and unnecessary-to-predict tasks). In the first case, where participants remain in normal working conditions but their task completion status cannot be directly assessed, predictive models are employed, a methodology thoroughly discussed in Sections IV-B1 to IV-B3 of this article. Additionally, there are two special cases: situations where model-based prediction is either infeasible or unnecessary, which are examined in depth in Section IV-B4. For the general case, the prediction of individual participants' task progress enables the classification of participants into two distinct sets: those unlikely to meet deadlines and those capable of completing tasks on schedule or ahead of time. In the event of special cases, the process directly proceeds to the task successor selection stage without intermediate prediction steps. In the successor selection stage, based on the results of the prediction stage, tasks that cannot be completed on time are migrated out, and the corresponding participants are eliminated. Participants who can complete the task on time are combined with the reserve participant pool of the MCS system to form a candidate participant pool. From this pool, participants are selected to succeed in the migrated tasks. The key issues of these two stages are analyzed as follows.

1) *Task Execution Progress Prediction:* The primary challenges in this stage are how to account for the impact

of various factors on the prediction results when constructing the task execution progress prediction model. Additionally, due to the heterogeneity of participants' terminal devices, centralized training could reduce the accuracy of the model's predictions. Furthermore, it is essential to consider various scenarios in which participants may fail to complete tasks on time, including unexpected events during task execution and foreseeable potential issues. These factors will necessitate a more comprehensive assessment of whether a task should be migrated. To address these challenges, we propose an improved LSTM task progress prediction model with an attention layer and utilize personalized FL for model training. The basic idea and implementation process for this stage will be detailed in Section IV-B.

2) *Task Successor Selection:* In this stage, several appropriate successors for the task are selected based on task progress prediction results, considering relevant factors such as the historical task completion rates of candidate participants. To determine the task successor, we utilize a MARL algorithm enhanced with GAT and utilize FL for model training. The underlying concept and implementation details of this stage will be presented in Section IV-C.

B. Task Execution Progress Prediction Method

According to the task execution progress δ calculation method defined in Section III, each participant's mobile terminal device can calculate the task execution progress at different time points based on local data. The task execution progress of participant m at time point t is defined as p_t^m, and thus participant m can obtain a task execution progress time series $p_t^m = \{p_0^m, p_1^m, \ldots, p_n^m\}$. Therefore, predicting task execution progress can be modeled as a time series prediction problem. Time series prediction problems can be categorized in several ways, including univariate/multivariate, short-term/long-term, and stationary/non-stationary. The task execution progress prediction addressed in this article can be classified as a multivariate, short-term, and stationary time series prediction problem. Currently, there is substantial research on time series prediction. Traditional methods, such as ARIMA [28], are not suitable for complex multivariate time series prediction tasks. In contrast, machine learning methods, such as K-nearest neighbors (KNN) [29] and support vector machines [30], can handle more complex and larger time series prediction problems. However, deep learning methods, such as recurrent neural networks (RNN) [31], long short-term memory (LSTM) [32], outperform traditional methods and machine learning in terms of nonlinear feature extraction and capturing complex temporal dependencies. Among them, LSTM is particularly effective in capturing long-term dependencies in time series, making it well-suited for predicting task execution progress. However, LSTM still has limitations in capturing critical information from the sequence. To address this issue, some studies have integrated attention mechanisms into LSTM, significantly improving prediction performance [33]. Therefore, this article adopts the LSTM-attention mechanism to predict task execution progress.

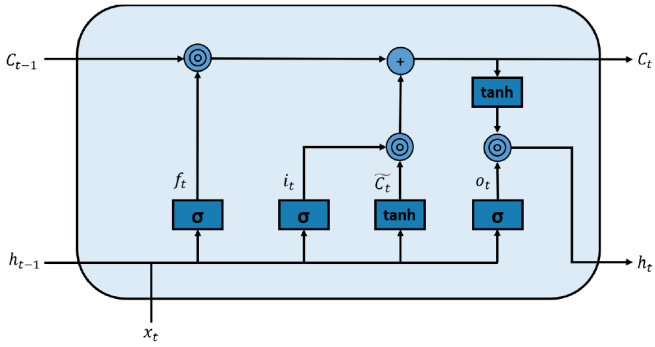

Fig. 2. Structure of memory unit of LSTM.

1) LSTM-Attention Model: LSTM has powerful memory capabilities and the ability to handle long-term dependencies, effectively addressing the limitations of traditional RNNs in modeling long-term dependencies. An independent LSTM unit structure is shown in Fig. 2. Each LSTM unit consists of a forget gate, an input gate, and an output gate, with the following calculation equations:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{4}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{6}$$

where f_t is the output of the forget gate, determining how much of the previous time step's memory cell information is retained. i_t is the output of the input gate, controlling the writing of new information, while C_t is the current time step's memory cell state, and $\tilde{C}_t$ is the candidate memory cell. o_t is the output of the output gate, controlling the hidden state at the current time step, and h_t represents the current hidden state. $\sigma(\cdot)$ is the sigmoid activation function. $W_f, W_i, W_C, W_o, b_f, b_i, b_c, b_o$ correspond to the weight matrices and bias terms for the different gates.

However, the LSTM primarily relies on recursive computation, and in some cases, the model may not effectively focus on the key parts of all input data. To address this issue, this article adds an attention layer after the LSTM layer. This layer computes the similarity between the hidden states, resulting in a weight coefficient α_t that measures the contribution of each time step to the prediction. The calculation process is as follows:

$$\alpha_t = \text{softmax}(w_q^T h_t) \tag{7}$$

where w_q^T represents the trainable weights, and the softmax operation is used to normalize the weights. After obtaining the attention weights for all time steps, the model combines the LSTM outputs into a new context vector through a weighted sum. This context vector is further processed by a dense layer to generate the final predicted output.

By introducing the attention mechanism, the model can dynamically assign different weights to the time steps in the input sequence, allowing it to better focus on the time steps that

Algorithm 1: Task Progress Prediction Model Trained by Personalized Federated Learning.

Input: Set of clients K, number of communication rounds T, number of local epochs E, learning rate η

Output: Global model w_G^T, personalized layer parameters θ_k^T

Initialize: global parameters w_G^0 at the cloud server
for $t = 0$ **to** T **do**
  for each client k in K **do**
    Download global model w_G^t
    Initialize local parameters: $w_k^t = w_G^t, \theta_k^t$ (personalized parameters)
    for $e = 0$ **to** E **do**
      $h_t = \text{LSTM}(w_k^t, x_t)$
      Attention layer computes $\alpha_t = \text{softmax}(w_q^T h_t)$, $c = \sum_{t=1}^{L} \alpha_t h_t$
      Fully connected layer predicts $y_G = \text{Dense}(c)$
      Personalized layer performs prediction $Y = \text{Adapt}(\theta_k^t, y_G)$
      Compute the loss, update local model and personalized layer parameters according to Eq. (9)
    end for
    Upload $w_k^{(t+1)}$ to the server, locally store $\theta_k^{(t+1)}$
  end for
  Cloud server aggregates model parameters according to Eq. (8)
end for
Return: Y, w_G^T, θ_k^T

contribute the most to the current prediction. This mechanism enables the LSTM to more accurately capture important features in the time series without significantly increasing computational complexity.

2) Personalized Federated Training: Due to the heterogeneity of the mobile devices held by participants, traditional centralized training methods may result in a single global model that fails to accurately capture the characteristics of each device, thus reducing prediction accuracy. Additionally, directly uploading raw data to the cloud server poses privacy risks and incurs significant communication overhead. To address these issues, this article adopts personalized FL to train the task progress prediction model. Based on a globally shared LSTM-attention model, a personalized layer is added for each client to capture local-specific patterns. The personalized layer consists of a single-layer dense network, avoiding excessive local computational burden while allowing the model to be personalized. When aggregating the model parameters from each client, only the parameters of the LSTM-attention model are aggregated, and the global model parameters are computed using a weighted average, as shown in the following equation:

$$w_G^t \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} w_k^t \tag{8}$$

where w_G^t represents the global model parameters at the tth communication round, n_k/N is the weight of client k, proportional to the amount of its local data, and w_k^t denotes the model parameters of client k at the tth communication round.

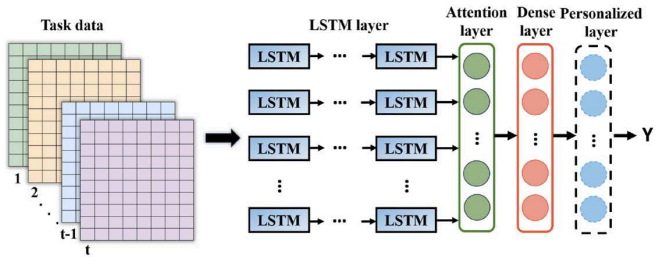

Fig. 3. Architecture of the task execution progress prediction model.

Each client k performs several gradient descent steps on its local dataset to update its local model parameters

$$w_k^{(t+1)} \leftarrow w_k^t - \eta \nabla F_k(w_k^t) \tag{9}$$

where w_k^t represents the model parameters of client k at the tth communication round, η is the learning rate, and $F_k(w_k^t)$ is the local loss function of client k. The parameter updates for the personalized layers follow the same procedure.

The algorithm flow after integrating personalized FL is shown in Algorithm 1, and the structure of the task progress prediction model is shown in Fig. 3.

Moreover, FL requires models to be trained locally on client devices. To ensure the model functions properly on participants' mobile terminals, we must analyze the computational overhead imposed on these devices. Our analysis examines two key metrics: floating-point operations (FLOPs) and memory usage.

a) *FLOPs:* The LSTM-Attention model comprises an LSTM layer, an attention layer, and two dense layers. The LSTM layer primarily performs matrix operations, with its FLOPs calculated as

$$\text{FLOPs}_l = 4 \times (D_l \times N_n + N_n^2) \times S_l \times B_l \tag{10}$$

where FLOPs_l denotes the FLOPs of the LSTM layer, the factor 4 accounts for the four gates in the LSTM, and D_l, N_n, S_l, and B_l represent the feature dimension, number of hidden units, time steps, and batch size of the LSTM-attention model, respectively. The attention layer's computation is more complex, encompassing the generation of query, key, and value vectors, dot-product attention computation, Softmax normalization, and weighted attention context. The FLOPs for the dense layers can be computed as described in the literature [34], using the equation

$$\text{FLOPs}_D = (2I_p - 1) \times O_p \tag{11}$$

where I_p and O_p denote the input and output dimensions of the LSTM-attention model, respectively. Based on this analysis, the total FLOPs for the LSTM-attention model is

$$\text{FLOPs}_p = \text{FLOPs}_l + \text{FLOPs}_a + \text{FLOPs}_D \tag{12}$$

where FLOPs_p represents the total FLOPs of the LSTM-attention model, and FLOPs_a and FLOPs_D denote the

FLOPs of the attention layer and dense layers, respectively. In this study, with parameters set as $D_l = 8$, $N_n = 32$, $S_l = 5$, and $B_l = 32$, the estimated FLOPs for a single forward pass of the LSTM-Attention model is approximately 0.865 MFLOPs. The backward pass typically requires 2–3 times the FLOPs of the forward pass. With a batch size of 32, the total FLOPs_p for the LSTM-Attention model is approximately 83.04 MFLOPs. Given that modern mobile devices (e.g., smartphones) have computational capabilities of up to 20 TFLOPs per second, they fully meet the training requirements of the LSTM-attention model.

b) *Memory Usage:* The memory usage of the LSTM-attention model primarily includes storage for weights and biases, runtime tensors such as activation values and attention scores during forward and backward propagation, and auxiliary data storage. The estimated memory footprint of the LSTM-attention model is approximately 51.5 MB, which is well within the capacity of modern mobile devices, ensuring seamless operation.

3) Input and Output of the Model: The input data for multivariate time series prediction models typically include historical observations and external features that influence the changes in the observed values. In our prediction task, the observed value is the progress of task execution. The external features selected for this study are summarized as follows.

a) *CPU Utilization, Memory Utilization, and Network Latency of the Terminal Device:* These three features affect the efficiency of data collection, data preprocessing, and data uploading on the terminal device, which indirectly impacts the task execution progress.

b) *Participant Movement Speed:* This feature directly influences the task execution progress.

The prediction model uncovers the implicit relationships between these factors to predict task execution progress.

4) Special Cases: The above considerations assume that participants are able to perform tasks normally, and the model is used to predict whether tasks can be completed on time. However, there are two special cases where prediction via the model is either not possible or unnecessary. For example, if a participant suddenly stops performing a task due to an unexpected malfunction of the mobile terminal or other subjective and objective reasons, such cases cannot be predicted by the model. Additionally, if the remaining battery life of the mobile terminal cannot support the timely completion of the task, the task will fail. This situation can be anticipated by retrieving the remaining battery life from the terminal device, and there is no need to build a model for prediction. In our strategy, if either of these two situations occurs, the task immediately enters the task successor selection stage of the strategy. The participant is eliminated, the task is migrated, and an appropriate successor is selected.

C. Task Successor Selection Method

1) Problem Formulation: In this stage, for tasks that cannot be completed on time, their remaining task slices are migrated from the current participant and combined into a new task. Therefore, a new task set $N' = \{N'_1, N'_2, \ldots, N'_{n'}\}$ can be generated from the set of participants who cannot complete tasks on time. Additionally, to avoid the problem of insufficient participants when selecting successors, this article introduces a system reserve participant pool. The set of participants who can complete tasks on time is combined with the reserve participant pool to form a candidate participant pool $M' = \{M'_1, M'_2, \ldots, M'_{m'}\}$. Subsequently, appropriate participants are selected from the candidate participant pool to succeed to the tasks in the new task set N'.

The goal of this article is to maximize the task completion rate (TCR) of the MCS system under certain constraints. Based on the results from the task execution progress prediction, tasks that can be completed on time are not considered, and only the completion rate of the migrated tasks needs to be focused on. Therefore, TCR can be defined as follows:

$$\text{TCR} = 1 - \frac{|N'_r|}{n'} \tag{13}$$

where $|N'_r|$ represents the number of remaining tasks in the task set N' that have not been succeed, and n' is the total initial number of tasks in the task set N'.

The problem is formalized as follows:

$$\max \ \text{TCR} \tag{14}$$

$$\text{s.t. } T_p^j + T_r^i + T_m^{i,j} + T_e^{i,j} \le T_d^j \quad \forall i \in m', j \in n' \tag{15}$$

$$\sum_{i=1}^{N_M} \text{CostAdd}_i + K' \cdot \text{Cost}' \le B \tag{16}$$

where max TCR represents the maximization of the task completion rate. Constraint (15) ensures that the execution and migration process of task i does not exceed its deadline. Constraint (16) ensures that the incentive costs of all selected task successors, combined with the cost of recruiting new participants from the reserve participant pool, do not exceed the migration budget of the MCS system.

The time $T_e^{i,j}$ required by participant i to complete task j is calculated as follows:

$$T_e^{i,j} = (1 - T_{ij} H_i) T_c^{i,j} \tag{17}$$

where T_{ij} is the task-participant type consistency matrix. A value of 1 at position (i, j) indicates that participant j has previously completed a task of the same type as task i, while 0 indicates that the task has not been completed before. In this article, it is assumed that all participants in the system's reserve participant pool satisfy the constraint (15). The cost of selecting a new participant from the pool, denoted as $Cost'$, is set as α times the incentive cost of the participant with the highest incentive cost among those capable of completing the task on time, where $\alpha > 1$.

2) Algorithm Design: Based on the above problem description, task successor selection can be regarded as a matching problem between multiple tasks and multiple participants. Currently, MARL has demonstrated superior performance in many-to-many matching scenarios such as edge computing offloading and drone swarm perception [35], [36], [37]. MARL is a

branch of RL that enables agents to learn optimal policies for solving complex decision-making problems. Currently, several mature MARL frameworks exist, including policy-based approaches such as MADDPG [38] and DAE [39], as well as value-function-based methods such as temporal message control (TMC) [40], value decomposition networks (VDN) [41], and Q-MIXing network (QMIX) [42]. Policy-based frameworks suffer from low sample efficiency and sparse reward issues, making them unsuitable for our task successor selection problem, which involves a large sample space. Among value-function-based approaches, some frameworks rely on inter-agent communication during training, introducing additional overhead and increasing the risk of privacy data leakage. The most widely used communication-free frameworks are VDN and QMIX. However, VDN employs a simplistic linear decomposition method, which can only represent limited joint action-value functions. This severely restricts the expressive power of centralized joint action-value functions and ignores additional information available during training. In contrast, QMIX introduces a mixing network to learn nonlinear relationships between joint and local action value functions while incorporating global state information during training, offering significant advantages over VDN.

For these reasons, this article adopts the QMIX framework, which avoids policy oscillations associated with independent learning, performance degradation in centralized methods, and the overhead of communication-based learning, making it well-suited for the successor selection scenario in MCS. However, in our scenario, both tasks and participants possess complex attributes, making it difficult for traditional QMIX's monotonic mixing network to capture intricate dependencies between them. Recent studies [43] have introduced strategies such as graph attention networks (GAT) to address MCS task allocation problems that can be abstracted as graph structures. GAT effectively captures task-participant dependencies through attention weights.

Since this article employs FL for distributed model training, participants do not share their data with the cloud server or other participants. Consequently, each participant can only access global task information and their own relevant data (e.g., historical task completion rates, geographic coordinates). Additionally, each participant makes decisions independently based on their local observations. Therefore, the task successor selection problem can be modeled as a decentralized partially observable Markov decision process (Dec-POMDP), represented by a seven-tuple $\langle C, S, A, P, R, O, \gamma \rangle$. Here, C denotes the finite set of agents, S represents the finite state space, and A is the joint action space. O is the joint local observation space, where $o_i \in O$ is the local observation of agent i, and agent i executes action $a_i \in A$ based on its local observation o_i. P is the state transition function, R is the reward function, and $\gamma \in [0, 1]$ is the discount factor. We provide specific definitions for the key elements of this seven-tuple as follows.

a) *State Space:* The state space consists of information related to tasks and participants, including the number of remaining tasks in N' at a given time slot t, as well as task-related features, such as the number of task slices, task type, etc. Additionally, it includes the number of candidate participants in M' at time t, and participant-related information, such as task completion rates, incentive responsiveness, etc. Therefore, the state set is defined as $S_t = \{N_t, M_t\}$, where N_t represents the remaining task set at time t, and M_t represents the candidate participant set at time t.

b) *Action Space:* The action of agent i is whether candidate participant i inherits task j in the current state, defined as $a_j \in \{0, 1\}$.

c) *Agents and Joint Action:* In this multiagent task, each candidate participant in the set M' involved in MCS is regarded as an agent, capable of observing all task information and their own attributes. The joint action is the collection of actions from all agents, defined as $A_c = \{a_1, a_2, \ldots, a_{m'}\}$, representing the successor selection results for all tasks.

d) *Observation:* The observation of agent i at time slot t is denoted as $o_t^i = \{o_{t,1}^i, o_{t,2}^i, o_{t,3}^i\}$, where $o_{t,1}^i$, $o_{t,2}^i$, and $o_{t,3}^i$ represent the candidate participant's own state information, global task information, and the remaining budget of the MCS system, respectively.

e) *Reward:* The reward consists of a base reward value and the matching degree between tasks and participants. The base reward value R_b is determined by whether the selected successor satisfies the constraints (15) and (16). If the constraints are satisfied, a positive reward is given; otherwise, a penalty is applied. The matching degree between tasks and participants is determined by the participant's historical task completion rate, the geographical consistency between the task and the participant, and whether the participant has previously completed a task of the same type. The matching degree S_m is calculated as follows:

$$S_m = T_{i,j} + H_i + \frac{1}{d_{i,j}}. \tag{18}$$

The reward function can be set to

$$r = \begin{cases} R_b + S_m, & \text{Satisfy (15), (16)} \\ -R_b, & \text{Not Satisfy (15), (16).} \end{cases} \tag{19}$$

3) Federated Training: Due to QMIX's centralized training and decentralized execution characteristics, when applied to MCS scenarios, both the agents' Q-networks and the mixing network are trained on the cloud server. This requires participants to upload their raw data to the cloud server for model training, which poses a significant risk of privacy leakage and imposes a substantial computational burden on the server. To address these limitations, the MobiTask strategy incorporates FL for model training. In this strategy, models are trained locally on individual devices, with only model parameters being transmitted to the cloud server for aggregation. This approach eliminates the need for raw data sharing, thereby enhancing privacy protection and data security while significantly reducing computational overhead on the server. The complete algorithm workflow is presented in Algorithm 2.

Algorithm 2: Task Successor Selection Algorithm.

Input: Set of clients K, number of communication rounds T, number of local epochs E, learning rate η

Output: Global Q-network $\theta_G^{'}$, TCR

Initialize: The cloud server initializes the global Q-network θ_G and the mixing network ϕ_G.
for $r = 1$ to R **do**
  Each candidate participant client executes in parallel:
    Download the global models θ_G.
    Initialize the local Q-network θ_L.
    Initialize the local experience replay $D \leftarrow \emptyset$.
  for $e = 1$ to E **do**
      Initialize time slot $t = 0$.
    for each $n \in N'$ **do**
        Construct the state $S_t = \{N_t, M_t\}$.
        Select the action a_t using the ϵ-Greedy algorithm.
        Calculate the reward value R_w.
        Store $\langle s_t, a_t, s_{t+1}, R_w \rangle$ into the experience replay D.
    end for
      Upload updated local Q-network θ_L parameters and Q-values to cloud server.
      Compute the global Q-value Q_t using the mixing network.
      Update each agent's Q-network using the global Q-value.
  end for
  Server-Side processing:
    computes global Q-value using mixing network and perform global model aggregation using Equations (8). Finally, update θ_G with Q_t.
    Compute the task completion rate (TCR) for the joint action A_c.
end for
Return: TCR, the final global Q-network $\theta_G^{'}$.

The training process initiates with the cloud server initializing QMIX's mixing network and global Q-network. Each agent (mobile device) downloads the initial Q-network parameters from the cloud server and maintains a local Q-network. During local training, agents observe environmental states as Q-network inputs to compute action-specific Q-values. To avoid local optima, agents employ an ϵ-greedy policy that probabilistically balances exploration and exploitation. The ϵ-greedy strategy selects the current optimal action with probability $1 - \epsilon$ for exploitation, while randomly choosing any valid action with probability ϵ to maintain exploration. Postexecution, the environment returns rewards and new states, with transition tuples (state, action, new state, and reward) stored in experience replay. During parameter updates, agents randomly sample mini-batches from their experience replay to compute target Q-values, then upload both their updated local network parameters and corresponding Q-values to the cloud server. The cloud server aggregates the parameters of all agents using (8) and updates the global Q-network, generating parameters w_G^t. Concurrently, the mixing network is updated using the local Q-values uploaded by all agents. Upon receiving the set of local Q-values, the mixing network computes the global Q-value Q_t, which reflects the collaborative effect among multiple agents. QMIX optimizes the behavior of individual agents by maximizing the global Q-value Q_t.

While conventional QMIX improves upon VDN's linear value decomposition through its mixing network, it remains inadequate for modeling complex interdependencies in MCS scenarios. To enhance the expressive capability of the joint action-value function, the MobiTask strategy integrates a GAT layer into the mixing network of the traditional QMIX framework. In this approach, tasks and participants are modeled as nodes within a graph structure, with edges between task-participant pairs representing potential assignment relationships. By computing attention weights, the GAT prioritizes participant features highly relevant to the task, such as historical task completion rates and geographical location compatibility, thereby improving the overall effectiveness of the joint action-value function. The mixing network updates via TD-error minimization

$$L(\theta) = \sum_{i=1}^{b} \left[\left(y_i^t - Q_t(s, a; \theta) \right)^2 \right] \tag{20}$$

where b is the batch size sampled from the experience replay buffer, $y_i^t = r + \gamma \max_{a'} Q_t(s', a'; \theta^-)$, $Q_t(s', a'; \theta^-)$ is the target network, θ^- represents the target network parameters, and γ is the discount factor.

In the MobiTask strategy, we employ the global Q-value Q_t to refine the federated global Q-network, which serves as a centralized representation of the decentralized local Q-networks maintained by individual agents, effectively embedding global coordination objectives into local decision-making processes. The updated global Q-network parameters are subsequently distributed across the agent population to synchronize local Q-network updates, completing a full cycle of federated training. This approach transforms QMIX's conventional centralized training paradigm into a distributed framework while preserving its core advantage of agent coordination through the mixing network, simultaneously mitigating both privacy leakage risks and computational burdens on cloud servers. The complete architecture of our task successor selection model is illustrated in Fig. 4.

Similar to the task execution progress prediction model, the computational overhead of the task successor selection model on end devices is analyzed from the perspectives of FLOPs and memory usage.

a) *FLOPs:* We need to estimate the computational overhead of the local Q-network running on participants' mobile terminals. This local Q-network, which consists of two dense layers forming a compact neural network, is responsible for making task-matching decisions. The local Q-network, responsible for task matching decisions, consists of two dense layers, forming a small-scale neural

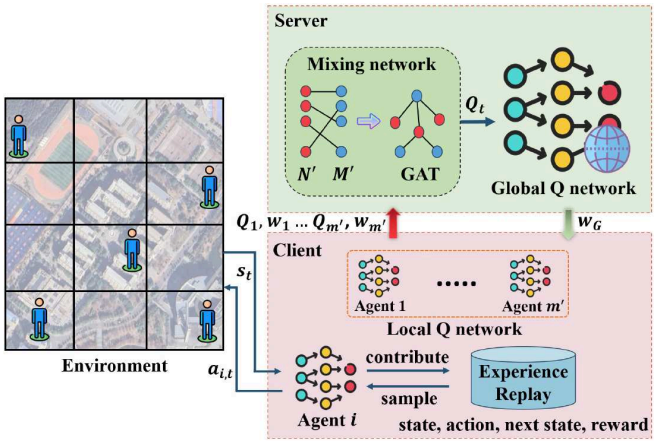

Fig. 4. Architecture of the successor selection model for migrated tasks.

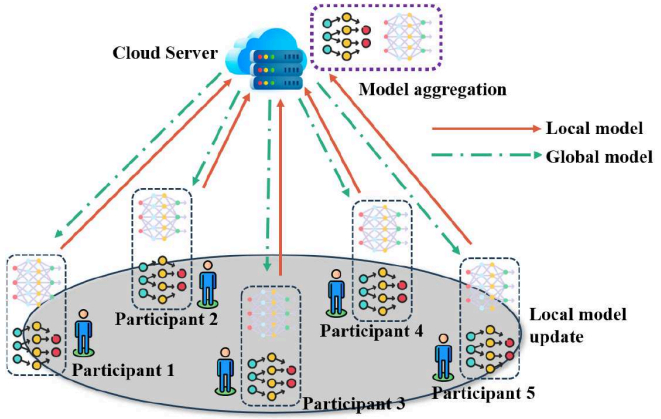

Fig. 5. FL architecture of MobiTask framework.

network. According to the literature [34], the FLOPs for a dense layer are calculated as:

$$\text{FLOPs}_{D'} = (2I_s - 1)O_s. \tag{21}$$

where $\text{FLOPs}_{D'}$ represents the FLOPs of the dense layer, and I_s and O_s denote the input and output dimensions of the Q-network, respectively. Based on the state space, action space, batch size, and the relationship between FLOPs for forward and backward propagation, the FLOPs for each batch of the local Q-network are estimated to be approximately 12.3 MFLOPs. The computational capabilities of current mobile devices fully meet the training requirements of this model.

b) *Memory Usage:* Regarding memory usage, as analyzed in Section IV-B2, the memory consumption during model training includes model parameters, intermediate tensors, and auxiliary data such as optimizer states. By calculating the model parameters and accounting for the memory occupied by the experience replay and gradient storage, the memory footprint of the locally running Q-network is approximately 20 MB, enabling normal operation on mobile devices.

D. FL Enhancement Mechanism

As introduced in Sections IV-B and IV-C, the proposed MobiTask strategy employs FL for model training, with the training paradigm illustrated in Fig. 5. However, conventional FL still exhibits limitations in participant privacy protection and training reliability. To address these challenges, this section designs an FL enhancement mechanism to improve the security and robustness of the MobiTask strategy.

1) Privacy Protection: The MobiTask strategy incorporates FL to enhance security and stability. While FL provides fundamental privacy protection by aggregating model parameters instead of sharing raw data, the uploaded parameters may still leak sensitive information through reverse engineering. To strengthen privacy guarantees, we propose integrating local differential privacy (LDP) into the FL framework.

In LDP, clients perturb private data locally before transmitting it to the cloud server, which processes the obfuscated data to produce outputs. Formally, a randomized mechanism F satisfies ϵ-LDP if for any two adjacent input datasets D and D'

$$\Pr[F(D) \in O] \leq e^{\epsilon} \cdot \Pr[F(D') \in O] \tag{22}$$

where O denotes an arbitrary output subset, and ϵ represents the privacy budget, where the value of ϵ influences the degree of privacy protection. A smaller ϵ corresponds to a higher level of privacy protection, while a larger ϵ results in weaker privacy protection but enhances the performance of the model. This ensures statistically indistinguishable outputs regardless of whether a specific individual's data is included, preventing adversaries from inferring personal information from observed results. Given the low-dimensional input features of our model and the limited computational resources of edge devices, we adopt the Laplace mechanism for LDP implementation. During local training, each client k adds Laplace noise to the gradient $\nabla F_k(w_k^t)$ of the current batch t

$$\tilde{\nabla} F_k(w_k^t) = \nabla F_k(w_k^t) + \text{Lap}(\Delta f/\epsilon) \tag{23}$$

where Δf denotes the gradient sensitivity, and $\text{Lap}(\lambda)$ represents Laplace-distributed noise with scale parameter λ.

By injecting Laplace noise during client-side gradient computation, the uploaded parameters satisfy rigorous ϵ-LDP guarantees, effectively shielding sensitive user data (e.g., movement trajectories and task behaviors) from reverse-engineering attacks.

2) Asynchronous Training: In MCS, MobiTask strategy employs FL for distributed model training. While infrastructure networks generally provide sufficient coverage to sustain federated training, participant mobility inevitably introduces significant latency or temporary disconnections between mobile devices and the cloud server due to distance constraints. This necessitates handling cases where participants fail to submit local model parameters within the current training round.

When a participant misses several rounds before reconnecting, their submitted model parameters become outdated compared with other devices' versions, potentially degrading global model quality. Thus, (8) cannot directly compute contributions

from delayed submissions. To address this, we introduce a dynamic decay mechanism to penalize delayed parameter updates. The modified global model aggregation formula becomes

$$w_G^t \leftarrow \sum_{k=1}^{K} \frac{n_k \cdot \beta^{\Delta\tau_k}}{N} w_k^t \tag{24}$$

where $\Delta\tau_k$ represents the number of delayed rounds for mobile device k, and β ($\beta < 1$) is the decay factor. This formula dynamically adjusts the contribution weight of a device based on its delay.

While this mechanism mitigates negative impacts from delayed submissions, devices experiencing prolonged disconnections may generate parameters too obsolete for meaningful contributions. Therefore, we enforce a validity threshold (e.g., $\tau = 3$ rounds) to discard excessively outdated parameters. This asynchronous training mechanism enhances the reliability of federated training in MCS, ensuring stable operation even in the presence of network instability caused by participant mobility.

V. Experiment

In this section, we evaluate MobiTask using the Geolife dataset and random dataset, comparing it with baselines on task completion rate and migration cost.

A. Experimental Settings

We conducted simulation experiments using the World Vaccine Progress dataset, the Geolife trajectory dataset, and a randomly generated dataset. The World Vaccine Progress dataset contains updated daily data on vaccination progress across countries, which are used to simulate the changes in task execution progress. In the World Vaccine Progress dataset, daily vaccination doses, after standardization, serve as a proxy for the task execution progress p_t^m, representing the proportion of task slices completed at a given time step. This mapping approach effectively captures the dynamic changes in task execution, while the varying vaccination rates across different countries in the dataset can simulate the behavioral heterogeneity of participants in MCS. Furthermore, the time-series characteristics of the dataset and its rich external features, such as administered doses and the proportion of fully vaccinated populations, align closely with the multivariate nature of task execution progress in MCS, facilitating the training of the LSTM-Attention model. The Geolife trajectory dataset, encompassing the trajectories of 182 users in Beijing from April 2007 to August 2012, fulfills the requirements for simulating geographical distances between participants and tasks.

To ensure the reliability and statistical significance of the experimental results, all experiments, including ablation and comparative studies, were conducted independently 10 times, with results averaged. In Figs. 6–11, 95% confidence intervals are used as error bars to reflect the precision of the mean estimates. The experimental parameter settings are shown in Table II. The experiments are conducted on a laptop equipped with an Intel(R) Core(TM) i7-155H 16-core CPU and 32GB of memory. The operating system used is Windows 11, and the experimental environment is Python 3.8.

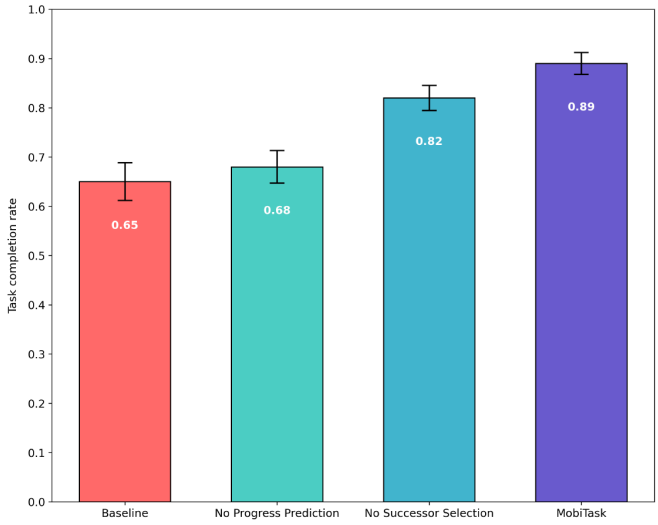

Fig. 6. Ablation experiments of MobiTask strategy.

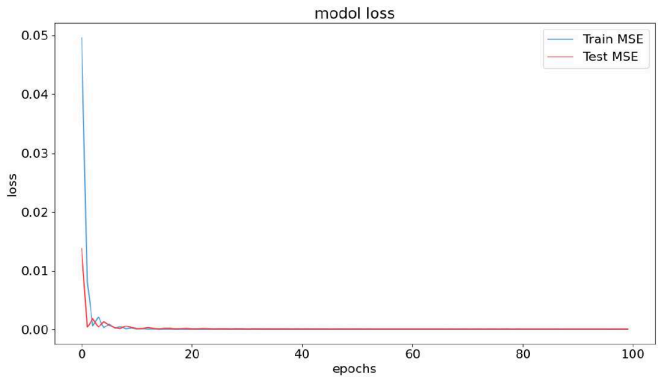

Fig. 7. Error curves of the prediction model on the training and testing datasets.

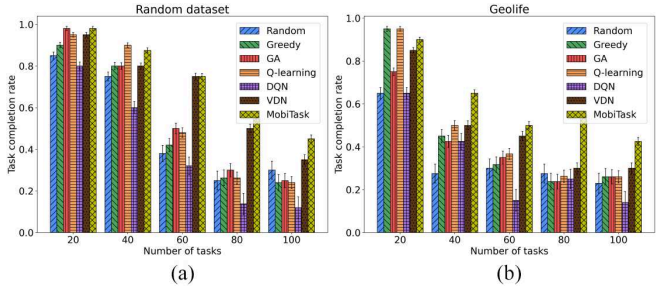

Fig. 8. Completion rate of tasks varies with the number of tasks $B = 1500$, $M = 50$. (a) Random. (b) Geolife.

B. Ablation Experiment

This section conducts an ablation experiment to validate the necessity of key components in the MobiTask strategy by systematically removing its core modules. As MobiTask primarily consists of two critical stages—task execution progress prediction and task successor selection—we compare the full strategy against three ablated variants.

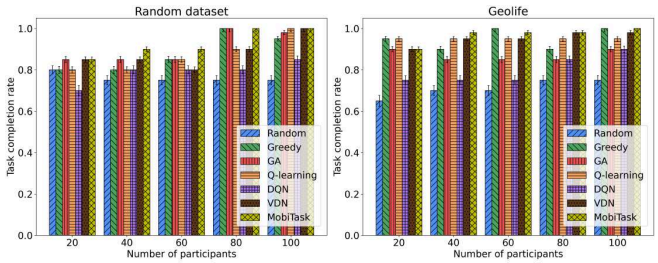

Fig. 9. Task completion rate varies with the number of participants $B = 2000$, $N = 50$. (a) Random. (b) Geolife.

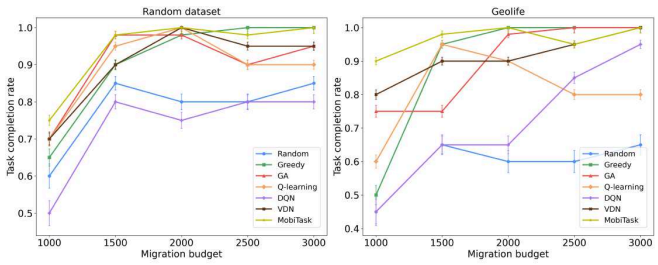

Fig. 10. Task completion rate varies with budget $M = 50$, $N = 50$. (a) Random. (b) Geolife.

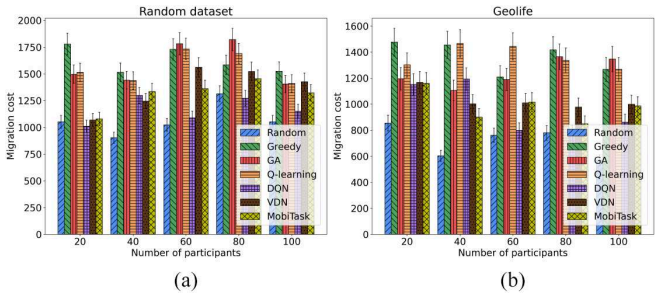

Fig. 11. Migration cost varies with the number of participants $B = 2000$, $N = 50$. (a) Random. (b) Geolife.

TABLE II
PARAMETERS SETTINGS

Experiment Parameter	Setting
Number of tasks N	20,40,60,80,100
Number of candidate participants M	20,40,60,80,100
Task deadline	[30, 100]
Remaining time of participant's own tasks	[0, 40]
Historical task completion rate of participants	[0.5, 1]
Distance between participants and tasks	[5, 50]
Movement speed of participants	[5, 10]
Task migration budget B	[1000, 3000]
Cost of selecting each participant as a successor	[10, 200]
Recruitment cost multiplier α	3

1) *Baseline:* No intervention is applied. Participants execute tasks under their original conditions without any system-level adjustments.
2) *No Progress Prediction:* The task execution progress prediction stage is removed. A random subset of participants is eliminated, and their tasks are migrated arbitrarily.
3) *No Successor Selection:* The task successor selection stage is removed. After eliminating participants unable to complete tasks on time, other participants are randomly selected to reexecute the tasks.

Experiments are conducted on the three datasets introduced in Section V-A, with parameters set as $N = 100$, $M = 100$, and $B = 2000$. We took the average of 10 independent runs. To evaluate scenarios without task migration, we use the global task completion rate as the primary metric. Simulation results are presented in Fig. 6. As illustrated in Fig. 6, the baseline group achieves the lowest task completion rate, as the absence of task migration inevitably leads to participant failures in meeting deadlines. The No Progress Prediction group exhibits the second-lowest task completion rate, with only a marginal improvement of 3% compared with the baseline group. This is because, without task execution progress prediction, it is impossible to accurately identify participants unable to complete tasks on time, and blind task migration may cause some tasks that could have been completed on time to exceed their deadlines. These results validate the necessity of the progress prediction module in MobiTask. The No Successor Selection group shows a 17% higher completion rate than baseline, but remains 7% lower than the full MobiTask strategy. This performance gap stems from the inefficiency of random successor assignment, which still permits delayed task executions. The results of the no successor selection group validate the rationality of incorporating the task successor selection module in the MobiTask strategy.

Collectively, the ablation experiment confirms the efficacy of each component in MobiTask's design.

C. Comparative Experiment

We design comparative experiments for both stages of the MobiTask: task execution progress prediction and task successor selection.

1) Task Execution Progress Prediction Stage Experiment: During the training of the task execution progress prediction model, the learning rate is set to 0.005, with 100 iterations and 64 hidden layer nodes. The mean square error (MSE) and mean absolute error (MAE) are chosen as the loss functions to evaluate the model's performance. The model is trained using a personalized FL approach. The error change curves during the global model's training and testing processes are shown in Fig. 7, demonstrating that the model successfully converges. We compare seven models: GRU, LSTM, transformer, GRU-attention, centralized-LSTM-attention (a centralized training LSTM-attention model), FL-LSTM-attention (an LSTM-attention model trained using normal FL), and the LSTM-attention model trained with personalized FL used in the proposed MobiTask. The experiment is conducted with 10 samples, and the average error values are calculated. The experimental results are shown in Table III. From the above experimental results, the LSTM-attention model trained using personalized FL in this article demonstrates a certain advantage in prediction accuracy compared with other models and training methods. The resulting MSE and MAE errors are 11% and 17%

TABLE III
PREDICTION RESULT ERRORS

Method	MSE	MAE
GRU	0.084	0.210
LSTM	0.087	0.227
Transformer	0.213	0.424
GRU-Attention	0.071	0.187
Centralized-LSTM-attention	0.055	0.149
FL-LSTM-attention	0.135	0.248
MobiTask(ours)	0.049	0.124

lower, respectively, compared with the best-performing models in other approaches.

2) Task Successor Selection Stage Experiment: For the task successor selection stage, we compare the MobiTask with the following six algorithms.

a) *Random:* A centralized random participant selection method for task succession.
b) *Greedy:* A centralized global greedy approach, which prioritizes selecting participants with the shortest remaining time on their current tasks for task succession.
c) *GA:* A global search method that starts from a random initial population and gradually improves candidate solutions through operations such as selection, crossover, and mutation to find the optimal or near-optimal solution.
d) *Q-Learning:* A model-free reinforcement learning algorithm that uses a Q-table to store state-action pairs and their corresponding Q-values to select the optimal action. The goal is to maximize long-term cumulative rewards.
e) *DQN:* An integration of deep learning with Q-learning, using a neural network to approximate the Q-value function and solving the computational and storage issues of traditional Q-learning in complex environments.
f) *VDN:* A MARL algorithm that promotes cooperative learning by aggregating the action-value functions of all agents into a centralized team value function through linear summation.

Comparative experiments were conducted on both a randomly generated dataset and the Geolife trajectory dataset, with results averaged over 10 independent runs. As shown in Figs. 8–11, error bars representing 95% confidence intervals are included to demonstrate the statistical reliability of task completion rates and migration costs. The goal of the MobiTask is to maximize the completion rate of migration tasks, which directly depends on factors such as the number of tasks to be succeeded (referred to as "tasks"), the number of candidate participants, and the task migration budget. We compare the performance of the MobiTask and other baseline algorithms by varying these factors. As shown in Fig. 8, for both datasets, as the number of tasks increases, the task completion rate decreases for all methods. This is because the number of candidate participants remains constant while the total number of tasks increases. Among them, the MobiTask is the least affected by changes in the number of tasks. On both datasets, when the task count reaches five times the number of candidate participants, it can still maintain a task completion rate of over 40%. This indicates that the MobiTask is more stable than the other baseline algorithms when the number of tasks increases significantly. When the task count is close to the number of candidate participants, the MobiTask performs at the forefront of all algorithms. As task quantities scale significantly, MobiTask strategy consistently maintains superior task completion rates compared with baseline methods. When increasing the task volume to 60, MobiTask achieves a 10% higher completion rate than the best-performing baseline.

In Fig. 9, we compare the task completion rates of different algorithms as the number of candidate participants increases. From the figure, it is evident that as the number of candidate participants increases, the task completion rate generally exhibits an upward trend. The MobiTask consistently maintains a high task completion rate, demonstrating greater stability compared with other baseline algorithms. In both datasets, the MobiTask is able to achieve over 80% task completion, highlighting the effectiveness of the multiagent collaboration approach designed in this article to maximize task completion rates. This approach outperforms traditional greedy algorithms and single-agent methods. While the baseline VDN algorithm employs a multi-agent collaboration framework, its linear summation approach for global Q-value computation fails to capture nonlinear agent interactions. In contrast, our QMIX-based solution introduces a mixing network that nonlinearly aggregates local Q-values, enabling more effective collaborative training among agents. Fig. 10 shows the performance of different algorithms as the migration budget varies. It can be observed that as the migration budget increases, the task completion rate also improves. When the migration budget exceeds 2000, some algorithms can achieve a 100% task completion rate. Under lower budget conditions, MobiTask still maintains a relatively high task completion rate. This is because MobiTask prioritizes selecting participants who are geographically closer to the task and have previously completed tasks of the same type. These participants tend to complete the task more quickly. Under low-budget conditions, the number of participants available for selection is limited for all methods. However, the probability of successfully completing the task is higher for participants selected by MobiTask compared with those selected by other methods.

In addition, we also compared the migration cost when the number of candidate participants changes, as shown in Fig. 11. From the figure, it can be seen that the migration cost for random selection and DQN is relatively low, as the task completion rates achieved by these two methods are also lower. Greedy selection tends to prioritize participants with shorter remaining execution times for their current tasks while overlooking the associated costs. MobiTask, on the other hand, is more likely to select existing participants to succeed in the task, resulting in relatively lower costs and better overall performance compared with other baseline methods.

To evaluate the performance improvement of the proposed GAT-QMIX model in task successor selection, we conducted a comparative analysis of GAT-QMIX and traditional QMIX in terms of TCR. Experiments were performed on the Geolife trajectory dataset, testing the performance of both models under

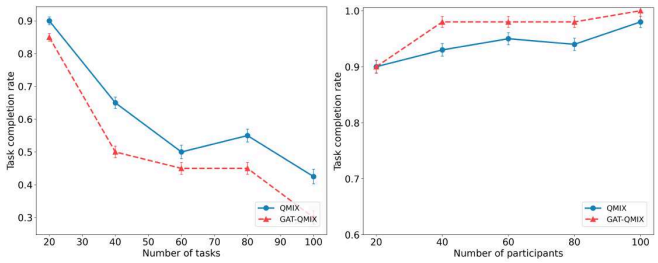

Fig. 12. Performance comparison between GAT-QMIX and QMIX. (a) $B = 1500$, $M = 50$. (b) $B = 2000$, $N = 50$.

varying task quantities and candidate participant numbers, with parameters consistent with those specified in Figs. 8 and 9. All experiments were averaged over 10 independent runs. Fig. 12(a) and (b) illustrates the TCR performance of GAT-QMIX and QMIX under varying task quantities and candidate participant numbers, respectively. As shown in Fig. 12(a), on the Geolife dataset, as the number of tasks increased from 20 to 100, GAT-QMIX consistently outperformed traditional QMIX, achieving an average TCR improvement of approximately 17.2%. Notably, when the task quantity was large, GAT-QMIX exhibited a TCR improvement of nearly 30% over QMIX. In Fig. 12(b), as the number of candidate participants increased from 20 to 100, both models approached a TCR of 100%. At $M = 80$, GAT-QMIX achieved a TCR of 98%, surpassing QMIX's 94% by 4.3%. These experimental results demonstrate that GAT-QMIX is more effective in task allocation under high-load scenarios. By incorporating a GAT layer, GAT-QMIX better captures the complex dependencies between tasks and participants, particularly when task or participant numbers increase, enabling more effective matching of task-participant pairs compared with traditional QMIX, thereby optimizing successor selection and enhancing task completion rates.

3) FL Privacy Budget Experiment: In Section IV-D1, we introduced the ϵ-LDP mechanism to enhance privacy protection in FL. However, the impact of different privacy budget values ϵ on model performance varies within the ϵ-LDP framework. To assess the effects of the ϵ-LDP mechanism integrated into the MobiTask strategy on both privacy protection and model performance in FL, we designed a privacy budget experiment. The experiment aimed to analyze the influence of varying privacy budget parameters ϵ on the performance of the task execution progress prediction model and the task successor selection model, while verifying that MobiTask maintains high task completion rates and prediction accuracy under privacy constraints.

Simulations were conducted using the World Vaccine Progress and Geolife trajectory datasets referenced in Section V-A, with baseline parameters set to $N = 100$, $M = 100$, and $B = 2000$. The privacy budget ϵ was varied across the range [0.1, 0.5, 1.0, 2.0, 5.0] to investigate performance transitions from strong privacy protection (low ϵ) to weak privacy protection (high ϵ). For each ϵ value, models underwent 300 training iterations with a learning rate of 0.005. Performance was evaluated using MSE and TCR to assess the impact of different ϵ values on the LSTM-Attention model for task execution progress prediction and the GAT-QMIX model for task successor selection, with results presented in Table IV.

TABLE IV
MODEL PERFORMANCE UNDER DIFFERENT PRIVACY BUDGETS

Privacy Budget ϵ	Model Performance	
	MSE	TCR
0.1	0.062	0.83
0.5	0.055	0.85
1.0	0.049	0.89
2.0	0.049	0.90
5.0	0.047	0.90

The results demonstrate that MobiTask, by incorporating local differential privacy, effectively balances participant privacy protection with model performance. At low ϵ values, the noise introduced by strong privacy protection significantly impacted prediction accuracy and task completion rates. However, as ϵ increased, performance degradation decreased rapidly, approaching the performance of the baseline group without privacy protection at $\epsilon = 2.0$. This indicates that MobiTask achieves a favorable trade-off between privacy protection and model performance.

VI. CONCLUSION

In this article, we propose MobiTask, an FL-based task migration strategy for MCS. The strategy consists of two main stages. In the task execution progress prediction stage, MobiTask employs an LSTM model enhanced with an attention mechanism to predict each participant's task execution progress, classifying them into two groups: those who can complete their tasks on time and those who cannot. In the task successor selection stage, MobiTask eliminates participants who cannot meet their task deadlines and reallocates their tasks. Subsequently, it employs a MARL algorithm incorporating GAT to select suitable successors from the candidate participant pool. Additionally, we adopt FL to train the models used in both stages, enhancing the strategy's security and stability. Finally, we conduct ablation experiments and comparative experiments on multiple datasets. The experimental results demonstrate that MobiTask is effective, exhibits stable performance, and outperforms other baseline methods. In future work, we plan to integrate participant mobility prediction into our strategy to more accurately estimate the arrival time of candidate participants at task locations, further improving the efficiency of task migration.

REFERENCES

[1] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 32–39, Nov. 2011.

[2] Y. Yang, B. Guo, J. Zhang, Z. Yu, and X. Zhou, "SentiStory: multi-grained sentiment analysis and event summarization with crowdsourced social media data," *Pers Ubiquit Comput.*, vol. 21, no. 1, pp. 97–111, Feb. 2017.

[3] J. Zhang, L. Han, and B. Guo, "Sparse mobile crowdsensing for gas monitoring in coal mine working face," *IEEE Internet Things J.*, vol. 11, no. 22, pp. 36633–36645, Nov. 2011.

[4] C. Marche, L. Perra, and M. Nitti, "Crowdsensing and trusted digital twins for environmental noise monitoring," in *Proc. IEEE Int. Mediterr. Conf. Commun, Netw., MeditCom.*, Madrid, Spain, 2024, pp. 535–540.

[5] Y.-A. Daraghmi, T.-H. Wu, and T.-U. İk, "Crowdsourcing-based road surface evaluation and indexing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 5, pp. 4164–4175, May 2022.

[6] J. Wang, L. Wang, Y. Wang, D. Zhang, and L. Kong, "Task allocation in mobile crowd sensing: state-of-the-art and future opportunities," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3747–3757, Oct. 2018.

[7] J. Lu, H. Liu, R. Jia, Z. Zhang, X. Wang, and J. Wang, "Incentivizing proportional fairness for multi-task allocation in crowdsensing," *IEEE Trans. Serv. Comput.*, vol. 17, no. 3, pp. 990–1000, May 2024.

[8] Z. Chen, M. Xu, and C. Su, "Online quality-based privacy-preserving task allocation in mobile crowdsensing," *Comput. Netw.*, vol. 251, Sep. 2024, Art. no. 110613.

[9] Z. Wang, Y. Cao, H. Zhou, L. Wu, W. Wang, and G. Min, "Fairness-aware two-stage hybrid sensing method in vehicular crowdsensing," *IEEE Trans. Mob. Comput.*, vol. 23, no. 12, pp. 11971–11988, Dec. 2024.

[10] W. Tan, L. Zhao, B. Li, L. Xu, and Y. Yang, "Multiple cooperative task allocation in group-oriented social mobile crowdsensing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3387–3401, Nov./Dec. 2022.

[11] S. Peng, B. Zhang, Y. Yan, and C. Li, "A multiplatform-cooperation-based task assignment mechanism for mobile crowdsensing," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 16881–16894, Oct. 2023.

[12] Q. Shen, M. Ma, and M. Li, "An extensible bounded rationality-based task recommendation scheme for from-scratch mobile crowdsensing," *IEEE Trans. Comput. Social Syst*, vol. 11, no. 6, pp. 7871–7880, Dec. 2024.

[13] J. Wang, X. Wang, and G. Zhao, "Task recommendation via heterogeneous multi-modal features and decision fusion in mobile crowdsensing," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 20, no. 3, pp. 78–98, Nov. 2023.

[14] G. Zhao, X. Wang, J. Wang, and J. Liu, "Task recommendation for mobile crowd sensing system based on multi-view user dynamic behavior prediction," *Peer-to-Peer Netw. Appl.*, vol. 16, no. 3, pp. 1536–1550, May 2023.

[15] J. Wang et al., "HyTasker: hybrid task allocation in mobile crowd sensing," *IEEE Trans. Mob. Comput.*, vol. 19, no. 3, pp. 598–611, Mar. 2020.

[16] S. Peng, G. Zhang, B. Zhang, Z. Yao, C. Liu, and C. Li, "A stable task assignment mechanism for multi-platform mobile crowdsensing," *IEEE Trans. Veh. Technol.*, vol. 74, no. 5, pp. 8079–8094, May 2025.

[17] H. Jiang, S. Wang, C. Tang, H. Wu, and R. Li, "Reschedulable task allocation strategy in cloud-edge-end cooperative mobile crowd sensing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Denver, CO, USA, 2024, pp. 3262–3267.

[18] H. Li, Z. Yu, and Y. Luo, "ContinuousSensing: a task allocation algorithm for human–robot collaborative mobile crowdsensing with task migration," *CCF. Trans. Pervasive. Comp. Interact*, vol. 6, no. 3, pp. 228–243, Sep. 2024.

[19] Y. Luo, Z. Yu, J. Ren, and B. Guo, "HuMachineSensing: A novel mobile crowdsensing framework with robust task allocation algorithm," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun., Int. Conf. Data Sci. Syst., Int. Conf. Smart City Int. Conf. Dependability Sensor, Cloud Big Data Syst. Appl., HPCC-DSS-SmartCity-DependSys.*, Haikou, Hainan, China, 2021, pp. 2386–2395.

[20] G. Bajaj and P. Singh, "MEW: A plug-n-play framework for task allocation in mobile crowdsensing," in *Proc. ACM Workshop Mob. Crowdsensing Syst. Appl., Part SenSys. (CrowdSenSys)*, New York, NY, USA, 2017, pp. 19–24.

[21] G. Yang, D. Guo, B. Wang, X. He, J. Wang, and G. Wang, "Participant-quantity-aware online task allocation in mobile crowdsensing," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 22650–22663, Dec. 2023.

[22] T. Hu, M. Xiao, C. Hu, G. Gao, and B. Wang, "A QoS-sensitive task assignment algorithm for mobile crowdsensing," *Pervasive Mob. Comput.*, vol. 41, pp. 333–342, Oct. 2017.

[23] B. Zhao, H. Dong, Y. Wang, X. Gao, and T. Pan, "An enhanced task allocation algorithm for mobile crowdsourcing based on spatiotemporal attention network," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 3, pp. 3803–3815, Jun. 2024.

[24] C. Li, X. Zeng, M. Zhang, and Z. Cao, "PyramidFL: a fine-grained client selection framework for efficient federated learning," in *Proc. Annu. Int. Conf. Mobile Comput. Netw. (MobiCom)*, New York, NY, USA, 2023, pp. 1–16.

[25] S. Dongare, A. Ortiz, and A. Klein, "Federated deep reinforcement learning for task participation in mobile crowdsensing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, 2023, pp. 4436–4441.

[26] Z. Chen, M. Simsek, and B. Kantarci, "Federated learning-based risk-aware decision to mitigate fake task impacts on crowdsensing platforms," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, 2021, p. 1.

[27] W. Zhang, Z. Li, and X. Chen, "Quality-aware user recruitment based on federated learning in mobile crowd sensing," *Tsinghua Sci. Technol.*, vol. 26, no. 6, pp. 869–877, Dec. 2021.

[28] A. Mubarak and E. Almetwally, "Modelling and forecasting of Covid-19 using periodical ARIMA models," *Ann. Data Sci.*, vol. 11, no. 4, pp. 1483–1502, Aug. 2024.

[29] J. Shi, X. Chen, Y. Xie, H. Zhang, and Y. Sun, "Delicately reinforced k-nearest neighbor classifier combined with expert knowledge applied to abnormity forecast in electrolytic cell," *IEEE Trans. Neural Netw. Learn. Sys.*, vol. 35, no. 3, pp. 3027–3037, Mar. 2024.

[30] N. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, pp. 24–38, May 2009.

[31] H. Dudukcu, M. Taskiran, Z. Taskiran, and T. Yildirim, "Temporal convolutional networks with RNN approach for chaotic time series prediction," *Appl. Soft Comput.*, vol. 133, no. 109945, Jan. 2023.

[32] Y.-Y. Chen, H.-N. Hung, S.-R. Yang, and C.-C. Yen, "On poisoning attacks and defenses for LSTM time series prediction models: speed prediction as an example," in *Proc. Int. Wireless Commun. Mob. Comput. Conf. (IWCMC)*, Ayia Napa, Cyprus, 2024, pp. 610–615.

[33] X. Wen and W. Li, "Time series prediction based on LSTM-Attention-LSTM model," *IEEE Access*, vol. 11, pp. 48322–48331, 2023.

[34] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, 2017, p. 1.

[35] T. Wang, A. Hussain, L. Zhang, and C. Zhao, "Collaborative edge computing for social internet of vehicles to alleviate traffic congestion," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 1, pp. 184–196, Feb. 2022.

[36] T. Cai, Z. Yang, Y. Chen, W. Chen, Z. Zheng, and Y. Yu, "Cooperative data sensing and computation offloading in UAV-assisted crowdsensing With multi-agent deep reinforcement learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3197–3211, Oct. 2022.

[37] R. Ding, Z. Yang, Y. Wei, H. Jin, and X. Wang, "Multi-agent reinforcement learning for urban crowd sensing with for-hire vehicles," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Vancouver, BC, Canada, 2021, pp. 1–10.

[38] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Long Beach, CA, USA, 2017, pp. 6380–6391.

[39] Y. Li, G. Xie, and Z. Lu, "Difference advantage estimation for multi-agent policy gradients," in *Proc. Mach. Learn. Res. (ICML)*, Baltimore, MD, USA, 2022, pp. 13066–13085.

[40] Q. Zhang, Q. Zhang, and J. Lin, "Succinct and robust multi-agent communication with temporal message control," in *Proc. Adv. Neural Inf. Process Syst. (NIPS)*, Vancouver, Canada, 2020, pp. 17271–17282.

[41] P. Sunehag et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS)*, Stockholm, Sweden, 2018, pp. 2085–2087.

[42] T. Rashid, M. Samvelyan, de Witt. Cs, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 7234–7284, Aug. 2020.

[43] C. Xu and W. Song, "Decentralized task assignment for mobile crowdsensing with multi-agent deep reinforcement learning," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 16564–16578, Sep. 2023.

Siyuan Yin received the B.E. degree in 2024 from China University of Mining and Technology, Xuzhou, China, where he is currently working toward the M.S. degree, both in computer science and technology.

His research interests include mobile crowdsensing and federated learning.

Haifeng Jiang received the Ph.D. degree in communication and information systems from China University of Mining and Technology, Jiangsu, China, in 2010.

He is currently an Associate Professor with the School of Computer Science and Technology, China University of Mining and Technology. His research interests include wireless network technology, mobile crowdsensing, and federated learning.

Chaogang Tang (Member, IEEE) received the B.S. degree in information security from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2007, and the Ph.D. degree in computer applications from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, in 2012, and the Department of Computer Science, City University of Hong Kong, Hong Kong, under a joint Ph.D. Program, in 2012.

He is currently working as a Lecturer with the China University of Mining and Technology, Jiangsu, China. His research interests include mobile cloud computing, fog computing, internet of things, and big data.

Shuo Xiao received the Ph.D. degree in traffic information engineering and control from Beijing Jiao Tong University, Beijing, China, in 2010.

He is currently a Professor in computer science and technology with China University of Mining and Technology, Jiangsu, China. His research interests include intelligent information processing, artificial intelligence and pattern recognition, and machine learning.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees from Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science with Freie Universität Berlin, Berlin, Germany, in 2015.

He is currently a Professor with the Center for Applied Mathematics, Tianjin University, Tianjin, China. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing, and deep learning.