

# Model-aided Federated Reinforcement Learning for Multi-UAV Trajectory Planning in IoT Networks

Jichao Chen<sup>1,2</sup>, Omid Esrafilian<sup>1</sup>, Harald Bayerlein<sup>2</sup>, David Gesbert<sup>1</sup>, and Marco Caccamo<sup>2</sup>

<sup>1</sup>Communication Systems Department, EURECOM, Sophia Antipolis, France

<sup>2</sup>TUM School of Engineering and Design, Technical University of Munich, Germany

{jichao.chen, h.bayerlein, mcaccamo}@tum.de, {omid.esrafilian, david.gesbert}@eurecom.fr

**Abstract**—Deploying teams of unmanned aerial vehicles (UAVs) to harvest data from distributed Internet of Things (IoT) devices requires efficient trajectory planning and coordination algorithms. Multi-agent reinforcement learning (MRL) has emerged as a solution, but requires extensive and costly real-world training data. To tackle this challenge, we propose a novel model-aided federated MRL algorithm to coordinate multiple UAVs on a data harvesting mission with only limited knowledge about the environment. The proposed algorithm alternates between building an environment simulation model from real-world measurements, specifically learning the radio channel characteristics and estimating unknown IoT device positions, and federated QMIX training in the simulated environment. Each UAV agent trains a local QMIX model in its simulated environment and continuously consolidates it through federated learning with other agents, accelerating the learning process. A performance comparison with standard MRL algorithms demonstrates that our proposed model-aided FedQMIX algorithm reduces the need for real-world training experiences by around three magnitudes while attaining similar data collection performance.

## I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have attracted growing interest in communication networks due to their high mobility, flexibility, and ease of deployment [1]. One of the most important applications is data harvesting from geographically dispersed Internet of Things (IoT) sensor devices, where UAVs acting as mobile base stations, can establish efficient line-of-sight (LoS) links with IoT devices. The performance of data harvesting depends on the dynamic trajectories of UAVs, and the behavior of each UAV can affect the tasks of other UAVs. Therefore, it is crucial to design effective trajectories for UAVs, while considering cooperation between them, in order to ensure efficient data collection.

Recently, deep reinforcement learning (DRL) algorithms have been extensively employed to design UAV trajectories for communication. The authors in [2] employ a centralized deep deterministic policy gradient (DDPG) approach to optimize multi-UAV trajectories, aiming to minimize the age of information (AoI) during a data collection mission. However, centralized learning leads to scalability issues with increasing agent numbers. The authors in [3] avoid this issue by utilizing a double deep Q-network (DDQN) approach based on centralized learning with decentralized execution (CLDE) for multi-UAV path planning to maximize collected data from IoT sensor nodes. The approach is focused on adapting to a wide range of scenario parameters, but incurs a high training cost.

QMIX, a classic multi-agent reinforcement learning (MRL) algorithm with a nonlinear value decomposition that we also base our approach on, is employed in [4] for multi-UAV trajectory planning to minimize the average AoI, outperforming other baselines like cluster-based and independent DQN-based approaches. Moreover, the authors in [5] develop a fully decentralized MRL framework to maximize data collection while minimizing AoI and maintaining a certain AoI threshold. This method utilizes a transformer for temporal modeling and introduces an intrinsic reward mechanism to enhance the exploration, achieving better results than other classic MRL algorithms. In contrast to our approach, all mentioned works have large training data requirements and assume full prior knowledge of all device or user positions.

In our work, we also make use of the idea of federated reinforcement learning (FRL) where learned information is exchanged between agents without uploading raw collected data to a central server, thereby accelerating learning without incurring too high communication overhead costs [6]. For instance, the authors in [7] utilize FRL to design multiple UAV trajectories for user localization, lowering localization error and increasing convergence speed. In another related application, the authors in [8] propose a distributed federated multi-agent DDPG algorithm to optimize trajectories for air and ground unmanned vehicles in emergency situations leveraging FRL to address data isolation and to accelerate the convergence.

Despite the satisfactory performance of the aforementioned algorithms in their respective domains, DRL-based trajectory planning algorithms may lose their benefits in real-world scenarios, since collecting real-world training data requires expensive interaction with actual physical systems [9]. Our proposed solution to this challenge is based on [10], a first attempt to design a model-aided DRL algorithm for UAV data harvesting, however only for the single UAV case in a simpler environment containing devices with unlimited data.

In this paper, we consider a scenario where multiple energy-limited UAVs cooperate to collect data from ground devices with only limited data volume. In our approach<sup>1</sup>, we make use of the popular QMIX algorithm [11] and the idea of FRL [6]. To the best of our knowledge, this is the first

<sup>1</sup>The final version of this paper has been accepted by IEEE GLOBECOM Workshops 2023. Code available at [https://github.com/Cirrick/Multi\\_UAV\\_Data\\_Harvesting](https://github.com/Cirrick/Multi_UAV_Data_Harvesting).

work that employs FRL with a learned environment model to design multi-UAV trajectories for IoT data collection. Our main contributions are summarized as follows:

- We exploit measurements collected by the UAVs to learn a digital twin of the real-world environment suitable for training MARL algorithms. To this end, we estimate unknown IoT device locations using particle swarm optimization (PSO) without requiring prior knowledge of the radio channel characteristics.
- Leveraging the learned environment, we introduce a model-aided QMIX-based algorithm to solve the multi-UAV data harvesting problem that reduces the amount of costly real-world training data by around three magnitudes compared to standard MARL algorithms.
- We further propose a model-aided federated QMIX (FedQMIX) algorithm to enhance convergence speed through federated learning consolidating the locally trained QMIX models at each UAV. This approach helps to distribute the computational load over all UAVs and to utilize all available resources efficiently.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider an urban environment where a set of UAVs  $\mathcal{I} = \{1, \dots, I\}$  are deployed to collect data from various stationary ground IoT devices. The mission time duration is discretized into  $T$  equal time slots with length  $\Delta t$ , which are chosen sufficiently short so that the velocity of each UAV can be assumed to remain constant within a single time slot. The position of the  $i$ -th UAV at time step  $t$  is denoted by  $\mathbf{p}_t^i = [x_t^i, y_t^i, h^i]^T \in \mathbb{R}^3$ ,  $t \in [0, T]$ , where  $h^i$  represents the altitude of the  $i$ -th UAV. We assume that each UAV flies at a different altitude for collision avoidance and that each UAV's altitude remains constant throughout mission duration. Each UAV flies from a predefined starting point  $\mathbf{p}_I$  and will return to a terminal point  $\mathbf{p}_F$  at the end of the mission. UAVs do not collect data from the devices at the final time step  $T$ .

A set of ground devices  $\mathcal{U} = \{1, \dots, K\}$  is distributed in the environment, with the  $k$ -th device located at  $\mathbf{u}^k = [x^k, y^k, 0]^T \in \mathbb{R}^3$ . For device  $k$ , the amount of remaining data in its buffer at time step  $t$  is denoted by  $D_t^k$ . The buffer of each device is initialized as  $D_0^k = D_{init}^k$  at the beginning of the mission. Moreover, the ground devices are divided into two distinct groups: devices with known locations (referred to as *anchor devices*)  $\mathcal{U}_{known}$ , and devices with unknown locations  $\mathcal{U}_{unknown}$ . Accordingly, we have  $\mathcal{U} = \mathcal{U}_{known} \cup \mathcal{U}_{unknown}$ .

### A. UAV Model

The action space of each UAV is defined as

$$\mathcal{A} = \left\{ \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{hover}}, \underbrace{\begin{bmatrix} 0 \\ c \\ 0 \end{bmatrix}}_{\text{north}}, \underbrace{\begin{bmatrix} -c \\ 0 \\ 0 \end{bmatrix}}_{\text{west}}, \underbrace{\begin{bmatrix} 0 \\ -c \\ 0 \end{bmatrix}}_{\text{south}}, \underbrace{\begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix}}_{\text{east}}, \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{no-op}} \right\}, \quad (1)$$

where  $c$  denotes the distance that the UAV can move within a single time step. The no-op action refers to no operation, i.e.,

no movement and no energy consumption, which can only be chosen when the UAV has drained its energy. Given the executed action  $\mathbf{a}_t^i \in \mathcal{A}$ , the position of the UAV evolves as

$$\mathbf{p}_{t+1}^i = \mathbf{p}_t^i + \mathbf{a}_t^i. \quad (2)$$

We assume the hover action consumes half the energy of the movement action as in [10]. Denoting the remaining battery of the  $i$ -th UAV at time step  $t$  by  $b_t^i \in \mathbb{R}$ , which evolves according to

$$b_{t+1}^i = \begin{cases} b_t^i, & \text{if } \mathbf{a}_t^i = \text{no-op}, \\ b_t^i - 0.5, & \text{if } \mathbf{a}_t^i = \text{hover}, \\ b_t^i - 1, & \text{otherwise.} \end{cases} \quad (3)$$

The data harvesting mission terminates when all UAV batteries are depleted.

### B. Channel Model

The channel gain between UAV  $i$  and device  $k$  at time step  $t$  is modeled as [10]

$$g_t^{i,k} = \begin{cases} \beta_{\text{LoS}} + \alpha_{\text{LoS}} \log_{10}(d_t^{i,k}) + \eta_{\text{LoS}}, & \text{if LoS,} \\ \beta_{\text{NLoS}} + \alpha_{\text{NLoS}} \log_{10}(d_t^{i,k}) + \eta_{\text{NLoS}}, & \text{if NLoS,} \end{cases} \quad (4)$$

where  $d_t^{i,k} = \|\mathbf{p}_t^i - \mathbf{u}^k\|_2$  is their absolute distance. Let  $z \in \{\text{LoS}, \text{NLoS}\}$  denote either a LoS or non-LoS (NLoS) condition,  $\alpha_z$  is a path loss constant,  $\beta_z$  is the average channel gain at the reference distance  $d_0 = 1\text{m}$ , and  $\eta_z$  represents the shadowing component modeled as a Gaussian distribution  $\mathcal{N}(0, \sigma_z^2)$ . Note that both the channel model and associated parameters are unknown and need to be learned by the UAVs. The signal-to-noise ratio (SNR) is given by

$$\text{SNR}_t^{i,k} = \frac{P10^{0.1g_t^{i,k}}}{\sigma^2}, \quad (5)$$

where  $P$  is the transmit power, and  $\sigma^2$  is the white Gaussian noise power at the receiver. Additionally, the channel gain and SNR between two UAVs can be modeled analogously. Furthermore, the information rate is given by  $R_t^{i,k} = \log_2(1 + \text{SNR}_t^{i,k})$  assumed to be constant within each time slot.

### C. Channel Access Protocol

We assume that the communication between the UAVs and ground devices follows a time-division multiple access (TDMA) approach. Denoting the data collection status by  $q_t^{i,k} \in \{0, 1\}$ ,  $q_t^{i,k} = 1$  indicates the  $i$ -th UAV collects data from the  $k$ -th device at time step  $t$ , and otherwise  $q_t^{i,k} = 0$ . We assume that each UAV can collect data from only one device at each time step, imposing the following constraint

$$\sum_{k=1}^K q_t^{i,k} \leq 1, \quad \forall i \in \mathcal{I}, t \in [0, T-1]. \quad (6)$$

Similarly, every device's data can only be collected by a single UAV, i.e.,

$$\sum_{i=1}^I q_t^{i,k} \leq 1, \quad \forall k \in \mathcal{U}, t \in [0, T-1]. \quad (7)$$

The value of  $q_t^{i,k}$  is set according to the max-rate rule in [3]: among all the ground devices, only the device with available data and the highest  $\text{SNR}_t^{i,k}$  is considered for data collection

by the  $i$ -th UAV. If the  $i$ -th UAV collects data from the  $k$ -th ground device, the achievable throughput is given by

$$C_t^{i,k} = \begin{cases} R_t^{i,k}, & \text{if } D_t^k \geq R_t^{i,k} \Delta t, \\ D_t^k / \Delta t, & \text{otherwise,} \end{cases} \quad (8)$$

which is limited by the information rate  $R_t^{i,k}$  when the remaining data of the device is sufficient; otherwise, the UAV collects all remaining data within one time slot duration  $\Delta t$ .

#### D. Problem Formulation

The objective of our algorithm is to design the trajectories for multiple UAVs to maximize the amount of data collected from ground devices within mission time. Utilizing the previously defined UAV mobility model and channel model, this data collection problem can be formulated as the following optimization problem

$$\max_{\times_i \mathbf{a}_t^i} \sum_{t=0}^{T-1} \sum_{i=1}^I \sum_{k=1}^K q_t^{i,k} C_t^{i,k} \Delta t, \quad (9)$$

$$\text{s.t. } \mathbf{p}_0^i = \mathbf{p}_I, \mathbf{p}_T^i = \mathbf{p}_F, \forall i \in \mathcal{I}, \quad (9a)$$

$$h^i \neq h^j, \forall i \neq j, i, j \in \mathcal{I}, \quad (9b)$$

$$b_T^i \geq 0, \forall i \in \mathcal{I}, \quad (9c)$$

$$(2), (3), (6), (7), \quad (9d)$$

where  $\times_i \mathbf{a}_t^i$  is the joint action of all the UAVs. (9b) ensures that all the UAVs fly at different altitudes, thereby avoiding collisions between them. (9c) guarantees that all the UAVs can reach the terminal positions with remaining battery at the end of the mission time. This optimization problem is challenging due to its non-convexity and is further complicated because of the unknown channel model and unknown device locations.

### III. DECENTRALIZED PARTIALLY OBSERVABLE MARKOV DECISION PROCESS AND QMIX

#### A. Dec-POMDP

To solve optimization problem (9), we first reformulate it as a decentralized partially observable Markov decision process (Dec-POMDP) which is defined as a tuple  $(\mathcal{I}, \mathcal{S}, \mathcal{A}_\times, P, R, \Omega_\times, \mathcal{O}, \gamma)$ , where  $\mathcal{I}$  denotes a set of  $I$  agents,  $\mathcal{S}$  describes the state space of the environment,  $\mathcal{A}_\times = \times_i \mathcal{A}^i$  is the joint action space, according to (1),  $\mathcal{A}^i = \mathcal{A}$  is the same for all the UAVs, and  $\Omega_\times = \times_i \Omega^i$  is the joint observation space. At each time step, given action  $\mathbf{a}^i \in \mathcal{A}$  executed by agent  $i$ , and the joint action  $\times_i \mathbf{a}^i \in \mathcal{A}_\times$ , the environment transitions from state  $s \in \mathcal{S}$  to next state  $s' \in \mathcal{S}$  according to the probability  $P(s' | s, \times_i \mathbf{a}^i)$ . All the agents share the same reward function  $R(s, \times_i \mathbf{a}^i, s') = r$ .

In a partially observable environment, each agent can only access its local observation  $o^i \in \Omega^i$  according to the observation probability function  $O(\times_i o^i, s', \times_i \mathbf{a}^i)$ , where  $\times_i o^i \in \Omega_\times$ . Such partial observability is introduced by a limited communication range between UAVs and devices, which is determined by the SNR level of the link. A UAV can communicate with a device if the SNR value of the link between them is greater than or equal to a threshold  $\text{SNR}_{thr}$ . The same assumption is also applied to the communication between two UAVs.

Moreover, we denote the individual local action-observation history of agent  $i$  by  $\tau_t^i = (o_0^i, \mathbf{a}_0^i, o_1^i, \dots, \mathbf{a}_{t-1}^i, o_t^i)$  and the joint action-observation history by  $\times_i \tau^i$ . Each agent takes action according to its policy  $\pi^i \in \Pi^i : \Omega^i \rightarrow \mathcal{A}^i$ , and the joint action-value function under joint policy  $\pi$  is defined as  $Q_{tot}^\pi(s, \times_i \mathbf{a}^i) = \mathbb{E}_\pi [\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \times_i \mathbf{a}_t^i = \times_i \mathbf{a}^i]$ , where  $\gamma \in [0, 1]$  is the discount factor. Additionally, the indices of all agents except agent  $i$  are denoted by  $-i$ .

**Action space.** The action space is defined in (1). Furthermore, we design a safety controller to only provide feasible actions for the agents to choose from. The safety controller ensures that no collisions with map boundaries occur and all agents reach the destination before their batteries run out by continuously comparing the distance to the destination and remaining battery levels. Note that agents with no remaining battery can only choose the no-op action. Consequently, we denote the feasible action space of agent  $i$  at time step  $t$  checked by the safety controller by  $\mathcal{A}_t^{i,sc}$ , and the minimum battery required to reach the destination by  $b_t^{i,sc}$ .

**Observation.** The observation of agent  $i$  at time step  $t$  is denoted by  $o_t^i = (o_{t,1}^i, o_{t,2}^i, o_{t,3}^i, o_{t,4}^i)$ . To be specific,  $o_{t,1}^i = \{\mathbf{a}_t^j\}_{\mathbf{a}_t^j \in \mathcal{A}_t^{i,sc}}$  consists of the feasible actions determined by the safety controller.  $o_{t,2}^i$  includes the features between agent  $i$  and all the devices, i.e.,  $o_{t,2}^i = \{\text{SNR}_t^{i,k}, \chi_t^{i,k}, D_t^k, d_{t,x}^{i,k}, d_{t,y}^{i,k}, q_t^{i,k}\}_{\forall k \in \mathcal{U}}$ , where  $\chi_t^{i,k} \in \{0, 1\}$  is a binary variable, indicating whether device  $k$  is reachable by agent  $i$ . Specifically, if  $\text{SNR}_t^{i,k} \geq \text{SNR}_{thr}$ , then  $\chi_t^{i,k} = 1$ ; otherwise,  $\chi_t^{i,k} = 0$  and  $D_t^k$  in  $o_{t,2}^i$  is set to zero. We denote the relative distance between agent  $i$  and device  $k$  along  $x$  and  $y$  axis by  $d_{t,x}^{i,k} = x_t^i - x^k$  and  $d_{t,y}^{i,k} = y_t^i - y^k$ , respectively. Similarly,  $o_{t,3}^i$  includes the features between agent  $i$  and other agents, i.e.,  $o_{t,3}^i = \{\text{SNR}_t^{i,j}, \chi_t^{i,j}, d_{t,x}^{i,j}, d_{t,y}^{i,j}, b_t^j\}_{\forall j \in \mathcal{I}-i}$ . Notably, if  $\chi_t^{i,j} = 0$ , then all elements related to agent  $j$  in  $o_{t,3}^i$  except the SNR value are set to zero. The last term of the observation includes individual features, i.e.,  $o_{t,4}^i = (b_t^i, b_t^{i,sc})$ .

**State.** The global state contains the information of all the UAVs and devices, regardless of the SNR value, i.e.,  $s_t = (s_{t,1}, s_{t,2})$ , where  $s_{t,1} = \{b_t^i, b_t^{i,sc}, x_t^i, y_t^i, \text{done}\}_{i \in \mathcal{I}}$  includes the features of all the UAVs,  $\text{done} \in \{0, 1\}$  indicates whether the UAV's battery is fully depleted, and  $s_{t,2} = \{D_t^k, x^k, y^k\}_{k \in \mathcal{U}}$  contains the features of all the devices. This global state is only available during centralized training.

**Reward.** The joint reward received at each time step  $t$  is defined as the amount of data collected by all the UAVs which is given by

$$r_t = \sum_{i=1}^I \sum_{k=1}^K q_t^{i,k} C_t^{i,k} \Delta t. \quad (10)$$

#### B. QMIX

To solve the Dec-POMDP, we adopt the popular QMIX [11] algorithm, which non-linearly factorizes the joint action-value function using a mixing network with leverage of the global state information, as described by

$$Q_{tot}(\times_i \tau^i, \times_i \mathbf{a}^i) = \text{Mix}(s, Q_1(\tau^1, \mathbf{a}^1), \dots, Q_I(\tau^I, \mathbf{a}^I); \theta), \quad (11)$$

where  $Q_i(\tau^i, \mathbf{a}^i)$  is the individual action-value function conditioned on the local observation-action history  $\tau^i$  and action  $\mathbf{a}^i$ , and  $\theta$  denotes the parameters of the QMIX model. We randomly sample a batch of  $B$  episodes from the replay buffer to train the QMIX model by minimizing the following loss

$$\mathcal{L}(\theta) = \sum_{b=1}^B \sum_{t=0}^{T-1} (y_{b,t}^{tot} - Q_{tot}(\times_i \tau_t^i, \times_i \mathbf{a}_t^i, s_t; \theta))^2, \quad (12)$$

where  $y_{b,t}^{tot} = r_{b,t} + \gamma \max_{\times_i \mathbf{a}^i} Q_{tot}(\times_i \tau_{t+1}^i, \times_i \mathbf{a}_{t+1}^i, s_{t+1}; \bar{\theta})$  denotes the temporal-difference (TD) target at time step  $t$  of the  $b$ -th episode,  $\bar{\theta}$  denotes the target network parameters.

#### IV. MODEL-AIDED FEDQMIX

Employing MARL algorithms directly in real-world scenarios is usually impractical since collecting large amounts of training data on physical systems is expensive and can lead to potential safety concerns [9]. To tackle these challenges, we propose a model-aided federated MARL algorithm named *model-aided FedQMIX* for planning multi-UAV trajectories in data harvesting missions while significantly reducing the requirement for real-world training data samples.

Our assumption about the environment setting lies between agnostic and fully informed. On the one hand, we do not assume prior knowledge of the wireless channel characteristics, which must be learned by the UAVs, nor do we presume knowledge of all devices' positions, which must be estimated to construct the simulated environment. On the other hand, a 3D map of the environment and position information of only a subset of devices (anchor devices) are known, which gives the UAVs the necessary knowledge to localize the devices with unknown positions and develop an accurate simulated model.

The proposed algorithm alternates between two parts: 1) learning a simulated environment from real-world measurements, and 2) training the QMIX model for trajectory planning via federated learning in the simulated environment. Specifically, UAVs collect measurements from ground devices in the real world, which are then used to learn the radio channel model and determine the unknown device locations, as introduced in Section IV-A. The learned information, combined with a 3D map, is used to build a simulated environment for each UAV agent. Subsequently, we train a global QMIX model by exploiting all the UAVs' resources via federated learning in the simulated environment, as detailed in Section IV-B.

##### A. Environment Learning

Akin to [10] but considering a multi-UAV setting, we employ a neural network to learn the radio channel and use particle swarm optimization (PSO) along with the learned radio channel to estimate unknown device locations.

During each real-world deployment and in addition to data collection, each UAV measures the channel gain between itself and the devices. Upon episode completion, all UAVs transmit the gathered measurements to a central server or a specified UAV for environment model learning. The channel

gain between the  $i$ -th UAV and the  $k$ -th device at time step  $t$  can be defined as a function  $\psi$  with parameters  $\vartheta$ , as follows

$$g_t^{i,k} = \begin{cases} \psi_{\vartheta}(d_t^{i,k}, \phi_t^{i,k}, \omega_t^{i,k} = 1) + \eta_{\text{LoS}}, & \text{if LoS,} \\ \psi_{\vartheta}(d_t^{i,k}, \phi_t^{i,k}, \omega_t^{i,k} = 0) + \eta_{\text{NLoS}}, & \text{if NLoS,} \end{cases} \quad (13)$$

where  $\phi_t^{i,k} = \arcsin(h^i/d_t^{i,k})$  is the elevation angle. The binary variable  $\omega_t^{i,k} \in \{0, 1\}$  indicates whether a measurement falls into LoS category for  $\omega_t^{i,k} = 1$  or NLoS otherwise. The shadowing effect  $\eta_z$  is characterized as  $\mathcal{N}(0, \sigma_z^2)$ . Note that both the function  $\psi$  and the parameters  $\vartheta$  are unknown and must be learned. Each measurement in (13) can be modeled as  $p(g_t^{i,k}) = (f_{t,\text{LoS}}^{i,k})^{\omega_t^{i,k}} (f_{t,\text{NLoS}}^{i,k})^{(1-\omega_t^{i,k})}$ , where  $f_{t,z}^{i,k} = \mathcal{N}(\psi_{\vartheta}(d_t^{i,k}, \phi_t^{i,k}, \omega_t^{i,k}), \sigma_z^2)$ . Akin to [10], the negative log-likelihood of the measurements is given by

$$\begin{aligned} \mathcal{L} = & \log \left( \frac{\sigma_{\text{LoS}}^2}{\sigma_{\text{NLoS}}^2} \right) \sum_{t=0}^T \sum_{i=1}^I \sum_{k=1}^K \omega_t^{i,k} \\ & + \sum_{t=0}^T \sum_{i=1}^I \sum_{k=1}^K \frac{\omega_t^{i,k}}{\sigma_{\text{LoS}}^2} \left| g_t^{i,k} - \psi_{\vartheta}(d_t^{i,k}, \phi_t^{i,k}, \omega_t^{i,k}) \right|^2 \\ & + \sum_{t=0}^T \sum_{i=1}^I \sum_{k=1}^K \frac{1 - \omega_t^{i,k}}{\sigma_{\text{NLoS}}^2} \left| g_t^{i,k} - \psi_{\vartheta}(d_t^{i,k}, \phi_t^{i,k}, \omega_t^{i,k}) \right|^2. \end{aligned} \quad (14)$$

Consequently, the problem of learning the channel model and estimating the device locations can be transformed into solving the optimization problem

$$\begin{aligned} \min_{\omega_t^{i,k}, \mathbf{u}^k, \forall t, \forall i, \forall k} \quad & \mathcal{L}, \\ \text{s.t.} \quad & \omega_t^{i,k} \in \{0, 1\}, \forall t, \forall i, \forall k. \end{aligned} \quad (15)$$

The above optimization problem is non-convex and hard to solve directly. Therefore, we decompose (15) into two sub-problems: i) radio channel learning, and ii) device localization. In phase one, the radio channel is learned by utilizing the known locations of anchor devices, and in the second phase, the learned channel from the previous step is combined with a PSO algorithm and the 3D map to localize the unknown devices. Due to the limited space, we refer to [10] for details.

Having estimated the unknown device locations and learned the radio channel, we utilize this information along with the map to build a simulated environment for training the MARL algorithm to design the trajectories for the UAVs.

##### B. QMIX Model Training via Federated Learning

Adapting the idea of federated learning to make efficient use of all UAVs' computational resources and accelerate the learning process, we create a digital replica of the environment locally at each UAV, also including virtual instances of all other UAVs. By running the same MARL algorithm simultaneously on each UAV and consolidating the trained models using federated learning, we obtain a global QMIX model.

While training locally in the simulated environments at each UAV, virtual agents may take different actions even if they have the same observations because of the random component in the  $\epsilon$ -greedy action selection strategy, leading to diverse

sets of state transitions in individual replay buffers. Federated learning can help to capitalize on the diversity of experiences in all UAVs' replay buffers by periodically consolidating locally learned QMIX models. This allows us to make efficient use of all available training experiences while avoiding the excessive communication overhead of centralized learning.

Specifically, each UAV  $i$  trains its QMIX network parameters  $\theta^i$  in its own simulated environment. Every  $N_{freq}$  episodes, all the UAVs periodically send their trained model to an aggregator, which computes a global model by averaging over local models as  $\theta = \frac{1}{I} \sum_{i \in \mathcal{I}} \theta^i$ . The aggregator then sends the global model back to each UAV, where training in their respective local simulated environments continues.

### C. Algorithm

The model-aided FedQMIX algorithm is summarized in Algorithm 1 and consists of three steps: 1) The UAVs employ the learned policy to generate trajectories for collecting data and measurements in the real world; 2) The acquired measurements are used to learn the radio channel and estimate unknown device locations, which are utilized to build the simulated environment; 3) The UAV agents train the QMIX model using federated learning in their respective simulations for a predefined number of episodes. The algorithm terminates after carrying out  $E_{max}$  real-world experiments.

## V. NUMERICAL RESULTS

We consider a 3D urban environment composed of city blocks with buildings of different heights. We design two types of map: the Return-Base Map (RBM) and the Reach-Destination Map (RDM), the top views of which are illustrated in Fig. 1(a) and Fig. 1(b), respectively. The RBM covers an area of 600m×800m with the same start and terminal positions for the UAVs located at the center of the map, while the RDM extends over a larger area of 1000m×1200m, where the UAVs are required to navigate from the start position situated at the lower left corner to the destination located at the upper right corner. The flying altitudes of the UAVs are set as 55m, 60m, and 65m, respectively. We adopt the same QMIX hyperparameters as in [11], except that we use the Adam optimizer with a learning rate of  $5 \times 10^{-4}$ . True propagation parameters are chosen similar to [10]. The federated learning aggregation period is set as  $N_{freq} = 50$  and a total of  $E_{max} = 30$  real-world episodes are executed during training. After completing a real-world episode, each UAV undergoes local training for  $N = 1000$  episodes in its simulated environment. This helps to reduce the need for expensive real-world experiments.

Fig. 1 shows two example trajectories generated by the model-aided FedQMIX algorithm for three UAVs. On both maps, the UAVs learn to efficiently divide the entire map into sub-regions that are then assigned to individual UAVs, without the need for centralized coordination. Total collected data is thereby maximized and energy wastage by UAVs congregating is avoided, which demonstrates the effectiveness of the learned cooperative behavior. In Fig. 1, the red crosses indicate the estimated locations of unknown devices in the final episode,

---

### Algorithm 1: Model-aided FedQMIX

---

```

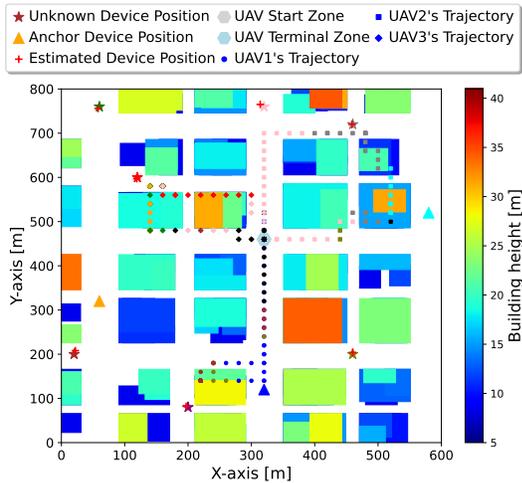
1: Initialize a set of  $I$  UAVs, replay buffers  $\mathcal{B}^i$ , the parameters  $\theta^i$ 
   of QMIX at the aggregator, local QMIX parameters  $\theta^i = \theta$ ,
   target network parameters  $\bar{\theta}^i = \theta^i$ , target network update
   period  $N_{target}$ , aggregation period  $N_{freq}$ .
2: for  $e = 0, 1, \dots, E_{max} - 1$  do
3:   1) Real-world experiment:
4:   UAVs use the policy derived from step 3) to plan trajectories
   for data collection while also gathering measurements.
5:   2) Learn the environment as described in Section IV-A
6:   3) Simulated-world experiment:
7:   for  $episode = 0, 1, \dots, N - 1$  do
8:     for Each UAV  $i \in \mathcal{I}$  in parallel do
9:        $t = 0$ , initialize state  $s_0$ 
10:      while  $b_t^i \geq 0, \forall j = 1, 2, \dots, I$  do
11:        for each simulated agent  $j = 1, 2, \dots, I$  do
12:           $\tau_t^j = \tau_{t-1}^j \cup \{(o_t^j, \mathbf{a}_{t-1}^j)\}$ 
13:          Choose action with  $\epsilon$ -greedy policy, i.e.,
           $\mathbf{a}_t^j = \begin{cases} \text{randomly select from } \mathcal{A}_t^{j,sc}, & \text{w.p. } \epsilon \\ \arg \max_{\mathbf{a}_t^j \in \mathcal{A}_t^{j,sc}} Q_j(\tau_t^j, \mathbf{a}_t^j), & \text{w.p. } 1 - \epsilon \end{cases}$ 
14:        end for
15:        Take joint action  $\times_j \mathbf{a}_t^j$ , observe  $\times_j o_{t+1}^j$ , get reward
           $r_t$  and next state  $s_{t+1}$ 
16:        Store  $(s_t, \times_j o_t^j, \times_j \mathbf{a}_t^j, r_t, s_{t+1}, \times_j o_{t+1}^j)$  in  $\mathcal{B}^i$ 
17:         $t = t + 1$ 
18:      end while
19:      Randomly sample a batch of  $B$  episodes from  $\mathcal{B}^i$ 
20:      for each time step  $t$  in each episode in the batch do
21:         $Q_{tot} = \text{Mix}(s_t, Q_1(\tau_t^1, \mathbf{a}_t^1), \dots, Q_I(\tau_t^I, \mathbf{a}_t^I); \theta^i)$ 
22:        Calculate target  $Q_{tot}$  using target network  $\bar{\theta}^i$ 
23:      end for
24:       $\theta^i \leftarrow \theta^i - \alpha \nabla \mathcal{L}(\theta^i)$  w.r.t.  $\theta^i$  using Eq. (12)
25:      if  $\text{mod}(episode, N_{target}) = 0$  then
26:        Reset  $\bar{\theta}^i = \theta^i$ 
27:      end if
28:    end for
29:    if  $\text{mod}(episode, N_{freq}) = 0$  then
30:      Update  $\theta = \frac{1}{I} \sum_{i \in \mathcal{I}} \theta^i$  and set  $\theta^i \leftarrow \theta, \forall i \in \mathcal{I}$ 
31:    end if
32:  end for
33: end for

```

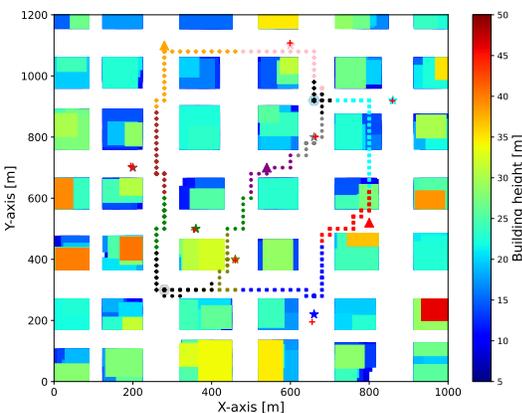
---

which are closely aligned with the actual device positions and showcase the accuracy of the learned environment model.

We compare our proposed algorithm with several benchmarks including the conventional QMIX algorithm without model learning, and fully decentralized independent Q-learning (IQL). Besides, we also train the QMIX model without federated learning, referred to as model-aided QMIX. In this approach, the training only occurs in one of the UAVs' simulated environments. Once the training is finished, the learned policy is shared with other UAVs. The performance comparison results of both maps are shown in Fig. 2. The IQL approach considerably underperforms QMIX due to the non-stationarity issue in independent learning methods [12]. The model-aided QMIX algorithm demonstrates a notable reduction in the need for real-world experiences while attaining comparable performance levels to the baseline QMIX algorithm. Significantly, our proposed model-aided FedQMIX algorithm shows superior performance compared with other approaches. It achieves the same performance as QMIX



(a) Return-Base Map (RBM)



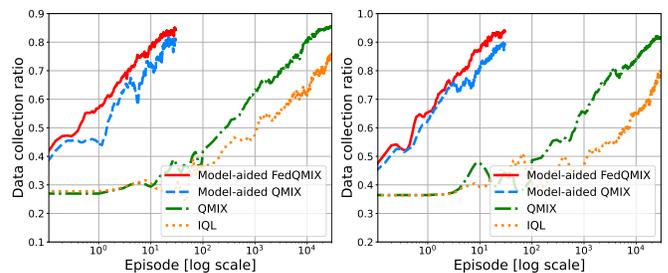
(b) Reach-Destination Map (RDM)

Fig. 1: Example trajectories, where the UAV's trajectory color corresponds to the device it is collecting data from, while black indicates no data collection. Anchor IoT devices are represented as triangles and devices with unknown locations as stars. a) RBM:  $K = 10$  devices with  $D_{init}^k = 16000$  initial data units to be picked up by three UAVs with  $b_0^i = 60$  initial battery units. b) RDM:  $K = 10$  devices with  $D_{init}^k = 20000$  initial data units to be picked up by three UAVs with  $b_0^i = 80$  initial battery units.

trained with real-world samples while requiring around three magnitudes less real-world data. Additionally, in comparison with model-aided QMIX, it converges faster and achieves better performance within the equivalent training timeframe. This can be attributed to the adoption of federated learning in training the QMIX model, which exploits the information from all UAVs' trained models, resulting in faster convergence and distributed computation load across UAVs.

## VI. CONCLUSION

We have proposed a novel model-aided FedQMIX algorithm for designing cooperative multi-UAV trajectories in data harvesting missions. By leveraging federated learning and training the QMIX model within a learned simulation environment, our approach has significantly accelerated the learning process and reduced the extensive requirement for real-world experiences while achieving the same data collection performance as the



(a) Return-Base Map (RBM) (b) Reach-Destination Map (RDM)

Fig. 2: Performance comparison of proposed model-aided FedQMIX and baseline algorithms. The x-axis indicates real-world training episodes on a logarithmic scale. Results are averaged over three random runs.

baseline methods. In future work, we plan to fully decentralize the environment model learning approach.

## ACKNOWLEDGMENTS

This work was partially funded by the HUAWEI France supported Chair on Future Wireless Networks at EURECOM, and by the German-French Academy for the Industry of the Future, under project 3CSI. Marco Caccamo was supported by an Alexander von Humboldt Professorship endowed by the German Federal Ministry of Education and Research.

## REFERENCES

- [1] J. Luo, Z. Wang, M. Xia, L. Wu, Y. Tian, and Y. Chen, "Path planning for UAV communication networks: Related technologies, solutions, and opportunities," *ACM Computing Surveys*, vol. 55, no. 9, 2023.
- [2] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghayeb, "Age of information aware trajectory planning of UAVs in intelligent transportation systems: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12 382–12 395, 2020.
- [3] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-UAV path planning for wireless data harvesting with deep reinforcement learning," *IEEE Open Journal of the Communications Society*, vol. 2, 2021.
- [4] X. Wang, M. Yi, J. Liu, Y. Zhang, M. Wang, and B. Bai, "Cooperative data collection with multiple UAVs for information freshness in the internet of things," *IEEE Transactions on Communications*, 2023.
- [5] H. Wang, C. H. Liu, H. Yang, G. Wang, and K. K. Leung, "Ensuring threshold AoI for UAV-assisted mobile crowdsensing by multi-agent deep reinforcement learning with transformer," *IEEE/ACM Transactions on Networking*, 2023.
- [6] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: Techniques, applications, and open challenges," *arXiv preprint arXiv:2108.11887*, 2021.
- [7] A. Shahbazi, I. Donevski, J. J. Nielsen, and M. Di Renzo, "Federated reinforcement learning UAV trajectory design for fast localization of ground users," in *European Signal Processing Conference (EUSIPCO)*, 2022, pp. 663–666.
- [8] S. Wu, W. Xu, F. Wang, G. Li, and M. Pan, "Distributed federated deep reinforcement learning based trajectory optimization for air-ground cooperative emergency networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 9107–9112, 2022.
- [9] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.
- [10] O. Esrafilian, H. Bayerlein, and D. Gesbert, "Model-aided deep reinforcement learning for sample-efficient UAV trajectory design in IoT networks," in *IEEE Global Communications Conference*, 2021.
- [11] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.
- [12] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.