Model-aided Deep Reinforcement Learning for Sample-efficient UAV Trajectory Design in IoT Networks

Omid Esrafilian, Harald Bayerlein, David Gesbert

Model-aided Deep Reinforcement Learning for Sample-efficient UAV Trajectory Design in IoT Networks

Omid Esrafilian, Harald Bayerlein, and David Gesbert
Communication Systems Department, EURECOM, Sophia Antipolis, France
{omid.esrafilian, harald.bayerlein, david.gesbert}@eurecom.fr

***Abstract*—Deep Reinforcement Learning (DRL) is gaining attention as a potential approach to design trajectories for autonomous unmanned aerial vehicles (UAV) used as flying access points in the context of cellular or Internet of Things (IoT) connectivity. DRL solutions offer the advantage of on-the-go learning hence relying on very little prior contextual information. A corresponding drawback however lies in the need for many learning episodes which severely restricts the applicability of such approach in real-world time- and energy-constrained missions. Here, we propose a model-aided deep Q-learning approach that, in contrast to previous work, considerably reduces the need for extensive training data samples, while still achieving the overarching goal of DRL, i.e to guide a battery-limited UAV towards an efficient data harvesting trajectory, without prior knowledge of wireless channel characteristics and limited knowledge of wireless node locations. The key idea consists in using a small subset of nodes as anchors (i.e. with known location) and learning a model of the propagation environment while implicitly estimating the positions of regular nodes. Interaction with the model allows us to train a deep Q-network (DQN) to approximate the optimal UAV control policy. We show that in comparison with standard DRL approaches, the proposed model-aided approach requires at least one order of magnitude less training data samples to reach identical data collection performance, hence offering a first step towards making DRL a viable solution to the problem.**

I. Introduction

Rapid innovation in producing low-cost commercial unmanned aerial vehicles (UAVs) has opened up numerous opportunities in the UAV market which is projected to reach 63.6 USD billion by 2025 [1]. One key application scenario is the future Internet of Things (IoT), in which harvesting data from wireless nodes that are spread out over wide areas far away from base stations (BSs) generally requires higher transmission power to communicate the information, reducing the network's operating duration by draining the sensor battery faster. A UAV that acts as a flying BS can describe a flight pattern that brings it in close range to the ground nodes, hence reducing battery consumption and increasing the energy efficiency of the data harvesting system. However, delivering this gain hinges on the availability of efficient methods to design a trajectory for the UAV, deciding when and where to collect data from ground nodes.

This work was partially supported by the French government, through the 3IA Côte d'Azur project number ANR-19-P3IA-0002, as well as by the TSN CARNOT Institute under project Robots4IoT.

The popularity of deep reinforcement learning (DRL) in this context can be explained by the fact that full information about the scenario environment (e.g. IoT sensor positions) is not a prerequisite. Further reasons include the computational efficiency of DRL inference, as well as the inherent complexity of UAV path planning, which is in general non-convex and often NP-hard [2], [3]. However, one of the greatest obstacles to deploying DRL-based path planning to real-world autonomous UAVs is the prohibitively extensive training data required [4], equivalent to thousands of training flights. In this work, we address this issue by proposing a so-called *model-aided DRL* approach that only requires a minimum of training data to control a UAV data harvester under a limited flight-time.

The training data demand of DRL methods for UAV path planning depends in large parts on the scenario complexity and the availability of prior information about the environment. On the one hand, works such as [5], where a deep Q-network (DQN) is trained to control an energy-limited UAV BS, assume absolutely no prior knowledge of the environment, requiring large amounts of training even in a simple environment as the DRL agent has to deduce the scenario conditions purely by trial and error. On the other hand, near perfect state information in works such as [6], where cooperative UAVs are tasked with collecting data from IoT devices in a relatively simple unobstructed environment, enables faster convergence and requires less training data. In this work, prior knowledge available to the UAV agent is in between the two extremes: while some reference IoT node positions are known (referred to as anchors), other node positions and the challenging wireless channel characteristics in a dense urban environment that causes alternation between line-of-sight (LoS) and non-line-of-sight (NLoS) links, must be estimated.

In the context of sample-efficient RL, model-accelerated solutions have been proposed previously for a variety of applications. A method called *imagination rollouts* to increase sample-efficiency for a continuous Q-learning variant has been suggested for simulated robotic tasks in [7]. Their approach is based on using iteratively refitted time-varying linear models, in contrast to a neural network (NN) model that we propose here. Learning a NN model in the context of stochastic value gradient learning methods has been proposed in [8].

Other works in the area of RL trajectory optimization for UAV communications have suggested other ways of reducing

training data demand. Li *et al.* [3] proposed a DRL method for sum-rate maximization from moving users based on transfer learning to reduce training time. In [9], the authors propose meta-learning on random user uplink access demands for distributed UAV BS control, reducing training time by around 50% compared to standard RL. Another possibility as proposed in [10] is to directly generalize training over a range of likely scenario parameters. The DRL agent then requires no retraining when scenario parameters randomly change at the cost of longer initial training and the requirement for the change being observable for the agents.

To the best of our knowledge, this is the first work that proposes model-based acceleration of the training process in DRL UAV path planning and also the first one that suggests the use of anchor nodes. Our contributions are as follows:

- We propose a novel model-aided deep reinforcement learning UAV trajectory planning algorithm for data collection from IoT devices that requires a minimum of expensive real-world training data.
- By introducing a device localization algorithm that exploits a limited number of reference device positions and a city 3D map, we show that our proposed method offers fast convergence even under uncertainty about device positions and without prior knowledge of the challenging radio channel conditions in a dense urban environment.
- We compare our model-aided approach to the baselines of standard DRL without any prior information as well as map-based full knowledge DRL and show, that our approach achieves a reduction in training data demand of at least one order of magnitude with identical data collection performance.

II. System Model and Problem Formulation

We consider a wireless communication system where a UAV-mounted flying BS is serving K static ground level nodes (IoT sensors) in an urban area. The k-th ground node, $k \in [1, K]$, is located at $\mathbf{u}_k \in \mathbb{R}^2$. The ground nodes are split into two groups: nodes with known locations $\mathbf{u}_k, k \in \mathcal{U}_{\text{known}}$, and nodes with unknown locations $\mathbf{u}_k, k \in \mathcal{U}_{\text{unknown}}$.

The UAV mission lasts for a maximum duration of T during which the UAV follows a trajectory with a constant velocity to maximize the amount of data collected from ground nodes. For the ease of exposition, we assume that the time period T is discretized into N equal time slots. The UAV position at time step n is denoted by $\mathbf{v}_n = [x_n, y_n, h]^{\mathrm{T}} \in \mathbb{R}^3$, where h represents the altitude of the drone. We also assume that the drone is equipped with a GPS receiver, hence the coordinates $\mathbf{v}_n,\ n \in [1, N]$ are known.

A. UAV Model

During the mission, the drone's position evolves as

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}_n\ ,\ \mathbf{a}_n \in \mathcal{A}, \tag{1}$$

where $\mathbf{a}_n$ is the UAV movement action, and $\mathcal{A}$ is the set of feasible actions for the UAV given by

$$\mathcal{A} = \left\{ \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{hover}}, \underbrace{\begin{bmatrix} c \\ 0 \\ 0 \end{bmatrix}}_{\text{right}}, \underbrace{\begin{bmatrix} -c \\ 0 \\ 0 \end{bmatrix}}_{\text{left}}, \underbrace{\begin{bmatrix} 0 \\ c \\ 0 \end{bmatrix}}_{\text{up}}, \underbrace{\begin{bmatrix} 0 \\ -c \\ 0 \end{bmatrix}}_{\text{down}} \right\}, \tag{2}$$

where c is the distance that the UAV travels within each time step. Moreover, the UAV is subject to a limited flying time depending on its battery budget. We indicate the remaining battery budget of the UAV at n-th time step by $b_n \in \mathbb{R}$ and it changes according to

$$b_{n+1} = \begin{cases} b_n - 0.5, & \mathbf{a}_n = \text{hover} \\ b_n - 1, & \text{otherwise.} \end{cases} \tag{3}$$

B. Channel Model

We now describe the radio channel model that is used for computing the channel gains between the UAV and the ground nodes. Note that the channel model and the channel parameters are unknown to the UAV. Classically, the channel gain between two radio nodes which are separated by distance d meters in dB is modeled as [11]

$$g_z = ß_z - 10\,\alpha_z \log_{10}(d) + \eta_z, \tag{4}$$

where α_z is the path loss exponent, $ß_z$ is the log of average channel gain at the reference point $d = 1\text{m}$, η_z stands for the shadowing component that is modeled as a Gaussian random variable with $\mathcal{N}(0, \sigma_z^2)$. $z \in \{\text{LoS}, \text{NLoS}\}$ emphasizes the strong dependence of the propagation parameters on the LoS or NLoS condition. Note that (4) represents the logarithm of the channel gain which is averaged over the small scale fading of unit variance.

C. Problem Formulation

We are seeking to find an optimal trajectory for the UAV to maximize the overall collected data from all ground nodes within the UAV mission time. We assume that the ground nodes are served by the drone in a time-division multiple access (TDMA) manner where all ground nodes have an equal communication time access to the channel and are served sequentially. The ground node scheduling is performed automatically by the UAV and is not part of the optimization problem. Hence, for the node k at time step n, the maximum throughput is given by

$$C_{k,n} = \frac{1}{K} \log_2 \left(1 + \frac{P 10^{0.1 g_{n,k}}}{\sigma^2} \right), \tag{5}$$

where K is the number of ground nodes and $\frac{1}{K}$ is the normalization factor capturing the TDMA channel sharing effect, $g_{n,k}$ is the channel gain between the k-th node and the UAV at time step n, P denotes the up-link transmission power of the ground node, and the additive white Gaussian noise power at the receiver is denoted by σ^2.

We can now formulate the problem of maximum data collection by taking into account the UAV mobility constraints as follows

$$\max_{\mathbf{a}_n} \sum_{k\in[1,K]} \sum_{n\in[1,N]} C_{k,n} \tag{6a}$$

$$\text{s.t. (1), (3)} \tag{6b}$$

$$\mathbf{v}_1 = \mathbf{v}_\text{I}, \mathbf{v}_N = \mathbf{v}_\text{F} \tag{6c}$$

$$b_N \geq 0, \tag{6d}$$

where (6a) is the total collected data from all nodes during the mission, $\mathbf{v}_\text{I}, \mathbf{v}_\text{F}$ are, respectively, the starting and the final points of the trajectory, and (6d) guarantees that there is enough battery power to reach the terminal point. This problem is challenging to solve, since the objective function (6a) is highly non-convex and also the channel model and some of the ground nodes locations are not available at the UAV side.

III. Markov Decision Process and Q-Learning

To solve problem (6), we first reformulate it as a Markov decision process (MDP) which is defined by a 4-tuple $(\mathcal{S}, \mathcal{A}, P_\mathbf{a}, R_\mathbf{a})$ with state space $\mathcal{S}$, actions space $\mathcal{A}$, the state transition probability function $P_\mathbf{a}$ giving the probability that action $\mathbf{a}$ in state s at time step n will lead to state s' in the next time step, and the reward function $R_\mathbf{a}(s, s')$ which yields the immediate reward received after transitioning from state s to state s' by taking action $\mathbf{a}$. In our problem, each state comprises two elements which is given by $s_n = (\mathbf{v}_n, b_n)$, and the action space $\mathcal{A}$ is defined in accordance with (2).

The reward function consists of two components

$$r_n = \sum_{k\in[1,K]} C_{k,n} - \lambda_n, \tag{7}$$

where $r_n \triangleq R_\mathbf{a}(s, s')$. The first term in (7) is the instantaneous collected data from all nodes at the n-th time step, and λ_n is a penalty imposed by the safety controller that guarantees the UAV will reach the terminal point $\mathbf{v}_\text{F}$. Specifically, the safety controller at each time step computes the shortest trajectory (a minimum set of actions) and the minimum required power for getting to the destination point from the current UAV location, then based on these values it declines or accepts the current action $\mathbf{a}_n$ chosen by the UAV. If action $\mathbf{a}_n$ is rejected, a penalty term will be added to the reward function. The shortest trajectory and the minimum required power computed at the n-th time step by the safety controller are denoted by $\mathcal{A}_n^{sc}$ and b_n^{sc}, respectively. Thus, the safety penalty λ_n is given by

$$\lambda_n = \begin{cases} \lambda, & b_n \leq b_n^{sc} \\ 0, & \text{else.} \end{cases} \tag{8}$$

The action chosen by the UAV at each time step is checked and modified (if necessary) by the safety controller as follows

$$\mathbf{a}_n = \begin{cases} \mathbf{a}_{n,1}^{sc}, & b_n \leq b_n^{sc} \wedge \mathbf{a}_n \notin \mathcal{A}_n^{sc} \\ \mathbf{a}_n, & \text{else,} \end{cases} \tag{9}$$

where $\mathbf{a}_{n,1}^{sc}$ is the first element of $\mathcal{A}_n^{sc}$.

To solve the MDP, we employ the popular Q-learning algorithm, a model-free RL technique, that enables us to directly compare our proposed method to the state-of-the-art from the literature. Note that, our aim is to reduce the real-world training data samples of Q-learning by model-aided acceleration with an *external* model that simulates the environment. Accordingly, the Q-learning algorithm is unchanged and follows the standard cycle of interaction between agent and environment to iteratively learn a policy $\pi(s)$ that tells the agent how to select actions given a certain state.

Q-learning relies on iteratively improving the state-action value function Q^π, a.k.a. Q-function. The Q-function represents an expectation of the total future reward when taking action $\mathbf{a}$ in state s and then following policy π. It is given by

$$Q^\pi(s, \mathbf{a}) = \mathrm{E}_\pi \left[\sum_{m=n}^{N} \gamma^{m-n} r_n | s_n = s, \mathbf{a}_n = \mathbf{a} \right], \tag{10}$$

with discount factor $\gamma^{m-n} \in [0, 1]$ striking a balance between the importance of immediate and future rewards.

In large state-action spaces, the Q-function is commonly approximated by a Deep Q-network (DQN) with the neural network parameters θ [5]. When training the DQN $Q^\pi(s, \mathbf{a}; \theta)$, instability can occur. Experience replay, where experience tuples $(s_n, \mathbf{a}_n, r_n, s_{n+1})$ are stored to be reused for training by the agent, and a separate target network with parameters $\hat{\theta}$ have become standard techniques to mitigate the risk of training instability [12]. Accordingly, the loss function at each time step to train the DQN is given by

$$\ell(\theta) = \mathrm{E}\left[\left(r + \gamma \max_{\mathbf{a}} Q^\pi(s', \mathbf{a}; \hat{\theta}) - Q^\pi(s, \mathbf{a}; \theta) \right)^2 \right], \tag{11}$$

where $Q^\pi(s_{n+1}, \mathbf{a}; \hat{\theta})$ is the target network and with time index n omitted and s_{n+1} abbreviated to s' for brevity.

IV. Model-aided Deep Q-Learning

Employing standard deep Q-learning is often not practical due to the tremendous amount of training data points required and the cost associated with obtaining these data points, i.e. through real-world UAV experiments. To ameliorate this problem, we propose an algorithm where the agent learns an environment model continuously while collecting real-world measurements. This model is then used by the agent to simulate experiments and supplement the real-world data.

More specifically in our scenario, the next state s_{n+1} given the current state s_n and action $\mathbf{a}_n$ can be computed from (1), (3). The reward function (7) consists of two parts: the safety penalty, which is known from (9), and the instantaneous collected data from the IoT node devices. Therefore we only need to estimate the instantaneous collected data from devices which according to (5), (4), is a function of ground node locations and the radio channel model. Hence, the approximation of the reward function boils down to ground node localization and radio channel learning from collected radio measurements.

The problem of simultaneous wireless node localization and channel learning has been studied in previously [13]. In this

section, we propose a new approach of model-free node localization by leveraging the 3D map of the environment. Akin to [13], a LoS/NLoS segmented radio channel is assumed. However, in contrast to [13], our goal here is to estimate the radio channel using a model-free method while localizing the ground nodes. To learn the radio channel, we use a neural network (NN). This network is utilised along with a particle swarm optimization (PSO) technique and a 3D map of the city to localize the wireless nodes with unknown positions.

A. Simultaneous Node Localization and Channel Learning

We assume the UAV follows an arbitrary trajectory denoted by $\chi = \{\mathbf{v}_n, n \in [1, N]\}$ for collecting received signal strength (RSS) measurements, where $\mathbf{v}_n$ represents the UAV's position in the n-th time interval. We also assume that the UAV collects radio measurements form all K nodes at each location. Let $g_{n,k}$ represent the RSS measurements (in dB scale) obtained from the k-th node by the UAV in the n-th interval. Assuming a LoS/NLoS segmented pathloss model that is suitable for air-to-ground channels in urban environments with buildings [11], we have

$$g_{n,k}=\begin{cases}\psi_{\boldsymbol{\theta}}(d_{n,k},\phi_{n,k},w_{n,k}{=}1)+\eta_{n,k,\text{LoS}} & \text{if LoS}\\ \psi_{\boldsymbol{\theta}}(d_{n,k},\phi_{n,k},w_{n,k}{=}0)+\eta_{n,k,\text{NLoS}} & \text{if NLoS,}\end{cases} \tag{12}$$

where $d_{n,k} = \|\mathbf{u_k} - \mathbf{v_n}\|$, and $\phi_{n,k} = \arcsin(\frac{\bar{d}_{n,k}}{d_{n,k}})$ is the elevation angle between the UAV at time step n and node k with $\bar{d}_{n,k}$ representing the ground distance between the ground node and the UAV. $\omega_{n,k} \in \{0, 1\}$ is the classification binary variable (yet unknown) indicating whether a measurement falls into the LoS or NLoS category. The function $\psi_{\boldsymbol{\theta}}(.)$ is the channel model parameterized by $\boldsymbol{\theta}$. Note that, neither function $\psi(.)$ nor parameters $\boldsymbol{\theta}$ are known and need to be estimated. $\eta_{n,k,z}$ stands for the shadowing effect with zero-mean Gaussian distribution with variance σ_z^2. The probability distribution of a single measurement in (12) is modeled as

$$p(g_{n,k}) = (f_{n,k,\text{LoS}})^{w_{n,k}}(f_{n,k,\text{NLoS}})^{(1-w_{n,k})}, \tag{13}$$

where $f_{n,k,z}$ has a Gaussian distribution with $\mathcal{N}(\psi_{\boldsymbol{\theta}}(d_{n,k}, \phi_{n,k}, w_{n,k}), \sigma_z^2)$.

Assuming that collected measurements conditioned on the channel and node positions are independent and identically distributed (i.i.d) [11], using (13), the negative log-likelihood of measurements leads to

$$\begin{aligned}\mathcal{L} =& \log\left(\frac{\sigma^2_{\text{LoS}}}{\sigma^2_{\text{NLoS}}}\right)\sum_{k=1}^{K}\sum_{n=1}^{N}\omega_{n,k}+\\ &\sum_{k=1}^{K}\sum_{n=1}^{N}\frac{\omega_{n,k}}{\sigma^2_{\text{LoS}}}\left|g_{n,k}-\psi_{\boldsymbol{\theta}}(d_{n,k},\phi_{n,k},w_{n,k})\right|^2+\\ &\sum_{k=1}^{K}\sum_{n=1}^{N}\frac{(1-\omega_{n,k})}{\sigma^2_{\text{NLoS}}}\left|g_{n,k}-\psi_{\boldsymbol{\theta}}(d_{n,k},\phi_{n,k},w_{n,k})\right|^2.\end{aligned} \tag{14}$$

The estimate of $\psi(.), \boldsymbol{\theta}$, and $\mathbf{u_k}$ can then be obtained by solving

$$\min_{\substack{\omega_{n,k},\,\mathbf{u}_k,\forall n,\forall k\\ \psi(.),\boldsymbol{\theta}}} \quad \mathcal{L} \tag{15a}$$

$$\text{s.t.} \quad \omega_{n,k} \in \{0, 1\}, \forall n, \forall k. \tag{15b}$$

The binary variables $\omega_{n,k}$ in objective function (14), and the fact that $\psi(.)$ is not explicitly known and is a nonlinear function of node locations, make problem (15) challenging to solve since it is a joint classification, channel learning and node localization problem. To tackle this difficulty, we split (15) into two sub-problems of learning the channel and localizing nodes. We also leverage the 3D map of the city for the measurements classification which will be discussed next.

1) Radio Channel Learning: Our aim is to learn the radio channel using collected radio measurements from the IoT nodes with known location (anchor nodes). Since the characteristic of the radio channel is independent of the node location and only affected by the structure of the city and the blocking objects in the environment, learning the radio channel from the nodes with known location can provide a good approximation of the radio channel. The measurements are classified by leveraging the 3D map of the city, since for a node with known location the classification variables $\omega_{n,k}$ can be directly inferred from a trivial geometry argument: for a given UAV position, the node is considered in LoS to the UAV if the straight line passing through the UAV's and the node position lies higher than any buildings in between. Having classified the measurements, we use a neural network with parameters $\boldsymbol{\theta}$ as an approximation of $\psi_{\boldsymbol{\theta}}(.)$. The neural network accepts an input vector $[d_{n,k}, \phi_{n,k}, w_{n,k}]^{\text{T}}$ and returns an estimate of the channel gain $\hat{g}_{n,k}$. Therefore, problem (15) just by considering the anchor nodes can be rewritten as follows

$$\min_{\substack{\boldsymbol{\theta}\\ k\in\mathcal{U}_{\text{known}},\forall n}} \quad \mathcal{L}. \tag{16}$$

This optimization is a standard problem in machine learning and can be solved using any gradient-based optimizer. The parameters obtained by solving (16) are denoted by $\boldsymbol{\theta}^*$.

2) Node Localization: Having learned the radio channel, we continue to localize the unknown nodes. The optimization problem (15) for the set of unknown nodes and utilizing the learned radio channel can be reformulated as follows:

$$\min_{\substack{\omega_{n,k},\,\mathbf{u}_k,\forall n\\ k\in\mathcal{U}_{\text{unknown}}}} \quad \mathcal{L}^* \tag{17a}$$

$$\text{s.t.} \quad \omega_{n,k} \in \{0, 1\}, k \in \mathcal{U}_{\text{unknown}}, \forall n, \tag{17b}$$

where $\mathcal{L}^*$ is obtained by substituting the learned channel model $\psi_{\boldsymbol{\theta}^*}(.)$ in (14). The binary random variables $\omega_{n,k}$, and the non-linear and non-convex objective function $\mathcal{L}^*$ make problem (17) hard to solve. We use the PSO algorithm which is suitable for solving various non-convex and non-linear optimization problems. PSO is a population-based optimization technique that tries to find the solution to an optimization problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (or objective

function). The algorithm is initialized with a population of random solutions, called particles, and a search for the optimal solution is performed by iteratively updating each particle's velocity and position based on a simple mathematical formula (for more details on PSO see [14]). As will be clear later, the PSO algorithm is enhanced to exploit the side information stemming from the 3D map of the environment which improves the performance of node localization and reducew the complexity of solving (17), since the binary variable $\omega_{n,k}$ can be obtained directly from the 3D map [13].

For ease of exposition, we first solve (17) by assuming only one unknown node. Then we will generalize our proposed solution to the multi-node case. To apply the PSO algorithm, we define each particle to have the following form

$$\mathbf{c}_j = [x_j, y_j]^{\mathrm{T}} \in \mathbb{R}^2, j \in [1, C], \tag{18}$$

where C is the number of particles and each particle is an instance of the possible node location in the city. Therefore, by treating each particle as a potential candidate for the node location, the negative log-likelihood (14) for a given particle can be rewritten as follows

$$\mathcal{L}^*(\mathbf{c}_j^{(i)}) = \log\left(\frac{\sigma^2_{\mathrm{LoS}}}{\sigma^2_{\mathrm{NLoS}}}\right)\left|\mathcal{M}_{\mathrm{LoS},1,j}\right| + \sum_{z\in\{\mathrm{LoS},\mathrm{NLoS}\}}\sum_{n\in\mathcal{M}_{z,1,j}}\frac{1}{\sigma_z^2}\left|g_{n,1}-\psi_{\boldsymbol{\theta}^*}(d_{n,k},\phi_{n,k},z)\right|^2, \tag{19}$$

where $\mathbf{c}_j^{(i)}$ is the j-th particle at the i-th iteration of the PSO algorithm, and $\mathcal{M}_{z,1,j}$ is a set of time indices of measurements collected from node 1 which are in segment z by assuming that the location of node 1 is the same as particle j. To form $\mathcal{M}_{z,1,j}$, a 3D map of the city is utilized. For example, measurement $g_{n,1}$ is considered LoS, if the straight line passing through $\mathbf{c}_j^{(i)}$ and the drone location $\mathbf{v}_n$ lies higher than any buildings in between. Therefore, the best particle minimizing (19) can be obtained from solving the optimization

$$j^* := \arg\min_{j\in[1,C]} \mathcal{L}^*(\mathbf{c}_j^{(i)}), \tag{20}$$

where j^* is the index of the best particle which minimizes the objective function in (20). In the next iteration of the PSO algorithm, the position and the velocity of particles are updated and the algorithm repeats for τ iterations. The best particle position in the last iteration is considered as the estimate of the node location.

Note that for the multi-node case, without loss of optimality, the problem can be transformed to the multi single-node localization problem and then each problem can be solved individually. This stems from the fact that the radio channel is learned beforehand and is assumed to have the same characteristics for all the UAV-node links (the radio channel characteristics is assumed to be independent of the node locations).

B. Algorithm

The proposed Algorithm 1 iterates between three phases: 1) the agent uses a policy obtained from its Q-network in the real world to collect RSS measurements from ground nodes. 2) The collected measurements are used to learn the radio channel and localize the unknown nodes as described in Sections IV-A1 and IV-A2, respectively. 3) The agent performs a new set of experiments in the simulated environment under the learned radio channel model and the estimation of the node locations to train the Q-network. Then, the agent repeats the first phase of the algorithm by generating a new policy using the trained Q-network and the procedure continues until convergence of the Q-network training.

The experience replay buffers for real world and simulated world experiments are denoted $\mathcal{B}$ and $\tilde{\mathcal{B}}$, respectively. A new episode in phase 1 and phase 3 starts by resetting the time index, the initial UAV position and the battery budget (lines 7 and 22). To train the Q-network, an ϵ-greedy exploration technique is used (line 36) with decay constant κ. β is the learning rate for primary network parameters θ. Target network parameters are updated every N_{target} episodes. In phase 3, the algorithm performs I sets of experiments in the simulated world, and the whole algorithm terminates after carrying out E_{max} real-world experiments.

V. Numerical Results

We consider a dense urban city neighborhood comprising buildings and regular streets as shown in Fig. 2. The height of the buildings is Rayleigh distributed in the range of 5 to $40\,\mathrm{m}$ and the true propagation parameters are chosen similar to [10]. The UAV collects radio measurements from the ground nodes every $5\,\mathrm{m}$ and we assume that the altitude of the UAV is fixed to $60\,\mathrm{m}$ during the course of its trajectory. The mission time of each episode is fixed to $N = 20$ time steps with a fixed step size of $c = 50\,\mathrm{m}$. We assume there are six ground nodes. Only the locations of anchor nodes $\mathbf{u}_1$ and $\mathbf{u}_2$ are known to the UAV in advance. The UAV starts from $\mathbf{v}_{\mathrm{I}} = [100, 100, 60]^{\mathrm{T}}$ and needs to reach the destination point $\mathbf{v}_{\mathrm{F}} = [300, 400, 60]^{\mathrm{T}}$ by the end of the mission. To learn the channel, we use a NN with two hidden layers where the first layer has 60 neurons with $tanh$ activation function, and the second layer 30 neurons with $relu$ activation function. The Q-network comprises 2 hidden layers each with 120 neurons and $relu$ activation function.

In Fig. 1, we compare the performance of the baseline Q-learning algorithm as explained in Section III and akin to [5], with the proposed model-aided Q-learning algorithm. Moreover, we show the result of an algorithm similar to [10], where the mixed-radio map of the nodes is embedded in the state vector. To compute the mixed-radio map, the individual radio maps of all nodes are combined. Individual radio maps are computed using the 3D map of the city and assuming perfect knowledge node positions and the radio channel. The model-aided algorithm outperforms the other approaches since it merely requires 10 real-world experiment episodes to converge to the same performance level as other algorithms. The algorithm introduced in [10] is superior to the baseline since it uses more information, i.e. the map and perfect knowledge of node positions and the radio channel model.

Algorithm 1 Model-aided deep Q-learning trajectory design

Initialize replay buffer $(\mathcal{B}), (\tilde{\mathcal{B}})$
Initialize Q-network and target network parameters
Initialize $t = 0$
for $e = 0$ to E_{max} **do**
  $t = t + 1$
  1) Real-world experiment:
  Initialize $s_0 = (\mathbf{v}_\mathrm{I}, b_{max})$, $n = 0$
  while $b_n \geq 0$ **do**
    $\mathbf{a}_n = \arg\max_{\mathbf{a}} Q^{\pi}(s_n, \mathbf{a}, \theta)$
    Validate $\mathbf{a}_n$ using the safety controller (9)
    Observe $r_n, s_{n+1}, \gamma_{1,n}, \cdots, \gamma_{K,n}$
    Store $(s_n, \mathbf{a}_n, r_n, s_{n+1})$ on $(\mathcal{B})$
    Memorize $(\mathbf{v}_n, \gamma_{1,n}, \cdots, \gamma_{K,n})$
    $n = n + 1$
  end while
  2) Learning the environment:
  Learn the radio channel as described in Section IV-A1
  Localize unknown nodes as described in Section IV-A2
  3) Simulated-world experiment:
  for $i = 0$ to I **do**
    $t = t + 1$
    Initialize $\tilde{s}_0 = (\mathbf{v}_\mathrm{I}, b_{max})$, $n = 0$
    while $b_n \geq 0$ **do**
      $\tilde{\mathbf{a}}_n = \begin{cases} \text{randomly select from } \mathcal{A} & \text{with probability } \epsilon \\ \arg\max_{\mathbf{a}} Q^{\pi}(\tilde{s}_n, \mathbf{a}, \theta) & \text{else} \end{cases}$
      Validate $\tilde{\mathbf{a}}_n$ using the safety controller (9)
      Compute $\tilde{r}_n$ from (7), and $\tilde{s}_{n+1}$ from (1), (3)
      store $(\tilde{s}_n, \tilde{\mathbf{a}}_n, \tilde{r}_n, \tilde{s}_{n+1})$ on $\tilde{\mathcal{B}}$
      for $m = 0$ to M **do**
        Sample $(s_m, \mathbf{a}_m, r_m, s_{m+1})$ uniformly from $\{\mathcal{B} \cup \tilde{\mathcal{B}}\}$
        $y_m = \begin{cases} r_m & \text{if terminal} \\ r_m + \gamma \max_{\mathbf{a}} Q^{\pi}(s_{m+1}, \mathbf{a}, \hat{\theta}) & \text{else} \end{cases}$
        $\ell_m(\theta) = \mathrm{E}\left[(y_m - Q^{\pi}(s_m, \mathbf{a}_m, \theta))^2\right]$
      end for
      $\theta = \theta + \beta \frac{1}{M} \nabla_\theta \sum_{m=0}^{M} \ell_m(\theta)$
      $n = n + 1$
    end while
    $\epsilon = \epsilon_{final} + (\epsilon_{start} - \epsilon_{final}) \exp(-\kappa t)$
    if $(t \mod N_{target} = 0)$ **then** $\hat{\theta} = \theta$
  end for
end for

Fig. 2 shows the final trajectory after convergence. The UAV starts flying towards the closest node and hovers above for several time steps in order to maximize the amount of collected data, and then reaches the destination $\mathbf{v}_\mathrm{F}$. Moreover, the estimate of unknown node locations obtained at the last episode of the training phase of Algorithm 1 are shown and confirmed to be very close the true positions.

VI. Conclusion

We have introduced a novel model-accelerated DRL path planning algorithm for UAV data collection from distributed IoT nodes with only partial knowledge of the nodes' locations. In comparison to two standard deep Q-learning algorithms, using either full or no knowledge of sensor node locations, we have demonstrated that the model-aided approach requires at least one order of magnitude less training data samples to reach the same data collection performance.

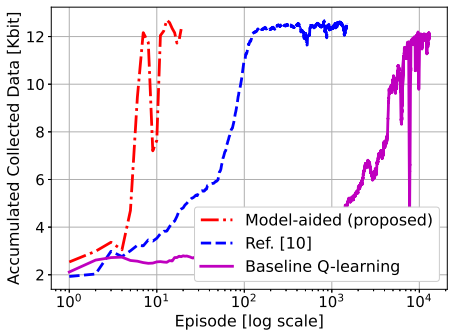

Fig. 1: Comparison of proposed model-aided, full-knowledge map-based [10], and no prior knowledge baseline Q-learning, showing accumulated collected data versus training episodes.

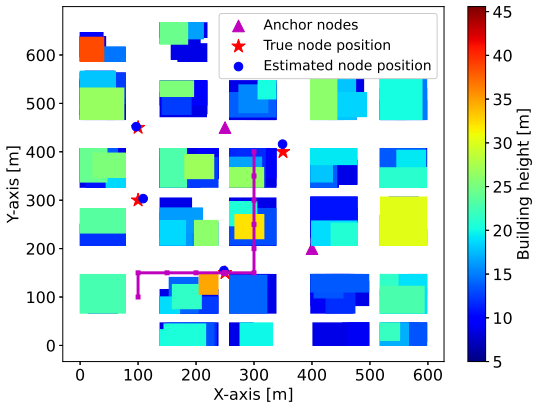

Fig. 2: Trajectory obtained by model-aided Q-learning and the estimates of unknown node locations in the final episode of Algorithm 1.

References

[1] L. Wood, "$63.6 bn drone service markets, 2025 - increasing use of drone services for industry-specific solutions - [news]," *Businesswire*, 17 Apr 2019. [Online]. Available: https://www.businesswire.com/news/home/20190417005302/en/

[2] Y. Zeng, Q. Wu, and R. Zhang, "Accessing from the sky: A tutorial on UAV communications for 5G and beyond," *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, 2019.

[3] X. Li, Q. Wang, J. Liu, and W. Zhang, "Trajectory design and generalization for UAV enabled networks: A deep reinforcement learning approach," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[4] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv:1904.12901*, 2019.

[5] H. Bayerlein, R. Gangula, and D. Gesbert, "Learning to rest: A Q-learning approach to flying base station trajectory design with landing spots," in *52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 724–728.

[6] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang, and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple UAVs," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3786–3800, 2021.

[7] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning (ICML)*, 2016.

[8] N. Heess, G. Wayne, D. Silver, T. Lillicrap, Y. Tassa, and T. Erez, "Learning continuous control policies by stochastic value gradients," in *28th International Conference on Neural Information Processing Systems*, 2015.

[9] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Distributed multi-agent meta learning for trajectory design in wireless drone networks," *arXiv:2012.03158*, 2020.

[10] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV path planning for wireless data harvesting: A deep reinforcement learning approach," in *IEEE Global Communications Conference*, 2020.

[11] J. Chen, U. Yatnalli, and D. Gesbert, "Learning radio maps for UAV-aided wireless networks: A segmented regression approach," in *IEEE International Conference on Communications (ICC)*, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] O. Esrafilian, R. Gangula, and D. Gesbert, "3D Map-based Trajectory Design in UAV-aided Wireless Localization Systems," *IEEE Internet of Things Journal*, 2020.

[14] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks (ICNN)*, 1995.