Trajectory Design for Unmanned Aerial Vehicles via Meta-Reinforcement Learning

Ziyang Lu*, Xueyuan Wang†, M. Cenk Gursoy*
*Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY
† School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China
Email:zlu112@syr.edu, xywang@cczu.edu.cn, mcgursoy@syr.edu

***Abstract*—This paper considers the trajectory design problem for unmanned aerial vehicles (UAVs) via meta-reinforcement learning. It is assumed that the UAV can move in different directions to explore a specific area and collect data from the ground nodes (GNs) located in the area. The goal of the UAV is to reach the destination and maximize the total data collected during the flight on the trajectory while avoiding collisions with other UAVs. In the literature on UAV trajectory designs, vanilla learning algorithms are typically used to train a task-specific model, and provide near-optimal solutions for a specific spatial distribution of the GNs. However, this approach requires retraining from scratch when the locations of the GNs vary. In this work, we propose a meta reinforcement learning framework that incorporates the method of Model-Agnostic Meta-Learning (MAML). Instead of training task-specific models, we train a common initialization for different distributions of GNs and different channel conditions. From the initialization, only a few gradient descents are required for adapting to different tasks with different GN distributions and channel conditions. Additionally, we also explore when the proposed MAML framework is preferred and can outperform the compared algorithms.**

I. Introduction

A. Background

1) Unmanned Aerial Vehicles: Unmanned aerial vehicles (UAVs) have been deployed in various applications due to their flexibility in positioning/placement and their ability to conduct tasks without human onboard. Indeed, due to their numerous benefits, UAVs in wireless networks have been intensively studied [1]. In [2], trajectory design in a UAV-assisted wireless network has been considered. In this work, the objective is to find trajectories with which the throughput is maximized while taking into account propulsion energy consumption. The authors tackle the problem with graph theory and utilize the shortest path algorithm. Other recent studies incorporate deep reinforcement learning (DRL) into the analysis. In [3], the optimization of the placement of multiple UAVs is addressed via DRL and the numerical results show that the proposed algorithm can outperform the benchmarks such as K-means algorithms. The work in [4] proposes a DRL-based framework for UAV trajectory design. During each time-limited episode, the goal is to have the DRL-assisted UAV to return to the starting point while maximizing the data collected from the ground nodes (GNs) during the flight. Prior work on DRL-assisted UAVs typically assumes fixed and known environment (e.g., fixed and known GN locations). If the environment varies, retraining from scratch is generally needed to have good performance.

In order to overcome this challenge, we propose a framework that utilizes meta-reinforcement in UAV trajectory design. Our goal is to let the UAV make adaptive decisions in different environments with the least amount of information.

2) Meta-Reinforcement Learning: The concept of meta learning is inspired by the fact that humans can learn quickly in an unseen task from their previous experiences in similar tasks. With meta learning, agents can learn from their experience in a set of similar tasks and automatically find the hyperparameters and architectures of the models for an unseen task with few training samples. Meta-reinforcement learning is basically meta-learning framework applied to the reinforcement learning algorithm.

A novel meta-learning algorithm called Model-Agnostic Meta-Learning (MAML) was first proposed in [5]. The benefit of MAML compared to the other meta-learning algorithms is that it is easy to implement and is independent of model structure. Instead of learning a task-specific model that exhibits strong performance in the corresponding task, MAML algorithm learns an efficient initialization of the deep neural network (DNN) for similar tasks through gradient descent. It is assumed that there exist tasks following a distribution $p(T)$, MAML algorithm continuously samples batches of similar tasks from $p(T)$ for learning the initialization ϕ. The goal is to obtain an efficient initialization ϕ^*, from which only one or a few gradient descents are required to adapt to an unseen task $T_i \sim p(T)$. The convergence of MAML is sensitive to hyper-parameters in the algorithm. Besides, implementing MAML involves second derivatives when performing backpropagation, incurring a large computational cost. To address the issue of stability, authors in [6] propose MAML++, a framework that contains various schemes for stabilizing the training of MAML. Regarding the computational cost, authors in [5] note that omitting the second derivative will not significantly diminish the performance and hence MAML can be reduced to first-order MAML (FOMAML). Several other first-order approximations of MAML are also proposed in recent work, including Reptile [7], Hessian-free MAML (HF-MAML) [8] and Evolution-Strategies MAML (ES-MAML) [9]. In our work, FOMAML is considered in the proposed framework.

In this paper, we address the UAV trajectory design problem with randomly distributed GNs. More specifically, in our setting, the UAV is given the task of collecting data from a set of GNs that are randomly distributed in the serving area. Our key contributions can be summarized as follows:

- We construct trajectories from a fixed starting point to a fixed destination while maximizing the overall information collected from all the GNs during the flight
- We take into account practical requirements such as maximum flight duration, UAV kinematic constraints (e.g., that limit the turning angle of the UAV), and collision avoidance with other UAVs (whose policies and trajectories are a priori unknown and whose position and speed can be acquired only within a certain sensing range).
- We design a meta-reinforcement learning agent that makes dynamic decisions for the trajectory without prior access to the GNs' spatial distribution and becomes aware of other UAVs in real time only if they are within the sensing range. It is important to note that there is no centralized control for the UAVs, and each UAV has its own decision-making policies.
- We provide comparisons with conventional DRL and joint-learning frameworks.
- We demonstrate that the proposed meta-reinforcement

learning framework can learn an efficient initialization and only a few gradient descents and training data are required for obtaining a task-specific model that provides strong performance in an unexperienced environment with previously unseen GN distribution.

II. Problem Statement

A. Channel Model

We assume that the line-of-sight (LOS) links are dominant in the air-to-ground channels due to the high altitude of the UAV [10]. The path loss can then be expressed as

$$L(d) = (d^2 + H_v^2)^{\Gamma/2} \tag{1}$$

where d is the horizontal distance between the GN and the UAV, H_v is the height of UAV, and Γ is the path loss exponent.

It is further assumed that antennas at the GNs are omnidirectional with the gain of $G_n = 0$ dB. The UAV is equipped with a receiver with a horizontally oriented antenna. The antenna gain provided by the UAV can then be expressed as

$$G_V(d) = \sin(\omega) = \frac{H_v}{\sqrt{d^2 + H_v^2}} \tag{2}$$

where ω is the elevation angle between the GN and the UAV. We note that the ensuing analysis is applicable to any choice of antenna gains, and the above antenna gains are selected for the sake of being concrete.

Now, the signal-to-noise ratio (SNR) at the UAV from the n^{th} GN can be formulated as

$$\text{SNR}_n = \frac{P}{\sigma^2} G_V(d_n) L^{-1}(d_n) = \frac{P}{\sigma^2} H_v (d_n^2 + H_v^2)^{-\frac{1+\Gamma}{2}}. \tag{3}$$

If we assume that the UAV connects to the n^{th} GN at time t, the transmission rate is,

$$\text{Rate}_t = \log_2(1 + \text{SNR}_n). \tag{4}$$

We assume that the height H_v of the UAV is fixed. Therefore, the transmission rate between the UAV and the connected GN varies depending on the horizontal distance d_n between them.

B. UAV Modeling and Collision Avoidance

We assume that there is one UAV collecting data from G GNs in the serving area and it has a maximum flying time of T. The UAV is required to arrive at the destination at the end of time slot T. The UAV flies from the fixed starting point (SP) to a fixed final area (FA) while maximizing the data collected from the GNs that are randomly distributed in the serving area.

In each time slot, the UAV can move horizontally with a turning angle that is assumed to be within $[-\pi/3, \pi/3]$ due to the kinematic constraints. We assume that the spatial distribution of GNs follows a two-dimensional Gaussian distribution around a cluster center. And the cluster center is randomly generated in the serving area.

We further assume that there exist other UAVs flying in the serving area, whose flight policies and trajectories are unknown beforehand. The data-collecting UAV considered in this work is equipped with sensors that can detect the location and speed of the other UAVs only within a certain sensing range. This information is required for collision avoidance.

We assume that the location information of the GNs is unavailable to the data-collecting UAV and the goal of this work is to design a framework that enables the UAV to determine effective trajectories under different spatial distributions of the GNs in the presence of other UAVs, without the knowledge of such dynamics in the environment.

III. Decision-Making UAV Agent

It is assumed that the UAV is equipped with a DRL agent for dynamically determining the trajectory. The DRL agent does not have prior access to the GNs' spatial distribution on the ground, and only becomes aware of the other UAVs in real time if they are within the sensing range. In time slot t, the DRL agent interacts with the environment by taking action a_t based on its current state s_t. Subsequently, it receives a reward r_t and learns a policy by updating the DNN parameters with the experience tuple (s_t, a_t, r_t).

A. Problem Formulation

As stated at the end of Section II, the goal for the UAV is to maximize the total data collected during each flight. It is required that the UAV reaches its destination within T time slots due to its power constraint. The second constraint is that the UAV should not leave the serving area for safety. Yet another constraint is that the UAV does not collide with the other UAVs. If any of the above three constraints is violated, the episode will be terminated.

It is assumed that each GN has the same amount of data and UAV will connect to one of the GNs at a time until all the data of the connected GN is collected. Before the flight or after collecting all the data from a GN, the UAV will connect to the next closest GN. Note that finding the next closest GN can be realized by measuring the signal strength of GN transmissions/reference signals.

B. State

In this work, we set the state as $s_t = (\frac{x_t}{L}, \frac{y_t}{L}, \frac{t}{T}, \frac{d_U}{step \times T}, \Phi, \text{data}_g, \text{ID}_g, \text{obs})$, where (x_t, y_t) is the current location of the data-collecting UAV, and t is the current time slot. In the fourth term $\frac{d_U}{step}$, d_U indicates the horizontal distance between the current UAV location and the center of the final area. d_U is then divided by the UAV movement step size (i.e., distance traveled within a unit time interval) to indicate the estimated arriving time to the destination. As seen in s_t, we normalize (x_t, y_t) to the side length L of the serving area, and normalize t and $\frac{d_U}{step}$ with the time constraint T. Φ denotes the facing direction of the UAV.

The sixth term data_g in the state denotes the remaining data of the connected GN. ID_g is an M-bit digital number indicating the ID of the connected GN, which allows the proposed framework to support up to 2^M GNs.

In the state, the last term $\text{obs} = \{x_{u1}/L, y_{u1}/L, v_{u1}, x_{u2}/L, y_{u2}/L, v_{u2}\}$ contains the location and velocity information of the other UAVs that are within the sensing range r_A of the UAV. Without loss of generality, we assume that the other UAVs have trajectories that follow random walk at the same speed. Note that this is an assumption that leads to a challenging scenario since it is difficult to predict the movements of other UAVs. We assume the UAV can sense up to two other UAVs ($U1$ and $U2$) within the sensing area. The UAV will collide with the other UAVs if their horizontal distance is within $2r_u$, where the r_u denotes the size of the UAVs.

C. Action

In each time slot t, UAV can select one of the five turning angles (uniformly quantized within $[-\pi/3, \pi/3]$). After taking one of the five possible actions, the location of the UAV will change to (x_{t+1}, y_{t+1}). If the next location (x_{t+1}, y_{t+1}) falls outside the serving area or leads to a collision, this episode will be terminated.

D. Reward

Reward function r_t is designed to encourage the UAV to reach the final area while maximizing the total sum-rate and avoiding any constraint violations (e.g., deadline violations, collisions, and flying outside the serving area) during the episode. We define the reward r_t of taking the action a_t in time slot t as

$$r_t = \epsilon \text{Rate}_t + P_{col} + P_{lost} + P_{time} + R_{goal} - 1 \tag{5}$$

where we set the coefficient $\epsilon = 50$ and the other terms are defined as follows:

1) P_{col}: If the next location of the UAV leads to a collision with another UAV, we set the collision penalty as $P_{col} = -50$, otherwise $P_{col} = 0$.
2) P_{lost}: If the next location is outside the serving area, the penalty is $P_{lost} = -50$, otherwise $P_{lost} = 0$.
3) P_{time}: If $t > T$ and the next location is still outside the final area, the deadline violation penalty is $P_{time} = -50$, otherwise $P_{time} = 0$.

If the UAV has arrived into the final area, there will be an extra reward $R_{goal} = \epsilon \sum_{t=0}^{T_f} \text{Rate}_t$ for encouraging the UAV to collect as much data as possible during the episode. Above, T_f is the time duration of the episode.

Finally, we give a small penalty to each step during the flight to encourage the UAV to reach the destination quickly after collecting all the data.

E. Policy Gradient

Experiments in the policy gradient algorithm are carried out in episodes, each of which contains H time slots. In time slot t, action a_t will be performed based on the current state s_t and then next state s_{t+1} and reward r will be obtained. Each episode can be seen as a Markov decision process (MDP) with the tuple (s_1, a_1, r_1, ...s_H, a_H, r_H). DNN parameters θ are then updated by minimizing the loss function

$$L(\theta) = -\frac{1}{H}\sum_{t=1}^{H}\left[\left(\sum_{i=t}^{H}\gamma^i r_i\right)\log(\pi_\theta(a_t|s_t))\right] \tag{6}$$

where γ denotes the discount factor in the policy gradient algorithm. $\pi_\theta(s_t)$ denotes the DRL policy with parameters θ, mapping the current state s_t to the probabilities of actions. In time slot t, agent will take the action with the highest probability for maximizing the total discounted reward of an episode.

IV. Meta-Reinforcement Learning

In this work, we propose a framework called meta-DRL that combines the techniques of meta-learning with deep reinforcement learning. Conventional (vanilla) DRL aims to train a task-specific model θ_i which can demonstrate promising performance in the corresponding task i. However, training such a DRL model can be expensive since it requires a large amount of data collected from the environment, which may not be easily accessible. Inspired by this, MAML seeks to acquire an initialization ϕ, from which only a single (or a few) gradient descent g_1^i is required for reaching the task-specific model θ_i, as shown in Fig. 1. i.e.

$$\theta_i = \phi - \alpha g_1^i = \phi - \alpha \nabla_\phi L_{T_i}(\phi) \tag{7}$$

where α denotes the adaptation learning rate and $\nabla_\phi L_{T_i}(\phi)$ denotes the gradient of loss over ϕ with the MDP trajectories sampled from task i with ϕ. In the remainder of this section, we assume that only one gradient descent is needed.

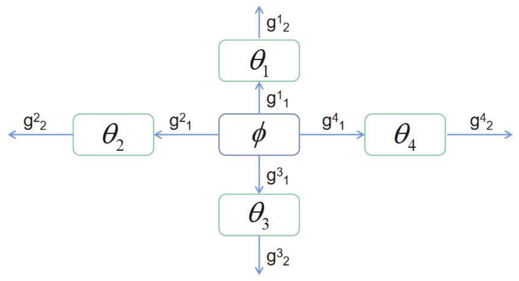


Fig. 1. Diagram of MAML.

In MAML, the loss function can be defined as

$$L(\phi) = \sum_{i=1}^{K} L_{T_i}(\theta_i) \tag{8}$$

where K is the number of tasks sampled from the same distribution $p(T)$ in every meta-DRL iteration. K is also referred to as meta-batch size.

According to the loss function in (8), MAML aims to minimize the total loss of the evolved parameters θ_i on task T_i. In other words, MAML looks for an initialization ϕ such that, after a single gradient descent, the T_i-specific models θ_i are obtained and each θ_i can demonstrate promising performance on the corresponding task T_i. With the goal to minimize $L(\phi)$, the update of ϕ can be expressed as

$$\phi = \phi - \beta \nabla_\phi L(\phi) = \phi - \beta \nabla_\phi \sum_{i=1}^{K} L_{T_i}(\theta_i) \tag{9}$$

where β denotes the meta learning rate.

Finding the gradient in (9) is not trivial. By applying the chain rule, we can rewrite the gradient as

$$\begin{aligned}\nabla_\phi \sum_{i=1}^{K} L_{T_i}(\theta_i) &= \sum_{i=1}^{K} \nabla_\phi L_{T_i}(\theta_i) \\ &= \sum_{i=1}^{K} \frac{\partial L_{T_i}(\theta_i)}{\partial \phi} = \sum_{i=1}^{K}\sum_{j} \frac{\partial L_{T_i}(\theta_i)}{\partial \theta_i^j}\frac{\partial \theta_i^j}{\partial \phi}\end{aligned} \tag{10}$$

where θ_i^j denotes the j^{th} parameter in θ_i. Now, (7) can be rewritten as

$$\theta_i^j = \phi^j - \alpha \frac{\partial L_{T_i}(\phi)}{\partial \phi^j}. \tag{11}$$

Therefore, each element of the last term $\frac{\partial \theta_i^j}{\partial \phi}$ in (10) can be computed as

$$\frac{\partial \theta_i^j}{\partial \phi^m} = \begin{cases} -\alpha \frac{\partial L_{T_i}(\phi)}{\partial \phi^j \partial \phi^m} & j \neq m \\ 1 - \alpha \frac{\partial^2 L_{T_i}(\phi)}{\partial {\phi^j}^2} & j = m. \end{cases} \tag{12}$$

The first-order MAML (FOMAML) was proposed by the authors in [5], in which the Hessian matrices in (12) are ignored, i.e, the value of (12) becomes 0 if $j \neq m$ and 1 if $j = m$. The work in [7] provides a proof and numerical results showing that FOMAML can achieve similar performance levels as MAML while significantly reducing the computational cost. In our work, we employ FOMAML in the proposed meta-DRL framework for UAV trajectory design.

With the approximation in FOMAML, gradient in (10) is simplified as

$$\sum_{i=1}^{K} \nabla_{\phi} L_{T_i}(\theta_i) = \sum_{i=1}^{K} \frac{\partial L_{T_i}(\theta_i)}{\partial \theta_i} = \sum_{i=1}^{K} \nabla_{\theta_i} L_{T_i}(\theta_i). \quad (13)$$

After applying the approximation in FOMAML, the gradient in (13) is simply the sum or average of the second-update gradients in each task T_i, i.e. g_2^i as shown in Fig. 1. Algorithm 1 describes the implementation of Meta-DRL with the method of FOMAML. Although we present the derivation with only two inner updates, it can be generalized to more inner updates according to [5]. For example, if I inner updates are performed, the first $I-1$ updates aim to obtain the task-specific parameter θ_i and the last update is the evaluation of θ_i in task T_i. FOMAML will collect the last gradients from all $\{T_i\}$ for updating the initialization ϕ.

Algorithm 1 Meta-DRL

```
1:  Initialize $\phi = \phi_0$
2:  while not done do
3:      Sample a batch of tasks $T_i$ with different environment.
4:      for all $T_i$ do
5:          Sample an episode of experiences $D = (s_1, a_1, r_1, ..., s_H, a_H, r_H)$ in task $T_i$ using DNN parameters $\phi$.
6:          Perform gradient descent using experience $D$ and compute the $T_i$-specific parameters $\theta_i$, as shown in (7).
7:          Sample another episode of experiences $D' = (s_1, a_1, r_1, ..., s_H, a_H, r_H)$ in task $T_i$ using DNN parameters $\theta_i$.
8:          Perform gradient descent using experience $D'$ and obtain $\theta_i^{'}$.
9:      end for
10:     Update $\phi \leftarrow \phi - \frac{\beta}{K}\sum_{i=1}^{K} \nabla_{\theta_i} L_{T_i}(\theta_i) = \phi - \frac{\beta}{K}\sum_{T_i}(\theta_i^{'} - \theta_i)$.
11: end while
```

Algorithm 2 Joint-Learning

```
1:  Initialize $\phi = \phi_0$
2:  while not done do
3:      Sample a batch of tasks $T_i$ with different environment.
4:      for all $T_i$ do
5:          Sample an episode of experiences $D = (s_1, a_1, r_1, ..., s_H, a_H, r_H)$ in task $T_i$ using DNN parameters $\phi$.
6:          Perform gradient descent using experience $D$ and update $\phi \leftarrow \phi - \alpha_2 \nabla_{\phi} L_{T_i}(\phi)$.
7:      end for
8:  end while
```

Benchmark Algorithms: We consider vanilla (conventional) DRL and joint learning as the benchmark algorithms in this work. The goal of joint learning is to find a global model that performs well across all the tasks. Algorithm 2 describes the implementation of joint learning. Joint-learning and meta-DRL share the same algorithmic framework for fair comparison. The only distinction is that, in joint-learning, only one update is performed in each sampled task T_i and this gradient will be used directly to update the neural network ϕ.

For vanilla DRL, we apply the method of conventional policy gradient to train a task-specific DNN in a selected task until convergence and evaluate its performance in a different task for comparison. Algorithm 3 is the vanilla DRL algorithm, which is also used for sampling the tuple $(s_1, a_1, r_1, ..., s_H, a_H, r_H)$ with the current DNN parameters in Algorithms 1 and 2.

Algorithm 3 Sampling experiences in each episode

```
1:  At the beginning of the first time slot, initialize UAV location as $(x_1, y_1)$=$(0, 0)$. Initialize state as $s_1$.
2:  for time slot $t = 1, ..., T$ do
3:      UAV chooses an action $a_t$ based on $s_t$ with the current DNN parameters.
4:      After taken $a_t$, $r_t$ can be obtained as illustrated in Section III.D.
5:      if UAV has not arrived the destination and no constraint violation has occurred, then
6:          UAV moves to the next location and obtain $s_{t+1}$ according to Section III.B.
7:      else
8:          Break.
9:          Record $(s_t, a_t, r_t)$.
10:     end if
11: end for
```

V. Numerical Results and Analysis

A. Simulation Setup

TABLE I
Experimental Parameters

Noise Power (σ^2)	10^{-6}
Size of Serving Area ($L \times L$)	$80m \times 80m$
Path Loss Exponent (Γ)	4
Communication Power (P^g)	0.1W
Length of Data-collecting UAV Step	10m
Length of other UAV Step	5m
Size of UAVs (r_u)	0.5m
Data-collecting UAV Sensing Range (r_A)	10m
Length of ID (M)	3
Meta-batch Size (K)	20
Episode Length (T)	50
Number of Inner Updates	20
Adaptation (Inner) Learning Rate (α)	0.02
Meta (Outer) Learning Rate (β)	0.03
Joint-learning Learning Rate ($\alpha2$)	0.00001
Discount Factor (γ)	0.9

Hyperparameters of the proposed meta-DRL framework are listed in Table I. We first generate 100 different tasks for training MAML initialization. In each task, we first randomly select 2 to 6 GNs and randomly place their cluster center in the serving area. There also exist three other UAVs for testing the ability of collision avoidance of the proposed framework, each of them starting from a random location and performing random walk within the serving area. Once the environment is generated, it will remain unchanged in a task. With the help of the proposed MAML framework, the goal of the UAV is to find a trajectory from the origin to the area in the top-right corner, while maximizing the data collected during the flight and avoiding collisions.

We randomly select a meta-batch of 20 tasks from a total of 100 tasks for each meta-DRL iteration, and then perform FOMAML according to Algorithm 1. For each inner update and adaptation update, we average the gradients obtained from 20 sampled episodes. It is critical to evaluate each gradient with multiple episodes to ensure the accuracy of each update.

For meta-DRL and the two benchmark algorithms, we use DNN with the same structure for fair comparison. The

DNN has two hidden layers consisting of 50 and 20 neurons respectively. We use ReLU as the activation function and Adam optimizer as the optimizer during the inner updates.

B. Numerical Results

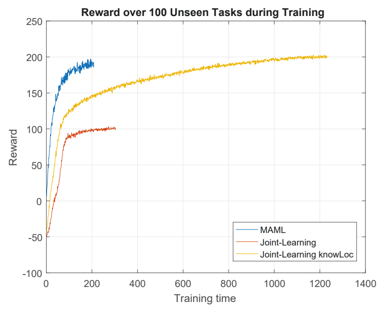


Fig. 2. Validation on 50 Unseen Tasks During Meta-DRL Training.

In order to track the performance of ϕ during the training of meta-DRL, we utilize ϕ in 100 unseen tasks with different GN positions and different trajectories of other UAVs (following a random walk) every 100 iterations. Utilizing ϕ in a task means that ϕ is used as the initialization of DNN and 20 gradient descents are performed in each task for adaptation. We depict the total reward of an episode after 20 gradient descents (averaged over 50 unseen tasks) in Fig. 2, which shows the efficiency of the initialization ϕ. It can be seen that at the beginning of the training, ϕ is inefficient and 20 updates can only lead to a negative reward. As the training proceeds, the proposed framework can learn an efficient ϕ that shows strong performance after adaptation in each task.

Meta-DRL and joint learning are compared in Fig. 2. The goal of joint-learning is to find a global model $\phi^{'}$ that is capable of handling all the tasks. We directly apply $\phi^{'}$ in the same 100 unseen tasks and depict its performance in Fig. 2. As shown in the figure, the DNN learned by joint-learning cannot perform as well as meta-DRL. This observation is expected since there does not exist a global model that can handle different environments with limited information. We then include the location information of the cluster center into the state, and joint learning now achieves comparable performance to the proposed MAML algorithm *but only with this additional information*. This observation does not contradict with our expectation. There can exist a global model when the available information increases and all the dynamics of the environment are known to the learning algorithm. We note that the proposed meta-learning algorithm does not require the knowledge of the cluster center location. Hence, in scenarios in which the location information of GNs is either required to be kept confidential or difficult to obtain, MAML algorithm can operate effectively and adapt to different tasks with limited training data.

Next, we compare the performances in two specific tasks with different numbers and distributions of the GNs. The trajectories of the other UAVs (with which collision should be avoided) are also different between the tasks. After the convergence of the three algorithms, we have a MAML initialization ϕ in the meta-DRL framework, a global model $\phi^{'}$ in joint-learning, and a task-specific model θ in vanilla DRL. Vanilla DRL is trained under Task I until convergence. Then, we assign ϕ, $\phi^{'}$ and θ to the DNN and let the DNN adapt in the two tasks.

Fig. 3(a) plots the total reward of each episode during 20 updates in Task I. It can be seen that vanilla DRL achieves the best performance in Task I since it is trained under this task and θ is already converged before the adaptation. Meta-DRL initialization ϕ does not perform well before the adaptation but it quickly increases to a reward level that is comparable to vanilla DRL. $\phi^{'}$ obtained from joint-learning has poor performance without adaptation (learning rate lr=0), indicating that there does not exist a global model that can handle all the tasks. Finally, we let $\phi^{'}$ adapt with the same learning rate (0.02) as meta-DRL, its performance starts increasing but its learning is much slower compared to meta-DRL, and its reward remains low.

The same experiment is performed in Task II and the reward during adaptation can be seen in Fig. 4(a). In this task, meta-DRL again starts from a low performance level but quickly adapts to the environment and achieves the highest performance. On the other hand, both joint-learning and vanilla DRL fail in Task II. It can be concluded that vanilla DRL cannot handle an unseen task as it is specifically trained for Task 1. On the other hand, meta-DRL is trained for obtaining an initialization, from which the performance can dramatically improve after a few updates.

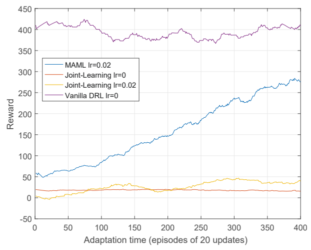
In Figs. 3(b), 3(c), 3(d) and 4(b), 4(c), 4(d), we plot the trajectories of the UAV after 20 gradient descents (400 episodes) from the corresponding initialization. It can be seen that meta-DRL can learn different trajectories after adaptation in different tasks. On the other hand, joint-learning and vanilla DRL fail to adapt to a different environment. Specifically, they fail to capture the distribution of the GNs, resulting in similar trajectories as those in Task I.

Another observation is that all three algorithms can avoid collision during the flights in different tasks. In this work, we include the information of the surrounding environment (obs in s_t), which is already sufficient for avoiding collisions. We can conclude that the initialization ϕ obtained from the proposed meta-DRL framework is not only an efficient initialization for maximizing the collected data but also a global model for avoiding obstacles. Therefore, if the available information for solving the problem is sufficient, meta-DRL may result in a global model that can handle different environments. If the information is insufficient and further trial and error are required in different environments, meta-DRL will result in a good initialization.

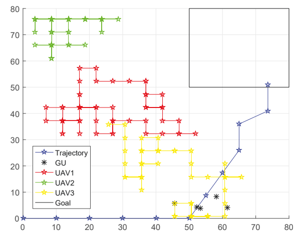
After the algorithms converge, we utilize different initializations in 200 newly generated tasks and compare their average performances on data collection in Table II, in terms of success rate, and reward. A trajectory is a success if the UAV reaches the final region within the time limit and without either collision or flying out of the map. And reward provides a measurement for the overall performance of the UAV.

From Table II, we have the following observations:

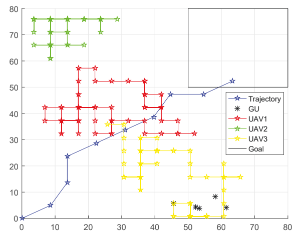
1) Comparing MAML with joint learning (no update), we can see that joint learning tries to obtain a global model which can handle different distributions of GNs, but it leads to a "diagonal-like" trajectory as shown in Fig. 3(c) and Fig. 4(c). Over the 200 unseen tasks, it collects less data compared to meta-DRL, but achieves a much higher success rate. Intuitively, joint-learning focuses on reaching the goal rather than collecting data, which results in a safer policy which is away from the edges of the serving area.
2) Adaptation from the global model leads to a better overall performance compared to directly applying the global mode to the tasks. But the performance is not comparable to meta-DRL. We can conclude that meta-DRL learns a more efficient initialization.
3) If the joint-learning algorithm takes into account enough information (i.e, knows the locations of the GNs), it



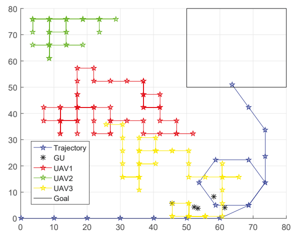
(a) Total Reward of Each Episode



(b) Meta-DRL Trajectory

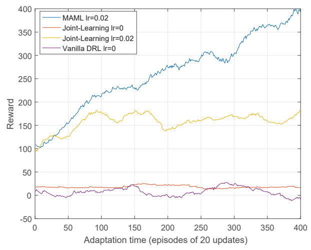


(c) Joint Learning Trajectory

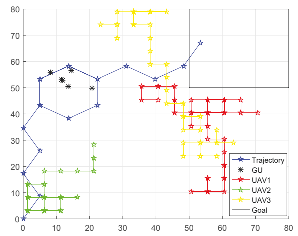


(d) Vanilla DRL Trajectory

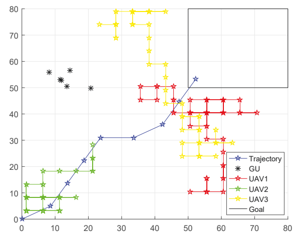
Fig. 3. UAV Trajectory with different algorithms in Task I



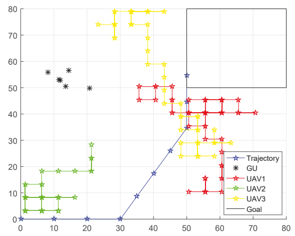
(a) Total Reward of Each Episode



(b) Meta-DRL Trajectory



(c) Joint Learning Trajectory



(d) Vanilla DRL Trajectory

Fig. 4. UAV Trajectory with different algorithms in Task II

TABLE II
AVERAGE PERFORMANCE OF DIFFERENT ALGORITHMS

Objectives	Avg Data Collected	Avg Success Rate	Avg Reward
meta-DRL(adapt with lr=0.02)	2.14	83.60%	173.36
Joint Learning (no update)	1.11	96.57%	93.42
Joint Learning(adapt with lr=0.02)	1.81	80.13%	139.50
Joint Learning knows cluster center (no update)	2.13	89.55%	178.69
Vanilla DRL	0.78	18.75%	-0.54

can learn a global model that can handle different GN distributions and achieve the maximum reward among all the algorithms.

4) Finally, the model obtained from vanilla DRL in a specific task cannot handle other unseen tasks, leading to the worst performance among all the algorithms.

VI. CONCLUSIONS

In this work, a meta-DRL framework is proposed for UAV trajectory design and we have compared its performance with joint-learning and vanilla DRL. Simulation results demonstrate that the proposed framework can learn an efficient initialization, from which only a few gradient descents and limited data is required to adapt to an unseen task. On the other hand, joint-learning cannot learn a global model that can handle different environments and vanilla DRL can only perform well in the environment for which it is trained.

It is worth noting that, if there is enough information about the environment, there may exist a global model that can handle different environments, and hence either joint-learning or vanilla DRL may work in different environments. However, the cost of the information can dramatically increase when the environment is complex or large-scale. When available information is limited, meta-DRL can outperform these benchmarks as shown in this work.

REFERENCES

[1] 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.

[2] Y. Tang, M. H. Cheung, and T.-M. Lok, "Trajectory design for multiple-UAV assisted wireless networks," *arXiv preprint arXiv:1912.12802*, 2019.

[3] J. Qiu, J. Lyu, and L. Fu, "Placement optimization of aerial base stations with deep reinforcement learning," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[4] H. Bayerlein, P. De Kerret, and D. Gesbert, "Trajectory optimization for autonomous flying base station via reinforcement learning," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[5] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[6] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," *arXiv preprint arXiv:1810.09502*, 2018.

[7] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," *arXiv preprint arXiv:1803.02999*, vol. 2, no. 3, p. 4, 2018.

[8] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1082–1092.

[9] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, and Y. Tang, "ES-MAML: Simple hessian-free meta learning," in *International Conference on Learning Representations*, 2019.

[10] X. Wang, M. C. Gursoy, T. Erpek, and Y. E. Sagduyu, "Learning-based UAV path planning for data collection with integrated collision avoidance," *IEEE Internet of Things Journal*, pp. 1–1, 2022.