Learning-Based UAV Trajectory Optimization with Collision Avoidance and Connectivity Constraints

Xueyuan Wang and M. Cenk Gursoy

***Abstract*—Unmanned aerial vehicles (UAVs) are expected to be an integral part of wireless networks, and determining collision-free trajectories for multiple UAVs while satisfying requirements of connectivity with ground base stations (GBSs) is a challenging task. In this paper, we first reformulate the multi-UAV trajectory optimization problem with collision avoidance and wireless connectivity constraints as a sequential decision making problem in the discrete time domain. We, then, propose a decentralized deep reinforcement learning approach to solve the problem. More specifically, a value network is developed to encode the expected time to destination given the agent's joint state (including the agent's information, the nearby agents' observable information, and the locations of the nearby GBSs). A signal-to-interference-plus-noise ratio (SINR)-prediction neural network is also designed, using accumulated SINR measurements obtained when interacting with the cellular network, to map the GBSs' locations into the SINR levels in order to predict the UAV's SINR. Numerical results show that with the value network and SINR-prediction network, real-time navigation for multi-UAVs can be efficiently performed in various environments with high success rate.**

***Index Terms*—Collision avoidance, decentralized algorithms, deep reinforcement learning, multi-UAV trajectory design, wireless connectivity.**

I. Introduction

Unmanned aerial vehicles (UAVs), also commonly known as drones, are aircrafts piloted by remote control or embedded computer programs without human onboard [1]. Recently, UAVs have found numerous applications, such as aerial inspection, photography, precision agriculture, traffic control, search and rescue, package delivery, and telecommunications. Based on the roles of UAVs, the following two scenarios are considered to integrate the UAVs into cellular networks: 1) UAV-assisted cellular networks, in which UAVs can be deployed as aerial base stations (BSs) to support wireless connectivity and improve the performance of cellular networks [2]; 2) cellular-connected UAV networks, in which the UAVs are regarded as aerial user equipments (UEs) that need to be supported by the ground communication infrastructure [3]. As aerial UEs, the UAVs need efficient trajectories and also should keep connected with ground base stations (GBSs) during their flights. Therefore, the trajectory of cellular-connected UAVs need to be carefully designed to meet their mission specifications, while at the same time ensuring that the communication requirements are satisfactorily met.

The authors are with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, 13244 (e-mail: xwang173@syr.edu, mcgursoy@syr.edu).

The material in this paper will be presented in part at the IEEE International Conference on Communications (ICC) 2021.

Trajectory optimization for cellular-connected UAVs has been investigated in the literature, in which a UAV has a mission of flying between a pair of given initial and final locations. The authors in [4]–[6] addressed the trajectory optimization problem with the goal to minimize the UAV's mission completion time. Particularly, the authors in [4] considered how to determine the optimal path for the UAV, subject to a quality of connectivity constraint in the GBS-to-UAV link specified by a minimum receive signal-to-noise ratio target. Techniques from graph theory and convex optimization were used to find the trajectory solution. In [5], the UAV is required to find a path during which it does not lose its cellular connection to one of the GBSs in the area by more than a given time period. Dynamic programming based approximate solution was proposed in this paper. The authors in [6] studied the trajectory optimization problem for a UAV with two different criteria for the connectivity constraint: 1) the maximum continuous time duration that the UAV is out of the coverage of the GBSs is limited to a given threshold; 2) the total time periods that the UAV is not covered by the GBSs is restricted. A double Q-learning method is proposed to solve the problem. Moreover, authors in [7] formulated a UAV trajectory optimization problem to minimize the weighted sum of its mission completion time and expected communication outage duration. A dueling double deep Q network with multi-step learning algorithm is proposed. A simultaneous navigation and radio mapping framework was also proposed to improve the performance. Additionally, an interference-aware path planning scheme for a network of cellular-connected UAVs was proposed in [8]. In particular, each UAV aims to achieve a tradeoff between maximizing energy efficiency and minimizing both wireless latency and the interference caused on the ground network along its path. A deep reinforcement learning algorithm, based on echo state network cells, was developed to solve the problem. In addition, trajectory design for cellular-connected UAVs has also been extensively investigated in [9]–[16]. However, none of the prior works considered multi-UAV networks, which is common in practice, along with collision avoidance constraints.

In scenarios involving multiple UAVs or more generally multiple autonomous systems, a fundamental challenge is to safely control the interactions with other dynamic agents in the environment. Specifically, it is important for the autonomous devices (e.g., robots and drones) to navigate in an environment with or without obstacles, and stay free of collisions with each other and the obstacles, based on local observations of the environment. Finding solutions to this problem is challenging, since one robot's action is based on others' motions (intents)

and policies which are in general unknown, and, furthermore, explicit communication of such hidden quantities is often impractical due to physical limitations. Earlier works have largely leveraged well-engineered interaction models to enhance the social awareness in robot navigation, e.g. [17] and [18], where the same policy is applied to all agents. The key challenge for these models is that they heavily rely on hand-crafted functions and cannot generalize well to various scenarios for crowd-like cooperation. As an alternative, reinforcement learning frameworks have been used to train computationally efficient policies that implicitly encode the interactions and cooperation among agents. Recent works, e.g., [19]–[22], have shown the power of deep reinforcement learning techniques to learn socially cooperative policies.

Different approaches for the collision avoidance of multiple UAVs have also been developed in the literature. For instance, a rolling horizon approach using dynamic programming was used to solve the problem in a multi-agent cooperative system in [23]. A neuro-dynamic programming algorithm is proposed in [24] for multi-UAV cooperative path planning. A mixed integer linear programming method is used in [25]. Partially observable Markov decision process based methods are applied in [26]–[28] for UAV collision avoidance. In addition, authors in [29] used reachable sets to represent the collection of possible trajectories of the obstacle aircraft. Once a collision is detected, a sampling-based method is used to generate a collision avoidance path for the UAV. The UAV in this paper was able to learn the position, velocity and receive other data from the obstacle aircraft. In [30], predictive state space was utilized to present the waypoints of the UAVs, with which initial collision-free trajectories are generated and then improved by a rolling optimization algorithm to minimize the trajectory length. However, considering collision avoidance in multi-UAV navigation with wireless communication requirements and addressing these challenges via deep reinforcement learning methods have not been adequately explored yet.

Motivated by these facts, we propose a decentralized deep reinforcement learning algorithm as a solution to the multi-UAV trajectory optimization problem with collision avoidance and wireless connectivity constraints. The contributions of the paper are listed as follows:

- We study multi-UAV trajectory optimization under realistic constraints, e.g., collision avoidance, wireless connectivity, and kinematic constraints, while also taking into account antenna patterns and interference levels. Since it is difficult to address this problem with standard optimization techniques (especially in a decentralized setting), we further reformulate it as a sequential decision making problem in the discrete time domain.
- We develop a decentralized deep reinforcement learning algorithm to learn the action policy for each UAV. More specifically, we optimize the value function of the Markov decision process (MDP) transformed from the formulated problem. Due to the high dimension and the continuity of the state space and action space, we design a value neural network to approximate the value function, and to encode the expected time to destination given the agent's joint state (including the agent's information, the nearby agents' observable information, and the locations of the nearby GBSs).
- Due to the fact that the UAVs do not communicate in the considered network, uncertainty exists in the UAVs' unobservable intents, which is critical for multi-UAV navigation problems. To address this uncertainty, we employ a velocity-filter approach to estimate the UAVs' intentions.
- We further design a signal-to-interference-plus-noise ratio (SINR)-prediction neural network to assist the value network to encode the interaction between the UAVs and the cellular network. Particularly, using accumulated SINR measurements obtained when interacting with the cellular network, the SINR-prediction network maps the nearby GBSs' locations into the SINR levels in order to predict the UAV's SINR.
- We delineate the initialization, refining, and training steps of the algorithm and describe the real-time navigation process. We extensively evaluate the proposed decentralized deep reinforcement learning algorithm. We demonstrate that with the introduction of the SINR-prediction network, testing environment is not restricted to be the same as the training environment. Furthermore, we show that real-time decentralized navigation of multiple UAVs can be efficiently performed with high success rate in various environments, e.g., environments with different antenna patterns, environments with obstacles or no-fly zones.

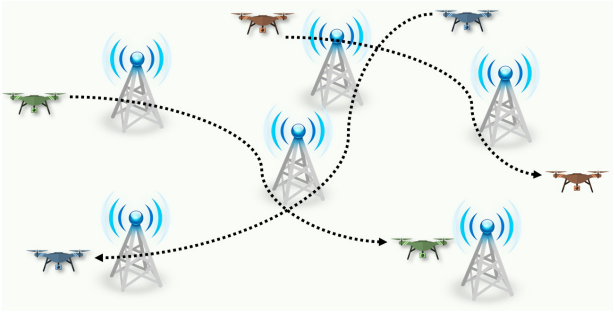


Fig. 1: An illustration of multi-UAV multi-GBS cellular networks.

The remainder of the paper is organized as follows: System model is introduced in Section II. Section III describes the multi-UAV trajectory optimization problem, including the considered constraints. Section IV focuses on the reinforcement learning framework for solving the proposed problem, and the approaches used to tackle the uncertainty in the environment. The decentralized deep reinforcement learning algorithm is presented in Section V in detail. In Section VI, numerical and simulation results are provided to evaluate the performance of the proposed algorithm. Finally, concluding remarks are given in Section VII.

II. System Model

In this section, we introduce the system model of the multi-UAV and multi-GBS cellular networks in detail. Note that in this section, unless specified otherwise, we remove the time

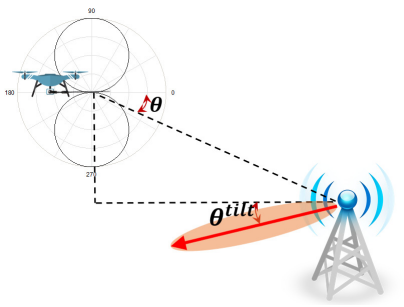


Fig. 2: Illustrations of the antenna patterns of the UAVs and the GBSs.

index e.g., in the position vector $\mathbf{p}(t) \rightarrow \mathbf{p}$, and the index for UAVs or GBSs, e.g., $\mathbf{p}_i \rightarrow \mathbf{p}$.

A. Deployment

We consider multi-UAV multi-GBS cellular networks as displayed in Fig. 1, in which J UAVs, with potentially different missions, need to fly from starting locations to destinations over an area containing K GBSs. Without loss of generality, we assume that the area of interest is a cubic volume, which can be specified by $C : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ and $\mathcal{X} \triangleq [x_{\min}, x_{\max}]$, $\mathcal{Y} \triangleq [y_{\min}, y_{\max}]$, and $\mathcal{Z} \triangleq [z_{\min}, z_{\max}]$. Let $\mathbf{p} = [p_x, p_y, H_V]$ denote the 3D position of the UAV, where H_V is the altitude of the UAVs which is assumed to be fixed for all UAVs. $\mathbf{p}^S = [p_{sx}, p_{sy}, H_V] \in \mathbb{R}^3$ and $\mathbf{p}^D = [p_{gx}, p_{gy}, H_V] \in \mathbb{R}^3$ are used to denote the coordinates of the starting points and destinations.

Each UAV's state is composed of an observable information vector and an unobservable (hidden) information vector, $\mathbf{s} = [\mathbf{s}^o, \mathbf{s}^h]$, where the observable state can be observed by other UAVs, while the unobservable state can not. In the global frame, observable state includes the UAV's position, velocity $\mathbf{v} = [v_x, v_y]$, and radius r, i.e., $\mathbf{s}^o = [\mathbf{p}, \mathbf{v}, r] \in \mathbb{R}^6$. The unobservable state consists of the destination $\mathbf{p}^D$, maximum speed $v_{\max}$, and orientation ϕ, i.e., $\mathbf{s}^h = [\mathbf{p}^D, v_{\max}, \phi] \in \mathbb{R}^5$. It is worth noting that the UAVs do not communicate with other UAVs. Hence, we address a more challenging non-communicating scenario.

In this cellular network, there are K GBSs providing wireless coverage simultaneously. The k^{th} GBS has transmit power P_{B_k}, and it is located at position $\mathbf{p}_{B_k} = [p_{x_{B_k}}, p_{y_{B_k}}, H_B]$, where H_B is the height of the GBS and is assumed to be the same for all GBSs.

B. Antenna Configuration

The GBSs and the UAVs are equipped with directional antennas with fixed radiation patterns, which are shown in Fig. 2.

1) GBS: We assume that the antenna elements of the GBSs are only directional along the vertical dimension but omni-directional horizontally [1]. Along the vertical dimension, the signal is usually downtilted toward the ground to cover the ground users and suppress the intercell interference. Therefore, the antenna gain can be expressed as [31]

$$G_B(d) = G_h + G_v \text{ (dB)} = 10^{-\min\left(-1.2\left(\frac{\arctan(\frac{H_B - H_V}{d}) - \theta^{tilt}}{\theta^{3dB}}\right)^2, \frac{G_m}{10}\right)} \quad (1)$$

where

$$G_h = 0 \text{ (dB)} \quad (2)$$

$$G_v(d) = -\min\left(12\left(\frac{\arctan(\frac{H_B - H_V}{d}) - \theta^{tilt}}{\theta^{3dB}}\right)^2, G_m\right) \text{ (dB)}. \quad (3)$$

Above, G_m is the maximum attenuation of the antennas, d is the horizontal distance between the UAV and the GBS, θ^{tilt} and θ^{3dB} represent electrical antenna downtilting angle and the vertical 3dB beamwidth of the antennas at the GBSs.

2) UAV: The UAVs are assumed to be equipped with a receiver with a horizontally oriented antenna, and a simple analytical approximation for antenna gain provided by UAVs can be expressed as [32]

$$G_V(d) = \sin(\theta) = \frac{H_V - H_B}{\sqrt{d^2 + (H_V - H_B)^2}} \quad (4)$$

where θ is elevation angle between the UAV and GBS, H_V is the UAV altitude, and H_B is the height of the GBS.

C. Path Loss

We assume that the path loss can be expressed as

$$L(d) = \left(d^2 + (H_B - H_V)^2\right)^{\alpha/2} \quad (5)$$

where α is the path loss exponent.

D. SINR and Connectivity

The UAVs receive signals from all GBSs, among one of which is the serving BS, and others contribute to the interference. The received signal from the k^{th} GBS to the i^{th} UAV can be expressed as $P_k G_{B_k}(d_{ik}) G_{V_i}(d_{ik}) L^{-1}(d_{ik})$. The experienced SINR at the i^{th} UAV if it is associated with the k^{th} GBS an be expressed as

$$\mathcal{S}_{r_{i,k}} \triangleq \frac{P_k G_{B_k}(d_{ik}) G_{V_i}(d_{ik}) L^{-1}(d_{ik})}{\mathcal{N}_s + \sum_{k' \neq k} P_{k'} G_{B_{k'}}(d_{ik'}) G_{V_i}(d_{ik'}) L^{-1}(d_{ik'})} \quad (6)$$

where $\mathcal{N}_s$ is the noise power. If the experienced SINR at a UAV is larger than a threshold $\mathcal{T}_s$, then the UAV is regarded as connected with the cellular network, and disconnected otherwise.

E. SINR Measurement

Along the path to destination, UAVs interact with the cellular network, measure the raw signal from GBSs, and obtain the instantaneous SINR $\mathcal{S}'_r([\mathbf{p}, \mathbf{s}_B]; h)$, where h includes the random small-scale fading coefficients with all GBSs, and $\mathbf{p}$ and $\mathbf{s}_B = [\mathbf{p}_{B_k}, \forall k]$ are the position of the UAV and positions of all GBSs, respectively. These measurements can be

obtained by leveraging the existing soft handover mechanisms with continuous reference signal received power (RSRP) and reference signal received quality (RSRQ) [7]. At each time t, over a very short time interval, during which the agents' locations can be approximately considered to be unchanged, it is assumed that the UAV performs N_m SINR measurements (each of which takes milliseconds). Then the empirical SINR can be obtained as

$$\widehat{\mathcal{S}}_{r_{(t)}} = \frac{1}{N_m} \sum_{n=1}^{N_m} \mathcal{S}'_{r_{(t)}}([\mathbf{p}(t), \mathbf{s}_B]; h_{(t),n}). \tag{7}$$

To average over the randomness arising from small-scale fading, we can consider large N_m and have $\lim_{N_m \to \infty} \widehat{\mathcal{S}}_{r_{(t)}} = \mathcal{S}_{r_{(t)}}$ by applying the law of large numbers. Therefore, as long as the UAV performs signal measurements sufficiently frequently so that $N_m \gg 1$, $\mathcal{S}_{r_{(t)}}$ can be evaluated by its empirical value $\widehat{\mathcal{S}}_{r_{(t)}}$.

III. Multi-UAV Trajectory Optimization

In this section, we first introduce the constraints and then formulate the multi-UAV trajectory optimization problem.

A. Constraints

1) Collision Avoidance: Collision avoidance is central to many autonomous systems. During flight, the UAVs should not collide with others, which means that the distance between two agents should be larger than their radius all the time, i.e.,

$$||\mathbf{p}_i(t) - \mathbf{p}_j(t)||_2 > r_i + r_j \quad \forall j \neq i, \forall t \tag{8}$$

where $\mathbf{p}_i(t)$ is the location of the i^{th} UAV at time t, and r_i is its radius. Note that this radius can also include a buffer zone in which no other UAV should be present.

2) Wireless Connectivity Constraint: To support the command and control and also data flows, UAVs have to maintain a reliable communication link to the GBSs. To achieve this goal, we consider the connectivity constraint for the UAVs, i.e., the maximum continuous time duration that the UAV is disconnected should not be longer than $\mathcal{T}_t$ time units. The maximum continuous disconnected time duration can be mathematically expressed as

$$T_O^{\max} = \max_{t \in [0,T]} t - T_L(t) \tag{9}$$

where T is the total travel time, and $T_L(t)$ is the last time that the UAV is connected with the cellular network before time t, i.e.,

$$\begin{aligned} T_L(t) = \max \quad & \tau \\ \text{s.t.} \quad & \tau \in [0, t] \\ & \mathcal{S}_r(\tau) \geqslant \mathcal{T}_s. \end{aligned} \tag{10}$$

Therefore, the connectivity constraint can be written as

$$\left(\max_{t \in [0,T]} t - T_L(t) \right) \leqslant \mathcal{T}_t. \tag{11}$$

3) Initial and Final Locations: Each UAV starts its mission from a given initial location and completes its flight at a given destination, i.e., $\mathbf{p}(0) = \mathbf{p}^S$ and $\mathbf{p}(T) = \mathbf{p}^D$.

4) Kinematic Constraint: Kinematic constraints need to be considered for operating UAVs. We impose the speed and rotational constraints as follows:

$$\mathbf{v}(t) = [v_s(t), \phi(t)] \tag{12}$$

$$\text{Speed limit: } v_s(t) \leqslant v_{\max} \tag{13}$$

$$\text{Rotation limit: } |\phi_(t) - \phi(t - \Delta t)| \leqslant \Delta t \cdot \mathcal{T}_r \tag{14}$$

where $\mathbf{v}(t)$, $v_s(t)$ and $\phi(t)$ are the UAV's velocity, speed and orientation at time t. $v_{\max}$ is the maximum of speed the UAV, and $\mathcal{T}_r$ is the maximum angle that a UAV can rotate in unit time period. This constraint limits the direction that a UAV can travel at a given time.

5) Association Constraint: Each UAV is associated with one GBS at a time, and the associated GBS is denoted by $a(t) \in \{1, ..., K\}$.

B. Problem Formulation in Continuous Time Domain

The goal of this work is to find trajectories for all UAVs in the network such that the travel/flight time of each UAV between the initial and final locations is minimized, while the constraints are satisfied. In the considered decentralized setting, the trajectory optimization problem for the i^{th} UAV can be formulated as

$$\begin{aligned} (\text{P0}): \quad & \underset{\{\mathbf{p}_i(t), a_i(t), \forall t\}}{\operatorname{argmin}} \quad T_i \\ & s.t. \quad ||\mathbf{p}_i(t) - \mathbf{p}_j(t)||_2 > r_i + r_j, \forall j \neq i, \forall t && \text{(P0.a)} \\ & \max_{t \in [0,T]} t - T_L(t) \leqslant \mathcal{T}_t && \text{(P0.b)} \\ & \mathbf{p}_i(0) = \mathbf{p}_i^S, \mathbf{p}_i(T_i) = \mathbf{p}_i^D && \text{(P0.c)} \\ & v_{s_i}(t) \leqslant v_{\max_i}, \forall t && \text{(P0.d)} \\ & |\phi_i(t) - \phi_i(t - \Delta t)| \leqslant \Delta t \cdot \mathcal{T}_r, \forall t && \text{(P0.e)} \\ & a_i(t) \in \{1, ..., K\}, \forall t && \text{(P0.f)} \end{aligned}$$

C. Problem Formulation in Discrete Time Domain

Since the UAV is not permitted to be disconnected continuously for more than $\mathcal{T}_t$ time units, it is sufficient to consider $\Delta t = \mathcal{T}_t / n_t$ as one time step and address the problem every n_t time steps. If, at these specific time instances, the experienced SINRs at all UAVs are higher than $\mathcal{T}_s$, we can guarantee that the connectivity constraint is satisfied. Now, the optimization problem can be represented in discrete time domain as follows:

$$\begin{aligned} (\text{P1}): \quad & \underset{\{\mathbf{p}_{i,t}, a_{i,t}, \forall t\}}{\operatorname{argmin}} \quad T_i \\ & s.t. \quad ||\mathbf{p}_{i,t} - \mathbf{p}_{j,t}||_2 > r_i + r_j, \forall j \neq i, \forall t && \text{(P1.a)} \\ & \mathcal{S}_{r_{i,t}} \geqslant \mathcal{T}_s, \text{ if } t \mid n_t && \text{(P1.b)} \\ & \mathbf{p}_{i,0} = \mathbf{p}_i^S, \mathbf{p}_{i,T_i} = \mathbf{p}_i^D, \forall i && \text{(P1.c)} \\ & v_{s_{i,t}} \leqslant v_{\max_i}, \forall t && \text{(P1.d)} \\ & |\phi_{i,t} - \phi_{i,t-1}| \leqslant \Delta t \cdot \mathcal{T}_r, \forall t && \text{(P1.e)} \\ & a_{i,t} \in \{1, ..., K\}, \forall t && \text{(P1.f)} \end{aligned}$$

where the integer-valued discrete time index t indicates time increments by Δt, and $t \mid n_t$ signifies that t is divisible by n_t.

It is obvious that the optimal cell association policy can be obtained as $a^*_{i,t} = \text{argmax}_{k\in\{1,\dots,K\}} S_{r_{i,t,k}}$. Then (P1) reduces to

$$\begin{aligned}(\text{P2}): \underset{\{\mathbf{p}_{i,t},\forall t\}}{\text{argmin}} \quad & T_i \\ s.t. \quad & (\text{P1}.a) - (\text{P1}.e).\end{aligned} \tag{15}$$

The non-communicating multi-agent navigation task can be formulated as a sequential decision making problem in a reinforcement learning framework [19]. The objective then is to develop policies, $\{\pi_i : \mathbf{s}^{jn}_{i,t} \mapsto \mathbf{v}_{i,t}, \forall i\}$ that select actions to minimize the expected time to destination while satisfying all the constraints, where $\mathbf{s}^{jn}_{i,t}$ and $\mathbf{v}_{i,t}$ are the joint state and the action of the agent, respectively. Now, the optimization problem can be reformulated as

$$\begin{aligned}(\text{P3}): \underset{\pi_i}{\text{argmin}} \quad & \mathbb{E}[T_i|\mathbf{s}^{jn}_i, \pi_j, \forall j \neq i] \\ s.t. \quad & (\text{P1}.a) - (\text{P1}.c) \\ & \mathbf{p}_{i,t} = \mathbf{p}_{i,t-1} + \Delta t \cdot \pi_i(\mathbf{s}^{jn}_{i,t-1}), \forall t \quad (\text{P3.d})\end{aligned}$$

where the expectation in the objective function in (P3) is with respect to other agents' unobservable states and policies, and (P3.d) is the agent's kinematics, which satisfy the kinematic constraints in (P1.d) and (P1.e). Further, we use the common assumption that each agent would follow the same policy [18] [19] [33], i.e., $\pi = \pi_i$.

IV. Reinforcement Learning Based Approach

In this section, we first introduce reinforcement learning (RL) formulation for the multi-UAV navigation problem. Then, we present the approaches used to tackle the uncertainty in the UAVs' unobservable intents, and the interaction between the UAVs and the cellular network.

A. Reinforcement Learning

RL is a class of machine learning methods for solving sequential decision making problems with unknown state-transition dynamics [19] [22]. Typically, a sequential decision making problem can be formulated as an MDP, which is described by the tuple $\langle S, A, P, R, \gamma\rangle$, where S is the state space, A is action space, P is the state-transition model, R is the reward function, and γ is a discount factor.

Since the action space in this work is continuous and the set of permissible velocity vectors depends on the agent's state, we choose to optimize the value function $V_\pi(\mathbf{s}^{jn})$ as in [19], instead of optimizing the commonly used action-value function $Q(\mathbf{s}^{jn}, \mathbf{v})$ (where $\mathbf{v}$ denotes the action). The state value function of an MDP is the expected return starting from time t following policy π, i.e.,

$$V_\pi(\mathbf{s}^{jn}_t) = \sum_{t'=t}^{T} \gamma^{t'} R_{t'}(\mathbf{s}^{jn}_{t'}, \pi(\mathbf{s}^{jn}_{t'})). \tag{16}$$

The optimal policy is to maximize the expected return:

$$\begin{aligned}&\pi^*(\mathbf{s}^{jn}_t) \\ &= \underset{\mathbf{v}_t}{\text{argmax}}\, R(\mathbf{s}^{jn}_t, \mathbf{v}_t) + \gamma \int_{\mathbf{s}^{jn}_{t+1}} P(\mathbf{s}^{jn}_{t+1}|\mathbf{s}^{jn}_t, \mathbf{v}_t) V^*(\mathbf{s}^{jn}_{t+1}) \text{d}\mathbf{s}^{jn}_{t+1},\end{aligned} \tag{17}$$

where $V^*(\mathbf{s}^{jn}_t) = \sum_{t'=t}^{T} \gamma^{t'} R_{t'}(\mathbf{s}^{jn}_{t'}, \pi^*(\mathbf{s}^{jn}_{t'}))$ is the optimal value function, $R(\mathbf{s}^{jn}_t, \mathbf{v}_t)$ is the reward received at time t, $P(\mathbf{s}^{jn}_{t+1}|\mathbf{s}^{jn}_t, \mathbf{v}_t)$ is the transition probability from time t to time $t+1$.

B. Reinforcement Learning Formulation

To estimate the high-dimensional, continuous value function, it is common to approximate it with a deep neural network (DNN) parameterized by weights and biases, $\boldsymbol{\xi}$. For notational simplicity, we drop the DNN parameters from the value function notation, i.e., $\mathcal{V}(\mathbf{s}; \boldsymbol{\xi}) = \mathcal{V}(\mathbf{s})$. And $\mathbf{s}$ is the joint state of an agent which is also the input of the DNN, and $\mathcal{V}(\mathbf{s})$ is the output of the value network given $\mathbf{s}$.

By detailing each of these elements and relating to (P1.a)-(P1.c) and (P3.d), the following provides an RL formulation for the multi-UAV navigation problem. Each UAV is an independent agent, and in the discussions below, we use agent instead of UAV.

1) State Space: In multi-agent multi-GBS cellular networks, the agents are able to observe the following information from the environment: 1) its own information vector $\mathbf{s}_{i,t}$ (for the i^{th} agent at time step t); 2) the observable state of the nearest $J_n < J$ agents $\mathbf{s}^{jno}_{i,t} = [\mathbf{s}^o_{j,t} : j \in \{1, 2, ..., J_n\}]$; 3) the location information of the nearest $K_n \leqslant K$ GBSs, which is assumed to be observed by the agents, and is denoted by $\mathbf{s}^o_B = [\mathbf{p}_{B_k} : k \in \{1, ..., K_n\}]$. All the information observed by the agent constitutes its joint state $\mathbf{s}^{jn}_{i,t} = [\mathbf{s}_{i,t}, \mathbf{s}^{jno}_{i,t}, \mathbf{s}^o_B], \forall t$.

2) Action Space: The action space is a set of permissible velocity vectors. Ideally, the agent can travel in any direction at any time. However, in reality the kinematic constraints in (12)-(14) restrict the agent's movement and should be taken into account. Then, based on the agent's current speed, orientation $[v_{s,i,t}, \phi_{s,i,t}]$ and the kinematic constraints, permissible actions $\mathbf{v} = [v_s, \phi]$ are sampled to built the action space $A_{i,t}$.

3) Reward Function: Similar to the formulation of the reward function defined in [34], [19], and [22], we define a sparse reward function, which awards the agent for reaching its goal, and penalizes the agent for getting too close or colliding with other agents, and also penalizes for getting close to be disconnected or already being disconnected from the cellular network. The reward function consists of three parts: the reward, R_c, that encourages the fast arrival to the destination and penalizes close encounters with other agents; the reward, R_s, that encourages keeping connectivity with the cellular network, and a constant movement penalty, R_t, that encourages the agents to reduce their flight time. For instance, at time step t, the reward functions for the i^{th} agent can be

expressed as follows:

$$R_{c_{i,t}}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}_{i,t}) = \begin{cases} 2, & \text{if } \mathbf{p}_i = \mathbf{p}_i^D, \\ -(1-(d_{t_{\min}} - r_i - r_j)/0.2), & \\ & \text{if } r_i + r_j < d_{t_{\min}} \leqslant 0.2 + r_i + r_j, \\ -1, & \text{if } d_{t_{\min}} \leqslant r_i + r_j, \\ 0, & \text{otherwise}, \end{cases} \quad (18)$$

and

$$R_{s_{i,t}}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}_{i,t}) = \begin{cases} -0.5, & \text{if } t \mid n_t \text{ and } \mathcal{T}_s \leqslant \mathcal{S}_{r_{i,t+1}} < \mathcal{T}_s + 0.1 \\ -1, & \text{if } t \mid n_t \text{ and } \mathcal{S}_{r_{i,t+1}} < \mathcal{T}_s \\ 0, & \text{otherwise}, \end{cases} \quad (19)$$

where $d_{t_{\min}}$ is the minimum distance to other agents within the next time step duration. Therefore, the overall reward function can be expressed as the sum

$$R_{i,t}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}_{i,t}) = R_{c_{i,t}}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}_{i,t}) + R_{s_{i,t}}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}_{i,t}) + R_t. \quad (20)$$

C. Estimation of the Agents' Unobservable Intents

The probabilistic state transition model in (17) is determined by the agents' kinematics as defined in (P3.d), other agents' hidden states, and the other agents' choices of action. Since the other agents' hidden intents are unknown, the system's state transition model is unknown as well. In addition, it is difficult to evaluate the integral, because the other agents' next state has an unknown distribution (that depends on their unobservable intents). We approximate this integral by assuming that the other agent would be traveling at a filtered velocity for a short duration Δt, which is regarded as a one-step lookahead procedure [17] [18] [19] [35]. This propagation step amounts to predicting the other agent's motion with a simple linear model, i.e., $\hat{\mathbf{v}}_{j,t} = \text{filter}(\mathbf{v}_{j,0:t})$. For the i^{th} agent, other agents' filtered velocities are included in the vector $\hat{\mathbf{v}}_{i,t}^{jno} = [\hat{\mathbf{v}}_{j,t} : j \in \{1, 2, ..., J_n\}]$. Then, the estimated next state of the i^{th} agent will be

$$\hat{\mathbf{s}}_{i,t+1,\mathbf{v}}^{jn} = [f(\mathbf{s}_{i,t}, \Delta t, \mathbf{v}), f(\mathbf{s}_{i,t}^{jno}, \Delta t, \hat{\mathbf{v}}_{j,t}^{jno}), \mathbf{s}_B^o] \quad (21)$$

where $f(\cdot)$ is the kinematic model. Then the policy becomes

$$\pi^*(\mathbf{s}_{i,t}^{jn}) = \underset{\mathbf{v}}{\text{argmax}}\, R_{i,t}(\mathbf{s}_{i,t}^{jn}, \mathbf{v}) + \gamma V^*(\hat{\mathbf{s}}_{i,t+1,\mathbf{v}}^{jn}). \quad (22)$$

D. SINR Prediction

Model-free RL requires no prior knowledge about the environment. This usually leads to slow learning process and requires a large number of agent-environment interactions, which is typically costly or even risky to obtain [7]. In addition, using only a value network to encode the interactions among the agents and the interactions between the agent and the cellular network is not easy. Actually, each real experience obtained from the agent and cellular network interaction not only can be used to get reward and refine the value network, but also can be used for model learning in order to predict the agent's SINR experienced at certain positions. More specifically, when flying in the environment, agents interact with the cellular network and obtain the empirical SINR $\widehat{\mathcal{S}}_r$. Since there is no need to use the exact SINR for connectivity measurement, this work uses the quantized SINR level, $L_w(\widehat{\mathcal{S}}_r)$, to check the agent's connectivity. With a finite set of measurements $\{\langle \mathbf{s}_B^{jn}, L_w(\mathbf{s}_B^{jn})\rangle\}$, where $\mathbf{s}_B^{jn} = [\mathbf{p}, \mathbf{s}_B^o]$, a DNN can be trained to predict the SINR level $L_w(\mathbf{s}_B^{jn})$.

A fully connected DNN with parameters $\boldsymbol{\xi}_w$ can be used to predict the agent's SINR level, i.e., $\boldsymbol{\xi}_w$ is trained so that $L_w(\mathbf{s}_B^{jn}) \approx \mathcal{L}_w(\mathbf{s}_B^{jn}; \boldsymbol{\xi}_w)$. The data measurement $\langle \mathbf{s}_B^{jn}, L_w(\mathbf{s}_B^{jn})\rangle$ only arrives incrementally as the agent flies to new locations and can be saved in a database (e.g., replay memory), and a minibatch is sampled at random from the database to update the network parameter $\boldsymbol{\xi}_w$. Note that the prediction of SINR levels might be highly inaccurate initially, but can be continuously improved as more real experience is accumulated.

V. Decentralized Deep Reinforcement Learning Algorithm

In this section, we present the proposed decentralized deep reinforcement learning algorithm as a solution to multi-UAV navigation with collision avoidance and wireless connectivity constraints, including the SINR-prediction neural network. The proposed algorithm is presented in Algorithm 1, and is referred to as RLTCW-SP (RL for Trajectory optimization with Collision avoidance and Wireless connectivity constraint and with SINR Prediction).

A. Parametrization

Since the optimal policy should be invariant to any coordinate plane, we follow the agent-centric parameterization as in [34], [19] and [22], where the agent is located at the origin and the x-axis is pointing toward the agent's destination. The states of the i^{th} agent after transformation is

$$\widetilde{\mathbf{s}}_i = [d_{g_i}, v_{\max_i}, \tilde{v}_{x_i}, \tilde{v}_{y_i}, r_i, \tilde{\phi}_i] \quad (23)$$

$$\widetilde{\mathbf{s}}_i^{jno} = [[\tilde{p}_{x_j}, \tilde{p}_{y_j}, H_V, \tilde{v}_{x_j}, \tilde{v}_{y_j}, r_j, d_j] : j \in \{1, 2, ..., J_n\}] \quad (24)$$

where d_g is the agent's distance to the goal, d_j is the agent's distance to the j^{th} agent, and $\tilde{p}$ denotes p in the new coordinate.

In addition, SINR experienced at an agent depends on the distance and the relative angular direction from the agent to the GBSs, while it does not depend on the positions in global coordinates. To remove this redundant dependence, the location information vector of all GBSs can be parameterized as

$$\widetilde{\mathbf{p}}_{B_k} = [d_{B_k}, \phi_{B_k}, \theta_{B_k}] \quad (25)$$

$$\widetilde{\mathbf{s}}_{B_i} = [\widetilde{\mathbf{p}}_{B_k} : k \in \{1, ..., K_n\}] \quad (26)$$

where $d_{B_k} = ||\mathbf{p}_{B_k} - \mathbf{p}_i||$ is the distance from the agent to the k^{th} BS, ϕ_{B_k} and θ_{B_k} are the horizontal and vertical angles of the k^{th} BS with respect to the agent.

Algorithm 1: RLTCW-SP Algorithm

Input: State-value pairs D

1 Initialize state-value pairs D
2 Initialize location-SINR pairs D_w
3 Initialize value network $\boldsymbol{\xi}$ with D
4 Initialize SINR-prediction network $\boldsymbol{\xi}_w$
5 **for** *episode = 0: total episode* **do**
6 **for** *n random training cases* **do**
7 Initialize $\mathbf{s}_{i,0} \forall i$
8 **while** *not all reached destinations* **do**
9 **for** *each agent i* **do**
10 **if** *not reached destination* **then**
11 $\mathbf{s}_{i,t}^{jn} \leftarrow$ observeEnvironment()
12 $A_{i,t} \leftarrow$ sampleActionSpace()
13 $c \leftarrow$ randomSample(Uniform (0,1))
14 **if** $c \leqslant \epsilon$ **then**
15 $\mathbf{v}_{i,t} \leftarrow$ randomSample($A_{i,t}$)
16 **else**
17 $\hat{\mathbf{v}}_{i,t}^{jno} \leftarrow$ filter($\mathbf{v}_{0:t-1}^{jn}$)
18 $\hat{\mathbf{s}}_{i,t+1}^{jno} \leftarrow$ propagate($\mathbf{s}_{i,t}^{jno}, \hat{\mathbf{v}}_{i,t}^{jno}$)
19 **for** *every* $\boldsymbol{a}$ *in* $A_{i,t}$ **do**
20 $\hat{\mathbf{s}}_{i,t+1} \leftarrow$ propagate($\mathbf{s}_{i,t}, \mathbf{a}$)
21 $\hat{L}_{w_{i,t+1}} = \mathcal{L}_w(\hat{\mathbf{s}}_{B_{i,t+1}}^{jn})$
22 $R_{i,t} \leftarrow$ getReward($\hat{\mathbf{s}}_{i,t+1}^{jn}, \hat{L}_{w_{i,t+1}}$)
23 $V_p = R_{i,t} + \gamma \mathcal{V}(\hat{\mathbf{s}}_{i,t+1}^{jn})$
24 $\mathbf{v}_{i,t} \leftarrow \text{argmax}_{\mathbf{a} \in A_{i,t}} V_p$
25 $R_{i,t}, \mathbf{s}_{i,t+1}, \mathcal{S}_{r_{i,t+1}} \leftarrow$ executeAction($\mathbf{v}_{i,t}$)
26 **for** *each agent i* **do**
27 $V_{i,0:T_i} \leftarrow$ updateValue($\mathbf{s}_{i,0:T_i}^{jn}, R_{i,0:T_i}, \boldsymbol{\xi}$)
28 $L_{w_{i,0:T_i}} \leftarrow$ getSINRlevel($\mathcal{S}_{r_{i,0:T_i}}$)
29 Update state-value pairs D with $\langle \mathbf{s}_{i,0:T_i}^{jn}, V_{i,0:T_i} \rangle$
30 Update location-SINR pairs D_w with $\langle \mathbf{s}_{B_{i,0:T_i}}^{jn}, L_{w_{i,0:T_i}} \rangle$
31 Sample random minibatch from D, and update value network $\boldsymbol{\xi}$ by gradient descent.
32 Sample random minibatch from D_w, and update SINR-prediction network $\boldsymbol{\xi}_w$ by gradient descent.
33 **return** $\boldsymbol{\xi}, \boldsymbol{\xi}_w$

Therefore, the joint state of the i^{th} agent after transformation is

$$\widetilde{\mathbf{s}}_i^{jn} = [\widetilde{\mathbf{s}}_i, \widetilde{\mathbf{s}}_i^{jno}, \widetilde{\mathbf{s}}_{B_i}]. \tag{27}$$

And the input of the SINR-prediction network becomes

$$\widetilde{\mathbf{s}}_{B_i}^{jn} = [[d_{B_k}, \phi_{B_k}, \theta_{B_k}] : k \in \{1, ..., K_n\}]. \tag{28}$$

B. Initialization

The value network $\boldsymbol{\xi}$ can be first initialized with imitation learning using a set of experiences to accelerate the convergence. More specifically, in this work, we use optimal reciprocal collision avoidance (ORCA) [18] to generate a number of trajectories that contain a large set of state-value pairs $\{\langle \mathbf{s}^{jn}, V \rangle\}^{N_1}$, where $V = \gamma^{t_g}$ and t_g is the time to reach the destination. The experiences are saved in memory D (line 1 in Algorithm 1). Then, the value network is initialized by supervised training on D (line 3). The value network is trained by back-propagation to minimize a quadratic regression error

$$\boldsymbol{\xi} = \underset{\boldsymbol{\xi}'}{\text{argmin}} \sum_{k=1}^{N_1} \left(V_k - \mathcal{V}(\mathbf{s}_k^{jn}; \boldsymbol{\xi}') \right). \tag{29}$$

If a set of location-SINR experiences can be downloaded from the cloud, we can save the downloaded dataset in memory D_w (line 2), $\{\langle \mathbf{s}_B^{jn}, L_w \rangle\}^{N_2}$, where L_w is the scaled SINR level that the agent experienced. Then, the SINR-prediction network can be initialized with $\boldsymbol{\xi}_w = \text{argmin}_{\boldsymbol{\xi}'} \sum_{k=1}^{N_2} \left(L_{w_k} - \mathcal{L}(\mathbf{s}_B^{jn}; \boldsymbol{\xi}') \right)$, which is trained by back-propagation (line 4). If no dataset is available, D_w is initialized with an empty list, and the SINR-prediction network is initialized with random network parameters.

C. Refining Process

After initialization, a refining process is performed using RL. Particularly, a set of random training cases is generated in each episode (line 6). In each training case, each agent navigates around others to arrive its destination, while interacting with the cellular network (line 10- line 25). It is worth noting that the agents navigate simultaneously and with no communication among each other. At each time step t, each agent first observes the environment, obtains the observable states of other nearby agents and the location information of the GBSs, and then obtains its joint state $\mathbf{s}_t^{jn}$ (line11). Then, based on its current velocity and kinematic constraints, each agent builds an action space A_t (line 12). Using an ϵ-greedy policy, each agent selects a random action with probability ϵ from A_t (line 15), or follows the value network greedily otherwise (lines 17-24). When following the value network to choose actions, each agent performs the following: 1) estimate other nearby agents' motion by filtering their velocities, and estimate their observable states $\hat{\mathbf{s}}_{t+1}^{jno}$ following equation (21) (lines 17-18); 2) predict its next SINR level $\mathcal{L}_{w_{t+1}}$ using the SINR-prediction network $\boldsymbol{\xi}_w$; 3) choose the best action in A_t which has the maximum V_p.

When all agents have arrived their destinations in each training case, trajectories $\mathbf{s}_{i,0:T_1} \forall i$ are then processed to generate a set of state-value pairs $\langle \mathbf{s}_{i,0:T_i}^{jn}, V_{i,0:T_i} \rangle$, where

$$V_{i,t} = \begin{cases} R_{i,t} + \gamma \mathcal{V}(\mathbf{s}_{i,1:t+1}^{jn}) & \text{if } t < T_i, \\ R_{i,t} & \text{if } t = T_i, \end{cases}$$

and a set of location-SINR pairs $\langle [\mathbf{p}_{i,0:T_i}, \mathbf{s}_B^o], L_{w_{i,0:T_i}} \rangle$. The new pairs are used to update D and D_w.

D. Training

We first use optimal reciprocal collision avoidance (ORCA) [18] to generate a number of trajectories that contain a large set of state-value pairs $\{\langle \mathbf{s}^{jn}, V \rangle\}^{N_1}$, where $V = \gamma^{t_g}$ and t_g is the time to reach the destination. The experiences, as input of Algorithm 1, are saved in memory D, which will be refined during training. To train the value network and SINR-prediction network, a set of training points is randomly sampled from the experience set, which contains state-value pairs for $\boldsymbol{\xi}$ or location-SINR pairs for $\boldsymbol{\xi}_w$ from many different trajectories. Then, the networks are finally updated by stochastic gradient descent (back-propagation) on the sampled subsets of experience.

E. Real-Time Navigation

With the trained value network and SINR-prediction network, agent can execute real-time navigation. This process is provided in Algorithm 2.

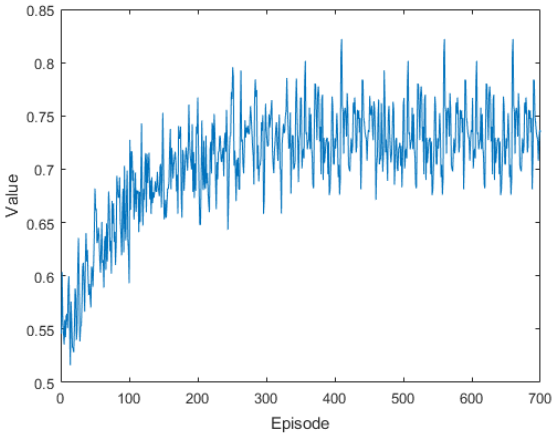
Algorithm 2: Real-Time Navigation

Input: $\boldsymbol{\xi}, \boldsymbol{\xi}_w$

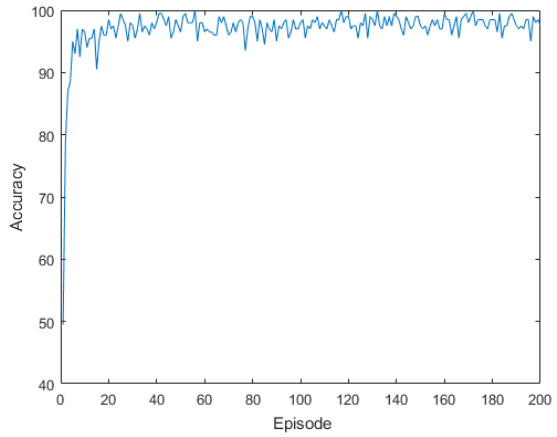
Initialize $\mathbf{s}_0$
while *not reached destination* **do**
  $\mathbf{s}_t^{jn} \leftarrow$ observeEnvironment()
  $A_t \leftarrow$ sampleActionSpace()
  $\hat{\mathbf{v}}_t^{jn} \leftarrow$ filter($\mathbf{v}_{0:t-1}^{jn}$)
  $\hat{\mathbf{s}}_{t+1}^{jno} \leftarrow$ propagate($\mathbf{s}_t^{jno}, \hat{\mathbf{v}}_t^{jn}$)
  for *every* $\boldsymbol{a}$ *in* A_t **do**
    $\hat{\mathbf{s}}_{t+1} \leftarrow$ propagate($\mathbf{s}_t, \mathbf{a}$)
    $\hat{L}_{w_{t+1}} = \mathcal{L}_w([\hat{\mathbf{p}}_{t+1}, \mathbf{s}_B])$
    $R_t \leftarrow$ getReward($\hat{\mathbf{s}}_{t+1}, \hat{\mathbf{s}}_{t+1}^{jno}, \hat{L}_{w_{t+1}}$)
    $V_p = R_t + \gamma \mathcal{V}(\hat{\mathbf{s}}_{t+1}^{jn})$
  $\mathbf{v}_t \leftarrow \text{argmax}_{\mathbf{a} \in A_t} V_p$
  $\mathbf{s}_{t+1} \leftarrow$ executeAction($\mathbf{v}_t$)
return $\mathbf{v}_{0:T-1}$, $\mathbf{s}_{0:T}$

VI. Numerical Results

In this section, we present the numerical results to evaluate the performance of the proposed algorithms. In the illustrations of environment and trajectories in this section, the GBSs are marked by blue triangles, and the yellow areas indicate the communication coverage zones where the agents are able to connect with the cellular network (i.e., $\mathcal{S}_r \geqslant \mathcal{T}_s$). Agents' trajectories are displayed as a list of dots in different colors, and the destinations are marked with crosses. In each flight trajectory, there are four possible outcomes for the agent/UAV: 1) success, if the agent arrives its destination successfully; 2) collision, if it collides with others; 3) disconnection, if the continuous disconnected time is larger than the threshold $\mathcal{T}_t$; 4) stuck, if the agent freezes and stops moving and consequently does not reach the destination. In addition, we also compute the additional average time (referred to also as average more time) needed to reach the destination, when compared with the lower bound (attained when the UAV goes straight towards the destination at the maximum speed). Therefore, we use success rate (SR), collision rate (CR), disconnection rate (DR) and average more time (AMT) to show the performance of the algorithms.



(a) Value of the value netowrk.



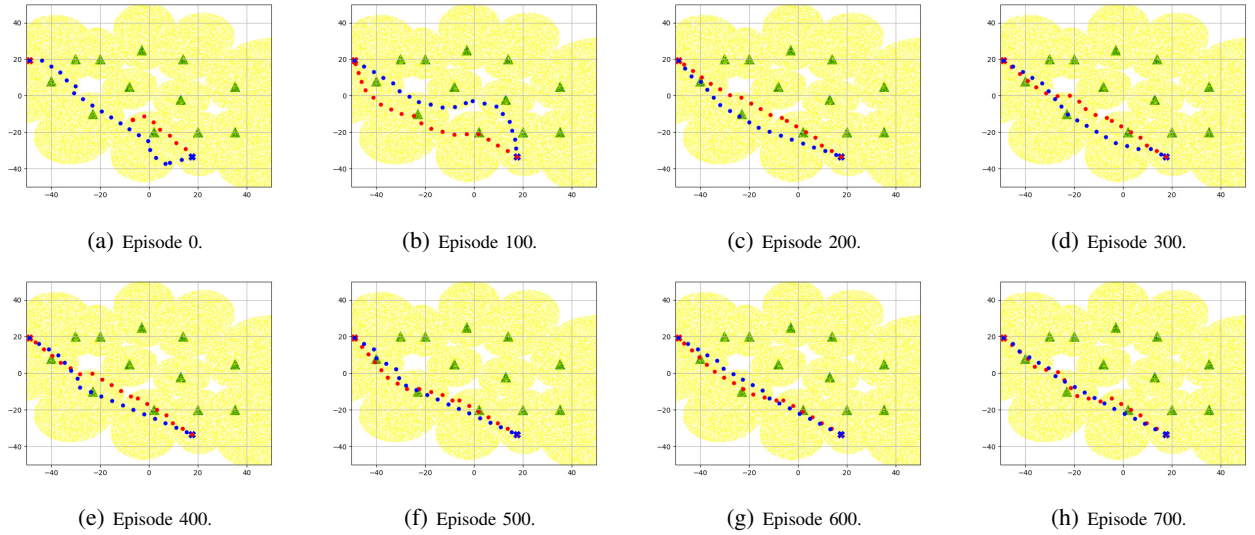
(b) Accuracy of the SINR-prediction network.

Fig. 3: Value of the value network and accuracy of the SINR-prediction network as functions of the number of episodes.

A. Environment Setting and the Networks

Since the agents fly at the same altitude, the area of interest becomes two-dimensional. In the simulations, we consider an area with 12 GBSs deployed. The GBSs transmit with power $P_B = 1$ dBW, have a height of $H_B = 32$ m, and the antenna patterns are set with $\theta^{tilt} = 10^\circ$ and $\theta^{3dB} = 15^\circ$. The UAVs are assumed to fly at a fixed altitude of $H_V = 50$ m. The noise power is $\mathcal{N}_s = 10^{-6}$, and the SINR threshold is $\mathcal{T}_s = -3$ dB. Each UAV, as an independent agent, is able to observe the nearest 8 GBSs' locations and at most 4 other agents' observable states.

We construct the value network via a three-layered DNN of size (64,32,16). The exploration parameter ϵ linearly decays from 0.5 to 0.1. The replay memory capacity is 30000 for the 2-agent scenario and 100000 for scenarios with more than two agents. The SINR-prediction network is constructed via a



(a) Episode 0. (b) Episode 100. (c) Episode 200. (d) Episode 300.

(e) Episode 400. (f) Episode 500. (g) Episode 600. (h) Episode 700.

Fig. 4: Trajectory examples at different episodes during training.

three-layered DNN of size (32,16,8). A standardization layer is utilized after the input layer of both networks. ReLU activation function is used for the input layer and two hidden layer for both networks. Both networks use Adam optimizer, and have learning rate 0.01, batch size 200, and a regularization parameter 0.0001.

B. Convergence in Training

Fig. 3 shows the value of the value network and accuracy of the SINR-prediction network as functions of the number of episodes during training for a 2-agent scenario. Fig. 3(a) shows that the value converges after around 200 episodes. From Fig. 3(b), we can see that the accuracy converges after around 20 episodes, since in each episode 50 random trajectories are generated for each agent, during which more than 15000 location-SINR pairs are collected and used to train the SINR-prediction network.

The trajectory optimization process for two UAVs is displayed in Fig. 4. At episode 0, the SINR-prediction network is initialized with random weights and bias, and is not able to predict the accurate SINR level. Besides, the policy has not been refined by RL. As a result, the two agents are easily getting disconnected or stuck. After 100 episodes of training, the SINR-prediction network is well-trained and able to predict the SINR levels with 97% accuracy. Also, the value network is trained with refined state-value pairs. Thus, the agents can reach their destinations, but with long trajectories to avoid collisions and disconnection. As the training proceeds, the policy improves, leading to shorter expected trajectories. Table I presents the AMT (for the successful trajectories) in different episodes, and we clearly observe the declining AMT values.

Separately, we also compute the AMT in two different scenarios for comparison: 1) the connectivity constraint is not considered for the two-agent trajectory design (CADRL [19]); and 2) the collision avoidance constraint is not considered if there is only one agent. The AMT in these two scenarios are 0.578s, and 0.354s, respectively.

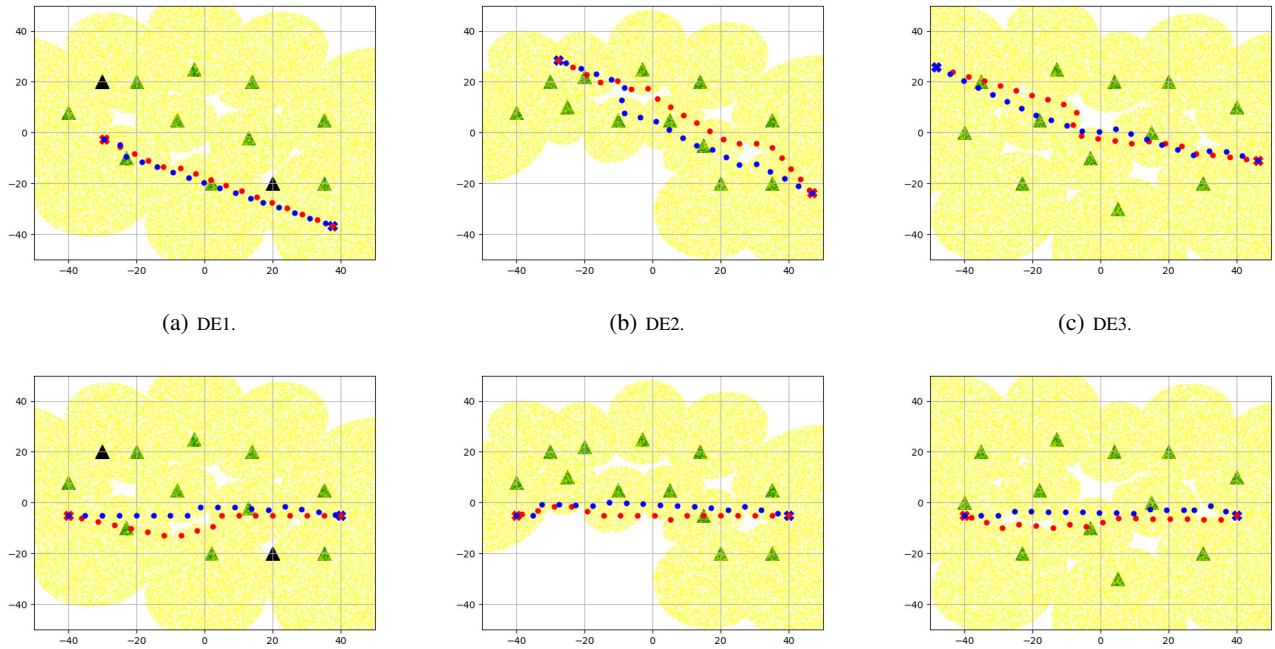
C. Testing of Navigation in Different Environments

In the proposed RLTCW-SP algorithm, an SINR-prediction network is trained to predict the SINR level. In an ideal scenario, the antenna pattern information of GBSs may be available to the agents, and then the agents are able to predict the SINR with that information. In this subsection, we compare the performances of the following three algorithms: 1) the agents are able to get the antenna pattern information of the GBSs, and then predict the SINR directly (referred to as RLTCW-AW); 2) the proposed RLTCW-SP algorithm, which uses the location-SINR memory, and trains an SINR-prediction network to predict the SINR level; 3) the agents do not predict the SINR and only use the value network to encode the interaction with the cellular network (referred to as RLTCW). The navigation test is done in three types of environments: 1) the same environment as in the training; 2) the same environment but two BSs are not operational for the UAV (due to congestion, malfunction, resource allocation to ground users, or GBS activation schedule), an illustration of which is presented in Fig. 5(a); 3) different environment with different GBS deployment, illustrations of which are presented in Figs. 5(b) and 5(c). Fig. 5 also presents examples of trajectories that the agents perform using the proposed RLTCW-SP algorithm in a challenging scenario in which the destination of one UAV is the starting point of the other UAV. Environments displayed in Figs. 5 (a) (b) and (c) are referred as DE1, DE2 and DE3 (using DE as the abbreviation for different environment).

The performance of the three algorithms in different environments are presented in Table II. As expected, the RLTCW-AW with the perfect knowledge of antenna patterns has the best performance, and the RLTCW-SP algorithm has slightly lower performance which is due to the potential inaccuracies in the SINR prediction, while the performance of RLTCW is substantially lower compared to the other two, due to very high DR (disconnection rate). In addition, the SR (success rate) performance of the proposed RLTCW-SP algorithm decreases only slightly in different environments, and how large the

TABLE I: Average more time need to reach destination at different episodes.

Number of Episodes	0	100	200	300	400	500	600	700
Average More Time (s)	2.45	1.723	0.972	0.8707	0.8007	0.772	0.766	0.712



(a) DE1. (b) DE2. (c) DE3.

(d) DE1. (e) DE2. (f) DE3.

Fig. 5: Illustrations of different environments used in navigation testing, and trajectory examples when using proposed RLTCW-SP algorithm. In (a) and (d), the two BSs in black are not operational for the UAVs.

TABLE II: Performance of different algorithms in different environments in terms of success rate (SR), collision rate (CR), and disconnection rate (DR) **(all rates are in %)**.

	Same Environment			DE1			DE2			DE3		
	SR	CR	DR	SR	CR	DR	SR	CR	DR	SR	CR	DR
RLTCW-AW	94.8	2.8	0	95	5	0	87.6	6.2	0.6	92.2	6.8	0
RLTCW-SP	94.02	4	0.06	93.4	5.8	0.07	86.1	7.4	0.7	91.8	6.2	0.1
RLTCW	70.4	3.4	23.4	77.95	3.3	18.2	59.3	4.8	28.4	70.6	4.2	22.9

decrease is depends on which environment is used in testing. When there are large and wide out-of-coverage zones in the environment (as shown in Fig. 5(b)), the SR performance decreases more. The reason is that the wide out-of-coverage zones are more likely to make the agent get stuck at the edge and not be able to decide which direction to go.

D. Navigation in Different Settings

In this subsection, we present simulation results on the trajectories when the GBSs have different antenna patterns and when the UAVs fly at different heights. The SINR threshold is $\mathcal{T}_s = -4$ dB in this subsection. In. Figs. 6 (a) and (d), we provide two different trajectory examples when we have $H_V = 50$ m, $\theta^{tilt} = 10^\circ$ and $\theta^{3dB} = 15^\circ$. In Figs. 6 (b) and (e), UAV altitudes are increased to $H_V = 100$ m, and we notice that due to larger path loss and smaller antenna gains, coverage zones shrink, which in turn potentially increases the length of the trajectories. In Figs. 6 (c) and (f), GBSs have larger downtilting angle and 3dB beamwith of the main lobe. In this case, the UAVs experience smaller received power from the main link and potentially larger interference, leading to substantially smaller SINR levels. Therefore, the coverage zones in Figs. 6 (c) and (f) are smaller than those in Figs. 6 (a) and (d) and even Figs. 6 (b) and (e). In all cases, we note that UAVs successfully find different trajectories to meet the connectivity requirements and adapt to different coverage zones.

E. Navigation in Environments with Obstacles/No-Fly Zones

The proposed RLTCW-SP algorithm can also be used for navigation in environment with obstacles that are regarded as non-moving agents. For instance, the trained networks for the 2-agent scenario can be used for one agent navigation in an environment with obstacles or no-fly zones. More specifically,

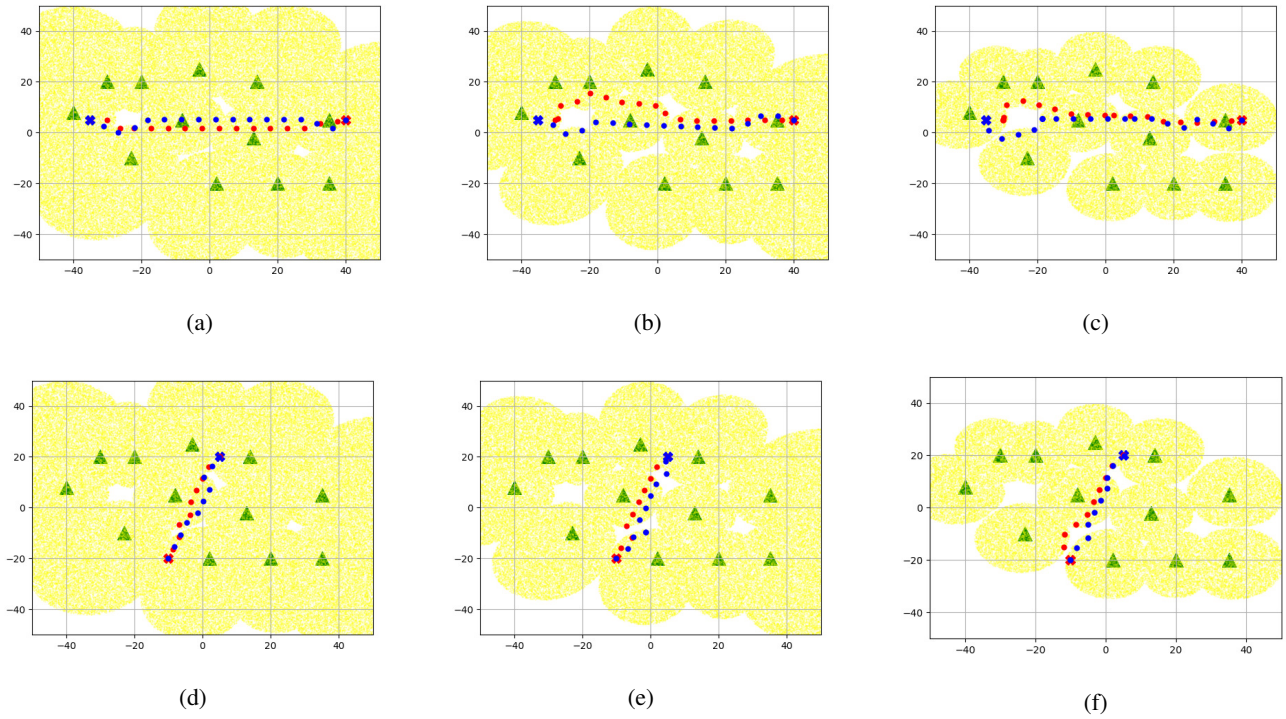


Fig. 6: Trajectory examples in environments with different settings, i.e., in (a) and (d), $H_V = 50$ m, $\theta^{tilt} = 10^\circ$ and $\theta^{3dB} = 15^\circ$; in (b) and (e), $H_V = 100$ m, $\theta^{tilt} = 10^\circ$ and $\theta^{3dB} = 15^\circ$; and in (c) and (f), $H_V = 50$ m, $\theta^{tilt} = 15^\circ$ and $\theta^{3dB} = 35^\circ$.

the agent can observe the nearest obstacle, and takes the obstacle's location in the joint state for choosing actions. Fig. 7 displays two illustrations. In this setting, obstacles can be considered as actual obstacles (e.g., tall buildings or structures) or they can model no-fly zones for the UAVs.

F. Navigation with More Than Two UAVs

Fig. 8 and Fig. 9 display the illustrations for 5-agent and 8-agent navigation scenarios, respectively. The SR, CR, DR and AMT performances are presented in Table III. We note that the CR increases when more agents are in the environment. As mentioned before, the agents can observe a maximum of 4 nearest agents in the environment. Therefore, for the 8-agent scenario, several agents are non-observable and as a result the CR can increase when compared with scenarios involving smaller number of agents. On the other hand, when there are more agents in the same area, the interactions become more complex, and harder for the algorithm to handle. However, we note that the performance regarding the SR is still above 90%. Table III also shows that the agents need more time to reach the destination when there are more agents in the environment.

TABLE III: Performance for 2/5/8-agent navigation.

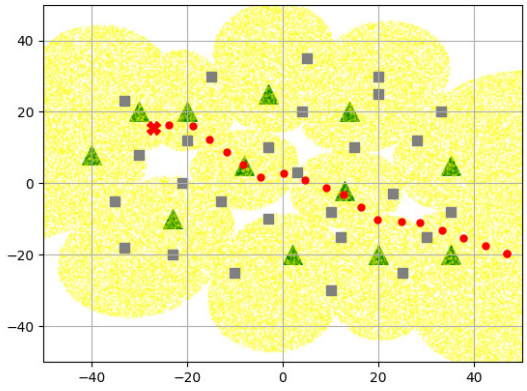
	SR(%)	CR(%)	DR(%)	AMT(s)
2-Agent	93.02	4	0.06	0.712
5-Agent	91.12	6.32	0.8	1.001
8-Agent	90.075	7.25	0.25	1.28

VII. Conclusion

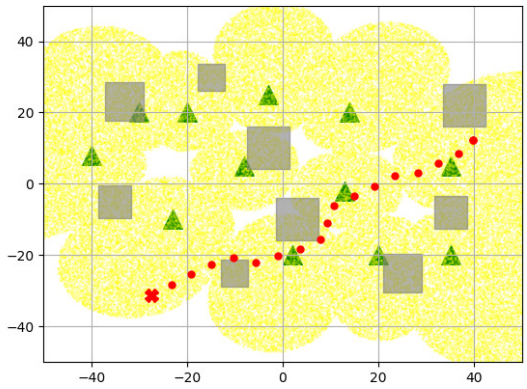
In this work, we have studied multi-UAV trajectory optimization with collision avoidance and wireless connectivity constraints. In establishing the wireless connectivity, we have taken into account the antenna radiation patterns, path loss, and SINR levels. We have formulated trajectory optimization as a sequential decision making problem and proposed a decentralized deep reinforcement learning algorithm. In particular, a value neural network has been developed to encode the expected time to the destination given the agent's joint state. An SINR-prediction neural network has been designed, using accumulated SINR measurements obtained when interacting with the cellular network, to map the GBS locations into the SINR levels in order to predict the UAV's SINR levels. We have investigated the performance in terms of success rate, collision rate, disconnection rate, and average more time. In the numerical results, we have considered various scenarios (e.g., with GBS deployments different from the setting in the training environment, different UAV heights, different antenna patterns, and obstacles/no-fly zones) and we have shown that with the value network and SINR-prediction network, real-time navigation for multi-UAVs can be efficiently performed in different environments with high success rates.

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.

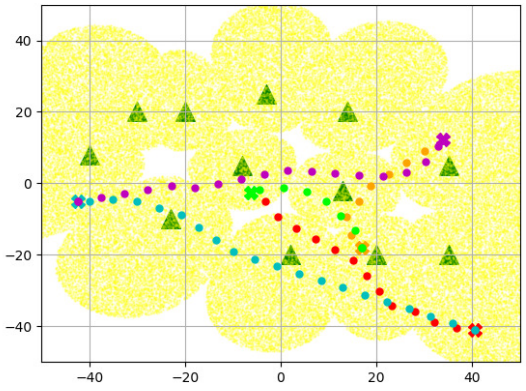


(a) Example 1.

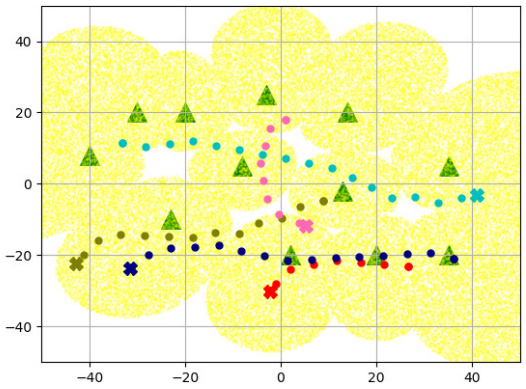


(b) Example 2.

Fig. 7: Trajectory examples in environments with obstacles/no-fly zones.



(a) Example 1.



(b) Example 2.

Fig. 8: Trajectory examples for 5-agent navigation.

[2] C. Liu, M. Ding, C. Ma, Q. Li, Z. Lin, and Y. Liang, "Performance analysis for practical unmanned aerial vehicle networks with LoS/NLoS transmissions," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6.

[3] M. M. Azari, F. Rosas, and S. Pollin, "Cellular connectivity for UAVs: Network modeling, performance analysis and design guidelines," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2019.

[4] S. Zhang, Y. Zeng, and R. Zhang, "Cellular-enabled UAV communication: A connectivity-constrained trajectory optimization perspective," *IEEE Transactions on Communications*, vol. 67, no. 3, pp. 2580–2604, 2019.

[5] E. Bulut and I. Guevenc, "Trajectory optimization for cellular-connected UAVs with disconnectivity constraint," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[6] B. Khamidehi and E. S. Sousa, "A double Q-learning approach for navigation of aerial vehicles with connectivity constraint," *arXiv preprint arXiv:2002.10563*, 2020.

[7] Y. Zeng, X. Xu, S. Jin, and R. Zhang, "Simultaneous navigation and radio mapping for cellular-connected UAV with deep reinforcement learning," *arXiv preprint arXiv:2003.07574*, 2020.

[8] U. Challita, W. Saad, and C. Bettstetter, "Interference management for cellular-connected UAVs: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2125–2140, 2019.

[9] S. Zhang and R. Zhang, "Trajectory optimization for cellular-connected UAV under outage duration constraint," *Journal of Communications and Information Networks*, vol. 4, no. 4, pp. 55–71, 2019.

[10] Y. Zeng, X. Xu, and R. Zhang, "Trajectory design for completion time minimization in UAV-enabled multicasting," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2233–2246, 2018.

[11] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[12] S. De Bast, E. Vinogradov, and S. Pollin, "Cellular coverage-aware path planning for UAVs," in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2019, pp. 1–5.

[13] B. Khamidehi and E. S. Sousa, "Power efficient trajectory optimization for the cellular-connected aerial vehicles," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1–6.

[14] H. Yang, J. Zhang, S. Song, and K. B. Lataief, "Connectivity-aware UAV path planning with aerial coverage maps," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

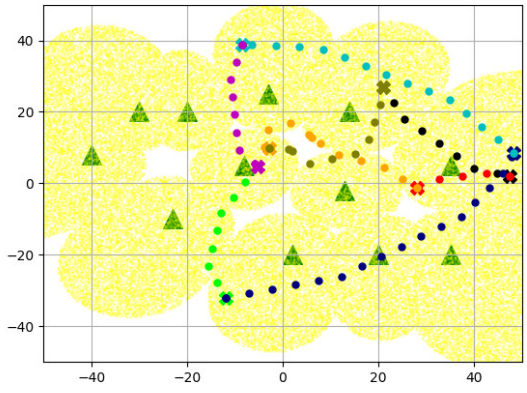
[15] S. Zhang and R. Zhang, "Radio map based 3D path planning for cellular-connected UAV," *IEEE Transactions on Wireless Communications*, 2020.

[16] X. Mu, Y. Liu, L. Guo, and J. Lin, "Non-orthogonal multiple access for air-to-ground communication," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 2934–2949, 2020.

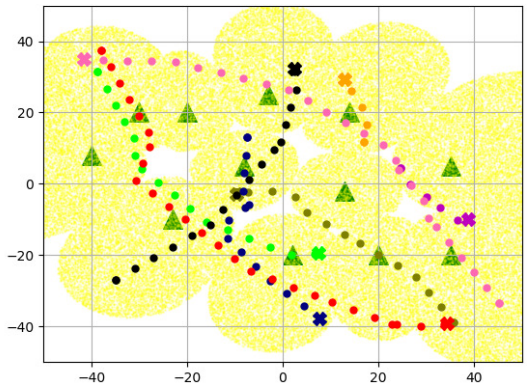
[17] J. van den Berg, Ming Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[18] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[19] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-



(a) Example 1.



(b) Example 2.

Fig. 9: Trajectory examples for 8-agent navigation.

communicating multiagent collision avoidance with deep reinforcement learning," *CoRR*, vol. abs/1609.07845, 2016. [Online]. Available: http://arxiv.org/abs/1609.07845

[20] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *CoRR*, vol. abs/1703.08862, 2017. [Online]. Available: http://arxiv.org/abs/1703.08862

[21] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052–3059.

[22] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *arXiv preprint arXiv:1910.11689*, 2019.

[23] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 1, 2003, pp. 25–30 Vol.1.

[24] D. Bauso, L. Giarre, and R. Pesenti, "Multiple UAV cooperative path planning via neuro-dynamic programming," in *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, vol. 1, 2004, pp. 1087–1092 Vol.1.

[25] B. D. Song, J. Kim, and J. R. Morrison, "Rolling horizon path planning of an autonomous system of UAVs for persistent cooperative service: MILP formulation and efficient heuristics," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 241–258, 2016.

[26] T. B. Wolf and M. J. Kochenderfer, "Aircraft collision avoidance using Monte Carlo real-time belief space search," *Journal of Intelligent & Robotic Systems*, vol. 64, no. 2, pp. 277–298, 2011.

[27] S. Temizer, M. Kochenderfer, L. Kaelbling, T. Lozano-Pérez, and J. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *AIAA guidance, navigation, and control conference*, 2010, p. 8040.

[28] H. Bai, D. Hsu, M. J. Kochenderfer, and W. S. Lee, "Unmanned aircraft collision avoidance using continuous-state POMDPs," *Robotics: Science and Systems VII*, vol. 1, pp. 1–8, 2012.

[29] Y. Lin and S. Saripalli, "Collision avoidance for UAVs using reachable sets," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 226–235.

[30] T. Yu, J. Tang, L. Bai, and S. Lao, "Collision avoidance for cooperative UAVs with rolling optimization algorithm based on predictive state space," *Applied Sciences*, vol. 7, no. 4, p. 329, 2017.

[31] J. Ikuno, M. Wrulich, and M. Rupp, "3GPP TR 36.814 (v9. 0.0): Evolved universal terrestrial radio access (E-UTRA); further advancements for E-UTRA physical layer aspects," *Tech. Rep.*, 2010.

[32] J. Chen, D. Raye, W. Khawaja, P. Sinha, and I. Guvenc, "Impact of 3D UWB antenna radiation pattern on air-to-ground drone connectivity," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Aug 2018, pp. 1–5.

[33] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016. [Online]. Available: https://doi.org/10.1177/0278364915619772

[34] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *CoRR*, vol. abs/1809.08835, 2018. [Online]. Available: http://arxiv.org/abs/1809.08835

[35] J. Snape, J. v. d. Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.