A Reinforcement Learning Simulator for Multi-UAV Based Network Coverage Problem

Dorian Tonnis, Loïc Desgeorges, Isabelle Guérin Lassous, Laëtitia Matignon

A Reinforcement Learning Simulator for Multi-UAV Based Network Coverage Problem

Dorian Tonnis
Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS UMR5205
Villeurbanne, France
dorian.tonnis@etu.univ-lyon1.fr

Loïc Desgeorges, Isabelle Guérin Lassous
Université Claude Bernard Lyon 1, ENS de Lyon, CNRS, LIP UMR5668
Lyon, France
firstname.name@univ-lyon1.fr

Laëtitia Matignon
Université Claude Bernard Lyon 1, CNRS, INSA Lyon, LIRIS UMR5205
Villeurbanne, France
laetitia.matignon@univ-lyon1.fr

***Abstract*—UAV-based wireless networks can be deployed to provide a network coverage to users who have no or poor network connection. Unlike traditional model-based approaches that require predefined assumptions before UAV deployment, reinforcement learning (RL) offers a promising alternative but requires a realistic simulator for training the proposed strategies. None of the existing open-source simulators provide both realistic wireless communications while enabling the training, in a cluttered environment, of multi-UAVs movement strategies using RL. Thus, in this paper, we present a simulator where we focus on improving the modeling of the network access part, *i.e.*, the communications between the UAVs and the users, by integrating signal propagation, physical rate adaptation and medium access sharing models. We evaluate the performance of a standard independent RL algorithm trained in our simulator across various use case scenarios, and compare the results obtained using different learning objectives. Results show that the classical formulation, commonly found in the literature and based on a a simplified wireless network model, underperforms in terms of communication quality.**

***Index Terms*—Reinforcement learning training simulator, wireless network coverage, UAV deployment, multi-agent system, wireless communications**

I. Introduction

Thanks to the recent democratization of UAVs (Unmanned Aerial Vehicles), they can be used in a variety of applications. Especially, from the perspective of next-generation wireless networks (6G), networks based on a fleet of autonomous UAVs have emerged as a promising tool for several reasons [1]. In particular, thanks to their fast and flexible deployment, UAV-based wireless networks can provide a network coverage to users who lack or have poor connectivity. However, many challenges arise when a fleet of UAVs must be deployed to provide network services to users. The UAVs need to be positioned correctly relatively to both users and other UAVs so that the network they form performs well, while typically aiming to maximize users coverage and minimize the number used UAVs.

Among the existing solutions to the network coverage problem and deploying a UAV-based wireless network, many rely on constructive models that need to be chosen *a priori* before deployment (*e.g.,* [2]). However, the environment in which the UAVs must be deployed may be unknown and may vary from one mission to another. The users and their density may also be unknown and variable, as well as the traffic demand. Reinforcement learning (RL)-based approaches have received increasing research interest in recent years for UAV-based wireless networks due to their ability to learn adaptive and decentralized control strategies in complex and dynamic environments [3], [4].

An important factor for training RL strategies is the availability of a realistic simulator. A few UAV-specific simulators exist that are integrated with RL frameworks, *e.g.,* Aerial Gym [5] or the OmniDrones platform [6] of ISAAC Sim. But in these simulators the wireless communications are not modeled. Other simulators have been developed for training RL strategies for wireless network solutions. For instance, in [7], the authors develop a bridge that connects *OpenAI Gym*, an open-source library providing a unified interface to wrap simulators into RL environments, with *ns-3*, a well-known open-source network simulator. This enables RL agents to interact with realistic network simulations. Others couple *OpenAI Gym* with GNU radio tools [8] or with simulation of dynamic spectrum access and jamming [9]. These simulators are dedicated to wireless networks and do not include any 3D scenarios with potential obstacles (like buildings) nor the possibility of having UAVs moving in these 3D scenarios. Few approaches manage to simulate both UAV navigation and network components for training RL strategies. Recently [10] provides a simple UAV simulation to optimise network coverage and learn UAV navigation strategies to deliver services to customers. Unfortunately, the code of this simulator is not open-source and has not been shared after request. In [11], the authors have developed an advanced simulator with 3D scenarios including obstacles, users, potentially mobile, and UAVs that can communicate with users. The code respects *OpenAI Gym* standard interface and is open source. On the other hand, the modelling of wireless communications is very simplistic as it uses a range-based communication model meaning that a UAV and a user can communicate as soon as their distance is lower than a predefined threshold. Moreover, the physical data rate used to communicate is constant whatever the distance between the UAV and the user.

In this paper, we propose a simulator to train and test RL strategies for deploying a UAV-based wireless network in order to provide a network coverage to users who have no or

poor network connection. Our work is based and extends the simulator initially developed in [11]. Our contributions are the following:

- we have completely refactored the simulator of [11] to a more recent version of Python (Python 3.13) and to comply with the *PettingZoo* [12] standard interface for multi-agent RL environments;
- under the assumption that UAVs and users communicate with the Wi-Fi technology, we have improved the communication model by integrating i) a signal propagation model taking into account potential obstacles, ii) a physical rate adaptation ensuring that the used physical data rate is not constant and adapts according to the radio conditions, and iii) a medium access sharing model that, based on the Wi-Fi medium sharing, estimates the users' throughput;
- we assess how one standard multi-agent RL algorithm performs when trained within our simulator under various use case scenarios, and compare the results with those obtained under a simple wireless network model like the one considered in [11].

The paper is organised as follows. In Section II, we describe and define the network coverage problem. We then describe, in Section III, the simulator we developed to train and test RL-based UAV navigation solutions. In Section IV, we present the RL-based UAV navigation algorithm we used to test our simulator. The results obtained on different scenarios with the chosen RL-based algorithm are discussed in Section V. We conclude in Section VI.

II. Problem Statement

In this paper, we focus our work on the network coverage application where a fleet of UAVs (also called agents hereafter) is used as communication relays [13], as illustrated in Fig. 1. Thus, UAVs can extend the connectivity of an existing network infrastructure to unconnected areas. UAVs are connected to this network infrastructure and relay the packets to and from unconnected users if the UAVs are positioned in such a way that they can communicate with them. The UAVs can also, when needed, directly communicate when they are within communication range. In this paper we focus on the access part, *i.e.,* on the communications between the UAVs and the users. We assume that all UAVs are equipped with a long range wireless communication technology, *e.g.,* a satellite technology or a mobile technology -5G- and can efficiently and directly communicate with the network infrastructure through a base station. We also assume that the users are located on a geographical area and that their location is unknown to UAVs and the network infrastructure. The geographical area can include obstacles, such as, for instance, buildings or mountains. These obstacles can impede the radio signal propagation, and thus the communication quality between the UAVs and the users. The location and the shape of the obstacles are also unknown to UAVs and the network infrastructure.

In this paper, we seek to answer the following problem, called hereafter **the network coverage problem**: how to position UAVs so that users can efficiently receive data from the network infrastructure? As explained hereafter, the UAV deployment will either seek to maximize the number of connected users or seek to maximize the mean user throughput on the access part (*i.e.,* between UAVs and users). Fig. 1 shows a solution to the network coverage problem with 2 clusters of users and 2 UAVs with the objective to maximize the mean user throughput.

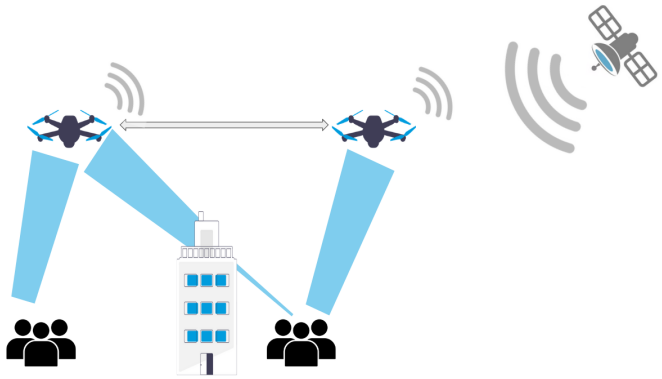


Fig. 1: A solution to the network coverage problem with 2 clusters of users and 2 UAVs. The UAVs search for their best position to provide, on the access part, a good connectivity to the network infrastructure (that can be accessed by a satellite in the figure). The width of the communication links between a UAV and users (blue line) represents the user throughput. This illustration highlights the impact of obstacles on signal quality.

As the environment and the users' location are unknown, we propose a reinforcement learning (RL)-based approach to solve the network coverage problem, where UAVs learn to adapt their positions in order to maximize a reward-based objective, *e.g.,* covering the most users or ensuring a high average user throughput.

III. Simulator

Training RL strategies requires the availability of a realistic simulator, as the more realistic the simulator is, the more likely the solution will be to make the right decisions in a real environment. In this section we present our simulator[1] designed to train RL-based UAV navigation policies for the network coverage problem, incorporating a realistic communication model for the access part.

A. Baseline simulator refactoring

Our work is based on the open-source simulator presented in [11], where multiple UAVs can move in a 2D or 3D grid-based environment including obstacles and user clusters (potentially dynamic). Since this simulator was outdated, we first performed a refactoring to ensure its compatibility with a more recent version of Python (*i.e.,* Python 3.13). Additionally, we adapted it for integration with the *PettingZoo* API [12],

[1] available at https://gitlab.liris.cnrs.fr/lmatigno/drone.

providing a standardized interface for a wide variety of multi-agent RL environments, enabling seamless integration with RL libraries for easy development and benchmarking. Fig. 2 shows a screenshot of our simulator with a specific 3D scenario.

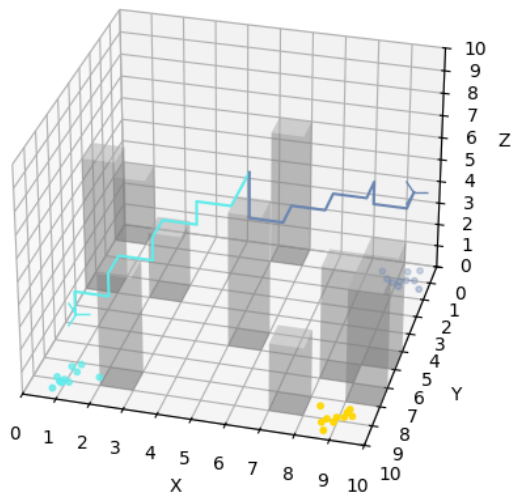


Fig. 2: Screenshot of a 3D environment with our simulator. Gray elements are obstacles, colored crosses represent UAVs and the colored lines their trajectory, users are shown as colored dots, depending on whether they are connected to a UAV. The yellow color means that the users are not covered by any UAV.

B. A more realistic communication model

In the simulator presented in [11], radio communications are very simply modelled and based solely on connectivity: according to the distance between the user and the UAV, they are either connected or not. Moreover, the physical data rate used to communicate, *i.e.,* the speed at which data is transmitted over the radio medium, is constant. This does not correspond to the reality of radio technologies, which very often adapt the physical data rate according to the radio quality of the transmissions. To improve the communication model initially used in [11], our simulator includes:

1) a signal propagation model;
2) a physical rate adaptation model;
3) a medium access sharing model.

We first considered that Wi-Fi communication technology was used between UAVs and users, and thus adapted the previous models accordingly. Any other wireless technology can be considered by adapting the models. We also assume that UAVs operate on orthogonal radio channels.

Firstly, concerning the signal propagation model, we use the path loss model defined in [14] and inspired from [15]. We chose this model because it simply takes into account the impact of obstacles on signal quality. We remind that the path loss, denoted PL and expressed in dB, relates to the reception power P_{Rx} (of the the received signal) and the transmission power P_{Tx} (of the emitted signal) as follows:

$$P_{Rx} = P_{Tx} - PL \tag{1}$$

The path loss without obstacles is formulated as follows:

$$PL = \begin{cases} PL_0 + 10\mu \log_{10}\left(\frac{d}{d_0}\right) + X_g & \text{if } d \geq d_0 \\ PL_0, & \text{otherwise} \end{cases} \tag{2}$$

where PL_0 is the path loss at the reference distance d_0 calculated using the Friis free-space path loss model, d is the distance between the emitter and the receiver, μ is the path loss exponent that depends on the environment (we use $\mu = 2$ in free space), X_g represents the variations of the path loss caused by shadowing effects and/or multiple paths.

In order to better take into account the impact of the different obstacles on the communication quality like in [14], we add a dissipation model inside the obstacles leading to high loss of the signal strength when the signal goes through an obstacle. The resulting path loss is expressed as follows:

$$PL = \begin{cases} PL_0 + 10\mu \log_{10}\left(\frac{d-d_{obst}}{d_0}\right) \\ \quad +X_g + PL_{obst}(d_{obst}), & \text{if } d - d_{obst} \geq d_0 \\ PL_0 + PL_{obst}(d_{obst}), & \text{otherwise} \end{cases} \tag{3}$$

where d_{obst} is the length of the path inside the obstacle (we assume that $d_{obst} \geq 1m$) and PL_{obst} is the path loss in dB due to obstacle attenuation. By using a simplified model of the path loss, this latter is defined as:

$$PL_{obst}(d_{obst}) = 10\mu_{obst} log_{10}(d_{obst}) + K \tag{4}$$

where μ_{obst} is the path loss exponent for the obstacle and K is a constant.

Secondly, we include a physical rate adaptation model in our simulator. Indeed, in Wi-Fi (and in many wireless technologies), the physical data rate is not constant but dynamic: the emitter adapts its physical date rate when transmitting according to its estimation of the communication quality. We chose a simple physical rate adaptation based on the reception power, but any other model can be integrated on the simulator. Table I shows the selected physical data rate according to the value of the reception power. We consider that a user u is connected to UAV i if the reception power at u of the signal received from i, denoted $P_{R_x,i}(u)$, is superior to $-70dBm$:

$$c_i(u) = \begin{cases} 1 & \text{if } P_{R_x,i}(u) \geq -70 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Thirdly, we integrate a radio medium sharing model of the Wi-Fi technology. Indeed, this radio medium sharing has a great impact on the users' throughput and we believe that it should be considered as soon as user throughput is taken into account in the solution to the coverage problem. In addition, the presence of different physical data rates also has a strong impact on the user throughput [16], as illustrated in Fig. 3. In this figure, a UAV serves two users. User u_1 is close to the

TABLE I: Correspondence table between the reception power and the physical data rate.

Reception Power (dBm)	Physical data rate (Mbit/s)
$-70 \leq P_{Rx} < -66$	6.5
$-66 \leq P_{Rx} < -63$	15.0
$-63 \leq P_{Rx} < -60$	30.0
$-60 \leq P_{Rx} < -55$	40.5
$-55 \leq P_{Rx} < -50$	52.0
$-50 \leq P_{Rx} < -40$	60.0
$-40 \leq P_{Rx}$	90.0

UAV and due to the good radio conditions, the UAV transmits to user u_1 with a high physical data rate, denoted $\rho_1(u_1)$. On the other hand, user u_2 is further away from the UAV and the radio conditions are poor. Therefore, the UAV transmits to user u_2 with a low physical data rate $\rho_1(u_2)$. In Wi-Fi, when the single-user transmission is used[2], the UAV transmits to a single user and the other user has to wait for the end of this transmission before being served. If the traffic rates for the two users (*i.e.,* the rate at which the UAV receives data for the users) are identical and each transmitted packet contains N bits of user data, then users u_1 and u_2 will receive their data with a throughput equal to $\frac{N}{T}$, with $T = t_1 + t_2$ the total time for UAV to transmit data to all its users, and $t_1 = \frac{N}{\rho_1(u_1)}$ (resp. $t_2 = \frac{N}{\rho_1(u_2)}$) the time for UAV to transmit data to user u_1 (resp. u_2). We see that, in this example, the throughput of user u_1 is impacted by the low physical rate $\rho_1(u_2)$ used to transmit to user u_2. Note that the use of the CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) mechanism in Wi-Fi implies the same radio medium sharing if the users were transmitting to the UAV. The set of notations used in this paper is in Table II.

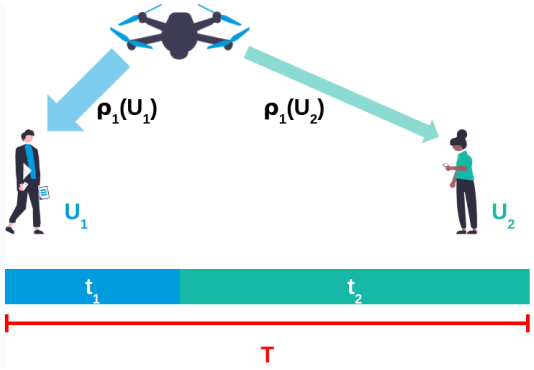


Fig. 3: Radio medium sharing illustration with one UAV and 2 users.

IV. UAV NAVIGATION ALGORITHM

In this section, we detail the RL approach adopted to address our network coverage problem and the algorithm we used in Section V. However, the simulator remains flexible thanks to its integration with *PettingZoo*, and can accommodate other RL algorithms as well.

[2]which is often the case of the Wi-Fi cards embedded in off-the-shelf UAVs.

TABLE II: Table of notations

Notation	Definition
V	set of UAVs
U	set of users
O_i	set of observations for agent i
o_i	observation of agent i
x_i, y_i, z_i	coordinates of the UAV i
A_i	set of actions for agent i
$\mathcal{T}$	transition function
$\mathcal{R}$	reward function
π_i	policy of agent i
Q	action-value function
PL	Path loss
$P_{R_x,i}(u)$	reception power at u from i
P_{T_x}	Transmission power
$\rho_i(u)$	physical data rate to transmit from i to u
TH_i	mean throughput of users connected to UAV i
$c_i(u)$	connection of user u to UAV i

A. Multi-Agent Sequential Decision Process

In our problem, a set of UAVs learn to take decisions within an environment over multiple time steps to achieve a specified goal. The first step with RL is to model the problem under the mathematical framework for sequential decision process. In our context of multi-agent system, it is here a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) defined by:

- V a finite set agents (or UAVs),
- S the set of global states of the environment,
- O_i the set of observation of agent i, as each agent partially observes the state of the environment, according to an observation function,
- A_i the action space of agent i, and $A = A_1 \times \cdots \times A_{|V|}$ is the joint action space,
- $\mathcal{T} : S \times A \times S \rightarrow [0, 1]$ the transition function representing the dynamics of the environment, *i.e.,*, how the environment state changes in response to all agents' actions,
- $\mathcal{R} : S \times A \times S \rightarrow \mathbb{R}$ the reward function, where all agents receive the same reward.

The most common assumption in RL is that the dynamics of the process, *i.e.,* the transition and reward functions, are *a priori* unknown to the agents. Thus they learn from interactions with the environment, where at each time step, each agent i makes partial observation o_i of the current state s of the environment and chooses an action a_i. Then the environment transitions to a new state $s' \sim \mathcal{T}(.|s, a)$ where a is the joint action, and each agent receives a reward $r = \mathcal{R}(s, a, s')$. An **episode** is a complete sequence of interactions between the agents and the environment, starting from an initial state and ending at a terminal state or after completing a maximum number of T time steps.

B. Independent Learners

To solve our multi-agent decision process, we use a basic approach called independent learning (IL) [17], which applies single-agent RL to each agent independently. IL is commonly used and has been shown to perform competitively in diverse learning tasks [18], [19]. Independent learners do not explicitly

observe the other agents; the effects of others' actions are part of the environment dynamics from the perspective of each learning agent and especially, the common reward in the Dec-POMDP provides some insight into how others behave. Each agent i learns its own policy $\pi_i : O_i \times A_i \rightarrow [0, 1]$, where $\pi_i(o_i, a_i)$ gives the probability that agent i selects action a_i when observing o_i.

C. Problem Modeling

We now explicitly define the observations, actions, and rewards for our network coverage problem. In the following, the set of users in the environment is noted U.

a) Discussion about the reward function: The reward function defines the learning objective that is, for each UAV, to learn a movement policy that ensures, for all users, an efficient network coverage. We are considering two strategies to model the reward.

Reward 1: The majority of state-of-the-art approaches (*e.g.,* [10], [11]) aim to maximize the total number of connected users with a reward function based on this number. Following a similar approach, we propose a first formulation of the reward that is function of the number of users not connected:

$$r = -\sum_{i \in V} \sum_{u \in U} \mathbb{1}_{c_i(u)=0} \tag{6}$$

Reward 2: The previous reward only considers the connectivity as the main metrics to evaluate the quality of the communication. However, this does not take into account the fact that connected users may have low throughput. To address this, we propose to use in the reward a weighted mean throughput per UAV:

$$r = T_h - \rho_{max} \tag{7}$$

where ρ_{max} is the maximal physical data rate[3], and T_h is a weighted mean throughput per UAV:

$$T_h = \sum_{i \in V} \frac{\sum_{u \in U_i} N_i(u)}{T_i} \frac{|U_i|}{|U|} \tag{8}$$

where U_i is the set of users connected to UAV i, $N_i(u)$ is the number of bits sent by UAV i to user u, and T_i is the total time for UAV i to transmit data to all its users (cf. Fig. 3). The used weight is function of the proportion of users covered by each UAV. The mean throughput per user is then easily computed from T_h as it corresponds to $\frac{T_h}{|U|}$.

b) Observation space: We consider that each UAV agent can observe its current and previous position in the environment (that is assumed to take discrete values), and the current and previous mean throughput per UAV. At each time step t, the observation of agent i is then:

$$o_i^t = < x_i^t, y_i^t, z_i^t, T_h^t, x_i^t - x_i^{t-1}, y_i^t - y_i^{t-1}, z_i^t - z_i^{t-1}, T_h^t - T_h^{t-1} > \tag{9}$$

where x_i^t, y_i^t and z_i^t are the coordinates related to the position at t of the UAV i, and T_h^t is the mean throughput per UAV at t (cf. Eq. 8). Current and previous observations help the agent infer hidden state information in partially observable environments.

c) Action space: At each time step, each UAV can move in one of the possible directions by ds meters or do nothing[4]. The actions are discrete, thus the action set for each UAV i is defined as follows:

$$A_i = \{(+ds, 0, 0), (-ds, 0, 0), (0, +ds, 0), (0, -ds, 0), (0, 0, +ds), (0, 0, -ds), (0, 0, 0)\}. \tag{10}$$

Algorithm 1 Independent Deep Q-networks (IDQN) for n agents

Initialize n Q-functions with random weights $\theta_1, ..., \theta_n$
Initialize n target $\hat{Q}$-functions with weights $\hat{\theta}_1 = \theta_1$,...,$\hat{\theta}_n = \theta_n$
Initialize n replay buffer B_i
for each episode **do**
  Collect initial observations $o_1, ..., o_n$
  for each time step $t = 1..T$ **do**
    Each agent i chooses an action a_i from o_i using an ϵ-greedy policy based on $Q(o_i, a_i; \theta_i)$
    Execute joint action $a = (a_1, ..., a_n)$, observe reward r and next observations $o'_1, ..., o'_n$
    Store transition (o_i, a_i, r, o'_i) of each agent i in replay buffer B_i
    for each agent $i = 1..n$ **do**
      Sample random mini-batch of b transitions (o_j, a_j, r_j, o'_j) from B_i
      Compute prediction $\hat{v}_j = Q(o_j, a_j; \theta_i)$
      Compute target:

$$v_j = \begin{cases} r & \text{if } t = T \\ r + \gamma \max_{a'} \hat{Q}(o'_j, a'; \hat{\theta}_i) & \text{otherwise} \end{cases}$$

      Update θ_i by minimizing the loss with gradient descent: $L(\theta_i) = \frac{1}{b} \sum_j (v_j - \hat{v}_j)^2$
      Update target weights: $\hat{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \hat{\theta}_i$
    end for
    $o_1 \leftarrow o'_1$,...,$o_n \leftarrow o'_n$
  end for
end for

D. RL Algorithm

Among the most classic mono-agent RL algorithms is Q-Learning [20] which learns an optimal action-value function, noted Q^*, while interacting with the environment. Q^* gives, for each observation-action pair, the maximum expected return (*i.e.*, cumulative sum of rewards) achievable by following an optimal policy thereafter. Once Q^* is learned, the optimal policy π^* can be obtained by choosing, for each observation,

[3]We use a negative reward function to favor initial exploration of UAVs.

[4]The agent does not move if the chosen action leads to an obstacle.

the action that maximizes Q^*, and is defined as $\pi^*(o) = \texttt{argmax}_{a \in A} Q^*(o, a)$. As Q-learning is not designed to cope with scenarios with large or continuous state spaces, Deep Q-Networks (DQN) [21] is now the preferred approach. It extends Q-learning by introducing a neural network to approximate the action-value function.

Following our IL approach (cf §IV-B), we use the Independent DQN algorithm detailed in Algorithm 1. Each agent i applies DQN independently and trains its own action-value function $Q_i(o_i, a_i; \theta_i)$ conditioned on its individual observation o_i and action a_i and parameterized with neural network weights θ_i. Each agent i collects experience samples (o_i, a_i, r, o_i') via interaction with its environment (line 8). These samples are stored in a replay buffer (line 9) and used to update the weights θ_i (and so to train Q_i) (line 14). Target networks $\hat{Q}_i$ with weights $\hat{\theta}_i$ are also used to address the challenges of moving target values. The agents follow the ϵ-greedy exploration-exploitation strategy (line 7) which chooses the greedy action[5] with probability $1 - \epsilon$, and with probability ϵ chooses a random action.

V. Scenarios and Simulation Results

We use our simulator, described in Section III to train and test the navigation algorithm described in Section IV-D. Parameters specific to the simulation and the learning part are given in Table III.

A. Training, evaluation and metrics

The RL algorithm is trained on E_{train} episodes. We compare training with Reward 1 and with Reward 2 (cf. §IV-C).

To evaluate the performance of the learned policy after the training, we run the greedy policy, at the end of each training, during 10 episodes of evaluation and plot different performance metrics:

- the mean number of users connected at each time step of the episode given by $m_1 = \sum_{i \in V} \sum_{u \in U} \mathbb{1}_{c_i(u)=1}$
- the mean throughput per user at each time step of the episode given by $m_2 = \frac{T_h}{|U|}$ with T_h defined in Eq. 8.

UAV position traces within the environment during one episode of evaluation are also recorded.

B. Mono UAV scenario

We are first considering a 3D scenario with a single UAV, one cluster of 10 users, and one obstacle in the topology which corresponds to a wall (cf. Fig.4). The cluster of users is at the same position during the training and evaluation phase.

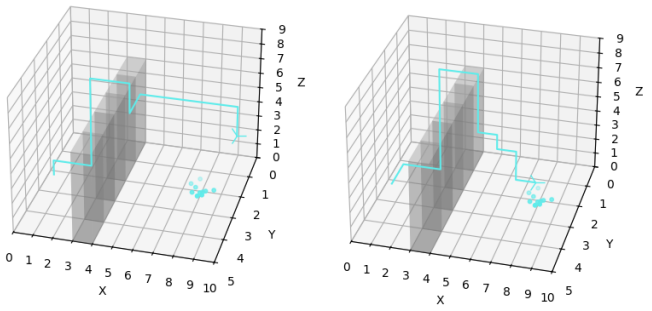
Trajectory of the UAV: In Fig. 4, it can be observed that the UAV has successfully learned to avoid the obstacle to approach and serve correctly the cluster of users. However, with Reward 1, maximizing the number of connected users, the UAV merely reaches the point where a connection is established with the users, without considering the quality of the communication. In contrast, with Reward 2, the UAV moves closer to the users to provide a better communication quality.

[5]A greedy action when observing o_i is the one that yields the highest Q value.

TABLE III: Simulation and learning parameters

Simulation Parameters	**Values**
cell size	$ds = 10m$
path loss at the reference distance	$PL_0 = 46.43$
variations of the path loss	$X_g = \mathcal{N}(0, 2^2)$
pathloss exponent in free space	$\mu = 2$
pathloss exponent for obstacle	$\mu_{obst} = 4$
constant	$K = 0$

RL Parameters	**Values**
discount factor	$\gamma = 0.95$
learning rate	$\alpha = 0.0001$
decaying ϵ strategy	exponential $(1 \rightarrow 0.1)$
episode duration	$T = 100$
training episodes	$E_{train} = 5000$ for Mono UAV $E_{train} = 10000$ for Multi UAVs with static clusters $E_{train} = 50000$ for Multi UAVs with dynamic clusters
replay buffer size	$B = 10\ 000$
minibatch size	$b = 32$
target soft update rate	$\tau = 0.001$
hidden layers for DQN	64×64
activation function for DQN	ReLu
optimiser	Adam



(a) Reward 1 (b) Reward 2

Fig. 4: Mono UAV scenario: Trajectory of the UAV after training according to each reward.

Result of the performance metrics: In Fig. 5, the evolution of the number of users connected (left) and the average user throughput (right) is given for the two rewards, averaged over 10 episodes. The shaded area around the mean value represents the standard deviation from the mean to illustrate the variability. It can be observed that, in both cases, the number of connected users at the end of the episodes is identical: 10 out of 10, meaning that every user is connected to the UAV regardless of the chosen reward. However, the throughput in the access network differs significantly. The approach with Reward 1 is suboptimal in terms of throughput, as the policy learned by the UAV results in a mean user throughput that is approximately half of what could be achieved. In contrast, when the reward function accounts for communication quality, as in Reward 2, the resulting user throughput is substantially higher. This difference owes to the simulator ability to model the performance of the communication and to Reward 2 taking into account the quality of the communication performance.

C. Two independent UAV scenario

In this scenario, we consider two UAVs that can be deployed to serve two clusters of 10 users each, located at the opposite

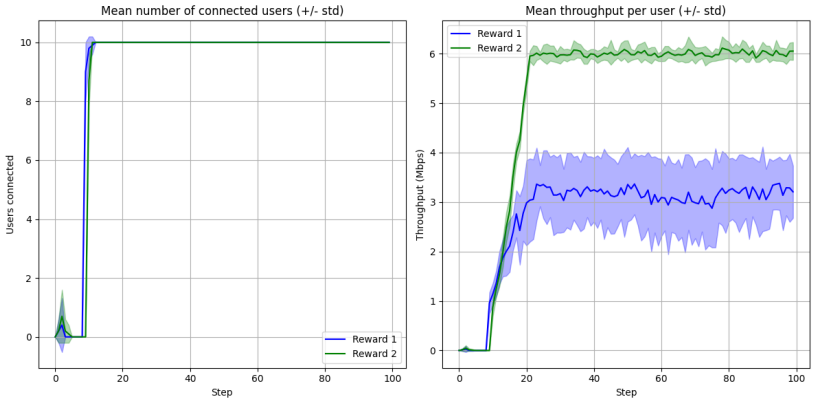


Fig. 5: Mono UAV scenario: Comparison of our performance metrics after training, averaged over 10 episodes, according to each reward.

corners of the area (bottom-left and top-right). To reduce computational time, the scenario is modeled using a 2D topology of size 20×20 cells.

Results with static clusters: In this subsection, we assume that the clusters of users are at the same position during the learning and evaluation phase. Fig. 6 plots the metrics for Rewards 1 and 2.

For both rewards, it can be seen that the 20 users are connected to a UAV, indicating that the UAVs collaborate effectively to distribute the users among themselves. This is due to the rewards' definitions that are collaborative by nature (and which implicitly rely on perfect communications between UAVs, without packet loss and delay). However, when analyzing the communication quality more closely, it becomes clear that Reward 2 is more efficient, with higher mean throughput per UAV compared to Reward 1. As in the mono UAV scenario, with Reward 1, the UAV stops as soon as it is connected to the users and does not seek to improve the communication quality, unlike Reward 2.

Results with dynamic inter-episode clusters: In this scenario, we focus on the case where the user clusters are static during the duration of an episode, but at the start of a new episode (during training and evaluation), they are randomly located within a given area of the environment. We only consider Reward 2 as it gives better throughput than Reward 1. We can see in Fig. 7 that the UAVs adapt their trajectory at each episode according to their observations and given the different positions of the user clusters. This randomness introduces variability in the results, requiring more extensive learning from the UAVs. However, the algorithm adapts well to this uncertainty, as it achieves an average user throughput of over 5 Mbps.

VI. Conclusion

In this paper, we propose a simulator to train and test RL-based UAV navigation solutions for the network coverage problem. Our simulator is based on the simulator of [11]: this latter has been completely refactored and extended with a realistic access communication model. The proposed simulator has been tested with an Independent DQN algorithm to deploy UAVs in order to connect users to an existing network infrastructure. We show that our simulator can, thanks to its wireless communication model, train RL solutions with rewards leading to a better communication quality between UAVs and users than the one offered by the simulator of [11].

For future work, we plan to test more complex scenarios in terms of numbers of UAVs and users and their mobility, as in terms of numbers of obstacles. We will enrich the communication model with multi-user transmissions as the radio medium sharing between UAVs. We will also include the UAV energy consumption in the simulator to test UAV navigation solutions that also incorporate this parameter. We also plan to explore other multi-agent RL approaches focused on emergent communication among agents, in order to enable optimized communication between UAVs and consider dynamic network topology.

Acknowledgments

This work was supported by the *Federation Informatique de Lyon (FIL)*.

References

[1] Walid Saad, Mehdi Bennis, Mohammad Mozaffari, and Xingqin Lin. *Wireless Communications and Networking for Unmanned Aerial Vehicles*. Cambridge University Press, 2020.

[2] Jiangbin Lyu, Yong Zeng, Rui Zhang, and Teng Joon Lim. Placement optimization of uav-mounted mobile base stations. *IEEE Communications Letters*, 21(3):604–607, 2016.

[3] Ahmad Taher Azar, Anis Koubâa, Nada Ali Mohamed, Habiba A. Ibrahim, Zahra Fathy Ibrahim, Muhammad Kazim, Adel Ammar, Bilel Benjdira, Alaa M. Khamis, Ibrahim A. Hameed, and Gabriella Casalino. Drone deep reinforcement learning: A review. *Electronics*, 10:999, 2021.

[4] Yu Bai, Hui Zhao, Xin Zhang, Zheng Chang, Riku Jäntti, and Kun Yang. Toward autonomous multi-uav wireless network: A survey of reinforcement learning-based approaches. *IEEE Communications Surveys & Tutorials*, 25(4):3038–3067, 2023.

[5] Mihir Kulkarni, Theodor J. L. Forgaard, and Kostas Alexis. Aerial gym – isaac gym simulator for aerial robots, 2023.

[6] Botian Xu, Feng Gao, Chao Yu, Ruize Zhang, Yi Wu, and Yu Wang. Omnidrones: An efficient and flexible platform for reinforcement learning in drone control. *IEEE Robotics and Automation Letters*, 2024.

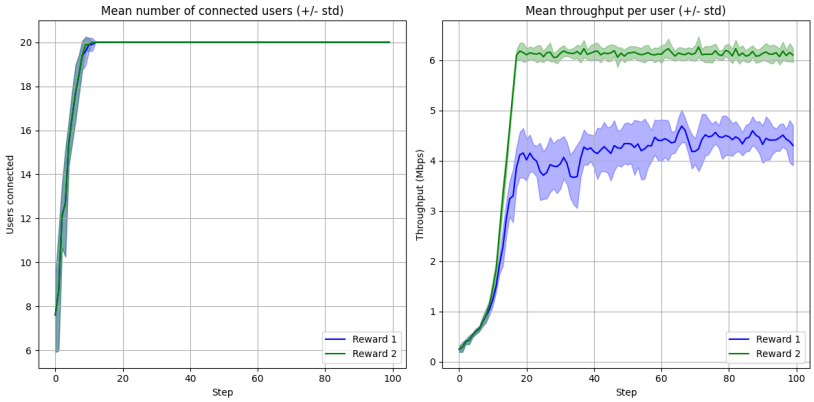


Fig. 6: Multi UAV scenario and static clusters: Comparison of our performance metrics after training, averaged over 10 episodes, according to each reward.

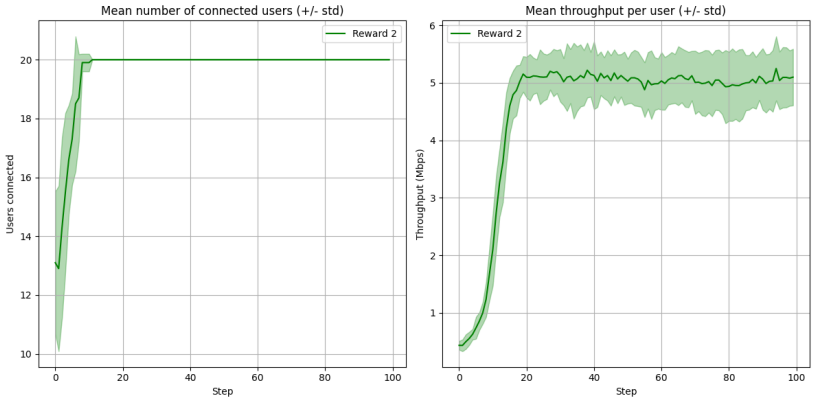


Fig. 7: Multi UAV scenario and dynamic inter-episode clusters: Results of the performance metrics after training, averaged over 10 episodes, with Reward 2.

[7] Piotr Gawlowicz and Anatolij Zubow. ns-3 meets OpenAI Gym: The playground for machine learning in networking research. In *Int. Conf. on Modeling, Analysis and Simu. of Wireless and Mobile Systems*, pages 113–120, 2019.

[8] Anatolij Zubow, Sascha Roesler, Piotr Gawlowicz, and Falko Dressler. GrGym: A Playground for Research on RL/AI Enhanced Wireless Networks. In *European Wireless*, 2022.

[9] Daniel Rosen, Illa Rochez, Caleb McIrvin, Joshua Lee, Kevin D'Alessandro, Max Wiecek, Nhan Hoang, Ramzy Saffarini, Sam Philips, Vanessa Jones, Will Ivey, Zavier Harris-Smart, Zavion Harris-Smart, Zayden Chin, Amos Johnson, Alyse M. Jones, and William C. Headley. RFRL Gym: A Reinforcement Learning Testbed for Cognitive Radio Applications. In *ICMLA*, 2023.

[10] Babatunji Omoniwa, Boris Galkin, and Ivana Dusparic. Communication-enabled deep reinforcement learning to optimise energy-efficiency in uav-assisted networks. *Vehicular Communications*, 43, 2023.

[11] Damiano Brunori, Stefania Colonnese, Francesca Cuomo, and Luca Iocchi. A reinforcement learning environment for multi-service uav-enabled wireless systems. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, March 2021.

[12] J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

[13] Samira Hayat, Evşen Yanmaz, and Raheeb Muzaffar. Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint. *IEEE Communications Surveys & Tutorials*, 18(4):2624–2661, 2016.

[14] Alexandre Bonnefond, Olivier Simonin, and Isabelle Guérin-Lassous. Extension of flocking models to environments with obstacles and degraded communications. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9139–9145. IEEE, 2021.

[15] Theodore S Rappaport. *Wireless communications: principles and practice*. Cambridge University Press, 2024.

[16] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, volume 2, pages 836–843 vol.2, 2003.

[17] Ming Tan. *Multi-agent reinforcement learning: independent vs. cooperative agents*, page 487–494. 1997.

[18] Christian Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 11 2020.

[19] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Proceedings of NeurIPS*, 2022.

[20] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.