DEEP REINFORCEMENT LEARNING BASED ENERGY-EFFICIENT MULTI-UAV DATA COLLECTION FOR IOT NETWORKS

SEYED SAEED KHODAPARAST

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

September 2021

Abstract

Unmanned aerial vehicles (UAVs) are regarded as an emerging technology, which can be effectively utilized to perform the data collection tasks in the Internet of Things (IoT) networks. However, both the UAVs and the sensors in these networks are energy-limited devices, which necessitates an energy-efficient data collection procedure to ensure the network lifetime. In this thesis, we propose a multi-UAV-assisted network, where the UAVs fly to the ground sensors and control the sensor's transmit power during the data collection time. Our goal is to minimize the total energy consumption of the UAVs and the sensors, which is needed to accomplish the data collection mission. We formulate this problem into three sub-problems of single UAV navigation, sensor power control, and multi-UAV scheduling, and model each part as a finite-horizon Markov Decision Process (MDP). We deploy deep reinforcement learning (DRL)-based frameworks to solve each part. Specifically, we use the deep deterministic policy gradient (DDPG) method to generate the best trajectory for the UAVs in an obstacle-constrained environment, given its starting position and the target sensor. We also deploy DDPG to control the sensor's transmit power during data collection. To schedule activity plans for each UAV to visit the sensors, we propose a multi-agent deep

Q-learning (DQL) approach by taking the energy consumption of the UAVs on each path into account. Our simulations show that the UAVs can find a safe and optimal path for each of their trips. Continuous power control of the sensors achieves better performance than the fixed power and fixed rate approaches in terms of the sensor's energy consumption and the data collection completion time. In addition, compared to the two commonly used baselines, our scheduling framework achieves better and near-optimal results in the simulated scenario.

Acknowledgements

I would like to thank my supervisors, Prof. Ping Wang and Prof. Uyen Trang Nguyen, for their guidance and support throughout my research. I am grateful for their patience and encouragement over the past two years. I would like to thank my examiners, Prof. Hina Tabassum and Prof. Jinjun Shan for diligently engaging with my thesis and providing valuable suggestions. I would like to acknowledge Dr. Xiao Lu, who helped me progress in my research and publish my first paper. Finally, thanks to my family for all their long-distance support during these hard times of the pandemic.

Table of Contents

List of Tables

List of Figures

Abbreviations

UAV Unmanned Aerial Vehicle

IoT Internet of Things

MDP Markov Decision Process

RL Reinforcement Learning

DRL Deep Reinforcement Learning

DDPG Deep Deterministic Policy Gradient

DQL Deep Q-learning

WSN Wireless Sensor Network

LoS Line-of-Sight

MSTP Multiple Traveling Salesman Problem

CH Cluster Head

CVRPTW Capacitated Vehicle Routing Problem with Time Windows

DDQN Double Deep Q-network

AoI Age of Information

UV Unmanned Vehicle

DNN Deep Neural Network

G2A Ground-to-Air

Chapter 1

Introduction

This chapter introduces the Unmanned Aerial Vehicles (UAVs) as mobile data collectors that can be utilized in the Internet of Things (IoT) networks. We discuss the energy constraint issue in the data collection problem and clarify our motivations for this thesis work. In Section 1.2, we present the contributions of this work. In Section 1.3, we describe the organization of this thesis. Finally, in Section 1.4, we present our related publication.

1.1 Data Collection with UAVs

Over the past few years, unmanned aerial vehicles (UAVs), commonly known as drones, have been increasingly used in a broad range of applications, including military services, surveillance and monitoring, telecommunications, and good's delivery [1], [2]. Their inherent desired features such as low cost, flexible maneuvering, and ease of deployment [3], make the UAVs a perfect replacement for human operators in scenarios where it might be costly or

hazardous.

With the recent advancements in the field of Internet-of-Things (IoT), wireless sensor networks (WSNs) have been widely deployed as a surveillance technology to monitor the surrounding environment, e.g., temperature and air pollution [4], [5]. These IoT sensors are usually energy-constrained devices with small transmit power and may not be able to transmit their data over a long distance. In that respect, UAVs, as mobile data collectors equipped with communication interfaces, can be employed to fly close to the dispersed sensors and collect their data [6]. Compared to ground data collection schemes, UAVs are more efficient as they can easily adjust their path to reach a sensor by avoiding terrestrial obstacles [7]. In addition, due to their high altitude, the UAVs have a higher chance of exploiting the Line-of-Sight (LoS) link to the ground sensors [8], resulting in superior link quality.

However, despite all the advantages that UAVs have brought into communication networks, they have limited energy storage, posing a crucial challenge. After dispatching from the origin, they need to navigate towards the sensors while avoiding collisions with any obstacle. Also, using a single UAV to perform data collection from a large set of sensors distributed in a wide area may take a long time, which results in task failure due to time constraints of the collected data. Collaborative multi-UAV approaches can be utilized to increase the chance of accomplishing the mission while reducing its completion time [9].

On the other hand, when the UAV arrives at the sensor's location and hovers above it to start the data collection, the communication link may experience different channel gains depending on the multi-path propagation environment. Due to the sensor's limited energy

storage, the sensor's transmit power needs to be controlled optimally to prolong its lifetime and provide reliable communication.

The complexity of the problem mentioned above makes it hard to be solved using traditional optimization techniques, which are usually time-consuming and need a complete model of the environment. Deep reinforcement learning (DRL) allows us to find the solution by letting the agents, in our case, the UAVs, interact with the environment without taking any unrealistic assumptions or prior knowledge about the system model. In that way, the UAVs can make their own decisions after some trial and error during the training phase. DRL differs from conventional supervised learning methods in which the agent is given a labeled dataset to learn to perform actions; because there is no such pre-existing labeled data in DRL. The data we use for training a DRL agent is generated during its interaction with the environment. The agent explores its environment through multiple interactions, and after evaluating its options, searches for the most rewarding actions. DRL also differs from unsupervised learning methods since the objective is to maximize the received rewards in the long run, but there is no such reward in unsupervised learning.

Motivated by the above observations, we formulate the energy-efficient data collection problem as a finite-horizon Markov decision process (MDP). To overcome the computational complexity caused by the state space's high dimensionality, we propose a multi-UAV deep reinforcement learning (DRL)-based approach to minimize the energy consumption on both the UAVs and sensors sides. In particular, we develop and combine three different DRL frameworks for single UAV navigation, sensor power control, and multi-UAV scheduling,

respectively. We consider a delay-tolerant scenario where each sensor caches the sensed data and transmits it to a UAV upon the request [10].

1.2 Contributions

The main contributions of this work are summarized as follows:

1. We first propose an obstacle-aware navigation framework based on deep deterministic policy gradient (DDPG) [11] method, which enables the UAV to adjust its trajectory and speed in a continuous manner from any given starting point towards its destination sensor with minimum energy consumption. Our navigation framework does not need to be retrained for new environment instances, as it learns the relationship between the UAV and its surroundings, including the obstacles and the target sensor.

2. We propose a DDPG-based approach to control each sensor's transmit power while sending the data to the UAV. Unlike most of the related works, the transmit power of the sensor is not supposed to have discrete levels and can adaptively be changed depending on the multi-path propagation environment.

3. We propose a multi-UAV scheduling framework by incorporating the data provided by the navigation framework, with the aim to minimize the UAVs' overall energy consumption to accomplish the data collection. A multi-agent deep Q-learning (DQL) [12] algorithm is developed to generate the activity plan of each UAV by providing a list of sensors that needs to be visited in order, considering the UAVs' energy consumption.

4. We conduct simulations to evaluate the performance of our proposed frameworks. Our simulation results show that the UAVs learn to navigate safely in the environment with a low collision rate. In addition, the power control model can successfully control the trade-off between the UAV's hovering and communication power and the sensor's transmit power, and the multi-UAV scheduling framework can achieve better performance comparing to the random and greedy baselines.

1.3 Organization of the Thesis

The thesis is organized as follows: In Chapter 2, we review the related works regarding the UAV data collection problem. Chapter 3 presents the background knowledge about the DRL algorithms used in this thesis. Chapter 4 describes the system model of the considered scenario and the proposed DRL frameworks utilized to solve the data collection problem, along with our experimental setup and simulation results. Finally, the conclusion of the thesis and future works are presented in Chapter 5.

1.4 Related Publication

Khodaparast, S. S., Lu, X., Wang, P., and Nguyen, U. T. Deep reinforcement learning based energy efficient multi-uav data collection for IoT networks. *IEEE Open Journal of Vehicular Technology 2* (2021), 249–260.

Chapter 2

Literature Review

The UAV data collection has attracted increasing research attention in recent years. Since the nature of the problem is complex as it involves both trajectory planning and data collection, research works have focused on different aspects such as flight time, energy constraints, and age of information.

Several existing approaches have exploited various optimization techniques. References [13], [14] focus on minimizing the task completion time in multi-UAV systems. The authors in [13] simplify the original non-convex optimization problem with the infinite number of variables into two sub-problems. They first propose a UAV-sensor association mechanism by leveraging the Multiple Traveling Salesman Problem (MTSP) and solve it by applying the genetic algorithm. Their objective function considers the longest flight distance of UAVs to make it as short as possible, which improves the overall fairness of associating the sensors to the UAVs. After turning the original problem into multiple single-UAV data collection problems, they

minimize the collection time of each UAV by separately optimizing sensor transmit power, UAV's speed, and the collection position to achieve the goal of minimizing the completion time. In [14], the authors adopt a two-step approach that first applies K-means clustering to optimize the number and locations of cluster heads (CHs) which are responsible for collecting data from their associated sensors. Then, finding the best trajectories by assigning CHs to UAVs is modeled by MTSP and solved by a heuristic genetic algorithm which achieves a sub-optimal solution with low complexity. However, both [13] and [14] ignore the energy constraints of the UAVs and the impact of small-scale fading.

In [6], the authors investigate the multi-UAV energy consumption minimization problem in a sensor network where the aggregated data of different cluster heads have time deadlines. To solve the problem, the authors propose a bi-level solution. In the first step, similar as [14], the number and locations of the cluster heads are optimized based on a customized K-means clustering. Subsequently, the problem is changed to finding the best routes for the UAVs, minimizing their total energy consumption, and respecting the time deadlines. This problem is NP-hard and can be assimilated to a capacitated vehicle routing problem with time windows (CVRPTW) [15]. Taking into account the energy consumption of the sensors, reference [16] aims to minimize the overall energy consumption of a single UAV and a set of IoT sensors. To this end, the authors jointly optimize the UAV's stop positions, the subset of sensors that transfer data at each stop, and the UAV's trajectory to navigate between the stops, under the constraints of the individual energy availability of the sensors.

The works above consider the path loss model for the communication channel. To present

a more accurate and realistic model of the communication link between the UAV and the sensors, references [17], [18] adopt a Rician fading channel model. The authors in [17] optimize three-dimensional trajectory jointly with communication scheduling to maximize the minimum transmission rate of the sensors. The three-dimensional (3D) trajectory modeling allows the system to incorporate more practical elevation angle-dependent Rician fading channels. In [18], the authors optimize the UAV trajectory and allocation of resources to maximize the total number of served IoT devices where the collected data has deadlines. However, the proposed approaches in [17] and [18] only apply for single-UAV systems.

The above-reviewed optimization techniques require the knowledge of complete system information in advance and can only perform data collection in a pre-scheduled manner. Hence, these approaches are not applicable in dynamic environments where instant system information may not be available. To handle data collection in a time-varying environment, research efforts have resorted to reinforcement learning (RL) and deep reinforcement learning (DRL) approaches, which enable the UAVs to acquire solutions without knowing the complete system information. References [10], [19]–[23] investigate the data collection problem for single-UAV systems. The authors in [10] devise a Q-learning-based mechanism in a grid environment, which achieves the optimal energy efficiency of a single UAV flying between the centers of the grid to collect data from a particular sensor. The UAV's action is defined as its next hovering location and the associated sensor to collect the data from. Reference [22] develops a double deep Q-network (DDQN)-based trajectory planning mechanism to maximize the throughput over the whole data collection mission subject to the maximum

flight time and navigation constraints of not entering no-fly zones, obstacle avoidance, and safe-landing in designated landing areas. However, the approaches introduced in [10] and [22] only work for discrete flying actions.

The focus of references [19]–[21], [23] is to minimize the age of information (AoI) of the data collected from the distributed sensors by utilizing DQN-based algorithms. Reference [23] proposes an AoI-based trajectory planning algorithm for fresh data collection. The traffic generation pattern of the IoT devices is unknown and follows an exponential distribution. However, the communication channel is not accurately modeled, and the authors assume the UAV can successfully collect IoT data as long as the device is in its coverage range. The authors in [20] introduce a DQN-based approach to optimize the UAV's trajectory and the communication scheduling of the sensors with the objective of minimizing the weighted sum-AoI. The limited energy storage of the sensors and the time constraint for the UAV's operation are considered in this work. Reference [19] devises a similar algorithm that achieves the same goals of [20] by optimizing the UAV's trajectory and the scheduling of the sensors under the additional energy constraint of the UAV. The authors in [21] jointly minimize the average AoI of the sensors, their packet drop rate, and the UAV's energy consumption. The UAV is used as a mobile relay to collect the sampled data at the sensor and send it to the base station. However, the main problem with DQN-based trajectory planning approaches is that they can only be applied to problems with a discretized set of actions. In these cases, similar to most optimization approaches, the environment is usually divided into equally-sized grids. The UAV can only move to one of its adjacent cells, which is not a realistic assumption.

In addition to the above-reviewed designs for single-UAV systems, research efforts have also been devoted to addressing data collection in multi-UAV systems with DRL approaches. Reference [24] considers a scenario with multiple Unmanned Vehicles (UVs) and charging stations, where the UVs can either collect data or charge at the distributed charging stations. The authors devise a sequential DRL model called "PPO + LSTM" for simultaneous task assignment and trajectory planning to maximize data collection ratio and geographic fairness with minimized total energy consumption of the UVs. The system uses a single agent that generates actions for all UVs. In [25], the authors propose a multi-agent DQL algorithm to maximize the minimum throughput by the joint optimization of path design and channel resource assignment in a UAV-enabled wireless powered communication network. Each UAV owns an independent DQN for its actions, while the other UAVs are considered part of the environment. References [26] aims to minimize the overall flight time of the UAVs under their individual energy constraints. The authors develop an option-based hybrid DRL method for a multi-UAV scenario, where each UAV can recharge its battery when its energy is not sufficient to complete the mission. However, the impact of obstacles on the navigation of the UAVs is ignored.

In a more related publication, reference [27] addresses obstacle-aware navigation based on a joint learning approach. The proposed method obtains the shortest and safest trajectory in an obstacle-constrained scenario based on DDPG. It then determines the best schedule for visiting the sensor nodes based on Q-learning. The proposed approach is applicable for a single UAV and does not consider the power consumption of the sensors. Additionally, to

our best knowledge, none of the existing DRL-based data collection designs for multi-UAV systems takes into account the impact of small-scale fading. Being aware of the limitations, we aim to address the continuous trajectory planning for multi-UAV data collection in a practical obstacle-constrained environment with small-scale fading and power consumption of both UAVs and sensors taken into consideration.

Chapter 3

DRL Preliminaries

In this chapter, we present the background of reinforcement learning frameworks used in the proposed solution. We introduce Markov Decision Processes (MDP) as the framework to model our problem. Then, we explain the two DRL algorithms used to solve this problem.

We consider a finite-horizon MDP defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P})$ with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R} := \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and the state transition probability $\mathcal{P} := \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The agent interacts with the environment by performing actions in discrete time steps and receiving a reward afterwards. In particular, at each time step t, the agent observes state s_t, takes an action a_t, receives a reward r_t based on s_t and a_t, and then goes to the next state s_{t+1} with the probability of $\mathcal{P}(s_{t+1}|s_t, a_t)$.

The objective of RL is to find a policy that maximizes the future rewards without knowing the state transition probability by only using the experience gained in the interaction with the environment, which is denoted by the state transition tuple (s_t, a_t, r_t, s_{t+1}). A policy $\pi(s)$

is defined as a mapping from state s to an action or a distribution over actions, depending on whether it is deterministic or stochastic. In fact, the policy controls the agent's actions and therefore, the rewards it receives from the environment. We define the return R_t as the discounted sum of future rewards from the current state s_t up to a terminal state at time T by

$$R_t = \sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau, \tag{3.1}$$

where γ is the discount factor ranging from 0 to 1. The discount factor balances the importance given to the immediate and future rewards. Higher values of γ indicate that the agent cares more about the future rewards, whereas the lower values show the opposite. Our goal is to find a policy that maximizes the expected return from the start time step, which is expressed as

$$\pi^* = \arg\max_{\pi} \mathbb{E}[R_0|\pi]. \tag{3.2}$$

We define the action-value function $Q(\cdot)$ to approximate the expected value of R_t when starting from state s_t, taking action a_t, and then following the policy π:

$$Q^\pi(s, a) = \mathbb{E}[R_t|s_t = s, a_t = a]. \tag{3.3}$$

The well-known algorithm in RL, Q-learning (QL) [28], is a tabular method that aims to find the optimal Q-values for each state-action pair (s, a) by iteratively improving the estimated Q-function with the following update

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]), \tag{3.4}$$

where α is the learning rate. To ensure the exploration of different state-action pairs during

training, the agent uses an ϵ-greedy policy that takes a random action with a probability of ϵ and the greedy action $\arg\max_a Q(s,a)$ with probability of $1-\epsilon$. After finding the optimal Q-values, the optimal policy can be achieved by taking the greedy action, as expressed in

$$\pi^*(s) = \arg\max_a Q^*(s,a). \tag{3.5}$$

However, QL is not efficient when the state and action spaces are large, as we need to store the Q-value for each state-action pair in a table. To overcome the curse of dimensionality, DRL techniques can be utilized by combining deep neural networks (DNNs) and RL. Now, we review the two DRL algorithms used in this thesis, i.e., DQL and DDPG.

In [12], Mnih *et al.* introduced DQL which makes use of two DNNs, namely the online network and the target network, to approximate the $Q(\cdot)$ function by minimizing the loss L

$$L(\theta^Q) = \mathbb{E}\Big[(Q(s_t, a_t|\theta^Q) - y_t)^2\Big], \tag{3.6}$$

where θ^Q denotes the weights of the online DNN, and y_t known as the target is expressed as

$$y_t = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\theta^{Q'}). \tag{3.7}$$

where $\theta^{Q'}$ shows the weights of the target network, which has the same structure as the online DNN. However, its weights are duplicated from the online network periodically and are held fixed in between to ensure the stability of the target values. Moreover, training samples in RL are correlated, which does not meet the independent and identical distribution need for training the DNNs. Experience replay mechanism was introduced to resolve this issue. In particular, we use a replay buffer to store the state transition tuples (s_t, a_t, r_t, s_{t+1}).

To update the DNN, we use a mini-batch randomly sampled from the replay buffer. Due to the randomness of the samples, experience replay can reduce the correlation between sequentially generated samples and improve the convergence behavior of the DNN. In addition, learning from the samples multiple times improves the data efficiency. Fig 3.1 shows a trained Q-network, where the neural network outputs the Q-values for each action.

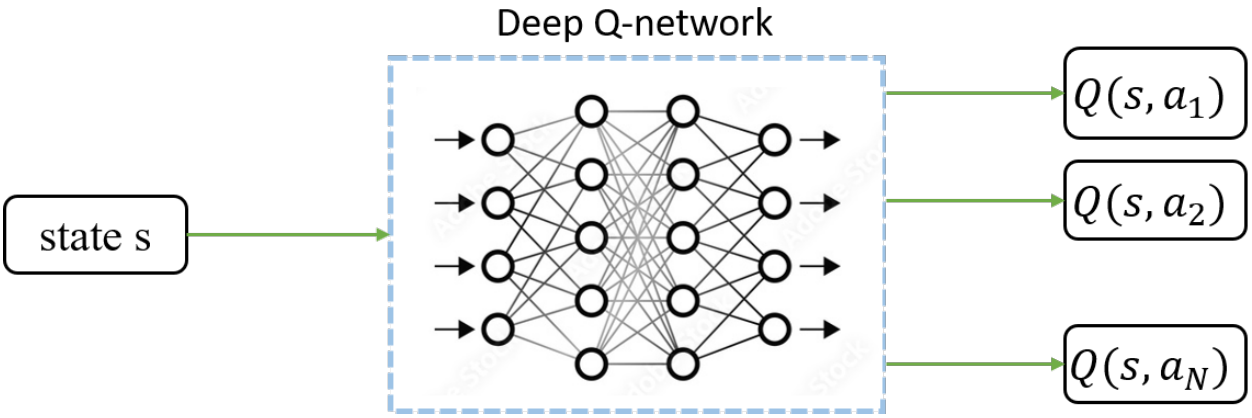

Figure 3.1: Deep Q-learning

The problem with DQL is that it only works for control problems with a discrete and low-dimensional action space. It is hard to apply DQL to continuous control because it needs to figure out the action that maximizes the $Q(\cdot)$ function in (3.5), which is quite difficult. However, UAV trajectory planning and sensor power control are both continuous control tasks.

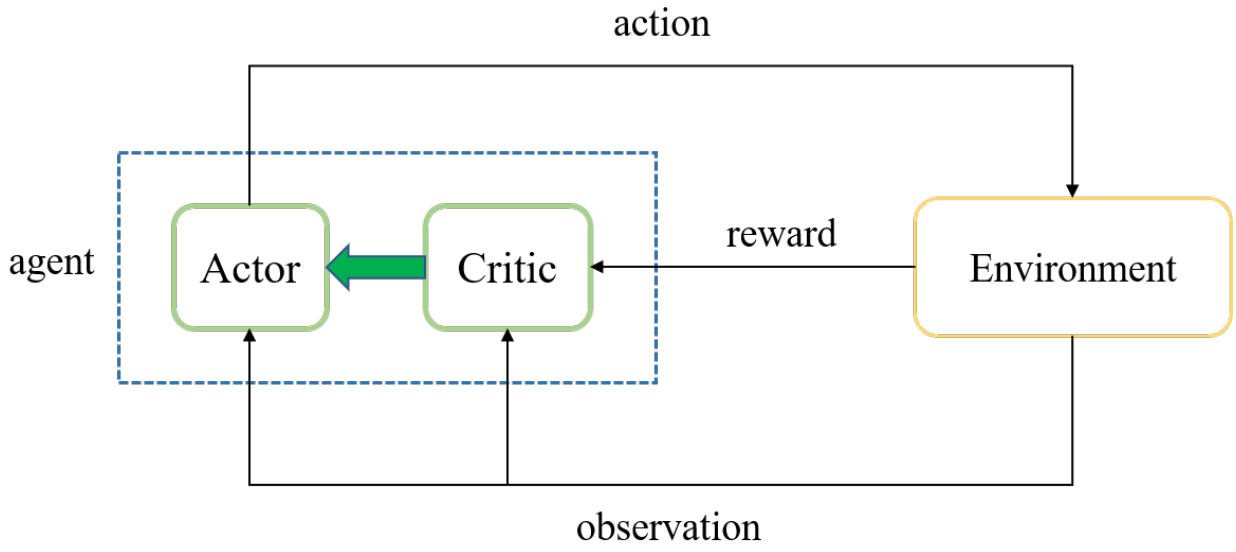

Figure 3.2: Actor-Critic architecture

DDPG [11] as an actor-critic method was designed for this purpose. The actor-critic architecture is shown in Fig. 3.2, where both actor and critic are implemented by DNNs. The critic is acted as $Q(s_t, a_t|\theta^Q)$ with parameters θ^Q and uses the same update as DQL for training. The critic's output is given to the actor network as feedback on the previous action. The actor uses a deterministic parameterized policy $\pi(s_t|\theta^\pi)$ with parameters θ^π to generate an action given state s_t. Our goal is to learn a policy that maximizes the expected return from the start state:

$$J = \mathbb{E}[R_0]. \tag{3.8}$$

The actor can be updated by applying gradient ascent to the objective in (3.8) with respect

to the actor parameters:

$$\nabla_{\theta^\pi} J = \mathbb{E}\Big[\nabla_a Q(s, a|\theta^Q)|_{a=\pi(s)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)\Big], \tag{3.9}$$

where ∇ is the operator that takes the gradients with respect to the parameters.

As in DQN, DDPG uses experience replay and target networks to address the same issues. However, the target networks in DDPG use "soft" updates rather than directly copying the weights periodically. The weights of these networks are updated by slowly tracking the weights of the online networks:

$$\begin{aligned} \theta^{Q'} &\leftarrow \eta\theta^Q + (1-\eta)\theta^{Q'} \\ \theta^{\pi'} &\leftarrow \eta\theta^\pi + (1-\eta)\theta^{\pi'} \end{aligned} \tag{3.10}$$

where $\eta << 1$ controls the update rate of the actor and the critic target networks.

Another major challenge of learning a deterministic policy is exploration, as it outputs a specific action given a state. To resolve that, we add noise sampled from a noise process $\mathcal{N}$ to our actor policy. Therefore, the action taken by the agent in the training phase can be expressed as follows:

$$a_t = \pi(s_t|\theta^\pi) + \mathcal{N}_t \tag{3.11}$$

Chapter 4

DRL-based Data Collection

In this chapter, we tackle the energy-efficient UAV data collection problem in the IoT networks. In the first section of this chapter, we introduce the system model. In the next section, we present our proposed methods for each sub-problem. We first formulate each sub-problem to an MDP problem, and then, propose the DRL algorithms to solve each MDP. The results of our proposed DRL methods are investigated in section 4.3.

4.1 System Model

In this thesis, we consider a cooperative multi-UAV data collection task with N sensors randomly located on the region and M UAVs to collect the data gathered by the sensors. Fig. 4.1 depicts such a configuration with two UAVs. The position of UAV m and ground sensor n can be characterized by the coordinate (x_m, y_m, z_m) and $(x_n, y_n, 0)$, respectively. All the UAVs start from the origin (e.g., charging station), fly towards the sensors, and

go back to the origin when the data collection is finished. We assume that when a UAV arrives at the sensor's location, it hovers above the sensor and activates it by sending a query signal. During each communication round, the UAV can acquire the channel gain by exchanging signals and, based on that, controls the transmit power of the sensor. After the sensor has sent all its data, the UAV flies to the next target until all the data is collected. We adopt a discrete-time system in which time is divided into slots of T_{nav} and T_{com} length for the navigation and communication tasks, respectively. In addition, we assume that the energy storage of the UAVs is sufficient to accomplish the data collection mission within the predefined time horizon, while our goal is to minimize their total energy consumption during the mission.

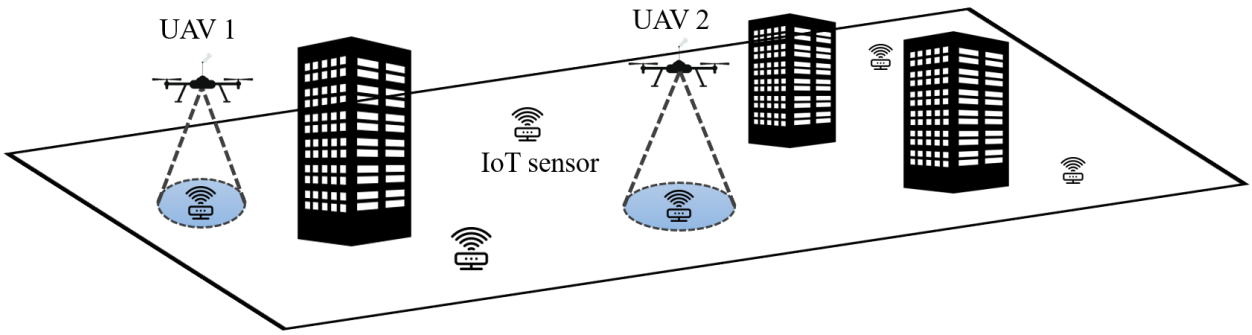

Figure 4.1: Illustration of our system model

4.1.1 Environment and Navigation Model

Since autonomous navigation in a real-world environment can be complex, we consider a virtual environment with dense obstacles to match the complexity of the urban areas. Unlike

most of the existing virtual environments investigated in the literature, which are usually considered as a grid-world, we focus on a free space environment with obstacles. The UAV has the freedom to choose any direction and speed to reach its target, whereas in the grid-world, the UAV's mobility is limited to a finite set of actions. Our goal is to train a UAV to fly from any arbitrary starting location to any arbitrary destination avoiding collision with the obstacles and flying out of the borders of the environment.

For simplicity, we assume that the UAV flies at a fixed altitude H. Therefore, the position of UAV m in the 3D space is characterized by (x_m, y_m, H), and its movement is restricted to $x-y$ plane. The control profile of the UAV consists of its speed and orientation angle, and the UAV's dynamic at time slot t can be characterized by

$$\begin{aligned} x_m^{t+1} &= x_m^t + v_m^t \times T_{nav} \times cos(\phi_m^t) \\ y_m^{t+1} &= y_m^t + v_m^t \times T_{nav} \times sin(\phi_m^t), \end{aligned} \tag{4.1}$$

where v_m^t and ϕ_m^t are the UAV's speed and orientation angle, respectively.

Similar to [29], we assume that the UAV knows its current location and the destination by using GPS signals. To illustrate the UAV's perception of the surrounding environment, we assume that the UAV is equipped with range finders that can identify the distance from itself to obstacles in multiple directions. Fig. 4.2 shows a UAV equipped with five range finders, where the angle between each two adjacent range finders is $\pi/4$. We know that if an obstacle's height is less than the UAV's altitude, it can simply be avoided and has no effect on the trajectory of the UAV. Without loss of generality, we assume that all the obstacles in the environment have heights greater than the UAV's altitude. For the sake of simplicity, we

assume that the environment is surrounded with obstacles so that the UAV can determine its distance from the borders by using the same set of range finders.

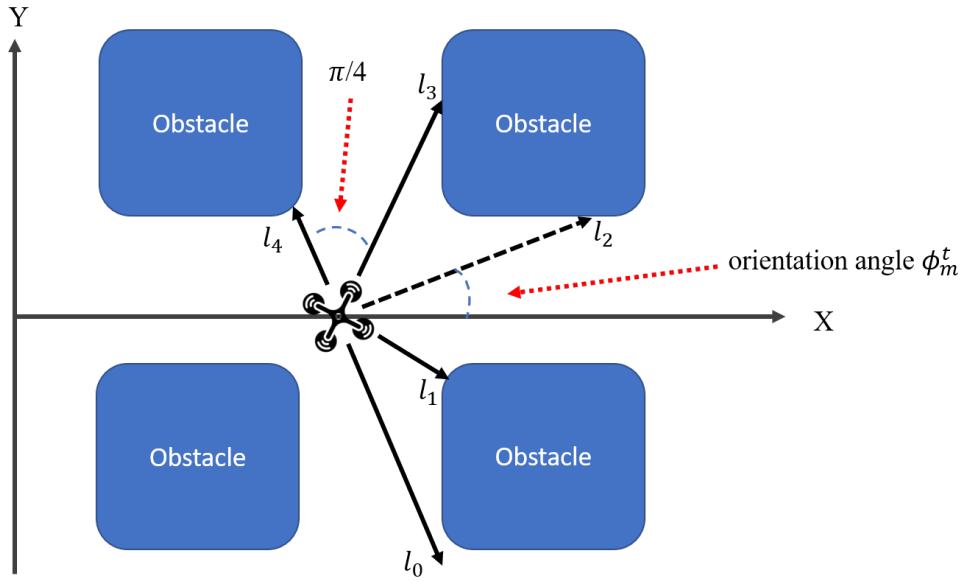

Figure 4.2: A UAV equipped with five range finders. $l_0 \sim l_4$ denotes distances returned by the five virtual range finders

4.1.2 UAV Energy Consumption Model

The energy consumption of the UAV includes two main components, namely the propulsion energy and the communication-related energy. The communication-related energy is used in communication circuits of the UAV to send, receive and process the signals, which is negligible comparing to the propulsion energy [30]. We assume the communication-related power is a constant denoted by P_c. The propulsion energy is needed to keep the UAV flying and hovering, which can be expressed as a function of its speed [31], i.e.,

$$P_{nav}(v) = P_0\left(1+\frac{3v^2}{U_{tip}^2}\right) + P_i\left(\sqrt{1+\frac{v^4}{4v_0^2}}-\frac{v^2}{2v_0^2}\right)^{\frac{1}{2}} + \frac{1}{2}d_0\rho\kappa A v^3, \tag{4.2}$$

where P_0 and P_i are constant parameters representing the blade profile power and induced power in hovering status, U_{tip} denotes the tip speed of the rotor blade, v_0 is the mean rotor-induced velocity during hovering. Moreover, the parameters d_0, κ, ρ and A represent the fuselage drag ratio, rotor solidity, air density and rotor disc area, respectively. To obtain the power consumption during hovering, we let $v = 0$ in (4.2), which gives us the hovering power

$$P_h = P_0 + P_i. \tag{4.3}$$

We should note that as the UAV hovers above a sensor, it is using the communication-related power to receive the data. Therefore, the total power consumption in the data collection status is expressed as

$$P_{dc} = P_h + P_c. \tag{4.4}$$

4.1.3 Channel Model and Data Rate

During data collection, the UAV hovers above the sensor at a high altitude, which increases the chance of establishing a line-of-sight (LoS) link. Hence, for the uplink ground-to-air (G2A) channel, we adopt the Rician fading channel consisting of a deterministic LoS link and a random multipath fading component, where the Rician factor is affected by the surrounding environment. We assume that the channel between UAV m and sensor n remains unchanged

within each time slot and at time slot t can be modeled as [17]

$$h_t[m,n] = \sqrt{\beta_t[m,n]}g_t[m,n], \tag{4.5}$$

where $\beta_t[m,n]$ is the large-scale average channel power gain accounting for signal attenuation, which includes the pathloss and shadowing, and $g_t[m,n]$ is the small-scale fading coefficient. Let $d_t[m,n]$ denote the distance between UAV m and sensor n, which is given by

$$d_t[m,n] = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2 + H^2}. \tag{4.6}$$

We can express the average channel power gain $\beta_t[m,n]$ as

$$\beta_t[m,n] = \beta_0 d_t^{-\alpha}[m,n], \tag{4.7}$$

where β_0 is the average channel power gain at a reference distance of $d_0 = 1$ m, and α is the pathloss exponent which usually ranges from 2 to 6. Due to the existence of the LoS path, the small-scale coefficient follows the Rician distribution [17] with unit power (i.e., $\mathbb{E}[|g[m,n]|^2] = 1$) and probability density function:

$$p_{g[m,n]}(g) = 2g(\chi+1)\exp\Big(-\chi-(\chi+1)g^2\Big)I_0\Big(2\sqrt{\chi(\chi+1)}g\Big), \tag{4.8}$$

where χ denotes the Rician factor of the channel, and $I_0(\cdot)$ is the 0th order modified Bessel function of the first kind. The Rician factor is the ratio between the power in the direct path and the power in the other scattered paths, which is affected by the environment and the UAV-ground elevation angle. Since the UAVs hover above the sensors to receive data, the elevation angle is approximately equal to $\pi/2$, and we assume that the Rician factor is fixed for a particular environment.

In the considered scenario, the UAV sends an activation signal to wake up the sensor. When the connection has been established, the UAV gets to know the channel gain from the sensor to the UAV based on the exchanged pilot signals. We assume that the UAV and ground sensors use advanced communication technology and are equipped with sectored antennas. Thus, we can write the achievable data rate as

$$K_t[m,n] = B\log\left(1+\frac{P_t G_t |h_t[m,n]|^2}{\sigma^2}\right), \tag{4.9}$$

where B,P_t, G_t, σ^2 are the bandwidth of the channel, transmit power of the sensor, the antenna power gain of the G2A link, and the noise variance on the UAV side at time t, respectively. Since each UAV hovers above the sensor and the sensors are located far enough from each other, the impact of signal interference has not been considered in this work. In a more complex scenario where we have interference from the other sensors, which can change the data rate, the UAV must take into account the variable interference to properly allocate the transmit power to achieve the desired data rate.

We use ζ_t to denote the available data to be collected at the sensor at time step t, which can be expressed as

$$\zeta_t^n = \zeta_{t-1}^n - T_{com} K_{t-1}[m,n], \tag{4.10}$$

where T_{com} is the time length of each time step. In addition, the initial data amount to be collected at each sensor is denoted by Λ, i.e., $\zeta_0^n = \Lambda$.

4.2 Proposed Methods

Our objective is to minimize the overall energy consumption of the UAVs and the sensors, which is needed to accomplish the data collection task. To achieve this objective, we divide the original problem into three sub-problems of single UAV navigation, sensor power control, and multi-UAV scheduling. Single UAV navigation enables a UAV to design a safe and energy-efficient trajectory from a starting point towards a given destination. The sensor power control framework adjusts the sensor's transmit power which is needed to send all the data considering the energy consumption of the sensor and the hovering UAV. Having learned how to fly to the sensors and collect their data, the scheduling framework provides UAVs with their associated sensor at each time step. The scheduling plan is designed to find an optimal path (i.e., a sequence of sensors) for each UAV to visit to minimize the overall energy consumption of UAVs needed to accomplish the mission while balancing the load of each UAV. We translate each sub-problem into an MDP problem and then present the DRL frameworks considered to solve each part.

4.2.1 Single UAV Navigation

Suppose UAV m is located at the start point (x_m^0, y_m^0, H), aiming to fly towards sensor n located at $(x_n, y_n, 0)$. For the purpose of navigation, the UAV needs to use its sensory observation of the environment to avoid the obstacles, as well as its relative position to the sensor to reach the destination. The sensory observation of the UAV is obtained by using the range finders. The relative position to the sensor can be measured by GPS signals. Since the

UAV flies at a fixed altitude, we just consider the variable axes, and hence, the relationship between the UAV and the sensor at time step t can be written as

$$(\psi_x^t, \psi_y^t) = (x_m^t, y_m^t) - (x_n, y_n). \tag{4.11}$$

We can denote the overall state of the UAV by $s_{nav}^t = (\psi_x^t, \psi_y^t, v_m^t, \phi_m^t, l_0^t, \ldots, l_4^t)$, where $l_0^t \sim l_4^t \in [0, 100]$ show the output of the range finders. The UAV can change its speed and orientation angle along the path, which can be expressed as

$$\begin{aligned} v_m^{t+1} &= v_m^t + \Delta v_m^t, \\ \phi_m^{t+1} &= \phi_m^t + \Delta \phi_m^t, \end{aligned} \tag{4.12}$$

where Δv_m^t and $\Delta \phi_m^t$ represent the throttle and steering signals. We denote the action of the UAV by $a_{nav}^t = (\Delta v_m^t, \Delta \phi_m^t)$. Note that both elements in the action vector are continuous variables.

The navigation task in an obstacle-constraint environment is complex. The reward function must be somehow designed to guide the UAV to reach its destination while considering the energy consumption and obstacle constraints. Our proposed reward function is composed of 4 parts: transition, energy consumption penalty, obstacle penalty, and finishing reward. The transition reward is designed as

$$r_{trans} = \lambda_1 \Delta d, \tag{4.13}$$

where λ_1 is a positive constant, and Δd shows the reduced distance to the sensor after taking the action. Δd is positive when the UAV becomes closer to the sensor, motivating the UAV to head for the sensor.

We define the energy consumption penalty as

$$r_e = -\lambda_2 T_{nav} P_{nav}(v), \tag{4.14}$$

where λ_2 is a positive constant, and $P_{nav}(v)$ is the power consumption of the UAV defined in (4.2). The UAV receives this penalty to adjust its speed and trajectory so that the energy consumption along the way is minimized.

To encourage the UAV to avoid the obstacles, the obstacle penalty is designed as

$$r_{obs} = -\lambda_3 e^{-\lambda_4 l_{min}}, \tag{4.15}$$

where l_{min} is the minimum value among the range finders. If the UAV becomes really close to an obstacle in either of the range finder directions, the penalty would increase exponentially. To further encourage the UAV to move towards the sensor, it would get a large constant positive reward r_{finish} when it arrives at the destination. The overall reward function for the navigation framework is formulated as follows

$$r_{nav} = r_{trans} + r_e + r_{obs} + r_{finish}. \tag{4.16}$$

When there are multiple UAVs working together for data collection, one UAV will consider other UAVs as obstacles if they fly at the same altitude. In this case, the aforementioned navigation approach can also be applied to avoid the collision among UAVs.

By defining the states, actions, and reward function, our navigation problem has been formulated into an MDP. Now, to deal with the continuous control problem of navigation, we deploy the DDPG algorithm as the framework to solve our problem, which is shown in Fig 4.3.

As mentioned in Chapter 3, DDPG is an actor-critic architecture, which uses neural networks to approximate the policy and Q-value function. The neural network which approximates the Q-value function denoted by $Q(s, a|\theta^Q)$ is called critic since it evaluates the actions taken by the actor. The other network $\pi(s|\theta^\pi)$ takes in a new observation along with the feedback from the critic to output a new action; this is why this neural network is called actor.

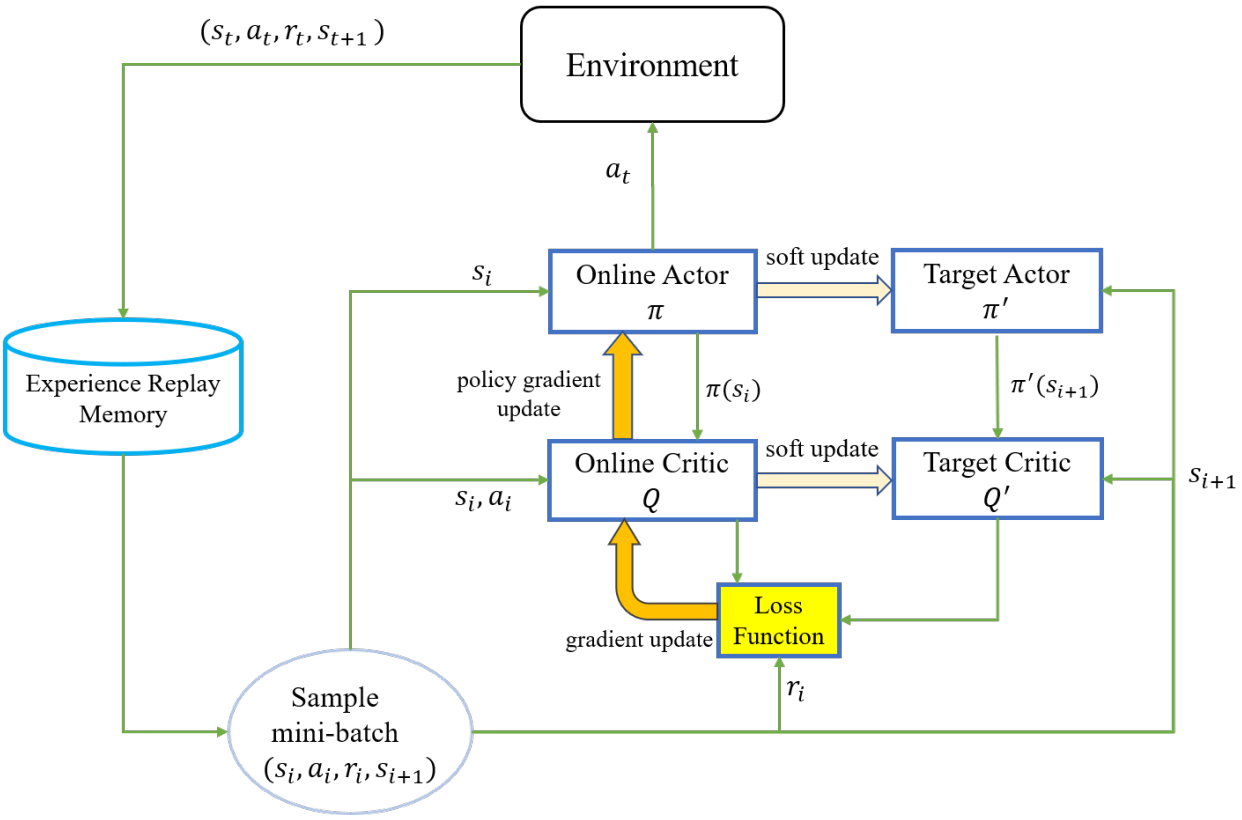

Figure 4.3: DDPG algorithm workflow

DDPG algorithm is formally presented in Algorithm 1. The algorithm works as follows. The weight vectors θ^Q and θ^π of the actor and critic networks are randomly initialized at the start of the algorithm (Line 1). As described earlier in Chapter 3, we use target networks π'

and Q' to improve learning stability. The target networks have the same structure as the online networks, and we initialize their weights in the same way as their online networks (Line 2). The target networks use soft updates to track the online network weights slowly (Line 14).

Algorithm 1 DDPG Algorithm

1: Randomly initialize critic $Q(s, a|\theta^Q)$ and actor $\pi(s|\theta^\pi)$ with weights θ^Q and θ^π
2: Initialize target network Q' and π' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\pi'} \leftarrow \theta^\pi$
3: Initialize the replay buffer W
4: **for** episode = 1,2, ... **do**
5: Receive the initial observation state s_1
6: **for** t = 1,2, ..., T_{max}^{DDPG} **do**
7: Select action $a_t = \pi(s_t|\theta^\pi) + \mathcal{N}_t$ according to the current policy and exploration noise
8: Execute action a_t, receive reward r_t and observe the new state s_{t+1}
9: Store transition (s_t, a_t, r_t, s_{t+1}) in W
10: Sample a random mini-batch of I transitions (s_i, a_i, r_i, s_{i+1}) from W
11: Set $y_i = r(s_i, a_i) + \gamma Q'(s_{i+1}, \pi'(s_{i+1}|\theta^{\pi'})|\theta^{Q'})$
12: Update critic by minimizing the loss:
$$L(\theta^Q) = \frac{1}{I}\sum_i \left[(Q(s_i, a_i|\theta^Q) - y_i)^2\right]$$
13: Update actor using the sampled policy gradient:
$$\nabla_{\theta^\pi} J \approx \frac{1}{I}\sum_i \left[\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=a_i} \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_{s_i}\right]$$
14: Update the target networks:
$$\theta^{Q'} \leftarrow \eta\theta^Q + (1-\eta)\theta^{Q'}$$
$$\theta^{\pi'} \leftarrow \eta\theta^\pi + (1-\eta)\theta^{\pi'}$$
15: **end for**
16: **end for**

During the training process, the agent can interact with the environment to learn the optimal policy. The agent takes its actions according to the actor network and a noise process $\mathcal{N}_t$ (Line 7). We add the noise process to ensure the agent's exploration in the environment.

Otherwise, we will not be able to try different actions, since the deterministic policy outputs just a single action. After taking the action, the agent receives a reward r_t and observes the next state s_{t+1}. We use a replay buffer to store the transition tuples (s_t, a_t, r_t, s_{t+1}) (Line 9). Then, a mini-batch is randomly sampled to train the actor and critic networks (Line 10). To achieve the goal of maximizing the accumulative future rewards the agent gets during its interaction with the environment, we update the critic and actor network weights by minimizing the loss function in (3.6), and maximizing the return in (3.8), respectively.

4.2.2 Sensor Power Control

After arriving at the target sensor, the UAV hovers above it to gather the data. At each time step, the UAV sets a proper transmit power for the sensor. Since we have assumed the UAV can measure the channel gain at each time step, we take the channel gain into the state space to provide the UAV with better decisions. Low channel gain indicates that the sensor must transmit more power for a certain amount of data rate. Also, the amount of data remained at the sensor must be considered, since it helps the UAV achieve the goal state in which the remaining data must be zero. The state can be written as $s^t_{spc} = (g_t, \zeta_t)$ where g_t is the channel gain at time t, and ζ_t is the remaining data at the sensor. The action is denoted by $a^t_{spc} = P_t$, where P_t is the transmit power of the sensor controlled by the UAV.

For the reward function, we need to motivate the UAV to collect data, i.e., having a high data rate. In addition, we must avoid using high transmission power if the channel is not in good condition. We should note that during data collection, the UAV itself is spending power for communication and hovering as stated in (4.4). With all these into consideration,

we design the reward function as follows:

$$r_{spc} = \lambda_5 T_{com} K_t - \lambda_6 P_{dc} - \lambda_7 P_t, \tag{4.17}$$

where K_t is the data rate in (4.9), P_{dc} is the UAV's power consumption during data collection, and P_t is the transmission power of the sensor. λ_5, λ_6 and λ_7 are the coefficients that control the trade-off between these three components.

The formulated MDP problem is also a continuous control task that can be solved by the DDPG algorithm. To avoid repetition, we leave out the details of the algorithm and refer to Algorithm 1.

4.2.3 Multi-UAV Scheduling

The data collection is a task where the UAVs need to plan their destinations in a way that minimizes their navigation energy consumption. We utilize the scheduling framework to organize trip plans for each of the UAVs to visit the sensors. We assume that the UAVs have sufficient energy storage so that none of them runs out of battery within the predefined duration considered for the data collection mission. Our objective is to minimize the overall energy consumption of the UAVs, which is required to finish the data collection mission within the considered duration.

We consider the stop points for each UAV, namely the N sensors and the starting locations. We will denote the set of stop points by $\Omega = \{\omega_0, \omega_1, \ldots, \omega_N\}$, where ω_0 represents the starting point, and the rest of them are the locations of the sensors. In each time step, the UAV can choose one of the stop points as its current destination and fly towards it by incorporating

the trained navigation framework explained in section 4.2.1. We use the same time-division system (T_{nav}) as the navigation framework to let the UAVs have the flexibility to choose their target in each time step. This allows our solution to be extended to more general and dynamic scenarios, as we will discuss it in section 5.2.

We denote the status of sensor n by $\nu(n) = \frac{\zeta_t^n}{\zeta_0^n}$, which is the normalized remaining data at the sensor and has a value between 0 and 1. The value of 1 means that the sensor's data has not been collected yet, while 0 means that a UAV has collected all the sensor's data. In this work, we only consider the propulsion energy consumed by the UAV as the criteria to determine the scheduling plan. Thus, the UAV's energy consumption during data collection while hovering at the sensor location does not impact the scheduling plan and can be ignored.

The state of UAV m can be written as $s_{sch,m}^t = (x_m, y_m, \nu(1), \ldots, \nu(N))$, where (x_m, y_m) is the UAV's current location, and $\nu(1), \ldots, \nu(N)$ denote the status of all the sensors, which can be acquired by the help of a central controller or through the information exchange among UAVs. The sensors' status helps the UAV decide for its next destination, as $\nu(n) = 0$ means the sensor does not need to be visited again. The action of UAV m is its choice for the next target location represented by $a_{sch,m}^t \in \Omega$.

We design the reward function to motivate the UAVs to find the best route starting from ω_0 and then returning to it. In order to design the reward function for each agent, we should consider the following points:

- Each UAV gets a penalty proportional to its power consumption $P_{nav}(v)$ at each time step, which is to motivate it to finish the task as early as possible.

- Each UAV gets a constant positive reward λ_9 when its destination is an unvisited sensor.

- To motivate the UAVs to fly back to the starting location, they get constant positive reward λ_{10} during their flight towards the starting location after collecting all the data, i.e., $\nu(n) = 0$ for all the sensors.

With all these in mind, we express the reward function of UAV m as

$$r^t_{sch,m} = -\lambda_8 T_{nav} P_{nav}(v) + \lambda_9 + \lambda_{10} \tag{4.18}$$

where $P_{nav}(v)$ can be obtained by incorporating the navigation framework after the scheduling framework determines the destination of the UAV. As we explained earlier, the navigation framework guides the agent towards its destination and can provide the power consumption information.

Having individual rewards may benefit one, while punishing the others. However, the UAVs need to work cooperatively to schedule their visiting tour. Therefore, by sharing the same reward, we can ensure that they work as a team for the data collection. We define the overall reward at time step t as

$$r^t_{sch} = \sum_m r^t_{sch,m}. \tag{4.19}$$

Our goal is to find the best destination at each time step to optimize the UAVs' total energy consumption while ensuring all the data to be collected in the predefined duration. We utilize the DQL framework to find the best scheduling plan for the UAVs, since each agent has a discrete and finite set of actions. DQL has a simpler implementation than the actor-critic

architectures. However, since we are dealing with multiple agents simultaneously, we modify the original DQL algorithm and propose the multi-agent DQL framework, where each agent has a separate Q-network that controls its policy. We have presented the multi-agent DQL solution in Algorithm 2. Our algorithm works as follows.

Algorithm 2 Multi-agent DQL Algorithm

Randomly initialize Q-network $Q_{sch,m}(s, a|\theta^Q)$ with weights $\theta^{Q_{sch,m}}$ for all the UAV agents
Initialize target network $Q'_{sch,m}$ with weights $\theta^{Q'_{sch,m}} \leftarrow \theta^{Q_{sch,m}}$
Initialize the replay buffer W_m and ϵ for each UAV
for $episode = 1, 2, ...$ **do**
  Receive the initial observation state s_1^m for each agent
  for t = 1,2, ...,T_{max}^{DQL} **do**
    for m in {1, ..., M} **do**
      Select the action a_t^m for UAV m according to ϵ-greedy policy
    end for
    Execute the actions together, receive individual reward r_t^m and observe new state s_{t+1}^m for each agent
    Compute the overall reward in (4.19) by summing over the individual rewards
    for m in {1, ..., M} **do**
      Update the agent's reward and store the transition $(s_t^m, a_t^m, r_t^{sch}, s_{t+1}^m)$ in W_m
      Sample a random minibatch of I transitions (s_i, a_i, r_i, s_{i+1}) from W_m
      Set $y_i = r_i + \gamma \max_{a_{i+1}} Q'_{sch,m}(s_{i+1}, a_{i+1}|\theta^{Q'})$
      Update the $Q_{sch,m}$ by minimizing the loss:
$$L(\theta^{Q_{sch,m}}) = \frac{1}{I}\sum_i \left[(Q_{sch,m}(s_i, a_i|\theta^Q) - y_i)^2\right]$$
    end for
  end for
  Decay ϵ by a factor if $\epsilon > \epsilon_{stop}$
  Update the target networks when $episode\%N_{upd} = 0$
$$\theta^{Q'_{sch,m}} \leftarrow \theta^{Q_{sch,m}}$$
end for

In the beginning, we initialize the network weights for each of the UAVs. We also do

the same procedure for the target networks (Line 1,2). Unlike DDPG, the target networks are updated every N_{upd} episodes by fetching the original network weights (Line 20). As the training begins, in each time step, the agents choose their actions by using ϵ-greedy policy (Line 8), where each agent chooses a random action with probability ϵ; otherwise, the greedy action $\arg\max_a Q(s,a)$ is selected based on the agent's Q-network. After receiving all the actions, the system generates the reward and the next state for each agent (Line 10). We should note that according to our formulation in (4.19), the updated reward is given to the agents by summing over their individual rewards (Line 11). Then, each agent stores its transition tuple in the replay buffer, samples a random tuple, and updates the network weights by minimizing the sample loss. (Line 13-16). We decay the ϵ in each episode to reduce the rate of exploration as the agent begins to learn a better policy (Line 19). The workflow of the algorithm is shown in Fig 4.4.

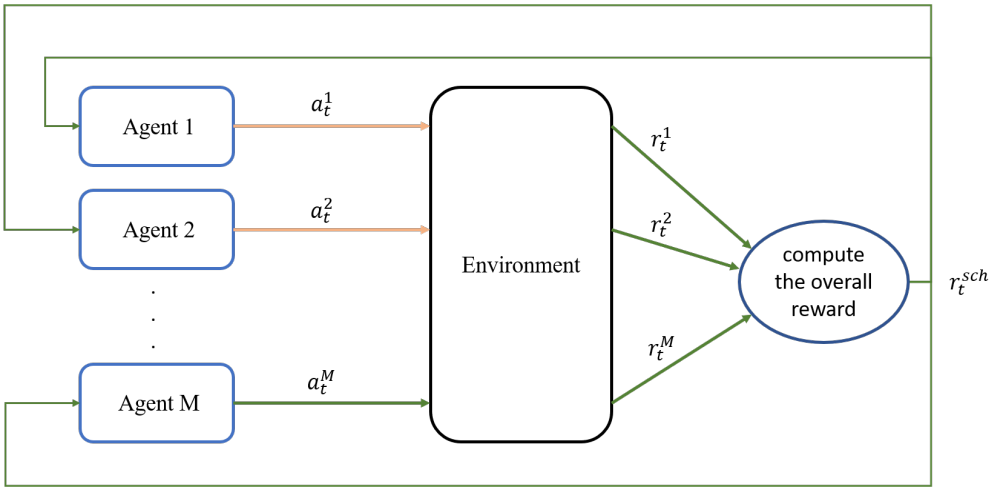

Figure 4.4: Multi-agent DQL algorithm workflow

4.3 Evaluation

In this section, we discuss the experimental setting and the performance of the proposed frameworks. Our simulations are performed with Python 3.7 and Tensorflow 2.0 on an Intel Core i7-9700CPU with 16G RAM. The simulation parameters regarding the UAV's propulsion power and communication channel are shown in Table 4.1. We have chosen these parameters according to the existing literature in [13], [27]. The values of the coefficients λ_1 to λ_{10} and r_{finish} in (4.16), (4.17), and (4.18) are carefully set through many trials that well balance the trade-off between different terms in the reward functions. If a coefficient has a too large value, the agent will pay more attention to the corresponding reward term, ignoring the others. For example, if we set λ_1 in (4.16) to a large value compared to the other terms, the agent may ignore the obstacle avoidance constraint and crash into obstacles more often.

For our DDPG model, we use a 2-layer feedforward neural network architecture with 400 and 300 neurons in each layer for the actor and critic. Furthermore, the Adam optimizer [32] is used to update the network parameters, and similar to [11], learning rates of 10^{-4} and 10^{-3} are chosen for the actor and critic networks, respectively. In our implementation, $\mathcal{N}_t$ has normal distribution with mean of 0 and variance of 0.2 to explore the environment. We use minibatch sizes of 64, the discount factor is $\gamma = 0.99$ and the soft target update rate is $\eta = 0.001$. We use the same architecture and parameters as the critic network in DDPG for each of the agents in the multi-agent DQL model. The ϵ starts from the value of 0.5 and decays by the factor of 0.999 until reaching $\epsilon_{stop} = 0.05$. We update the target networks every 10 episodes ($N_{upd} = 10$) to stabilize the learning process.

Parameter	Description	Simulation Value
f_c	Carrier frequency	2 GHz
B	Bandwidth	1 MHz
σ^2	Noise power	-100 dB
χ	Rician factor	1
α	Pathloss exponent	2
Λ	Data size of each sensor	100 Mbits
U_{tip}	Tip speed of the rotor blade	200
v_0	Mean rotor induced velocity in hovering	7.2
d_0	Fuselage drag ratio	0.3
ρ	Air density	1.225
s	Rotor solidity	0.05
A	Rotor disc area	0.79
P_0	Blade profile power in hovering	580.65
P_i	Induced power in hovering	790.67

Table 4.1: Paramters used in the simulations

4.3.1 Single UAV Navigation

We simulate the virtual obstacle-constrained environment, in which the UAV starts from a random location and is expected to fly to a destination. The target area is a 2D square of size $600 \times 600\ m^2$ with obstacles randomly located in it. Here, we aim to reach the destination by avoiding obstacles and minimizing the energy consumption of the UAV. The UAV's flight altitude H is set to 50 m. The reward is instantiated as: $\lambda_1 = 0.3$, $\lambda_2 T_{nav} = 0.002$, $\lambda_3 = 50$, $\lambda_4 = 0.1$ and $r_{finish} = 50$.

First, we illustrate the convergence of the proposed method. We trained the DDPG model for 5000 episodes, each of which has a maximum of 30 time steps. In Fig. 4.5, the convergence of the trained model with different learning rates of actor and critic networks is presented.

Figure 4.5: Convergence of the DDPG model during the training for the navigation task: (a) the average return received by the UAV in the last 100 episodes (b) the average propulsion energy (in joule) consumed by the UAV in the last 100 episodes.

Fig. 4.5(a) shows the average return formulated in (3.1) obtained by the UAV. Specifically, since we are randomly choosing the starting and finishing points in the area with different distances between them, the return received by the UAV in each episode can be different. Therefore, we average over the last 100 episodes to better understand the convergence of our model. We can see that the UAV learns to obtain the maximum accumulated reward in the obstacle-constrained environment with the original learning rates. In Fig. 4.5(b), we present the average propulsion energy consumed by the UAV in the last 100 episodes. We can see that as the model achieves a better return, the propulsion energy of the UAV decreases. The energy penalty defined in (4.14) causes the UAV to get higher rewards by decreasing the energy consumption on its path to the destination. Although the learning rates of 10^{-3} and 10^{-2} for the actor and critic networks perform better at the beginning, they result in lower

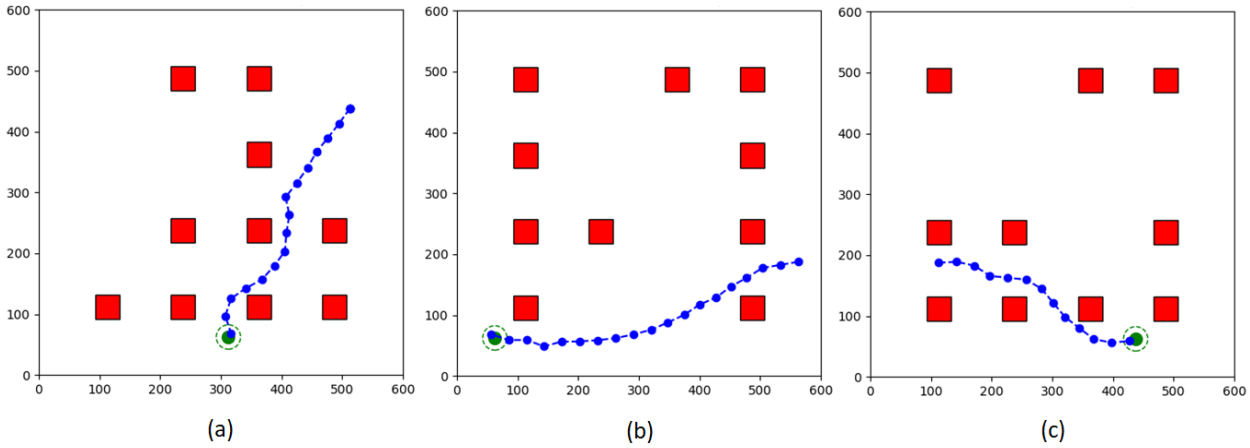

Figure 4.6: UAV's Trajectory in 3 difference environment configurations.

performance comparing to learning rates of 10^{-4} and 10^{-3}. High learning rates cause the gradient updates to be large, which leads to a sub-optimal solution. Hereinafter, we set the learning rate as the latter ones.

We should mention that the training time complexity of the single UAV navigation framework is independent of the number of obstacles and only changes based on the number of range finders to which the UAV is equipped. Instead of using the obstacles' positions in our state representation, we incorporate the range finder readings, which helps the UAV maintain its relative distance to the obstacles.

In Figure 4.6, we present the trajectory followed by the trained UAV for 3 different environments, where the green circle shows the hovering location above the sensor, the red squares show the positions of the obstacles, and the blue dots indicate the trajectory of the UAV at each time step. We change the locations of the obstacles, starting position, and the destination and observe that the trained DRL agent is flexible to different scenarios, because

we considered the relative position to the destination and the range finders as our state, which helps the agent make proper decisions in different scenarios. By adopting continuous control actions, the UAV is able to change its path when getting close to an obstacle.

We also test the model for a period of 1000 episodes to validate the successful navigation without any collision to the obstacles. The UAV reaches the target safely for 90.8% of the test episodes.

4.3.2 Sensor Power Control

We consider the scenario where the UAV hovers above a sensor location to collect its data. Considering the reward function in (4.17), our goal is to jointly optimize the UAV and sensor's power consumption during data collection. We investigate the impact of different coefficients, which can control the trade-off between the two types of power consumption. We assume that the transmit power of the sensor has a maximum value of 0.1 W. We set the maximum number of time steps in each episode to 100. Unless stated, we use the following parameters for the reward function defined in (4.17): $\lambda_5 T_{com} = 0.1$, $\lambda_6 P_{dc} = 0.1$, $\lambda_7 = 10$.

In Fig. 4.7, we illustrate the average return received by the agent during training. Since the agent may experience different channel gains in each episode, we average over the last 100 episodes to smooth the curve. Compared to the navigation task, the DDPG model converges faster (about 10 times). This is mainly due to the fact that we have a much simpler state representation and action space, which speeds up the convergence. Our state consists of the values of channel gain and the normalized remaining data, and our action is the transmit power of the sensor.

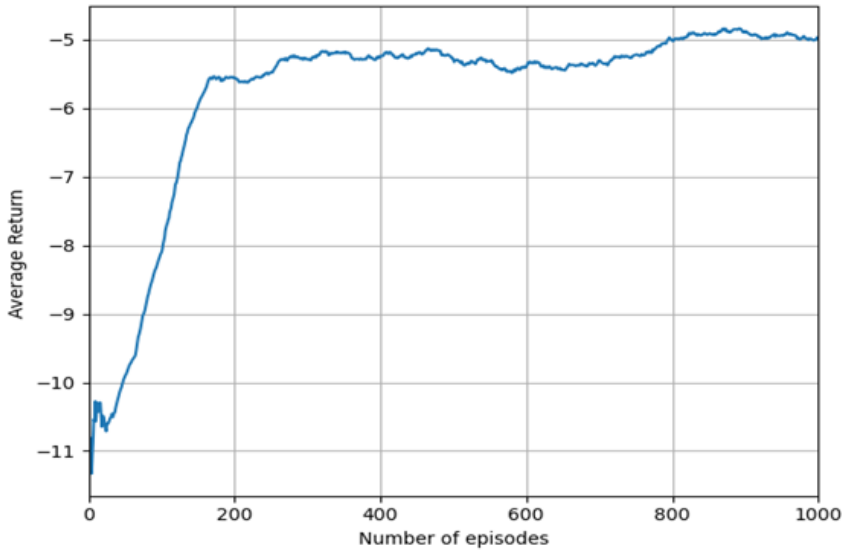

Figure 4.7: Convergence of the DDPG model during the training for the sensor power control.

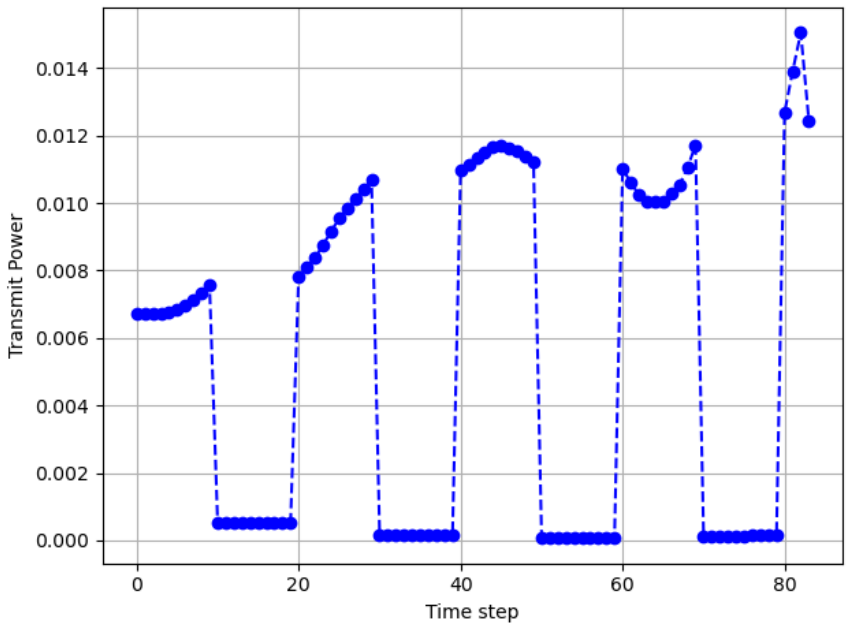

Figure 4.8: The DDPG agent's output power during an episode

To better understand the relationship between the output power and the channel gain, we temporarily ignore the Rician distribution and assume that the channel gain in each episode

can only switch between 0.1 and 1 values, where 0.1 indicates a very poor channel condition and 1 indicates a good channel condition. In Fig. 4.8, the sensor's output power is illustrated in each time step of an episode. In the first 10 time steps, the channel gain is set to 1 (i.e., good channel condition), and after each 10 time steps, it switches to the other value, i.e., the channel alternates between poor and good conditions. The agent outputs a higher power as the channel link experiences better quality, whereas it outputs close-to-zero power when the channel condition is poor to avoid the waste of sensor energy. Also, as the agent approaches the end of the data collection period, it outputs a higher power to finish the task in the predefined time.

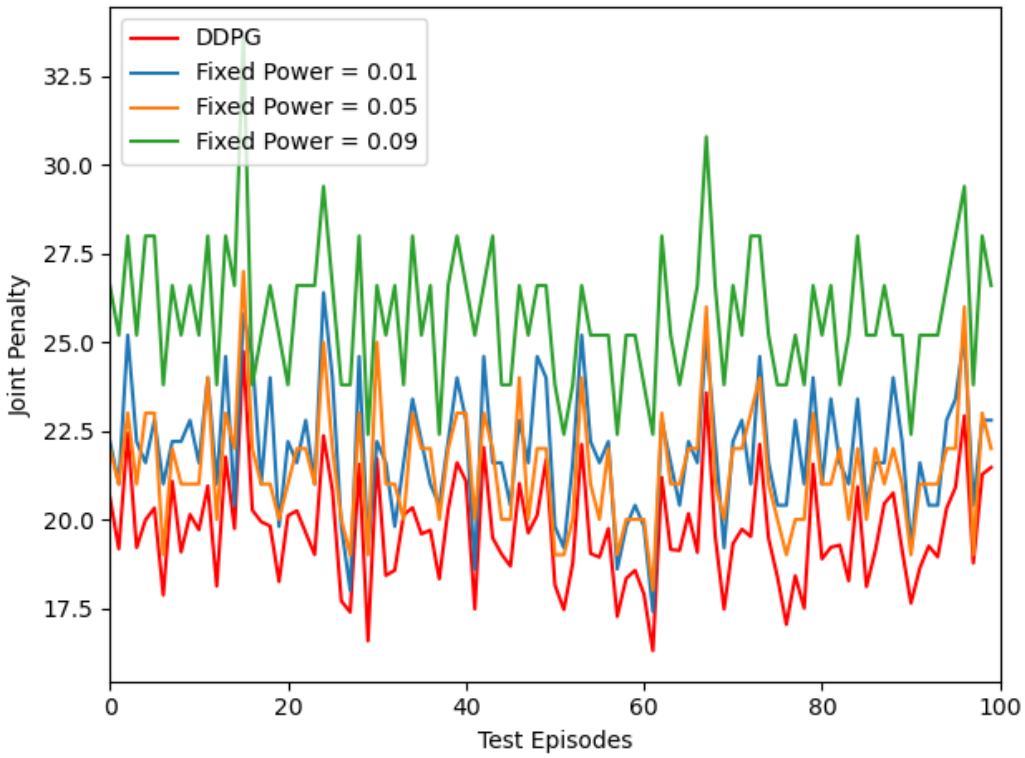

Figure 4.9: Joint penalty received by the UAV during the power control.

We compare the reward received by our adaptive model to some fixed power scenarios in the testing phase. Since the accumulated data collected by the UAV is constant in each episode and equal to 100 Mbits, the accumulation of the first term $\lambda_5 T_{com} K_t$ in the reward function (4.17) would be the same for all the scenarios. We added this term to the reward to motivate the UAV to collect data from the sensor. Hence, we take it out for comparison purposes and focus on the joint penalty $\lambda_6 P_{dc} + \lambda_7 P_t$ received by the agent. The penalty considers the power consumed by both the UAV and the sensor jointly at the same time. We choose the three values of 0.01 W, 0.05 W, and 0.09 W out of the allowable power range to account for the small, medium, and high fixed power scenarios. For ease of exposition, we present the results for 100 episodes. In all the scenarios, the environment experienced by the agent, i.e., the channel gains it receives in a specific episode and time step, is exactly the same. Fig. 4.9 shows the result for the case of $\lambda_6 P_{dc} = 0.5$ and $\lambda_7 = 10$. As we can see, the DDPG model achieves better performance against the fixed power approaches. We should note that as the transmit power of the sensor increases, we are likely to have a higher data rate, and the time required to collect all the data from the sensor decreases. As a consequence, the UAV spends a smaller amount of power for its communication and hovering. The model has learned the trade-off between P_{dc} and P_t successfully and can adaptively adjust the sensor's power by considering the environment.

In Fig. 4.10, we compare the sensor power consumption and the completion time of the DRL agent against a fixed power approach in which the sensor outputs the average power of the DRL agent. In particular, we average the transmit power of the DRL agent in each

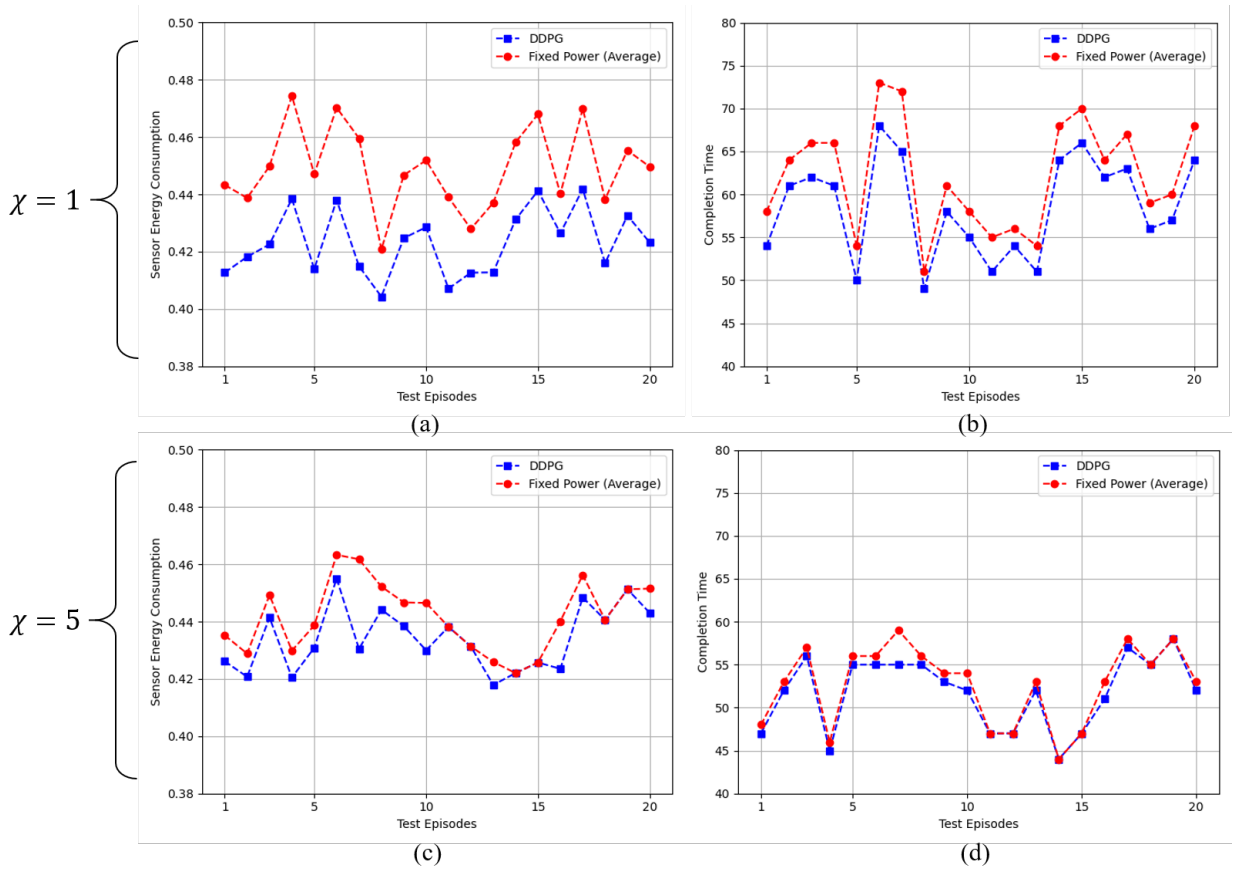

Figure 4.10: Comparison between the DRL agent and the fixed power approach in terms the sensor energy consumption (in joule) and completion time (number of time steps): in (a) and (b) the environment has a Rician factor of 1, and in (c) and (d) the Rician factor is set to 5.

episode and set the fixed power to that value. Also, the channel gains observed by both approaches are the same to make the comparison fair. Our goal is to see how the overall energy consumption and completion time in each episode change if the sensor transmits data with the average power of the DRL agent, regardless of the channel conditions. We perform our experiment in two environments with different Rician factors (χ=1 and $\chi = 5$). For both environments, it can be seen that the DDPG agent outperforms the fixed power

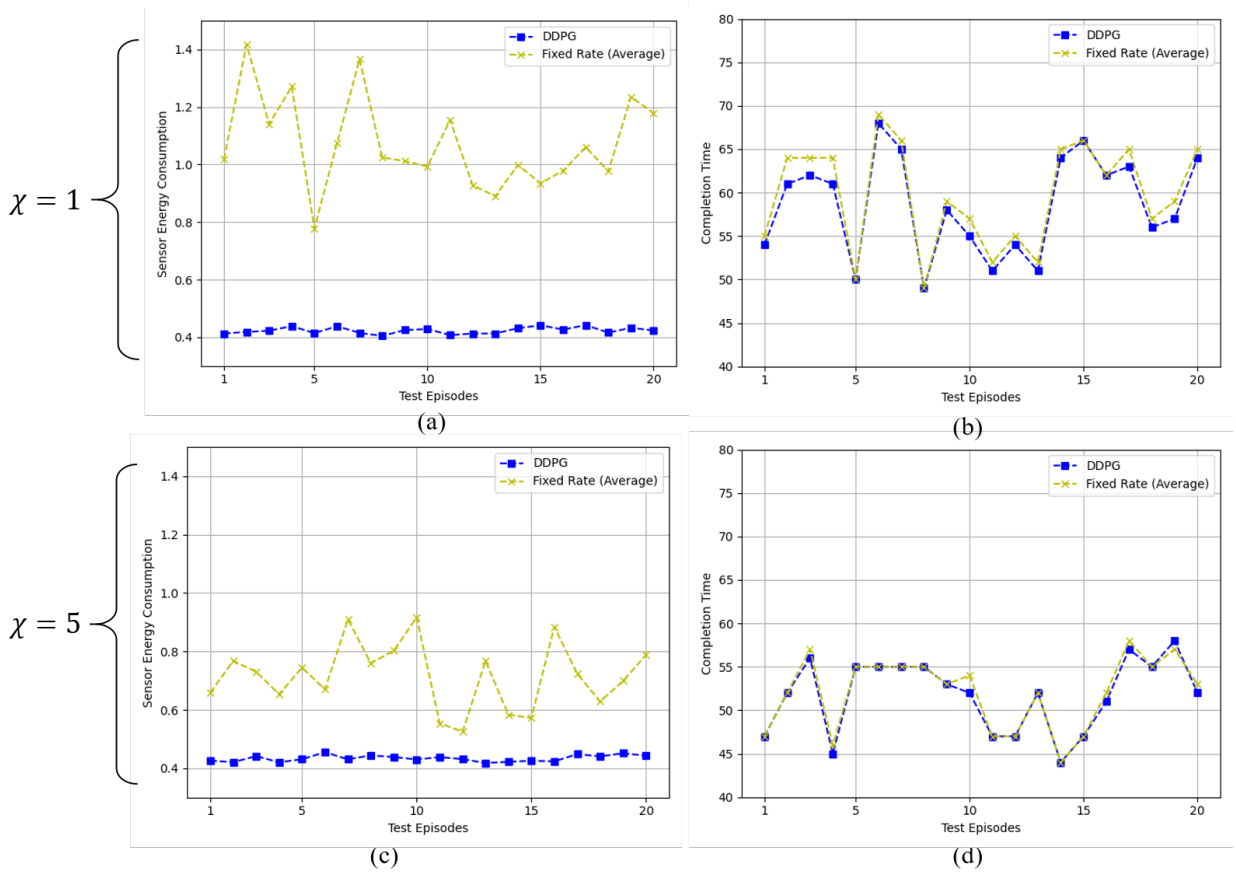

Figure 4.11: Comparison between the DRL agent and the fixed rate approach in terms the sensor energy consumption and completion time: in (a) and (b) the environment has a Rician factor of 1, and in (c) and (d) the Rician factor is set to 5.

approach by adaptively controlling the sensor's transmit power according to the varying channel gain. The DRL agent can finish the data collection task earlier, while the sensor consumes less energy than the other method. As the completion time decreases, the UAV also spends less energy for hovering. Therefore, our DDPG approach is successfully optimizing the energy consumption of the sensor and the hovering UAV. We should mention that as the Rician factor increases, as shown in Fig. 4.10 (c) and (d), the variations in the channel gain

decrease because of the fact that the UAV receives more power in the direct path rather than the scattered paths. This explains why the performance of the DDPG and the fixed power approach becomes closer to each other.

We adopt a similar comparison between the DDPG algorithm and a fixed rate approach, where in each episode, the fixed data transmission rate is set to the average rate of the DDPG agent. In Fig. 4.11, we see the fixed rate approach consumes much higher energy, although the completion time is close to the DDPG agent. The reason is that when the link between the UAV and the sensor has a low channel gain, the fixed rate approach needs to output a high power in order to ensure the same rate, which leads to the high energy consumption of the sensor. Similar to the fixed power method, increasing the Rician factor makes the performance of the approaches close to each other.

4.3.3 Multi-UAV Scheduling

In this section, we study the behavior of the multi-agent scheduling framework in the selected scenario, shown in Fig. (4.13). The black spot at the bottom left shows the departure point for the UAVs (ω_0), the red squares show the positions of the obstacles, and the green circles are the sensor locations ($\omega_1 - \omega_6$). The UAVs start their trip from ω_0 and fly to the sensors to collect their data. For ease of illustration, we have considered 2 UAVs; however, the presented framework can handle a general case with more UAVs. Each training episode lasts 100 time steps. The reward is instantiated as: $\lambda_8 = 0.001$, $\lambda_9 = 1$, $\lambda_{10} = 1$.

Fig. 4.12 depicts the convergence of the model in terms of the overall accumulated reward received by the agents and mission completion time in each episode of the training phase. As

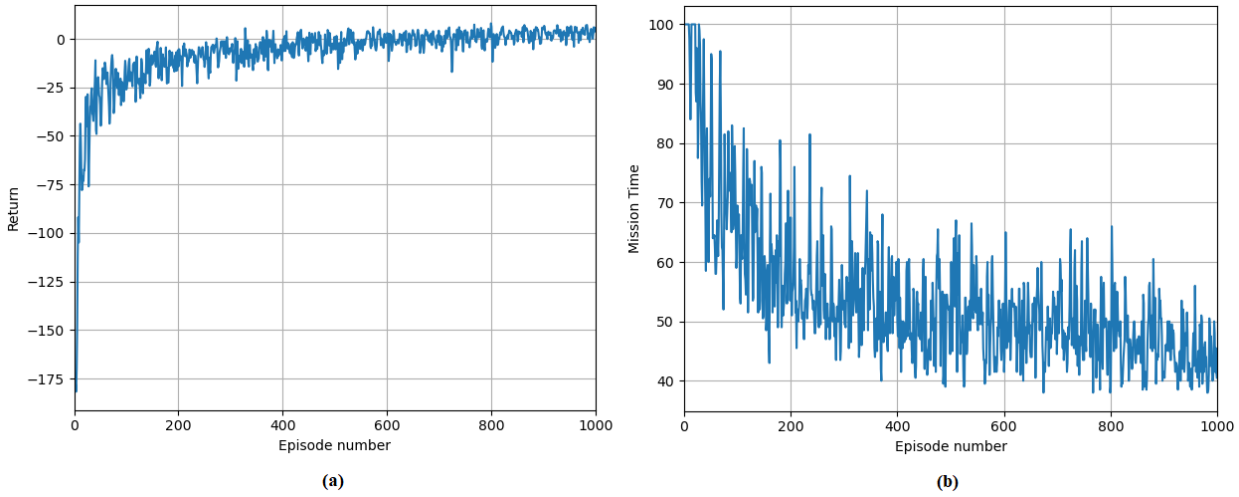

Figure 4.12: Convergence of the DQL model during the training in terms of (a) the overall accumulated reward received by the agents (b) mission completion time

we can see, at beginning episodes, the agents do not know how to interact with each other and receive a high amount of penalty, but as we progress in training, their policies improve, resulting in a lower penalty. The fluctuations in the figure result from using ϵ-greedy policies, which makes the agents choose random actions during training to explore the effect of taking different actions. In Fig. 4.12 (b), the completion time in terms of the number of time slots is shown for each training episode. At the beginning of the training, the episodes take the maximum allowed number of time steps (100), which means that UAVs have not learned to accomplish the data collection mission. However, during the training, the collection time decreases to around 40, and the UAVs can cooperatively collect all the data and return to the starting location.

Since we included the status of all sensors in the state representation, the training time complexity of each agent increases with respect to the number of sensors. In addition, we

should point out that as the number of UAVs increases, the agents need more time to learn the complex interaction with the other agents. Therefore, the number of both UAVs and sensors has an impact on the training time.

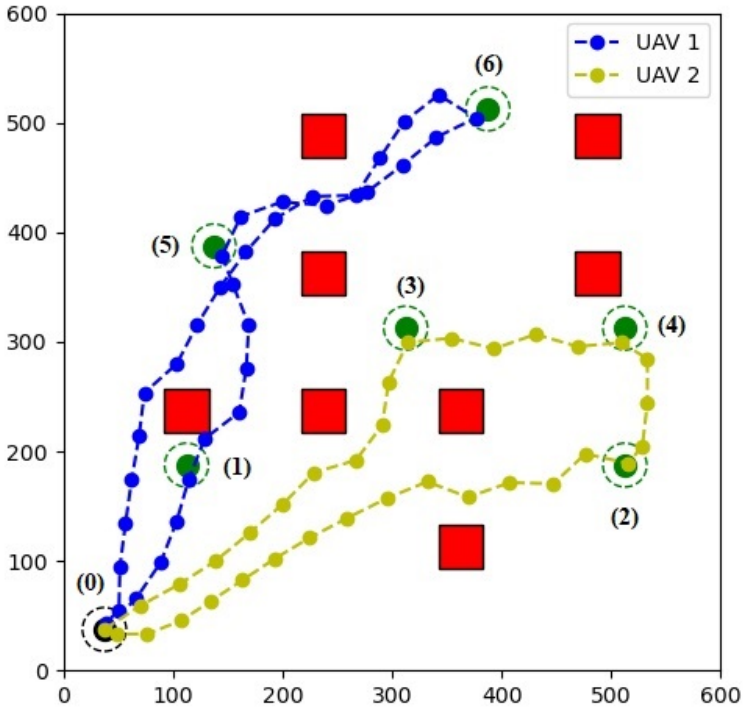

Figure 4.13: Considered Scenario for data collection: the black spot shows the departure point of the UAVs (ω_0). The green circle illustrates the sensor locations ($\omega_1 - \omega_6$).

To evaluate our proposed DRL solution, we compare the multi-agent approach to the random and greedy baselines in terms of the total energy consumption of the UAVs. To make a fair comparison between these methods, we assume they use the same navigation framework to design their trajectory from a starting point to a destination. In the random approach, the agents choose the unvisited sensors randomly, whereas in the greedy approach, the agents fly immediately to the next unvisited sensor with the lowest energy consumption

path. We also find the optimal solution by doing an exhaustive search. In all the solutions, the UAVs are required to collect data from the sensors evenly to make use of their storage capacity and decrease the data collection time. The results have been shown in Table 4.2. The 2nd and 3rd columns indicate the scheduled trip for UAV 1 and UAV 2 respectively, e.g., $0 \to 1 \to 5 \to 6 \to 0$ means that the UAV departs at ω_0, and then visits sensors $\omega_1, \omega_5, \omega_6$, and finally goes back to the departure point ω_0. Our measure for this comparison is the total energy consumption of both UAVs when flying according to the scheduled trip, shown as the cost in the last column of Table 4.2. We see that the DQL model achieves a closer result to the optimal solution comparing to the other two methods.

Method	UAV 1	UAV 2	Total energy consumption
DQL	$0 \to 1 \to 5 \to 6 \to 0$	$0 \to 3 \to 4 \to 2 \to 0$	95803
Random	$0 \to 1 \to 3 \to 2 \to 0$	$0 \to 4 \to 6 \to 5 \to 0$	113926
Greedy	$0 \to 1 \to 3 \to 4 \to 0$	$0 \to 5 \to 6 \to 2 \to 0$	97953
Optimal	$0 \to 1 \to 5 \to 6 \to 0$	$0 \to 2 \to 4 \to 3 \to 0$	92785

Table 4.2: Total energy consumption of the UAVs for different methods

Chapter 5

Conclusion

5.1 Summary of the Thesis

In this work, we have proposed a DRL-based approach to solve the data collection problem for multiple UAVs in IoT networks with the aim to minimize the energy consumption on both the UAVs and sensors sides. We have divided the original problem into three sub-problems of single UAV navigation, sensor power control, and multi-UAV scheduling and solved each sub-problem by utilizing a DRL algorithm. In the navigation framework, the DDPG algorithm has been deployed to enable each UAV to generate its trajectory autonomously by adjusting its speed and orientation angle, given the starting and finishing positions. The reward function has been designed to make the UAVs avoid the obstacles and the region borders with the help of their sensory observations. For the sensor power control part, we have used the DDPG method to control the sensor's transmit power adaptively based on the current channel gain and the remaining data. To schedule the trip plans for each UAV, we have proposed the multi-agent DQL algorithm by incorporating the navigation framework and minimizing the UAVs' overall energy consumption during their flight on the scheduled paths. Our simulation

results have shown that the navigation framework enables the UAVs to fly safely towards their target by avoiding obstacles. The learned policy can perform appropriately in different map scenarios due to the formulation of states and the reward function. Also, by controlling the sensor's power, we can obtain better performance than the fixed power and fixed rate approaches in terms of the total sensor power consumption and the data collection completion time. Finally, our scheduling framework enables the UAVs to make autonomous decisions for their next target sensor based on the sensors' current status, achieving better performance than the random and greedy baselines.

5.2 Future Works

We can extend our data collection scenario to more general settings. One can be adding multiple charging stations to the map, where the UAVs need to recharge their battery to be able to continue their data collection mission. In this case, the battery level of the UAV must be considered in the DRL states, which allows the UAVs to make decisions based on their battery level, i.e., when their battery level is low, they return to the charging station to reload the battery. We can also consider sensors with different data time windows, where a UAV must collect their data within the time window; otherwise, the data is lost.

We should mention that the scheduling scenarios we have discussed so far can also be formulated as a capacitated vehicle routing problem with time windows. However, the MDP formulation presented in section 4.2.3 allows us to consider dynamic data collection scenarios, where the sensors can collect data dynamically as the UAVs are flying over the region. In

this case, the time windows of the newly collected data can be announced to the UAVs, and DRL methods can be appropriately utilized. Also, when a UAV starts the data collection from a specific sensor, it can pause the data collection and fly towards another sensor with critical data collection demands.

Furthermore, we can try to replace the feedforward neural network architecture with more sophisticated ones. In [33], recurrent neural architecture is investigated to deal with partially observed environments. When the agent's observation of the environment does not fully represent the MDP states, memorizing the observations can help make better decisions. In this case, the agent makes its next decision based on the history of the observations and not only based on the current observation. This memory might be helpful in the navigation and sensor power control, as we take the whole observed history into consideration.

Bibliography

[1] M. Mozaffari and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2334–2360, 2019.

[2] K. Kuru and *et al.*, "Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform," *Ieee Access*, vol. 7, pp. 15 804–15 831, 2019.

[3] J. Wang and *et al.*, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *Ieee vehIcular technology magazIne*, vol. 12, no. 3, pp. 73–82, 2017.

[4] G. Ding and *et al.*, "An amateur drone surveillance system based on the cognitive internet of things," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 29–35, 2018.

[5] A. Kadri and *et al.*, "Wireless sensor network for real-time air pollution monitoring," in *2013 1st international conference on communications, signal processing, and their applications (ICCSPA)*, IEEE, 2013, pp. 1–5.

[6] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Energy-efficient multi-UAV data collection for IoT networks with time deadlines," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[7] C. Y. Tazibt and *et al.*, "Wireless sensor network clustering for UAV-based data gathering," in *2017 Wireless Days*, IEEE, 2017, pp. 245–247.

[8] C. H. Liu and *et al.*, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059–2070, 2018.

[9] Y. Wang and *et al.*, "Multi-UAV collaborative data collection for IoT devices powered by battery," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, 2020, pp. 1–6.

[10] S. Fu and *et al.*, "Energy-efficient UAV enabled data collection via wireless charging: A reinforcement learning approach," *IEEE Internet of Things Journal*, 2021.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[12] V. Mnih and *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] Y. Wang and *et al.*, "Minimizing data collection time with collaborative UAVs in wireless sensor networks," *IEEE Access*, vol. 8, pp. 98 659–98 669, 2020.

[14] S. Alfattani, W. Jaafar, H. Yanikomeroglu, and A. Yongacoglu, "Multi-UAV data collection framework for wireless sensor networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2019, pp. 1–6.

[15] J. K. Lenstra and A. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, 1981.

[16] M. B. Ghorbel, D. Rodrıéguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, "Joint position and travel path optimization for energy efficient wireless data gathering

using unmanned aerial vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2165–2175, 2019.

[17] C. You and R. Zhang, "3d trajectory optimization in rician fading for UAV-enabled data harvesting," *IEEE Transactions on Wireless Communications*, vol. 18, no. 6, pp. 3192–3207, 2019.

[18] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 34–46, 2020.

[19] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2020, pp. 716–721.

[20] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2019, pp. 1–6.

[21] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "Deep reinforcement learning for efficient data collection in UAV-aided internet of things," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, 2020, pp. 1–6.

[22] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "UAV path planning for wireless data harvesting: A deep reinforcement learning approach," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[23] C. Zhou, H. He, P. Yang, F. Lyu, W. Wu, N. Cheng, and X. Shen, "Deep RL-based trajectory planning for AoI minimization in UAV-assisted IoT," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, IEEE, 2019, pp. 1–6.

[24] C. Piao and C. H. Liu, "Energy-efficient mobile crowdsensing by unmanned vehicles: A sequential deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6312–6324, 2019.

[25] J. Tang, J. Song, J. Ou, J. Luo, X. Zhang, and K.-K. Wong, "Minimum throughput maximization for multi-UAV enabled WPCN: A deep reinforcement learning method," *IEEE Access*, vol. 8, pp. 9124–9132, 2020.

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

[27] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, "A UAV-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling," *IEEE Access*, vol. 8, pp. 110 446–110 460, 2020.

[28] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[29] C. Wang and *et al.*, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2124–2136, 2019.

[30] C. Zhan and Y. Zeng, "Aerial–ground cost tradeoff for multi-UAV-enabled data collection in wireless sensor networks," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1937–1950, 2019.

[31] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: `http://arxiv.org/abs/1412.6980`.

[33] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.