Task Offloading with Safe and Stable Machine Learning in Next Generation Wireless Networks

by

Luciana Farias De Oliveira Nobrega

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of

Master of Applied Science in
Electrical and Computer Engineering
Concentration in Applied Artificial Intelligence

Abstract

The advent of next-generation wireless networks has necessitated efficient and safe offloading of computational tasks to meet the demands of users for low latency and high reliability. This study focuses on task offloading with Machine Learning (ML) solutions in Unmanned Aerial Vehicle (UAV)-aided Multi-Access Edge Computing (MEC) systems, specifically applied to smart agriculture. The system involves closed-loop communication where sensors collect data, offload it to processors, and return commands to actuators. Despite efforts to minimize computational task waiting and processing times, few studies address these issues alongside both information delay and freshness, particularly in UAV systems.

This thesis aims to jointly optimize data freshness and computational Turnaround Time (TAT) in UAV-aided MEC systems by proposing a UAV trajectory and UAV-MEC offloading strategy. We explore the trade-off between the Age of Loop (AoL) and computational TAT in dynamic network conditions, varying computational resources, data arrival rates, and packet workloads.

We employ a hierarchical-based Deep Reinforcement Learning (DRL) approach to simultaneously evaluate and optimize the freshness of information and TAT. The solution decouples the original problem into two sub-problems, each managed by a distinct Markov Decision Process (MDP). This approach allows for the independent optimization of UAV trajectories and UAV-MEC offloading strategies, providing real-time, specific information about the environment to enhance decision-making.

Our findings demonstrate that decoupling the problem and using a hierarchical-based DRL solution significantly improves performance. The hierarchical approach allows the agent to focus on learning optimal policies for trajectory and offloading decisions individually. Compared to the traditional, non-hierarchical DRL solution, it demonstrated excellent performance in unseen scenarios (i.e., those not included in the training process), highlighting its superior adaptability, stability and efficiency across various systems.

In conclusion, the hierarchical DRL approach provides a robust and stable solution for optimizing UAV trajectories and MEC offloading in next-generation wireless networks. This strategy enables efficient use of computational resources while minimizing AoL and TAT, particularly in dynamic and resource-constrained environments. The results underscore the importance of hierarchical learning for real-time, specific environmental information, enhancing decision-making and performance in UAV-aided MEC systems.

Acknowledgements

Firstly, I would like to express my gratitude to my supervisor, Dr. Erol-Kantarci, for her guidance, support, and encouragement throughout my research journey. Her insightful feedback and motivation have been instrumental in shaping this work, which has significantly enriched my academic experience.

I would also like to extend my heartfelt thanks to my friends and colleagues at the Networked Systems and Communications Research Lab (NETCORE), especially to Dr. Termehchi, Dr. Bao, and Dr. Pamuklu, for their wise feedback and suggestions on this work.

Furthermore, I would like to express my appreciation to the University of Ottawa for offering an environment conducive to my research and personal growth. I am especially grateful to all the professors who have guided and supported me throughout these years.

Dedication

First, to my dear parents, Flaviano and Lucia, for all the support and for showing me the value of hard work. To my dear husband, Rafael, for his endless love, patience and encouragement, who has been my anchor throughout this journey. Without their support, this work would not have been possible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

5G Fifth Generation 6, 7, 9, 11, 12

A2C Advantage in Actor-Critic 24

AI Artificial Intelligence 1, 2, 8, 25

AoI Age of Information 12, 22–25, 27–29

AoL Age of Loop ii, 12, 13, 30, 32, 41–44, 47–49, 52, 53, 55, 56, 58–70

AWGN Additive White Gaussian Noise 37, 55

BS Base Station 8, 23, 25, 28

CH Cluster Head 33, 36, 41

CPU Central Processing Unit 3, 4, 10–12, 24, 26, 28, 29, 34, 36, 39, 42, 54, 56, 60, 62, 66, 67, 70

DDQN Double Deep Q-Network 18

DNN Deep Neural Network 17, 21, 49

DP Dynamic Programming 16

DPPG Deep Deterministic Policy Gradient 25, 27, 29

DQL Deep Q-Learning 24, 28, 29

DQN Deep Q-Network 17, 18, 29

DRL Deep Reinforcement Learning ii, 4, 6, 17, 20, 28–30, 44–46, 69, 70

eMBB Enhanced Mobile Broadband 7

FDMA Frequency Division Multiple Access 37

FIFO First In, First Out 11, 39

GAE Generalized Advantage Estimation 19

HPPO Hierarchical Proximal Policy Optimization 48, 49, 53

IMT International Mobile Telecommunications 6, 7

IoT Internet of Things 1, 2, 6–11, 22, 30, 34, 70

IoTD Internet of Things Device 2, 3, 8, 10, 13, 23–25, 32–34, 36–39, 41–44, 46–49, 52, 54–56, 58, 62, 67

IRS Intelligent Reflecting Surface 29, 30

ITU International Telecommunication Union 6, 7

ITU-R ITU Radiocommunication Sector 7

KPI Key Performance Indicator 24, 25

LIFO Last In, First Out 11

LoS Line-of-Sight 10, 36, 37

MDP Markov Decision Process ii, 4, 14, 23, 44, 49, 69

MEC Multi-Access Edge Computing ii, 2–4, 6, 9, 10, 12, 24–30, 32–34, 36–40, 42–44, 46, 49, 53–57, 59, 60, 62–71

ML Machine Learning ii, 3, 4, 14, 57

mMTC Massive Machine Type Communications 7

MSE Mean Squared Error 21

NLoS Non-LoS 36, 37

PG Policy Gradient 19, 20

PPO Proximal Policy Optimization 18, 20, 24, 25, 29, 45, 46, 48, 49, 53, 69, 70

RL Reinforcement Learning 4, 6, 14, 15, 17, 18, 56

TAT Turnaround Time ii, 3, 4, 11, 28, 30, 32, 38–43, 53, 55, 56, 60–62, 64–70

TD Temporal-Difference 17, 19

TD3 Twin Delayed DDPG 24, 25

TRPO Trust Region Policy Optimization 19, 20

UAV Unmanned Aerial Vehicle ii, 1–4, 6, 8, 10, 12, 22–25, 27–30, 32–34, 36–39, 41–44, 46–49, 53–57, 59, 60, 62, 63, 66–71

URLLC Ultra Reliable Low Latency Communications 7, 9, 12

V2X Vehicle to Everything 7, 8, 25, 30

VM Virtual Machine 28

Chapter 1

Introduction

1.1 Motivation

Agriculture is an essential component of human society since it is the source of the food consumed and the origin of primary components utilized by industries. It is one of the main pillars of the global economy that supports human life and nutrition. However, the present requirements for sustainability, quality, and profit while the world still faces the challenge of expanding the food supply have accelerated the need for intensive agricultural methods.

With the advent of Internet of Things (IoT), sensors and actuators have been collaboratively used to achieve resource management, precision farming, and crop and livestock monitoring [1]. These applications demand the analysis of complex structured and unstructured data in the form of text, images and video. In this context, using Artificial Intelligence (AI) for data analysis in agriculture applications has led to various improvements in the field. For instance, AI-powered crop monitoring systems were able to reduce the number of pesticides used by 80% [2], whereas it is reported that an average farm using an IoT system can reduce water consumption by 8% on average [3], lowering costs and the impact on the environment.

UAVs can also be used for direct monitoring and detecting crop diseases, fires and pests with the use of high-resolution cameras [4]. In addition, UAVs facilitate precise and safe pesticide application [5]. Within this framework, agricultural robotics [6] refers to AI-powered robots and drones utilized for planting, harvesting, spraying and packing tasks. Moreover, with AI techniques, it is possible to do predictive analysis [7] to examine

data to forecast crop yields, climate and market demand, which allows farmers to make accurate decisions about planting. These applications of AI in agriculture aim to optimize crop yields, reduce waste, and improve resource efficiency, harvesting, and pricing and are summarized in Figure 1.1.

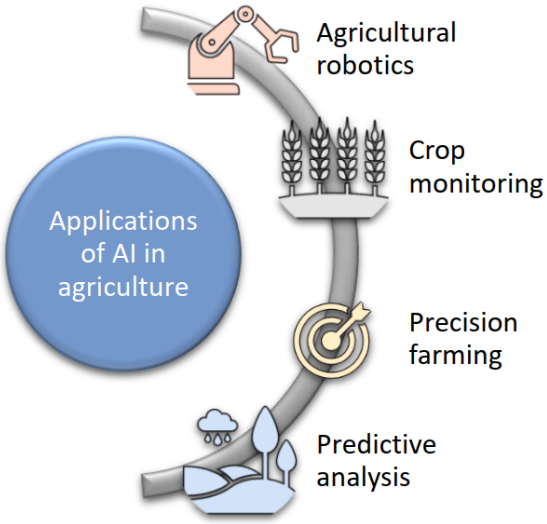


Figure 1.1: Common applications of AI in agriculture.

However, these applications might require highly complex data and computational-intensive tasks, which impose challenges in data collection, transmission, and processing, especially considering delay-constrained scenarios such as fire detection. Internet of Things Devices (IoTDs), such as sensors and actuators, have low processing power and limited transmission range, making it challenging to execute real-time applications effectively and connect to robust computational units [8]. To address these limitations, advanced network architectures are needed to support timely and efficient data processing.

The MEC is a type of such network architecture that brings cloud capabilities to the edge of the network, closer to the served devices, aiming to reduce communication latency. UAV-aided MEC systems [9] leverage UAVs with internal computing capability and facilitated connectivity with the edge of the network, supported by a powerful MEC server. They are a promising approach to deal with the challenges that arise in IoT networks. In this setup, sensors collect data that must be processed to make decisions for actuators. UAVs have high mobility, which allows them to get close to sensors and collect the sensing

data. They serve dual roles as processing units and relay to the MEC server. Additionally, the UAVs are responsible for the command downlink communication for the actuator. Hence, the UAVs facilitate the communication of ground IoTDs, while the MEC server offers additional computational power to the system.

However, there are limitations. UAVs have restricted processing capability and battery life. The MEC server may handle various processing tasks (not only from the UAV system), making it offer variable computational and storage resources over time. In addition, there is an extra communication delay when offloading the computational tasks to the MEC server, which can increase the response latency in time-sensitive applications. Finally, the UAVs path needs to be designed aiming for continuous and fair data collection among IoTDs. Therefore, the UAVs trajectories and task offloading strategy between UAVs and MEC server are essential variables to achieve optimal behaviour in this system.

1.2 Thesis Contribution

In this work, we optimize the freshness of information and computational resource usage in terms of average Central Processing Unit (CPU) TAT by designing a UAV trajectory for data collection and a UAV-MEC offloading strategy. We consider a closed-loop communication system where IoTDs, each representing a combined sensor and actuator node, collect environmental information in the form of data packets. These packets are collected by a UAV, forming one task for each device. Each task is then processed either locally by the UAV or offloaded for the MEC server. The resulting command must be returned to the origin IoTD. We assume that the MEC server has a faster CPU than the UAV. However, because the server handles various computational tasks, we aim to minimize its usage. More specifically, although the services for the UAV are given the highest priority on the MEC server, we strive to ensure that the server's CPU and storage resources remain available for other services as well. Moreover, it is essential to minimize the computational burden of the UAV in order to conserve its energy.

In our system, transmitting packets to the MEC server may decrease the freshness of information due to the transmission delay but reduce computational resource usage. In contrast, processing packets on the UAV avoids increased delays but might increase the computational TAT. In addition, the solution for UAV trajectory and UAV-MEC offloading should be adaptable regarding dynamic network conditions, variable computational resources at MEC and random data generation, offering stable results. Therefore, we propose to resolve it with a hierarchical-based ML algorithm, where the trajectory's definition will help to determine the task offloading strategy.

We summarize the main contributions of this work as follows. Firstly, we simulate a UAV-aided MEC system with smart farm applications, considering queuing, transmission and processing delays. Secondly, we formulate a joint optimization problem for information freshness and CPU TAT by designing a UAV trajectory and UAV-MEC offloading strategies. Thirdly, we present the problem formulation using a MDP and apply an on-policy DRL solution, highlighting its disadvantages in the given scenario. Fourthly, we reformulate the problem into two MDPs with a hierarchical dependency and propose a hierarchical-based DRL approach. Finally, we analyze the trade-off between information freshness and CPU scheduling time in the system, comparing the two proposed DRL solutions with heuristic baselines.

1.3 Thesis Organization

Chapter 2 provides background information about the communication model used in the work, including more details about the freshness of information, CPU turnaround time, and ML strategies, focusing on Reinforcement Learning (RL) and DRL. Chapter 3 includes a comprehensive literature review of freshness and computational-aware offloading and UAV trajectory optimization. Chapter 4 introduces the methodology used in this work and details the system model and proposed solution. The results and analysis are presented in Chapter 5. Finally, this work is concluded in Chapter 6.

List of Publications

[1] L. Nobrega, A. Termehchi, T. Bao, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "AoI-aware trajectory planning for smart agriculture using proximal policy optimization," in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024.

[2] L. Nobrega, A. Termehchi, T. Bao, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Hierarchical deep reinforcement learning with information freshness in smart agriculture applications," in *2024 IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2024.

Chapter 2

Background Theory

In this chapter, we are going to explore the integration of UAV-aided MEC systems within Fifth Generation (5G) and beyond networks, focusing on their application in IoT systems. We will delve into the challenges associated with these systems, particularly concerning the freshness of information and computational scheduling. Moreover, we provide an overview of relevant concepts regarding RL and DRL that will be used in the following chapters of this work.

2.1 UAV-aided MEC systems

2.1.1 5G and beyond

Evolving from inconsistent and analog voice communications to digital, fast, reliable, and massive data transmission, wireless communications mature to respond to both societal and industrial demands. The introduction of the 5G technology holds promise for connecting a more extensive array of devices to the Internet, facilitating faster data transmission, and handling substantial data loads with minimal latency. Because of that, new applications of virtual and augmented reality, smart cities, and video monitoring, to cite a few, were limited or even impossible on previous mobile network generations, but they are currently being enabled and further developed.

Formally defined by the International Telecommunication Union (ITU) agency, the recommendations for 5G were outlined in 2015 on the International Mobile Telecommunications (IMT)-2020 [10] and followed a multi-stakeholder approach. It means that

the specifications and standards of 5G technology were developed collaboratively with input from various stakeholders, including industry experts, policymakers, researchers, and telecommunications companies, ensuring a comprehensive and inclusive framework for its implementation and advancement. ITU was also responsible for the standardization of analogue cellular (1G), digital cellular (2G), IMT-2000 (3G) and IMT-Advanced (4G), and are currently defining the requirements for the Sixth Generation, with the IMT-2030 [11].

As the first step towards standardizing the new mobile communication generation, communications trends were observed. Specifically, the demands for higher data rates, ultra-low latency, high reliability, and an increased number of connected devices were identified. With this framework, the 5G key performance requirements are aggregated into three groups by the ITU Radiocommunication Sector (ITU-R), each one detailed below:

- **Enhanced Mobile Broadband (eMBB):** It considers the increased requirement for wireless connectivity, enabling stable and reliable connections with high data rates for both cell and edge users. Therefore, it allows new mobile uses such as high-definition video streaming [12] and augmented and virtual reality [13].
- **Ultra Reliable Low Latency Communications (URLLC):** It is formulated to fulfill strict reliability, latency and availability requirements, enabling varied applications that demand speedy and reliable data transfer, such as wireless industrial operations [14], remote medical procedures [15] and autonomous vehicles [16].
- **Massive Machine Type Communications (mMTC):** It aims to promote a broad collection of devices with reduced expenses, enhanced coverage, and prolonged battery longevity. In short, this usage scenario entails a substantial number of interconnected devices, typically transmitting data that is relatively insensitive to delays. The devices must be economical and possess prolonged battery durability. This paves the way for deploying interconnected sensors for various applications, ranging from smart agriculture [17] to urban traffic control [18].

2.1.2 IoT Networks

In IMT-2020, ITU observed the trend of more and more diverse devices (or "things") being connected and anticipated some of its uses, which over the following years have been studied and further developed. These applications include smart cities, agriculture, healthcare and Vehicle to Everything (V2X). Fundamentally, an IoT network comprises

web-enabled smart devices with embedded systems, commonly referred to as IoTDs, and might include sensors, actuators, and communication hardware, allowing them to collect, transmit, and respond to environmental data, being capable of responding to different tasks, including intelligent decision-making [19].

IoTDs and wireless IoT systems are crucial for the upcoming smart world, driving the emergence of smart applications and enhancing the potential of automated systems. This is possible thanks to the alignment of such networks with fast and reliable wireless communication and new AI methods for data analysis and decision-making. However, IoT networks and IoTDs involve specific challenges [20, 21, 22]. Some of the challenges are highlighted as follows:

- **Communication among a massive number of devices:** In complex systems such as those found in V2X or smart agriculture, numerous sensors and actuators gather crucial environmental data and respond accordingly. Given that a small space can host a multitude of these systems needing to communicate with each other, it is vital for the communication network to support a high volume of devices exchanging data simultaneously.
- **Low-cost devices:** Given that a massive number of devices are going to be used, it is straightforward to think that they need to be of reduced cost and uncomplicated maintenance to permit easier application and use. Therefore, some concessions related to, for example, low-cost antennas and computational processors, need to be made.
- **Battery limitations:** Contrary to other communication devices, such as cellphones, IoTDs have different battery needs. Because they might be deployed in remote areas, battery recharging is not as easy as in other devices. Hence, some IoTD target batteries that last 10 to 15 years [23]. Moreover, IoTDs such as UAVs, although they can be recharged with a higher frequency, spend much of their energy with flight and communication, reducing their use capability (i.e., they can be used only for a short period of time before discharging). Therefore, ensuring that the IoTDs has ways to reduce its energy consumption, such as transmitting only a little information at lower power or limiting its processing capability is crucial.
- **Communication coverage:** Some applications require that IoTDs be in locations with difficult access. For instance, smart farms may require sensors far from any Base Station (BS), i.e., outside its coverage area. Other applications need sensors inside buildings, basements or concrete walls, i.e., being kept indoors or in a place with harder communication with outside BS.

- **Big data applications:** In this framework, a considerable amount of data might be necessary for data analysis and decision-making. In this case, a reliable and fast communication model is required to avoid losing important information during transmission.

We can think of IoT platforms in open-loop communications, where the data is collected and transmitted to different devices or to a server and will be available for real-time monitoring or stored for future data analysis. In addition, closed-loop systems are also possible, where the data collected is processed to generate a feedback command to an actuator, which will affect the environment accordingly. For example, we can consider an automatic system for precision farming that works based on information (collected by sensors) about the humidity and crop growth rate, that are processed, and its result defines whether the irrigation system should be enabled [24]. This type of system is usually much more sensitive to delays in data transmission, computation and control resources [25].

In both cases, we can consider a cloud data mechanism as depicted in Figure 2.1 and described in [26], where the collected information is transmitted via a 5G network to the cloud or directly to other equipment for real-time monitoring. The cloud provides storage and processing capabilities, and if the specific application requires a feedback command, it will be sent to the IoT actuator.

2.1.3 Multi-Access Edge Computing

MEC is a key technology in enhancing the capabilities and performance of 5G networks, helping to meet its challenging requirements of throughput, latency and scalability [27]. It provides computational and data storage closer to the user, bringing these physical resources (in this work it is referred to as a MEC server) to the edge of the network. It saves data from travelling thousands of kilometres when it is destined for the same place: the content is sent, received, and stored close to each other. This geographical proximity makes it possible to have a very small latency, enabling time-constrained applications.

For instance, the URLLC 5G requirement of latency below 5 milliseconds, which is not viable with the conventional network architecture, became possible with MEC system [28]. Moreover, it allows transmitting computationally intensive tasks from computational or battery-constrained devices to more robust servers without increasing the latency excessively [29].

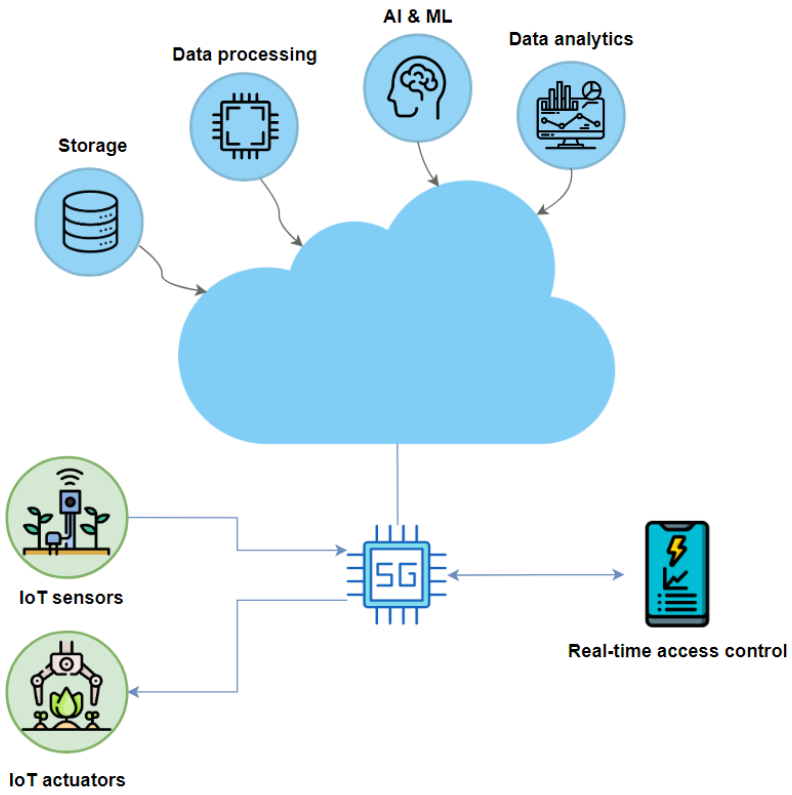


Figure 2.1: IoT data cloud mechanism [26].

2.1.4 UAV-Aided MEC Networks

The distance between IoTDs and MEC servers restricts their direct communication and exchange of information due to factors such as a lower probability of Line-of-Sight (LoS), high path losses, and limited transmission range of the devices. To address this issue, UAVs are employed to gather data from IoTDs and subsequently offload it to the bordering servers, while also allowing local data processing with its internal CPUs [30].

Indeed, UAVs can be employed in multiple applications beyond the interaction with ground IoT sensors and actuators. For instance, they can be equipped with cameras for monitoring or be used for goods transportation and communication support in post-disaster operations [31]. This is due to their ease of deployment and reprogramming during runtime, and their high degree of autonomy during flight [32].

In conclusion, combining UAVs flexibility and the additional use of powerful edge servers

offers a promising avenue for efficient data management in the context of 5G and beyond networks. Additionally, they present an innovative architecture to enable the use of IoT networks in the evolving landscape of smart agriculture.

2.1.5 Challenges

Computational Scheduling

When a computing process is admitted to be processed by the CPU, it goes to the ready queue, indicating its status is "Ready." Once the CPU becomes available, it serves a computing process on that queue, according to the current queuing discipline (First In, First Out (FIFO), Last In, First Out (LIFO), etc.) [33]. The time that the computing task spends on the ready queue is the waiting time. When the task starts to be processed, its status changes to "Run". The time necessary to process a task is the burst time, and after the task is processed, its status is updated to "Complete". The time at which the task arrives at the CPU is called the arrival time, and the time at which it finishes processing is the completion time. The delay between these two times is known as the TAT. In other words, the computational TAT is the total duration from the moment a process enters the CPU to the point it finishes execution. This definition is summarized in Figure 2.2. We emphasize that when the sole term TAT is used in this work, we are referring to computational or CPU TAT.

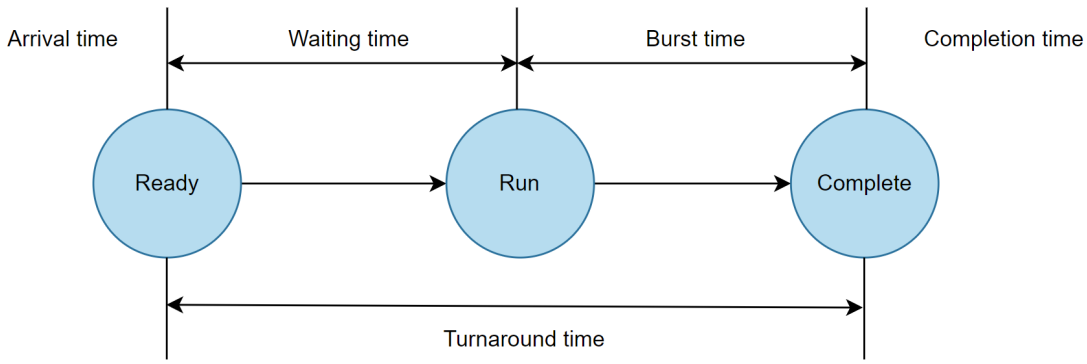


Figure 2.2: CPU turnaround time.

There are two basic classes of priority policies: the preemptive resume policy and the non-preemptive priority policy. The preemptive-resume policy allows a higher-priority

job to interrupt and take over the CPU from a lower-priority job. Conversely, the non-preemptive priority policy ensures that once a job begins processing, it will not be interrupted until it is completed, meaning a higher-priority job cannot interrupt an ongoing process.

In the context of UAV-aided MEC systems, we need to consider that the UAVs have limited batteries, and MEC servers play a crucial role in efficient data processing with powerful CPUs [34]. However, we must consider cases where the UAV cannot transmit all processing to the MEC server, and there is a need to define whether and which computational tasks to be offloaded [35]. For instance, considering a smart farm system where the MEC server needs to handle a massive number of processing tasks and the computational resources are limited, it is necessary to restrict the amount of resources that each task will use.

Freshness of Information

In the system considered in this work, maintaining up-to-date information is critical since obsolete data becomes less valuable over time for smart decisions [36]. For instance, the speed of detecting issues such as fire and pests is crucial as rapid detection facilitates prompt treatment [8]. To quantify this temporal aspect, we employ the Age of Information (AoI) metric, which represents the time elapsed since the generation of the last received data packet [37] and is used to quantify the freshness of information, where a lower AoI means a fresher information. Information freshness is crucial for 5G and beyond communication, particularly for URLLC applications, where latency-only design is insufficient [38].

In UAV-aided systems, the definition of the trajectory of the UAV used for data collection is essential for better freshness of information [39, 40], since the data collection can be optimized to cover the served area efficiently and according to the data generation. The offloading decision when the UAV serves as a relay for a MEC server is also relevant [41, 42] due to the impact it has on data transmission and processing.

Nevertheless, smart farms with integrated sensors and actuators, as proposed in [24], necessitate the consideration of a closed-loop communication, where a sensor collects data, offloads it to a processor, and the generated command returns to the actuator, which is often attached to the sensor. The AoL, an alternative to AoI introduced in [43], is pertinent to assessing data freshness in closed-loop scenarios where sensors and actuators are co-located. The work [43] considers two definitions for AoL: the downlink AoL, which considers the generation time of the most recent *command*, and the uplink AoL, which

considers the generation time of the *information* that generated the latest command. In this work, we assess only the uplink AoL.

For an instinctive explanation of uplink AoL, we have Figure 2.3. We assume that some information was generated at time t'_1. After a while, the information was collected and processed, and the resulting command was received at the corresponding IoTD at time $\hat{t}_1$. Up to this point, the uplink AoL increased with time, and when the command is received the uplink AoL drops to the difference between the current time ($\hat{t}_1$) and the time of generation of the data that originated the command (t'_1). After this point, the uplink AoL keeps increasing with time. We continue the example illustrated in Figure 2.3 with another command being created at t'_2 and the originated command returned at $\hat{t}_2$ shortly after. In this case, the updated uplink AoL is $\hat{t}_2 - t'_2$. Therefore, we can define the uplink AoL as the time difference between the current time and the latest received command generated by the freshest data packet.

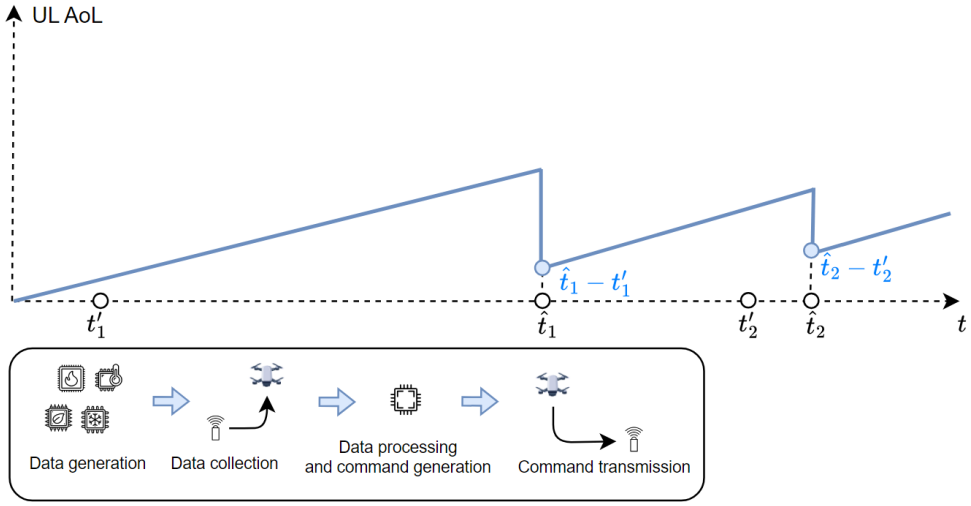


Figure 2.3: Uplink AoL behaviour over time (based on [43]).

2.2 Reinforcement Learning

2.2.1 Overview

The most intuitive way of learning is based on a try-and-error strategy. The idea of performing actions and receiving feedback (positive or negative) that will assist with the

next decisions is intuitive and straightforward. In RL, an agent executes an action based on the environment's current state. As depicted in Figure 2.4, for any time t, the environment will be in the state S_t, and the agent will define an action A_t accordingly. That action will modify the environment, making it go to state S_{t+1} and generate a reward R_{t+1} (i.e., a feedback signal), which represents how well the agent performed. By optimizing the rewards, the agent will be capable of performing efficient actions, according to their specific objective.

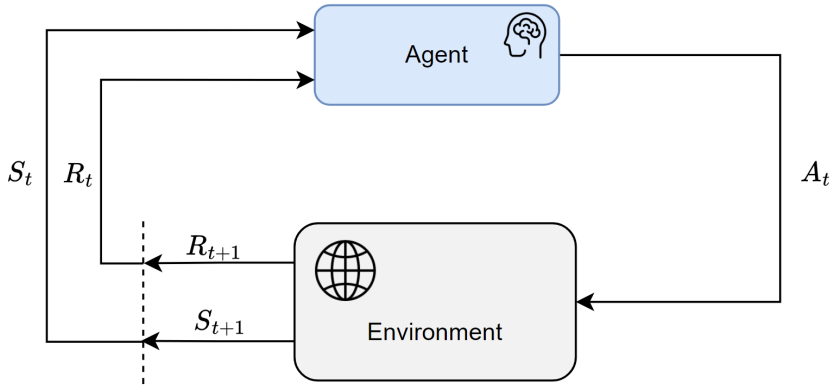


Figure 2.4: Agent-environment interaction in reinforcement learning [44].

In next-generation wireless networks, the use of RL techniques will be crucial, especially considering the increasing network complexity, strict requirements (delay, spectrum efficiency, reliability, etc.) to meet, and a massive number of parameters and subsystems to optimize. It is also worth mentioning that, although supervised ML strategies have been used in interesting wireless applications, such as channel estimation and beamforming optimization, showing positive outcomes, these methods require a substantial amount of labelled training and testing data [45], which is difficult to obtain in real-life scenarios.

MDPs provide a mathematical framework and enable the definition of an environment for RL. Its main idea relies on Markov's property where, given the present, the future is not dependent on the past. In other words, if the current state of an environment is available, the previous states can be omitted. A MDP is a tuple $\{\mathcal{S}, \mathcal{A}, P, R, \gamma\}$, representing the state space, action space, dynamics of the process, reward space and a discount factor, respectively.

A trajectory is a sequence of states and actions in the environment. Considering a trajectory τ of length T, we can define:

$$\tau = (S_0, A_0, S_1, A_1, \ldots, S_{T-1}, A_{T-1}, S_T). \tag{2.1}$$

A stochastic policy symbolizes the probability of the agent defining the action $a \in \mathcal{A}$ given the state $s \in \mathcal{S}$. It can be parameterized with the parameter θ.

$$\pi_\theta(a|s) = \Pr[A_t = a|S_t = s, \theta_t = \theta]. \tag{2.2}$$

The return represents the cumulative reward from time step t. It can be defined in the finite horizon if only a specific trajectory is considered, or in the infinite horizon if we consider all the rewards the agent obtains. This last case includes the discount rate $0 \leq \gamma \leq 1$, which avoids infinite returns in cyclic Markov processes by decreasing the relevance of delayed rewards compared to immediate rewards. Hence, the return over a trajectory τ is defined as below:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^t R_{t+k+1}. \tag{2.3}$$

2.2.2 Value Functions and Bellman Equations

The agent aims to maximize the expected return over the trajectory τ. Based on this, we define the value function to represent the expected return starting from a state s and always following the policy π.

$$V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s], \forall s \in \mathcal{S}. \tag{2.4}$$

It is an effective way to determine whether an action was efficient before a delayed reward comes. Likewise, the action-value function (or Q-function) represents the expected return starting from a state s, taking action a and following a policy π.

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a], \forall s \in \mathcal{S}, \forall a \in \mathcal{A}. \tag{2.5}$$

As aforementioned, the ultimate objective of a RL task is to maximize the total expected return. Hence, the optimal value and action-value functions are defined when they achieve the maximum according to the optimal policy, as below:

$$V^*(s) = \max_\pi V^\pi(s),\ \forall s \in \mathcal{S}, \tag{2.6}$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a),\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}. \tag{2.7}$$

The optimal value functions obey an identity known as the Bellman equation. It says that the value functions can be decomposed in the immediate reward, R_{t+1}, and the discounted value of the successor value functions. In other words, if the optimal action-value function is known for all possible actions a', we can achieve the optimal action-value (Q-value) by selecting the action a' that will maximize the expected value of $R_{t+1} + \gamma Q^*(S_{t+1}, a')$, or:

$$Q^*(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a]. \tag{2.8}$$

Given the optimal action-value function, we can find the optimal policy as:

$$\pi^* = \arg\max_{a \in \mathcal{A}} Q^*(s,a). \tag{2.9}$$

2.2.3 Common Approaches

To resolve 2.8, the optimal action-value function can only be computed when the system's dynamics are known. The idea is to use Dynamic Programming (DP) to evaluate the value functions and improve the policy iteratively [46]. However, in real-world applications, the model is not fully known.

Another strategy is to estimate the value of Q^π from experience [44]. For instance, we can define an agent following policy π that will visit a specific state multiple times and, when the number of times that state is experienced approaches infinity, we can have a proper approximation for the state-value function. Similarly, if the agent keeps separate averages for each action in each state, we obtain an approximation of the action-value function. The policy is improved greedily in order to maximize the Q-function. The methods that use this strategy of random sampling and average returns are the *Monte Carlo methods*. However, in scenarios with an excessive number of different states, it is not feasible.

Therefore, multiple RL methods learn from incomplete experiences and do not need to explore the environment in a full trajectory to improve their knowledge. Temporal-Difference (TD) learning learns the value function directly based on Bellman equation, with estimates of $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$ [47], as below:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)), \tag{2.10}$$

where α is the learning rate. SARSA [44] is an on-policy TD method, where the update of the Q-value is done based on the sequence $\ldots, S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \ldots$. Specifically, based on one state S_t, the action A_t is chosen to maximize $Q(S_t, A_t)$, the reward R_{t+1} and new state S_{t+1} are obtained, the new action A_{t+1} is is chosen following the current policy, and the action-value function is updated according to 2.10.

Q-learning [48] is an off-policy TD strategy, where the main difference to SARSA is that A_{t+1} is not obtained following the current policy. Instead, the value function is updated based on the maximum value, where A_{t+1} is not computed and the current policy is not relevant. This algorithm is defined by:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q(S_{t+1}, a) - Q(S_t, A_t)). \tag{2.11}$$

2.2.4 Deep Reinforcement Learning

In the aforementioned RL techniques, the value functions or policies can be stored in an array, or table. This is why they are called *tabular methods*. However, the table increases exponentially with the number of states and actions, hence making its use impractical for real-life applications. In DRL, Deep Neural Networks (DNNs) act as a function approximation for the value functions or policy. Its importance gains significance when we consider practical decision-making problems, where there is a high dimensional state-action space.

A well-known DRL algorithm is Deep Q-Network (DQN) [49], which approximates the Q-values for each action-state pair using a target DNN, an online DNN and an experience replay buffer. The online network interacts with the environment and collects more experience, whereas the target network is updated less frequently, avoiding short-term oscillations. The experience buffer allows the agent to learn from past experiences, rather than only learning from the most recent ones. Over time, with more experience, the DNNs become more accurate in predicting the Q-value of each action. Several extensions of DQN

were proposed in the past years, such as duelling DQN [50] and Double Deep Q-Network (DDQN) [51].

These methods are known as value-based methods (i.e., off-policy), in which the approximated value function is learned from taking different actions and the policy π is achieved by acting greedily to maximize the value function. This work focuses on policy-based learning (i.e., on-policy) methods, where the approximate value function is learned directly from actions taken using a current policy π. Therefore, we will give more details about it in the section 2.3.

2.3 Policy Gradient Algorithms

The RL method used in this thesis is based on Proximal Policy Optimization (PPO), a policy gradient method that optimizes policies by constraining updates to ensure stable and reliable performance. It is an on-policy method, meaning it does not use a replay buffer: once a batch of experience has been used, it is discarded. The general idea of policy gradient algorithms leading to the development of PPO is as follows.

2.3.1 Vanilla Policy Gradient

In policy-based algorithms, the agent is responsible for selecting an action from the current state based on a parameterized policy. As expressed in 2.2, the policy π_θ represents the probability, in time t, of the action a being selected given that the state is s, and the parameters of the current policy are θ.

As previously discussed, to find the optimal value of θ, θ^*, we need to maximize the expected return. If we consider the rewards computed from a trajectory generated under the policy π_θ, we define the expected return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t]\,. \tag{2.12}$$

We can employ *gradient ascent* to adjust θ in the direction indicated by the gradient $\nabla_\theta J(\theta)$. This helps us to determine the optimal θ for π_θ, which yields the highest return, as below:

$$\theta_{\text{new}} = \theta_{\text{old}} + \alpha \nabla_\theta J(\theta). \tag{2.13}$$

Using the Policy Gradient (PG) theorem, introduced in [52], we are able to estimate the gradient in 2.13 over the trajectory τ following the stochastic policy π_θ. As detailed in [53], there are multiple expressions for the estimation of the gradient. One common way to denote it is:

$$\nabla_\theta J(\theta) = \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(A_t|S_t)\hat{A}_t], \tag{2.14}$$

where $\hat{A}_t$ corresponds to the *advantage function* at time step t, based on trajectory τ, usually defined as $A(s,a) = Q(s,a) - V(s)$. The Q-value is calculated at the end of the trajectory; therefore, there are no estimates. The value function is represented as a critic neural network (that is frequently updated) and needs to be estimated. The idea is to have a value that indicates how well the performed actions were compared to what would normally happen in the agent's initial state s.

A vanilla policy gradient method can be implemented with automatic differentiation software using Equation 2.14 as an objective function. In this case, a positive advantage means that the actions performed better than expected; hence, the gradient $\nabla_\theta J(\theta)$ will be positive, and the probability of taking the same actions again increases. On the other hand, a negative advantage indicates that the actions taken were worse than expected; therefore, the gradient is negative, decreasing the probability of taking these actions again. With this behaviour, the policy becomes successively less random over the course of training. However, given that the estimation of the value function (and hence the advantage function) can be biased, Generalized Advantage Estimation (GAE) was proposed in [53] to improve the stability and efficiency of the estimation and of the PG methods. This is achieved by calculating a weighted sum of TD errors over multiple time steps.

2.3.2 Trust Region Policy Optimization

In [54], Trust Region Policy Optimization (TRPO) method was presented, motivated to improve the learning efficiency of PG methods by including a constraint in the objective function to avoid policy updates that move too far from the current policy. This is done using the Kullback-Leibler divergence, measuring the difference between the probability distribution of the updated and previous policies. The objective function in TRPO can be defined as:

$$\max_{\theta} \hat{\mathbb{E}}_t \left[r(\theta) \hat{A}_t \right] \tag{2.15}$$
$$\text{s.t. } \hat{\mathbb{E}}_t[\text{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta)] \leq \delta,$$

where $r(\theta) = \frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}$ is the probability ratio between the old and new policies and KL is the Kullback-Leibler divergence.

2.3.3 Proximal Policy Optimization

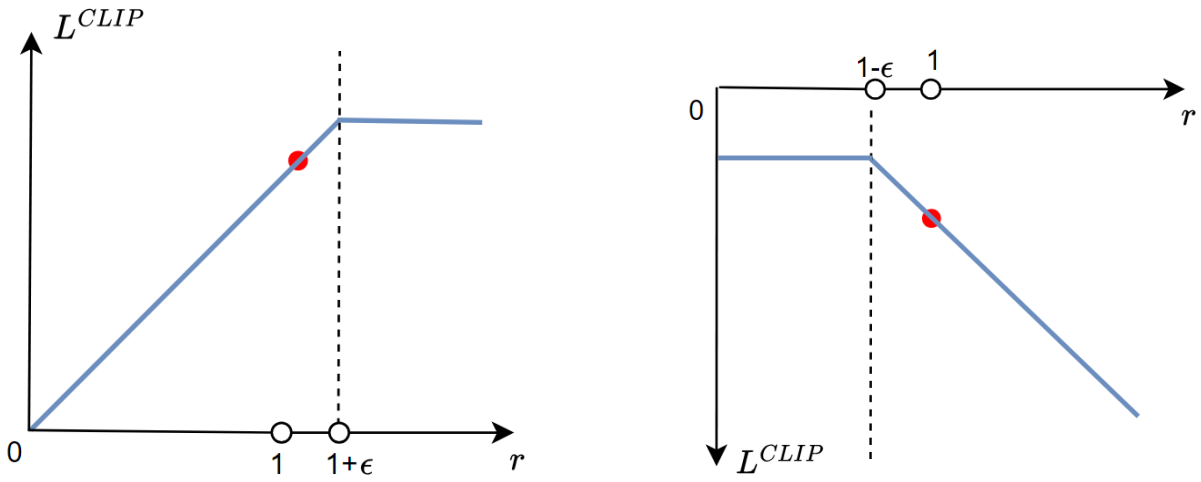
Although TRPO improved the overall performance of PG algorithms by constraining the update of the policies, the computation of the Kullback-Leibler divergence adds an overhead to the optimization process. Therefore, PPO was introduced in Schulman et al. [55] to simplify the policy update constraint by using a clipped surrogate objective and became a state-of-the-art DRL strategy. The main objective proposed in [55] was:

$$L_t^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min[r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] \right], \tag{2.16}$$

where ϵ is a hyperparameter. The goal of the clipping function in PPO is to prevent the policy update from making $r(\theta)$ outside the range of $[1-\epsilon, 1+\epsilon]$, losing the motivation for updating the policy excessively. Figure 2.5 illustrates the intuition behind this clip function. In Figure 2.5a, we analyze the case where $\hat{A}_t > 0$, i.e., the action performed better than expected, and the objective function will increase if the action becomes more likely. If $r(\theta) > 1$, the new policy already makes the action more probable than the old policy. Hence, if $r(\theta) > 1 + \epsilon$, the objective function is clipped to avoid benefiting the policy moving too drastically. In Figure 2.5b, the case for $\hat{A}_t < 0$ is depicted, where the action was worse than expected, and its probability should be decreased. In this case, if $r(\theta) < 1$, the action is already less probable, and if $r(\theta) < 1 - \epsilon$, the objective function is again clipped to avoid making excessive updates.

Thus, the objective in PPO does almost the same as TRPO objective, forcing the policy updates to be conservative if they move very far from the current policy. The difference in PPO is that it is done with a simple objective function, where the computation of the Kullback-Leiber divergence is not necessary.

The authors in [55] suggest using an entropy bonus to encourage exploration. Therefore, we can define a loss function in PPO as:



(a) Positive advantage ($A > 0$). (b) Negative advantage ($A < 0$).

Figure 2.5: Surrogate objective L^{CLIP} as a function of probability ratio [55].

$$L_t^{CLIP+S}(\theta) = \hat{\mathbb{E}}_t \left[\min[r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t] + cS[\pi_\theta(S_t)] \right], \tag{2.17}$$

where c is a hyperparameter coefficient for the entropy and $S[\pi_\theta(S_t)]$ is the entropy of the current policy. A DNN (actor network) represents the current policy and selects the actions. Another DNN (critic network) evaluates the actions taken by the actor and guides the policy updates. It estimates the value function ($V_\theta(S_t)$) and compares with the target value V_t^{targ}. The target value is computed using the observed rewards in a trajectory. The loss function for the critic network is the Mean Squared Error (MSE) between the estimated and the target value. For a trajectory τ with T steps:

$$L_t^{VF} = \frac{1}{T} \sum_{t=0}^{T-1} (V_\theta(S_t) - V_t^{\text{targ}})^2. \tag{2.18}$$

Chapter 3

Literature Review

In this chapter, we delve into the existing literature related to UAV freshness of information and computational scheduling optimization problems, focusing on UAV-assisted scenarios. We provide an overview of these studies and how the environment and solutions developed over time. We also describe what is missing in the literature and how this work expands the topic.

3.1 Freshness of Information

In precision farming scenarios, the freshness of information is relevant for effective decision-making. Obsolete information might result in imprecise judgments of the state of the environment, which could potentially lead to inaccurate decisions about targeted interventions (such as pest and disease detection) or resource coordination. As mentioned in previous chapters, although the freshness of information might be related to the transmission and processing delay, it also considers the timing of data generation, which justifies the synchronization between information creation and collection.

3.1.1 Traditional approaches

In UAV-assisted IoT network systems, the relationship between the UAV trajectory and freshness of information is well known in the literature.

Liu et al. analyzed the AoI and defined a relation between UAV trajectory and freshness of information when the UAV is used for data collection of the information generated by

ground nodes [56]. The authors first used a dynamic programming approach to find the optimal trajectory of one UAV. However, because the complexity of the algorithm increases with the network size, a genetic algorithm-based solution was developed to search the near optimal trajectory.

In [57], the authors considered a similar scenario, where one UAV collects data from clustered IoTDs and goes back to a data center to offload the packets. However, because the UAV had to serve multiple IoTDs simultaneously, the problem was to not only design the UAV trajectory but also define a multiple-access strategy for data gathering. The authors used a dynamic programming strategy to achieve minimal AoI and showed that that the AoI of each IoTD is related to the number of clusters, the UAV trajectory and the sensors uploading sequence.

In Hu et al. [58], two sub-problems are addressed for AoI minimization: joint energy transfer and data collection time allocation, and UAV trajectory planning. These problems can be decoupled because the time allocation at each sensor node is independent of the UAV trajectory. The solution for the first sub-problem, joint energy transfer and data collection time allocation, is obtained considering the Karush-Kuhn-Tucker conditions. This solution is then used as input for the second sub-problem. The second sub-problem, UAV trajectory planning, is solved using dynamic programming and ant colony heuristic algorithms.

Finally, in [59], the freshness of information in a multiple UAV system was studied. In this case, UAVs were used for data collection produced by ground devices and as a relay for transmitting this data to a BS. The trajectory of the UAVs and the offloading strategy from them to the BS were defined for the UAVs's AoI minimization.

3.1.2 Reinforcement Learning-based Approaches

The evolution of the freshness of information is influenced by the varied traffic patterns of IoTDs. For instance, with a consistent sampling rate, the AoI of a soil moisture sensor changes periodically as it regularly updates the soil moisture levels, whereas the AoI of a motion or fire sensor is event-driven, randomly updating only when an activity is detected. Therefore, the UAV trajectory planning should accommodate the dynamic nature of these traffic generation patterns.

Building upon these considerations, Zhou et al. [60] explored a scenario where one UAV was employed for data collection, in a system where the ground sensor nodes had an unknown traffic generation pattern. As such, they formulated the problem as an MDP

and solved it through Deep Q-Learning (DQL), achieving an AoI-based trajectory planning strategy. The authors obtained higher freshness when compared to greedy trajectory scheduling, motivating further research involving solutions with reinforcement learning techniques.

In this context, Tong et al. in [61] evaluated the influence of sensors' sampling and queuing processes on data freshness. More specifically, ground sensor nodes store the collected information in their buffers and an old packet is dropped when a fresher one arrives. By optimizing the UAV path, the authors were able to minimize the average AoI of the sensors, the packet drop rate and UAV energy consumption, also using a DQL-based strategy.

In [39], a Sarsa-based algorithm was proposed for data gathering. The objective was to jointly minimize the average AoI of sensor nodes and energy consumption of UAVs, considering the battery capacity and collision avoidance constraints.

Additional works that focused on UAV trajectory optimization for data freshness include systems considering safety constraints for UAV positioning resolving it using a Twin Delayed DDPG (TD3)-based algorithm [62]; and in ultra-dense networks, concentrating on downlink direction [63]. The authors in [64] minimize AoI by optimizing the UAV location and the order of visits of hovering points using a transformer-based approach. In addition, wireless energy transmission has been jointly evaluated with AoI to define UAV trajectory. Specifically, Chi et al. [65] designed the UAV path based on not only the need for data gathering but also the necessity of hovering over recharging stations; and [40] assumed a hovering time that was enough to collect the packets and to recharge the ground IoTDs.

The works surveyed until this point proposed solutions for freshness based solely on UAV path planning for data collection. However, AoI and other similar freshness metrics can be further optimized with different strategies.

For instance, Akbari et al. [66] considered a network where UAVs assist in task collection produced by ground sensors and act as relays for other static UAVs and to a MEC server. Both UAVs and the MEC server can process the collected tasks. Their objective was to minimize AoI by defining the routing strategy between processing nodes, using a federated learning DQL-based solution to optimize the placement of network function instances and routing, thereby enhancing freshness and reducing the energy consumption of IoTDs in transmission and CPU processing in UAVs. Additionally, Wang et al. [67] focused on AoI minimization through caching placement optimization using a predefined UAV trajectory. By employing a PPO-based algorithm, they achieved better Key Performance Indicator (KPI) results compared to DQL and Advantage in Actor-Critic (A2C) approaches.

Both of the above works demonstrate that the trajectory of UAVs is not the only factor for ensuring freshness. However, it is clear that for practical applications, the optimization of UAV trajectory is pivotal. Therefore, it is common in the literature to work on the joint optimization of trajectory and other factors. Hu et al. [68] shows that the selection of which data to process is also significant. In their analyzed system, ground sensors are dispersed over the environment and generate tasks, where UAVs cooperatively collect them and offload the sensing result to a BS. The objective is to minimize the AoI of each task, and it decreases based on the last time the task was executed (i.e., the target was sensed, and the result is transmitted to the BS). The proposed solution involves a distributed sense-and-send protocol, where the UAVs, at each time step, should select which tasks to collect and where it should move.

The authors in [69] considered a V2X network, where UAVs are utilized to collect and process traffic streams, supporting intelligent transportation systems, enabling critical services such as autonomous driving and accident prevention. A centralized AI agent, which employs a Deep Deterministic Policy Gradient (DPPG)-based solution, processes information from the vehicular environment and makes strategic decisions for the deployed UAVs. Specifically, this agent is responsible for learning the dynamics of the vehicular environment, including random vehicle arrival, managing both actions: the UAV trajectories and the scheduling data offloading from vehicles to the UAVs.

In the same context, [70] minimizes data age by simultaneously defining the UAVs' trajectories and the set of devices for data gathering at each step using a PPO-based algorithm, obtaining better performance when compared to off-policy methods. Similarly, Sun et al. [71] developed a TD3-based solution for energy consumption reduction and data freshness. They defined an optimal UAV trajectory and band allocation for IoTDs and UAV communication and were able to find a trade-off between the analyzed KPIs.

The data offloading strategy from the UAV to a central node (usually referred to as MEC server or BS) is also relevant for the freshness of information. Wu et al. [72] studied this matter by defining a sensing and transmission protocol for UAV data offloading to a BS, and optimized both UAV trajectory and subchannel allocation. Zhu et al. [41] optimized UAV hovering by balancing UAV movement and offloading schedules using different Actor-Critic frameworks. Their approach focused on minimizing AoI by combining UAV trajectory optimization with offloading decisions to the cloud and managing UAV-cloud bandwidth. By effectively coordinating these elements, they achieved significant improvements in UAV performance and data freshness.

3.2 Computational Resources

3.2.1 Computational Scheduling and Execution Delay

The use of MEC servers, which have more powerful computational capabilities, is a promising strategy for systems with devices that have limited resources. Offloading part or all of the computational tasks from these devices to an edge server enables energy savings and allows for more complex applications in next-generation communications. However, it brings new challenges, such as the decision about when and which tasks to offload, while managing the transmission delay and network bandwidth constraints, especially in time-sensitive applications. Therefore, in an era of complex data, networks and requirements, management is of growing significance.

In this context, Liu et al. [73] studied a system with one mobile device and one MEC server, where both could run computational tasks simultaneously. The MEC server's CPU capability is much higher than that of the mobile device, making it advantageous to offload tasks to the server to reduce processing time. However, the mobile device needed to decide whether to offload tasks to the MEC server. By taking into account channel conditions and the current data queue status, the objective was to minimize the delay from producing the task to returning the processing result back (therefore, considering both execution and transmission delays), by optimizing the offloading decision. A search algorithm was used for this optimization, yielding lower delays compared to a greedy approach, where the task was always offloaded to the first CPU that was found available.

Mao et al. in [74] expanded the above work, including multiple mobile devices competing for limited computational resources at the MEC server. The CPU-cycle frequency and scheduling decisions are optimized in order to minimize the power consumption of the MEC server and average execution delay. Specifically, they used Little's Law to calculate the average execution delay based on the average sum queue length. The problem is decomposed into three parts: *i*) optimizing mobile devices' transmission power to the MEC server and bandwidth allocation for queue size and transmission power minimization, *ii*) optimizing mobile devices' CPU-cycle frequency to minimize their processing energy consumption and queue size, and *iii*) optimizing MEC's CPU-cycle frequency to minimize its processing energy consumption and queue size. The proposed strategy effectively balances the trade-off between power consumption and execution delay, confirming the importance of joint radio and computational resource management. The authors highlighted that optimal system operation should consider both the task arrival rate and the competition among users for network and computational resources.

Tran and Pompili [75] worked on scenarios with multiple MEC servers, where the users can offload computational tasks to these servers as needed. The objective was to minimize the total task delay (i.e., the transmission and processing delay) and devices' energy consumption related to transmission. In other words, this involved deciding whether to offload tasks and determining the optimal MEC server for offloading. By focusing on minimizing users' delay and transmission energy consumption, they decoupled the complex problem into three manageable parts: task offloading decisions, uplink power allocation, and computing resource allocation. This approach allowed for faster optimization and effective resource management.

The authors in [76] included one UAV working as a relay to the computational scheduling at the MEC server. The UAV collects data from mobile devices and sends it to MEC server. The objective is to optimize UAV trajectory, the power of the users and the computational scheduling at the MEC server while minimizing the total delay of all users (considering the UAV flight duration, transmission latency from users to UAV and from UAV to MEC, and processing delay in the MEC server), with constraints related to energy consumption of UAV. The problem is also decoupled in two parts: the trajectory and power optimization; and computational scheduling optimization, where they are resolved iteratively. Simulations demonstrated the effectiveness of the solution.

Finally, [77] minimizes the delay between the task collection by one UAV and its processing conclusion, in a scenario where both UAV and MEC server can process the tasks. The UAV's trajectory, UAV's transmission power and task offload ratio (i.e., how many packets should be offloaded to the MEC) should be optimized. By using a DPPG-based algorithm, the authors were able to find a solution that minimizes the execution delay.

3.2.2 Computational Scheduling and Freshness of Information

While contributing to different strategies for optimizing computation scheduling, the above works did not consider the generation timing of the data, i.e., the freshness of information. Moreover, it is commonly assessed based on the delay in data transmission; however, depending on the computational scheduling strategy, AoI and similar metrics can be highly affected by execution delay as well.

Motivated by the above, the processing delay affecting the freshness of information was studied in [78]. More specifically, the authors developed closed-form expressions for AoI based on a scenario where one data source offloads tasks to a local server that could either process it or offload to a MEC server. After each task is processed, an acknowledgement signal should be sent from the local server to a final destination; hence, if the data was

processed at the MEC server, this signal should be transmitted to the local server before dispatching to the last destination. This study assessed the influence of message size, number of required CPU cycles, data rate and computing capacity on the average AoI. The transmission and processing queue were also considered. Results showed that partial offloading (i.e., processing part of the tasks in the local server, and the rest in the MEC server) brings the best results related to AoI.

Li et al. [79] considered a vehicle network sensing the environment and sending the data to a server for processing, where the sensing result should be sent back to the vehicle. Considering the delay related to data processing, the authors used a DQL-based strategy to optimize the processing order at the server aiming to minimize the freshness of information considering the vehicles' movement dynamics (i.e., how much the vehicle moved before receiving the sensing result).

The authors in [80] proposed a task offloading optimization to achieve a better freshness of information in a multiaccess UAV-aided computing system, where mobile devices generate computational tasks and offload to either a computational capable UAV or to any BSs connected to a MEC server. The mobile devices need to receive their computed tasks back, and offloading them to the UAV or to BS will increase the transmission delay but will potentially reduce the processing time while also reducing its energy consumption. Using a DRL-based algorithm, the authors optimized the computational offloading strategy and dealt with the trade-off of processing in energy consumption and freshness of information of the mobile devices.

Alabbasi and Aggarwal in [81] explored the joint optimization of freshness of information and task completion time, using a convex optimization approach to make the balance between these two metrics. Specifically, they studied a vehicular network in which cars collect information about the environment. The data is computed in a data center with multiple Virtual Machines (VMs) with the same computational capabilities. Finally, the output should be sent to an end user in a shared channel. In this case, the freshness of information, measured in terms of AoI, is essential because the vehicular network needs recent information for safe and accurate decisions. Moreover, the service provider needs to finish jobs as soon as possible to avoid computational and network congestion; hence, the time spent consuming resources on the server (processing queue, processing time and downlink transmission in the shared channel) must be lowered. Particularly, the authors refer to this time as the *completion time*, which is equivalent to the sum of the CPU TAT and the transmission delay. The optimization problem is related to the computational scheduling between the data center's multiple cores. It aims to minimize the weighted freshness of information and the completion time. The authors clearly explained the trade-off between both metrics in their environment, observing that the scheduling directly affects only the

computational phase; however, it indirectly influences both AoI and completion time.

3.3 Hierarchical Decomposition

In this section, we describe works that employed a hierarchical decomposition for a UAV trajectory or computational scheduling decision and solve it with a DRL strategy.

In [82], the authors considered a scenario with multiple UAVs collecting and processing data from mobile devices, located on the ground. The computation can be either done on the ground device or on any UAV, which is used to alleviate the computational burden on the devices. The objective is to minimize average delay by optimizing the UAVs' position and the offloading strategy. Realizing that the UAVs' position strategy highly influences the offloading decision, the authors decided to decouple the problem. Specifically, their solution involves defining the next position of the UAVs to minimize the expected average delay and, in the next time steps, with the UAVs in the same position, reduce the delay by optimizing the devices-UAV offloading. They used a DPPG and DQN strategy to optimize the UAVs position and the offload design, respectively. It was possible to find an efficient solution with the proposed method in different network configurations.

Yang et al. [42] minimize AoI and system energy consumption through UAV trajectory and device offloading optimization. The authors divide the problem into two steps: the first is solved with PPO, and the second is solved greedily. Likewise, Gong et al. in [83] extended the work and solved the problem of AoI minimization in an Intelligent Reflecting Surface (IRS)-aided system by optimizing the scheduling policy using PPO and, in sequence, using alternating optimization and relaxation methods to decide on the beamforming variable. The authors show that optimizing the scheduling first facilitates the second decision.

Finally, the authors in [84] explore UAV trajectory design and network resource control within a UAV-aided MEC system, concentrating on a scenario where sensors generate a substantial amount of data, resulting in increased strain on the communication model. They propose that the UAV should preprocess the gathered information to remove redundant data. The first problem addressed is optimizing the UAV path to maximize the amount of collected data, which is tackled using a DQL approach. The second problem involves optimizing the UAVs' CPU frequency, transmission power to the MEC, and bandwidth allocation to minimize the UAVs' power consumption, using Lyapunov optimization-based techniques. Both problems are resolved iteratively, and simulations show that the proposed solution efficiently reduces waiting times and queue lengths, whereas these parameters increase indefinitely in greedy solutions.

3.4 Overview

As analyzed in the reviewed works, different configurations of UAV-aided MEC systems have been widely explored, with applications for data collection and computation in systems such as IoT, IRS and V2X. It is clear that transmission and processing delays, the freshness of information and energy consumption are matters of concern and the optimization of UAV trajectory and computational scheduling strategies is essential for the efficiency in this scenario.

However, there is a lack of studies in closed-loop scenarios for UAV-aided MEC systems. For instance, smart farms with integrated sensors and actuators, as proposed in [24], necessitate the consideration of a closed-loop communication, where a sensor collects data, offloads it to a processor, and the generated command returns to the actuator, which is often attached to the sensor. In addition, although some efforts are being made to reduce the waiting time and processing time of computational tasks, only very few works do it jointly with delay or freshness of information, and the trade-off between process completion and/or turnaround time and freshness of information has not been explored in UAV systems.

Therefore, different than existing works, we use the AoL metric to evaluate the freshness of information in a closed-loop UAV-aided MEC communication system and, at the same time, consider the queuing delay and duration of data processing, measuring it in terms of TAT. We also investigate the trade-off between the AoL and computational TAT in a UAV-aided MEC system, providing a stable hierarchical-based DRL solution for UAV trajectory and task offloading optimization in a dynamic network, computational resources, data arrival rates and packet workload scenarios. Table 3.1 includes a comparative analysis between this work and the relevant references cited in this chapter. We focus the comparison by regarding the studies that analyzed the scenario in a closed-loop communication system, with decisions based on UAV trajectory and offloading to a central node, aiming to optimize both freshness of information and computational performance using a DRL solution. Unlike existing studies, our work addresses all of these aspects.

Table 3.1: Comparative analysis of related works

Ref.	Closed-loop comm.	Decision		Optimization		DRL solution
		UAV traj.	UAV offloading	Freshness of info.	Comp. efficiency	
[60]		✓		✓		✓
[61]		✓		✓		✓
[39]		✓		✓		
[62]		✓		✓		✓
[63]		✓		✓		✓
[64]		✓		✓		
[65]		✓		✓		✓
[40]		✓		✓		✓
[66]			✓	✓		✓
[67]				✓		✓
[68]		✓		✓		✓
[69]		✓		✓		✓
[70]		✓		✓		✓
[71]		✓		✓		✓
[72]		✓	✓	✓		✓
[41]		✓	✓	✓		✓
[73]	✓					
[74]	✓				✓	
[75]					✓	
[76]		✓			✓	
[77]		✓	✓			✓
[78]				✓		
[79]	✓			✓		✓
[80]	✓			✓		✓
[81]				✓	✓	
[82]		✓				✓
[42]		✓	✓	✓		✓
[83]				✓		✓
[84]		✓	✓		✓	✓
This work	✓	✓	✓	✓	✓	✓

Chapter 4

Methodology

In this chapter, we detail the methodology used in this thesis. The system model is presented, and the joint optimization problem involving the UAV trajectory and offloading strategy to the MEC server for minimization of both AoL and TAT is defined. Nevertheless, we explain the challenges related to the problem and explain why a hierarchical-based deep reinforcement learning solution would be beneficial. Finally, the proposed solution is presented.

4.1 System Model

In a smart farm environment, we consider a UAV-assisted MEC system as depicted in Figure 4.1. A single UAV with computational capabilities travels at a constant speed s_u and constant height h_u; its two-dimensional positions at any step $n \in \mathcal{N} = \{1, \ldots, N\}$ is defined as $c_u^n = \{x_u^n, y_u^n\}$. One MEC server is located on the border of the farm and is also able to perform computations.

In this scenario, multiple IoTDs are spread around the farm. Each one comprises a sensor and an actuator, where the sensor collects information about the environment. The actuator receives commands that will result in changes in the system, such as enabling crop irrigation, pesticide application, or automated feeders in aquaculture. The collection of IoTDs is denoted as the set $\mathcal{I} = \{1, \ldots, I\}$. Each IoTD i is located on the ground and has a fixed position $c_i = \{x_i, y_i\}$. The IoTDs are aggregated into clusters, where each IoTD belongs to only one cluster and generates packets that will originate different tasks, which are stored in their individual buffers. The set of clusters is denoted as $\mathcal{K} = \{1, \ldots, K\}$; all

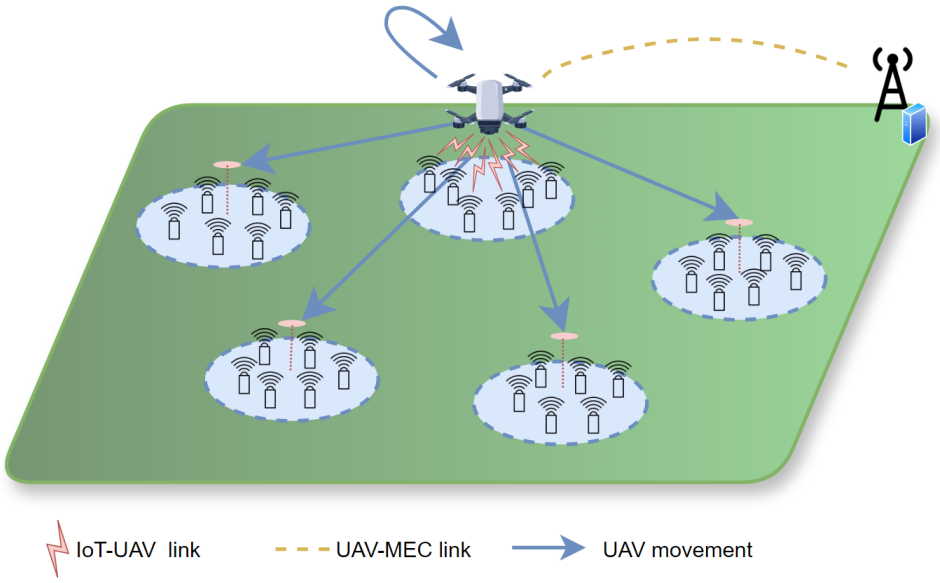


Figure 4.1: Proposed UAV-assisted MEC system. The communication links work in both uplink and downlink directions.

clusters $k \in \mathcal{K}$ are composed of the same number of IoTDs (I_k) and have a Cluster Head (CH), which is a point at height h_u with coordinates $c_k = \{x_k, y_k\}$ that the UAV hovers to communicate with IoTDs belonging to cluster k (defined as the set $\mathcal{I}_k$) and collect their data.

At each step n, the UAV selects one cluster $k \in \mathcal{K}$ to serve. It is determined by the one-hot variable $\rho_k^n \in \{0, 1\}$. Serving one cluster k means that the UAV will:

- travel to the CH k;
- collect the accumulated packets from all $i \in \mathcal{I}^n = \{\mathcal{I}_k | \rho_k^n = 1\}$; and
- aggregate the packets from the same IoTD, originating up to one computational task per IoTD.

Given the importance of the variable ρ in defining the next UAV position, we refer to it both as the UAV trajectory decision and as the cluster assignment decision.

The originated computational tasks are processed at either the UAV or the MEC server, and the UAV is responsible for determining where each task is going to be processed. This

is performed based on the variable $\alpha_{ik}^{n} \in \{0, 1\}$, where it is 1 if the task produced by the i^{th} IoTD of the k^{th} cluster is offloaded to the MEC server, and 0 otherwise. When the UAV collects all the packets from the served IoTDs, it immediately starts to process the tasks assigned to be computed locally (i.e., with $\alpha_{ik}^{n} = 0$) and begins forwarding the tasks designated for the MEC server. Each one of the processing units will independently perform computations and generate one command per task, that should be sent back to the source IoTD, whose actuator operates accordingly. Therefore, all the communication links depicted in Figure 4.1 work in both uplink and downlink directions.

Moreover, the UAV is responsible for not only facilitating the communication between IoTs and MEC server but also for providing extra computational power to the tasks collected by the sensors. It is also important to mention that there is no direct communication between IoTDs and the MEC server. Accordingly, as depicted in Figure 4.2, the commands processed at the MEC server must first be transmitted to the UAV and then forwarded to the respective IoTD. The same figure illustrates the internal details of communication and processing. The interconnection between antennas (TX/RX units) and internal CPUs is assumed to happen instantaneously. Next, we detail each part of the system model, with Table 4.1 depicting the list of main notations.

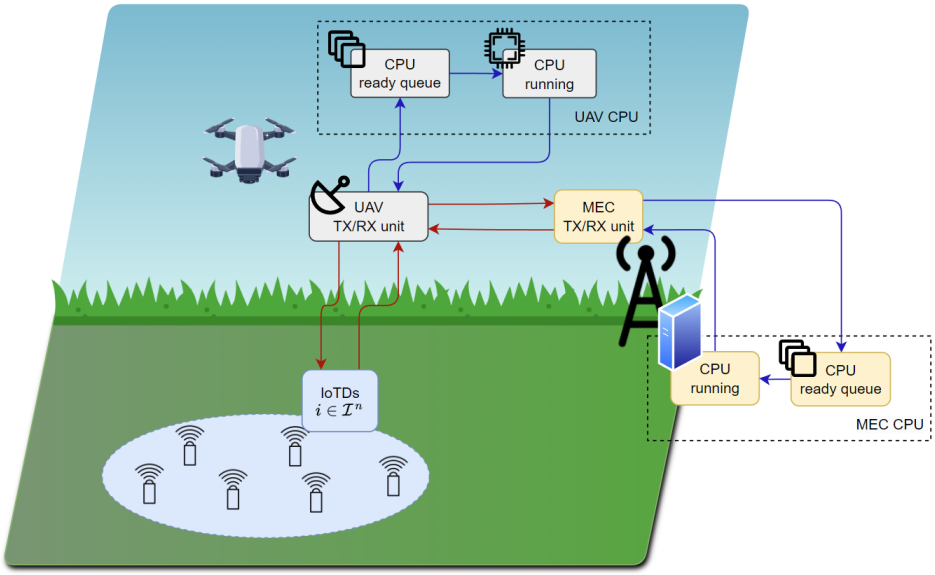


Figure 4.2: Diagram of possible paths of the computational tasks, where red arrows indicate the communication channels and blue arrows represent the interconnection between different components of the same device.

Table 4.1: List of Main Notations

Sets	**Size**	**Description**
$k \in \mathcal{K}$	K	Set of clusters
$n \in \mathcal{N}$	N	Set of steps
$i \in \mathcal{I}$	I	Set of IoTDs
$i \in \mathcal{I}_k$	I_k	Set of IoTDs in cluster k
$i \in \mathcal{I}^n$	I^n	Set of IoTDs in cluster k served at step n
$i \in \mathcal{I}^n_{\text{UAV}}$	I^n_{UAV}	Set of IoTDs with tasks processed at the UAV
$i \in \mathcal{I}^n_{\text{MEC}}$	I^n_{MEC}	Set of IoTDs with tasks processed at the MEC
Decision variables	**Domain**	**Description**
α^n_{ik}	$\{0, 1\}$	Task from IoTD i offloaded to MEC at n
ρ^n_k	$\{0, 1\}$	Cluster k served at n
Time variables	**Domain**	**Description**
t^n_{fly}	$\mathbb{R}^+$	Duration of UAV flight at n
t^n_{hov}	$\mathbb{R}^+$	Duration of UAV hover at n
t^n_{col}	$\mathbb{R}^+$	Duration of IoTDs-UAV transmission at n
$t^n_{\text{proc},i}$	$\mathbb{R}^+$	Burst time of task produced by i at n
$\hat{t}^n_{\text{UAV},i}$	$\mathbb{R}^+$	Waiting time at the UAV of task from i at n
$\hat{t}^n_{\text{MEC},i}$	$\mathbb{R}^+$	Waiting time at the MEC of task from i at n
IoTDs variables	**Domain**	**Description**
b^n_i	$\mathbb{N}$	Number of packets of the task produced by i in n
ψ^n_i	$\mathbb{N}$	Generation time of the recent collected packet
Ω^n_i	$\mathbb{R}$	TAT for task produced by i in n
Δ^n_i	$\mathbb{R}$	AoL IoTD i in n
J^n	$\mathbb{N}$	Number of tasks processed up to n
$\bar{\Omega}[n]$	$\mathbb{R}$	Average TAT over all tasks in n
$\bar{\Delta}[n]$	$\mathbb{R}$	Average AoL over all IoTDs in n
Parameters	**Domain**	**Description**
ϕ_i	$\mathbb{R}^+$	Packet size of task produced by IoTD i
C_i	$\mathbb{N}$	CPU-cycles necessary to process task (workload)
P^{tx}_i	$\mathbb{N}$	Transmission power of the IoTD i
P^{tx}_u	$\mathbb{N}$	Transmission power of the UAV
f_{UAV}	$\mathbb{N}$	CPU-cycle frequency of UAV
f_{MEC}	$\mathbb{N}$	CPU-cycle frequency of MEC server

4.1.1 Data Generation

The IoTDs constantly collect information about the environment. The data production is random and follows a Poisson process with a specific arrival rate for each cluster.

Each IoTD i gathers its packets and stores them in its local buffer until they are offloaded to the UAV. In this work, we consider that the buffers have infinite size; therefore, there is no risk of overflow. The packets produced by any $i \in \mathcal{I}$ have a fixed packet size ϕ_i and a number of necessary CPU cycles to process it completely, C_i, which is proportional to ϕ_i, as defined in [73].

4.1.2 Communication Model

Upon arriving at a CH $k \in \mathcal{K}$, the UAV sends a wake-up message to all $i \in \mathcal{I}^n$, requesting them to offload their data. This message is assumed to have a fixed duration of t_{w}. Each IoTD $i \in \mathcal{I}^n$ has its channel allocated to the UAV hence interference is neglected. We denote $b_i^n \geq 0$ as the number of packets accumulated in the buffer of $i \in \mathcal{I}^n$ in the moment the UAV arrives at the served cluster. We define $b_i^n = 0$, $\forall i \notin \mathcal{I}^n$.

Overview

For the uplink channel in IoTD-UAV and UAV-MEC communication links, we consider LoS and Non-LoS (NLoS) propagation models as calculated in [85]. The probability of LoS propagation is hence:

$$P^{LoS}(\theta) = 1 - P^{NLoS}(\theta) = \frac{1}{1 + a \cdot e^{-b(\theta - a)}}, \tag{4.1}$$

where θ denotes the communication angle between IoTD i and UAV or between the UAV and the MEC. The variables a and b are constants related to the communication environment.

In addition, the path loss effect is considered in all uplink channels, with

$$L(d_{iu}) = 20\log_{10}\left(\frac{4\pi f_c d_{iu}}{c}\right) + P^{LoS}\eta^L + P^{NLoS}\eta^N \tag{4.2}$$

and

$$L(d_{us}^n) = 20\log_{10}\left(\frac{4\pi f_c d_{us}^n}{c}\right) + P^{LoS}\eta^L + P^{NLoS}\eta^N \tag{4.3}$$

representing the IoTD-UAV channel and the UAV-MEC channel, respectively. Here, d_{iu} and d_{us}^n denote the three-dimensional distance between IoTD i and the UAV and between the UAV and the MEC server at n, respectively. The variables f_c, c, η^L, and η^N pertain to the carrier frequency, speed of light, and path loss for the LoS and NLoS propagation groups, respectively.

Communication between IoTDs and the UAV

The communication delay within the IoTD-UAV channel for one single task generated at IoTD i at step n and composed by b_i^n packets is based on Shannon's theorem [86] and is expressed as:

$$t_{iu}^n = \frac{b_i^n \phi_i}{r_{iu}} = \frac{b_i^n \phi_i}{B\log_2\left(1 + \frac{P_i^{\text{tx}} \cdot L(d_{iu})}{\sigma^2}\right)}, \forall i \in \mathcal{I}^n, n \in \mathcal{N}, \tag{4.4}$$

where B is the available bandwidth for communication, P_i^{tx} is the transmission power of the IoTD i, and $\sigma^2 = BN_o$, where N_o represents the power spectral density of the Additive White Gaussian Noise (AWGN). Nevertheless, assuming that the communication between IoTDs and the UAV is Frequency Division Multiple Access (FDMA)-based and that the bandwidth is equally divided among the devices for this communication, we have $B = W/I_k$, where W is the total available bandwidth.

The UAV collects all available data from all $i \in \mathcal{I}^n$, hence the delay for data collection at step n can be defined as the sum of the wake-up time message, with duration equal to t_{w}, and the maximum communication delay between the UAV and all $i \in \mathcal{I}^n$. Therefore:

$$t_{\text{col}}^n = t_{\text{w}} + \max_{i \in \mathcal{I}^n}(t_{iu}^n), \forall n \in \mathcal{N}. \tag{4.5}$$

Communication between UAV and the MEC Server

After receiving the collected information from the served IoTDs, the UAV aggregates the packets from the same IoTDs, originating up to one task per device, and defines which tasks

will be offloaded to the MEC server for processing. As aforementioned, this is represented by α_{ik}^n. Let $\mathcal{I}_{\text{MEC}}^n = \{\mathcal{I}^n | \alpha_{ik}^n = 1\}$ and $\mathcal{I}_{\text{UAV}}^n = \{\mathcal{I}^n | \alpha_{ik}^n = 0\}$ denote the set of IoTDs whose tasks will be processed at the MEC server and at the UAV, respectively. The transmission delay within UAV-MEC channel is calculated as:

$$t_{ius}^n = \frac{b_i^n \phi_i}{r_{us}^n} = \frac{b_i^n \phi_i}{W \log_2 \left(1 + \frac{P_u^{\text{tx}} \cdot L(d_{us}^n)}{\sigma^2}\right)}, \forall i \in \mathcal{I}_{\text{MEC}}^n, n \in \mathcal{N}, \tag{4.6}$$

where the transmission rate r_{us}^n is calculated according to the transmission power of the UAV, P_u^{tx}, and the path loss $L(d_{us}^n)$, where d_{us}^n is the distance between the UAV and MEC server at step n. For the transmission from the UAV to the MEC server, the full bandwidth is accessible, i.e., W.

Nevertheless, we consider that the tasks are sent one at a time, in order, and according to their origin IoTDs' index, where, for illustration, the task from IoTD 1 is sent before the task from IoTD 2. Based on this, we can obtain the time necessary for the task collected by $i \in \mathcal{I}_{\text{MEC}}^n$ to arrive on the MEC server. For served IoTDs with no collected packets (i.e., $b_i^n = 0$, where $i \in \mathcal{I}_{\text{MEC}}^n$), the arrival rate will be non-existent, hence, we define it as equal to zero. In other cases (i.e., $b_i^n > 0$, where $i \in \mathcal{I}_{\text{MEC}}^n$), we can easily calculate the arrival time based on the transmission delay computed in Equation 4.6. Thus, we define:

$$t_{\text{arr},i}^n = \begin{cases} 0 & \text{if } b_i^n = 0, \\ \sum_{i'=1}^{i} t_{i'us}^n \alpha_{i'k}^n & \text{if } b_i^n > 0. \end{cases} \quad \forall i \in \mathcal{I}_{\text{MEC}}^n, n \in \mathcal{N}. \tag{4.7}$$

Downlink Communication

Lastly, the communication delays within MEC-UAV and UAV-IoTD channels (i.e., the downlink direction) are respectively represented as t_{sui} and t_{ui} where both are fixed and identical for all $i \in \mathcal{I}^n$ and for all $n \in \mathcal{N}$. This generalization is possible because the commands sent back to the origins are assumed to be small and effortlessly transmitted in all cases.

4.1.3 Turnaround Time

In order to compute the TAT in this scenario, we need to consider that for each task, we have a time in the ready queue at its processing unit (i.e., the waiting time), $\hat{t}_i^n$, and a

time for processing (i.e., the burst time). With this, the TAT for each computed task can be obtained.

Burst Time

We calculate the delay to process one task with b_i^n packets based on the number of necessary CPU cycles to compute each packet and based on the computational power of the processing units. As aforementioned, the variable C_i represents the necessary CPU cycles to compute each packet collected by the IoTD i. Dividing it by the CPU clock speed, it is possible to obtain the burst time for one packet, and by multiplying it by the number of packets in one task, we calculate the burst time of that task.

Let the CPU-cycle frequency at the UAV and the MEC server be f_{UAV} and f_{MEC}, respectively, we have for each task produced by i in step n:

$$t_{\text{proc},i}^n = (1 - \alpha_{ik}^n) b_i^n \frac{C_i}{f_{\text{UAV}}} + \alpha_{ik}^n b_i^n \frac{C_i}{f_{\text{MEC}}}, \forall i \in \mathcal{I}^n, n \in \mathcal{N}, k \in \{\mathcal{K} | \rho_k^n = 1\}. \tag{4.8}$$

CPU Waiting Time at the UAV

All tasks processed at the UAV are computed sequentially and, similarly to the transmission order, rely on its origin IoTD's index, in a FIFO fashion. Because of that, the waiting time of the task produced by i in step n is the summation of the processing time of all tasks that are already in the CPU ready queue. That is, the summation of burst time ($t_{\text{proc},i'}^n$) for $i' \in [1, i-1]$, as below:

$$\hat{t}_{\text{UAV},i}^n = \sum_{i'=1}^{i-1} t_{\text{proc},i'}^n (1 - \alpha_{i'k}^n). \tag{4.9}$$

CPU Waiting Time at the MEC Server

A task processed at the MEC server must be sent to it from the UAV, wait for the processing time of the tasks already in the CPU ready queue, and only then be processed.

The waiting time is zero if it is the first task in the sequence (i.e., $i = I_{\text{MEC},1}^n$, where we use the notation $I_{\text{MEC},r}^n$ to represent the r-th task received at the MEC server). For subsequent tasks, it will depend on whether any task is still being processed when it

arrives in the processing queue and how many tasks are already waiting in the ready queue. Therefore, it is essential to track the arrival time and the burst time of each task in each step n.

In Figure 4.3, we summarize how the ready queue at the MEC server operates. Based on this, we define the waiting time for tasks processed at the MEC server:

$$\hat{t}^n_{\text{MEC},i} = \begin{cases} 0 & \text{if } i = I^n_{\text{MEC},1}, \\ \left[\sum_{i'=i_0}^{i-1} (t^n_{\text{proc},i'} \alpha^n_{i'k}) - (t^n_{\text{arr},i} - t^n_{\text{arr},i_0})\right]^+ & \text{otherwise,} \end{cases}$$

where $[x]^+ = \max\{x, 0\}$ and i_0 is the last $i \in \mathcal{I}^n_{\text{MEC}}$ with $\hat{t}^n_{\text{MEC},i} = 0$.

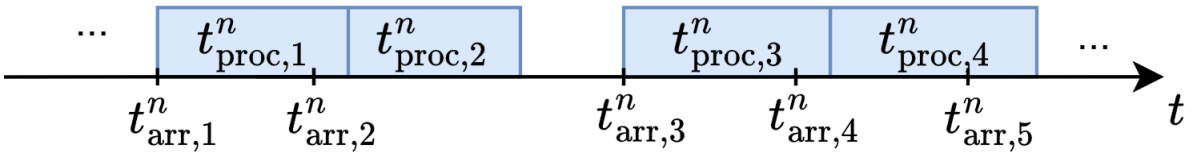


Figure 4.3: Example of MEC server ready queue at any step $n \in \mathcal{N}$, with five tasks. In this case, the waiting time of each task is $\hat{t}^n_1 = 0$, $\hat{t}^n_2 = t^n_{\text{proc},1} - (t^n_{\text{arr},2} - t^n_{\text{arr},1})$, $\hat{t}^n_3 = 0$, $\hat{t}^n_4 = t^n_{\text{proc},3} - (t^n_{\text{arr},4} - t^n_{\text{arr},3})$ and $\hat{t}^n_5 = t^n_{\text{proc},4} + t^n_{\text{proc},3} - (t^n_{\text{arr},5} - t^n_{\text{arr},3})$.

Average TAT

As aforementioned in previous chapters, TAT for each task can be obtained with the sum of waiting time and burst time. We calculate the TAT for the task produced by $i \in \mathcal{I}^n$ at step n:

$$\Omega^n_i = \begin{cases} \mathbb{1}_{\{b^n_i\}}[\hat{t}^n_{\text{UAV},i} + t^n_{\text{proc},i}] & \text{if } i \in \mathcal{I}^n_{\text{UAV}}, \\ \mathbb{1}_{\{b^n_i\}}[\hat{t}^n_{\text{MEC},i} + t^n_{\text{proc},i}] & \text{if } i \in \mathcal{I}^n_{\text{MEC}}, \end{cases} \tag{4.10}$$

where $\mathbb{1}_{\{b^n_i\}}$ is 1 if $b^n_i > 0$ and 0 otherwise, hence guaranteeing that TAT will be zero if no task was produced by i in step n ($\Omega^n_i = 0$ when $b^n_i = 0$).

Finally, we update the average TAT over all the processed tasks:

$$\bar{\Omega}[n+1] = \begin{cases} 0 & \text{if } J^n = 0, \\ \frac{1}{J^n} \sum_{\eta=1}^{n} \sum_{i=1}^{I} \Omega_i^\eta & \text{if } J^n > 0. \end{cases} \tag{4.11}$$

where $J^n = \sum_{\eta=1}^{n} \sum_{i=1}^{I} \mathbb{1}_{\{b_i^\eta\}}$ is the total number of tasks processed up to step n.

4.1.4 UAV Mobility Model

Given the definition $d_{ku}^n = \|c_u^n - c_k\|_2^2$ as the Euclidean distance between the UAV and the CH of cluster k, we define the time of flight of the UAV from its initial position c_u^{n-1} to the CH of the served cluster as:

$$t_{\text{fly}}^n = \frac{d_{ku}^{n-1}}{s_u}, \forall n \in \mathcal{N}, k \in \{\mathcal{K} | \rho_k^n = 1\}. \tag{4.12}$$

After arriving at the CH of $k \in \{\mathcal{K} | \rho_k^n = 1\}$, the UAV hovers over a certain point for enough time to collect the packets from the served IoTDs and until all the tasks are processed and returned to their origin. Therefore, we define the UAV hovering time at step n as:

$$t_{\text{hov}}^n = t_{\text{col}}^n + \max_{i \in \mathcal{I}^n} (t_i^n), \tag{4.13}$$

where t_i^n is the time elapsed between the collection of all tasks from the IoTD and the receipt of each command back at their origin, as below:

$$t_i^n = \begin{cases} \Omega_i^n + \mathbb{1}_{\{b_i^n\}} t_{ui} & \text{if } i \in \mathcal{I}_{\text{UAV}}^n, \\ \mathbb{1}_{\{b_i^n\}} [\sum_{i'=1}^{i} t_{i'us}^n \alpha_{ik}^n + \Omega_i^n + t_{sui} + t_{ui}] & \text{if } i \in \mathcal{I}_{\text{MEC}}^n. \end{cases} \tag{4.14}$$

4.1.5 Age of Loop

The AoL, specifically the uplink AoL in this context [43], measures the data freshness by evaluating the latency from the receivers' perspective in a closed-loop link. More specifically, it is the time elapsed from the generation time of the task that originated the most recent command received by any of the IoTD and the current time. The sole term AoL will be used for simplicity.

At the end of step n, we assume that the tasks from all $i \in \mathcal{I}^n$ are guaranteed to be processed, and their corresponding command is received back at the origin. Moreover, as aforementioned, each IoTD has up to one task at each step. Defining $t^n = t^n_{\text{fly}} + t^n_{\text{hov}}$ as the total duration of step n, the updated AoL in IoTD i is represented as:

$$\Delta_i^{n+1} = \begin{cases} \sum_{\eta=1}^{n} t^\eta - \psi_i^n & \text{if } b_i^n > 0, \\ \Delta_i^n + t^n & \text{otherwise.} \end{cases} \tag{4.15}$$

where ψ_i^n is the generation time of the most recently *collected* packet per IoTD. In the above equation, it is straightforward that the AoL of an IoTD $i \in \mathcal{I}^n$ with at least one packet at n is decreased to the difference between the current time (i.e., the moment step n ends) and the generation time of its most recent task. For the other IoTDs not served at n (i.e., $b_i^n = 0$ and/or $i \notin \mathcal{I}^n$), the AoL increases according to the step duration.

Finally, we define the average AoL over all IoTDs as:

$$\bar{\Delta}[n] = \frac{1}{I} \sum_{i=1}^{I} \Delta_i^n. \tag{4.16}$$

4.2 Problem Definition

4.2.1 Problem Formulation

The overall objective is to jointly minimize the average AoL and the average TAT by choosing the direction of the UAV and the UAV-MEC offloading strategy. We assume that the MEC server has a faster CPU than the UAV and handles various data on the farm, necessitating the reduction of computational and storage resource usage. In addition, it is essential to minimize the computational burden of the UAV in order to conserve energy. Considering $\rho^n = \{\rho_k^n | k \in \mathcal{K}\}$ and $\alpha^n = \{\alpha_{ik}^n | i \in \mathcal{I}, k \in \mathcal{K}\}$, the problem is formulated as:

$$
\begin{aligned}
\mathbf{P}: \min_{\{\rho^n, \alpha^n\}} \quad & \left[\frac{\vartheta}{\Theta^A} \bar{\Delta}[n] + \frac{(1-\vartheta)}{\Theta^T} \bar{\Omega}[n] \right] \\
\text{s.t. } & \mathcal{C}1: \rho_k^n \in \{0,1\},\ \forall k \in \mathcal{K}, n \in \mathcal{N} \\
& \mathcal{C}2: \sum_{k \in \mathcal{K}} \rho_k^n = 1,\ \forall n \in \mathcal{N} \\
& \mathcal{C}3: \alpha_{ik}^n \in \{0,1\},\ \forall i \in \mathcal{I}^n, k \in \mathcal{K}, n \in \mathcal{N} \\
& \mathcal{C}4: \sum_{i \in \mathcal{I}^n} \alpha_{ik}^n \leq I_k,\ \forall n \in \mathcal{N}, k \in \mathcal{K} \\
& \mathcal{C}5: \mathcal{I}^n = \{\mathcal{I}_k | \rho_k^n = 1\},\ \forall n \in \mathcal{N} \\
& \mathcal{C}6: \sum_{i \in \mathcal{I}} \alpha_{ik}^n = 0,\ \forall k \notin \{\mathcal{K} | \rho_k^n = 0\}, i \in \mathcal{I}, n \in \mathcal{N}
\end{aligned}
\tag{4.17}
$$

where the variables ϑ, Θ^A and Θ^T are the weight and normalization factors, respectively. The constraints $\mathcal{C}1$ to $\mathcal{C}3$ are related to the fact that ρ^n and α^n are binary decisions. Constraint $\mathcal{C}2$ limits the UAV to serve only one cluster per time step, and constraint $\mathcal{C}4$ bounds the UAV-MEC offloading decision up to the number of served IoTDs. $\mathcal{C}5$ states $\mathcal{I}^n$ definition according to ρ_k^n decision. Lastly, $\mathcal{C}6$ defines $\alpha_{ik}^n = 0$ for all tasks of $i \notin \mathcal{I}^n$. For these last two constraints, the relation between α and ρ is evident, where both are being defined simultaneously.

4.2.2 Challenges to solve the optimization problem

The optimization problem $\mathbf{P}$ is challenging due to the delicate trade-off between the AoL and TAT. For instance, serving the same cluster for consecutive steps results in a decreased accumulation of packets within the cluster's IoTDs, thus reducing the waiting time and processing time, consequently lowering the TAT. However, the AoL of the other clusters that are not visited will be increased, hence augmenting the average AoL.

Secondly, the trajectory and offloading decisions are coupled variables. Optimizing either alone is not sufficient for optimizing the overall objective. It means that even if an efficient trajectory is defined, AoL and TAT might still increase if an inadequate offloading strategy is followed. The opposite is also true: even with an optimal offloading strategy, AoL and TAT might still increase if the trajectory is not efficiently managed.

Finally, solving the formulated problem becomes more challenging when accounting for the system model's real-world scenario, which includes oscillating network conditions,

computational resources, and random data generation. Hence, we aim to identify a robust solution that adapts to these environmental variations.

4.3 Deep Reinforcement Learning Solution

4.3.1 MDP formulation

To address the challenges associated with the problem **P**, we use DRL. As an initial step toward this solution, we formulate a MDP based on **P** as:

State

Because the state space should define the current status of the environment, we define it as the average AoL of each cluster ($\Delta_k^n = 1/I_k \sum_{i \in \mathcal{I}_k} \Delta_i^n$), the distance from each cluster head to the UAV, the average of the generation time of the most recently collected task of each cluster ($\Psi_k^n = 1/I_k \sum_{i \in \mathcal{I}_k} \psi_i^n$), and the number of packets collected. With this, we aim to give enough information to help the agent define the next action. Therefore, we have the state space defined as:

$$\mathbf{s}_n = \{\Delta_{k \in \mathcal{K}}^n, d_{ku \in \mathcal{K}}^{n-1}, \Psi_{k \in \mathcal{K}}^{n-1}, \sum_{i \in \mathcal{I}} b_i^{n-1}\}. \tag{4.18}$$

Action

As aforementioned, the action should be the cluster assignment decision and the offloading decision. The agent should define both simultaneously, i.e., when the UAV arrives at the served cluster head (as defined by ρ^n), the offloading decision is already defined, and the UAV already knows which IoTDs will have their tasks offloaded to the MEC server and which ones will have their tasks processed locally. The action space can hence be defined as:

$$\mathbf{a}_n = \{\rho^n, \alpha^n\}. \tag{4.19}$$

Reward

Aiming to optimize problem $\mathbf{P}$, we take the immediate reward as the negative of it, or as below:

$$\mathbf{r}_n = -\left[\frac{\vartheta}{\Theta^A}\bar{\Delta}[n] + \frac{1-\vartheta}{\Theta^T}\bar{\Omega}[n]\right]. \tag{4.20}$$

4.3.2 Proximal Policy Optimization

The DRL solution proposed is based on the PPO algorithm, where an actor network defines the next action a_n according to the current environment state s_n, generating the reward r_{n+1} and changing the environment to state s_{n+1}. At each step, the environment information is stored in an experience buffer $\mathcal{B}$. After repeating this process for N times (the length of training trajectory), the return is computed for each trajectory in each step, and the advantage estimates are calculated for each trajectory using the newest critic network. Finally, the parameters of both networks are updated with K epochs, with the experience buffer split into M mini-batches. That sequence is performed again until the end of the simulation. This process is summarized in Figure 4.4 and in Algorithm 1.

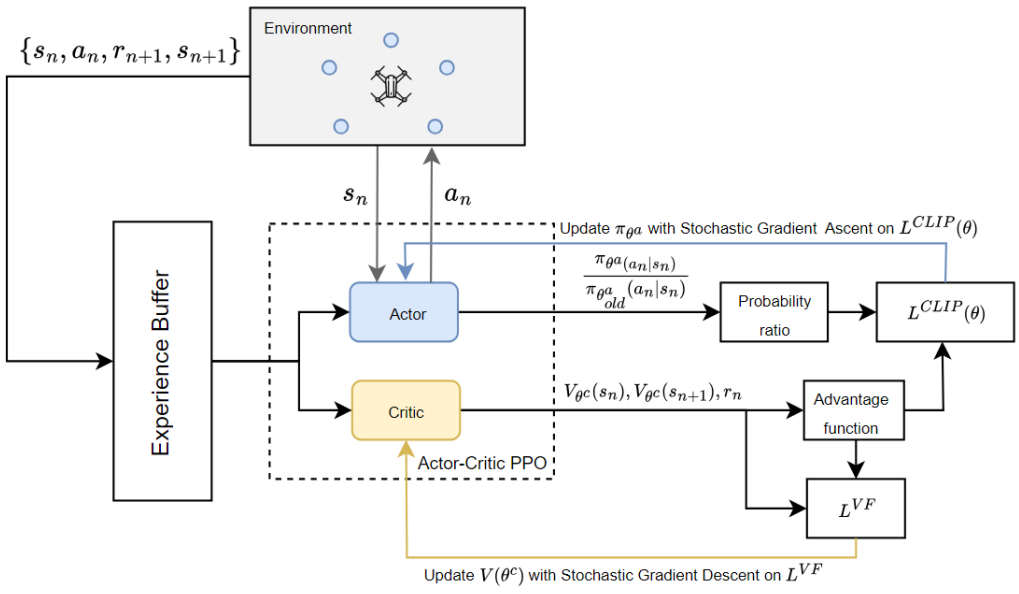


Figure 4.4: Proximal Policy Optimization diagram.

Algorithm 1 PPO solution (based on [55])

Initialize actor and critic network parameters θ_0^a and θ_0^c
for episodes $= 1, \ldots, E$ **do**
    for timestep $n = 0, \ldots, N-1$ **do**
        Observe state s_n and buffer $\mathcal{B}$
        Using the current policy $\pi_{\theta^a}(a_n|s_n)$, select action a_n
        Run a_n, go to next state s_{n+1} and receive reward r_{n+1}
        Store a_n, s_n, s_{n+1} and r_{n+1} in buffer $\mathcal{B}$
    end for
    Compute advantage estimates $\hat{A}_n$ for each trajectory using GAE [53]
    Compute the probability ratio $r(\theta) = \frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}$
    Update θ^a according to 2.17 and θ^c according to 2.18
    Clear buffer $\mathcal{B}$
end for

It is important to mention that, as explored in Chapter 3, multiple works [42, 67, 70] compared a PPO-based solution to different DRL approaches and achieved a more efficient result with PPO in similar scenarios. Therefore, it is not our objective to do a comparison study between different algorithms since we understand that other authors previously did it. Instead, we assume that PPO algorithm is suitable for our proposed framework due to its proven effectiveness and stability.

4.4 Hierarchical Deep Reinforcement Learning Solution

As aforementioned, although ρ and α are coupled variables in problem **P**, there is a hierarchical nature in performing each action at each time step. Notably, the UAV should first move to the served cluster and collect the tasks produced by the served IoTDs, and only then offload the data to the MEC server, as depicted in Figure 4.5.

Therefore, recognizing the hierarchical nature of problem **P**, we propose to decouple it into two sub-problems and utilize a hierarchical DRL based algorithm for both of them.

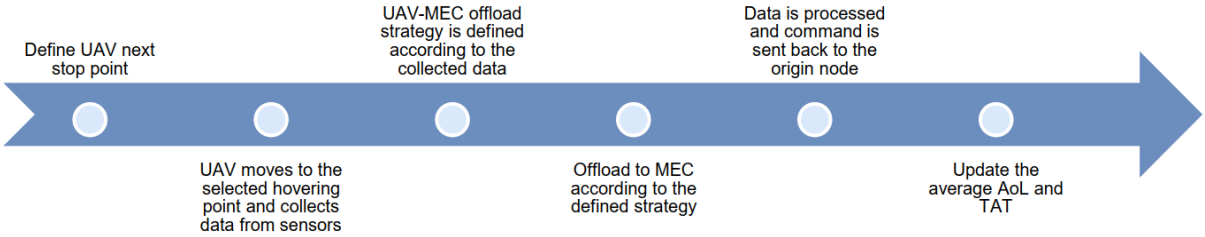


Figure 4.5: Proposed process sequence in the hierarchical solution.

4.4.1 Problem Decoupling

From (4.15), it is explicit that the only form to reduce the AoL for each IoTD is through ψ_i^n. Notably, the collection of fresher packets evidently diminishes the AoL. However, there is a limit to how much it can be reduced. If the UAV serves a cluster recently visited, the AoL reduction is less than when visiting a long-neglected cluster. Essentially, the aim is to maximize the difference between the current average generation time (Ψ_k^n) and the previous average generation time (Ψ_k^{n-1}) to encourage visits to clusters not recently attended to and collect fresher data. Accordingly, we formulate the first sub-problem as:

$$\begin{aligned} \mathbf{SP1}: \max_{\rho^n} \ & \Psi_k^n - \Psi_k^{n-1} \\ & \text{s.t. } \mathcal{C}1 \text{ and } \mathcal{C}2. \end{aligned} \tag{4.21}$$

Next, the agent must define an offloading strategy for the collected tasks. Therefore, we define the next optimization problem in the same way as $\mathbf{P}$, however, without including ρ^n as a decision variable:

$$\begin{aligned} \mathbf{SP2}: \min_{\alpha^n} \ & \left[\frac{\vartheta}{\Theta^A} \bar{\Delta}[n] + \frac{(1-\vartheta)}{\Theta^T} \bar{\Omega}[n] \right] \\ & \text{s.t. } \mathcal{C}3, \mathcal{C}4, \mathcal{C}5, \mathcal{C}6. \end{aligned} \tag{4.22}$$

It is worth mentioning that in **SP2**, the values of α^n and $\mathcal{I}^n$ are constrained by $\mathcal{C}5$ and $\mathcal{C}6$, respectively, according to ρ^n defined in **SP1**. In addition, two independent controllers will be necessary: one for trajectory design, and another one for offloading strategy definition.

4.4.2 Hierarchical Proximal Policy Optimization

To solve both **SP1** and **SP2**, we employ a hierarchical-based approach, Hierarchical Proximal Policy Optimization (HPPO), where the agent can learn different policies at different levels of abstraction and resolve the problem using different PPO controllers. This helps to handle the complexity of decision-making and capture dependencies between different stages of the process.

In the HPPO solution, there is a new level of step time discretization, the sub-steps $\dot{n}$ and $\ddot{n}$. Each sub-step happens sequentially and once per step. The details for each sub-step are presented below.

Sub-step $\dot{n}$

At the beginning of $\dot{n}$, the current environment state is collected and, similarly to equation 4.18, comprises the following information:

- The AoL of each IoTD: $\Delta^{\dot{n}}_{i\in\mathcal{I}} = \Delta^{n}_{i\in\mathcal{I}}$;
- The distance from each cluster head to the UAV: $d^{\dot{n}}_{ku\in\mathcal{K}} = d^{n-1}_{ku\in\mathcal{K}}$;
- The generation time of the latest collected task of each IoTD: $\psi^{\dot{n}}_{i\in\mathcal{I}} = \psi^{n-1}_{i\in\mathcal{I}}$;
- The previous number of tasks collected of each IoTD: $b^{\dot{n}}_{i\in\mathcal{I}} = b^{n-1}_{i\in\mathcal{I}}$.

It is worth mentioning that, at this point of the sub-step, the agent (i.e., the UAV) does not know about the specific information of each IoTD. All the information at this point is generic (related to clusters, not IoTDs) and is all based on information collected and accumulated based on previous steps.

With this, the agent's trajectory learning controller decides the next UAV position based on the current environment state, defining which cluster should be served. The sub-step $\dot{n}$ ends when the UAV has collected all the tasks from the served IoTDs, at which point the instantaneous AoL of each IoTD is updated to $\Delta^{\ddot{n}}_{i\in\mathcal{I}} = \Delta^{n}_{i\in\mathcal{I}} + t^{n}_{\text{fly}} + t^{n}_{\text{col}}$.

Sub-step $\ddot{n}$

The second sub-step, $\ddot{n}$, begins with the updated UAV position and with the knowledge of the collected packets in $\dot{n}$. Therefore, the following information can be collected from the environment at the beginning of $\ddot{n}$:

- UAV position: $c_u^{\ddot{n}} = c_u^n$;
- Number of collected packets: $\psi_{i\in\mathcal{I}}^{\ddot{n}} = \psi_{i\in\mathcal{I}}^n$;
- Generation time of collected tasks: $b_{i\in\mathcal{I}}^{\ddot{n}} = b_{i\in\mathcal{I}}^n$.

The offloading controller receives the environment's updated state. Obtaining more specific information about the collected packets is possible since the agent has already collected the IoTDs' packets. Finally, the UAV-MEC offloading decision is made.

HPPO Diagram and Algorithm

The process for HPPO is summarized in Figure 4.6 and Algorithm 2, where it is evident how the trajectory and offload controllers are decoupled, where each one has its own experience buffer (named $\mathcal{B}^T$ and $\mathcal{B}^O$, respectively), actor and critic networks, and independent policy updates. Similarly to PPO, the parameters of the DNNs are updated with K epochs and M mini-batches.

4.4.3 MDP Formulation - Trajectory Controller

Given that there is one independent controller for each decision, the MDP formulation for our HPPO-based algorithm is conducted as two separate MDPs, one for each controller. The MDP for the trajectory controller is hence defined as:

State

We define the state space as the average AoL of each cluster ($\Delta_k^n = 1/I_k \sum_{i\in\mathcal{I}_k} \Delta_i^n$), the distance from each cluster head to the UAV, the average of the generation time of the most recently collected task of each cluster ($\Psi_k^n = 1/I_k \sum_{i\in\mathcal{I}_k} \psi_i^n$), and the number of packets collected. Therefore, we have:

$$\mathbf{s}'_n = \{\Delta_{k\in\mathcal{K}}^n, d_{ku\in\mathcal{K}}^n, \Psi_{k\in\mathcal{K}}^n, \sum_{i\in\mathcal{I}} b_i^n\}, \forall k \in \mathcal{K}, n \in \mathcal{N}. \tag{4.23}$$

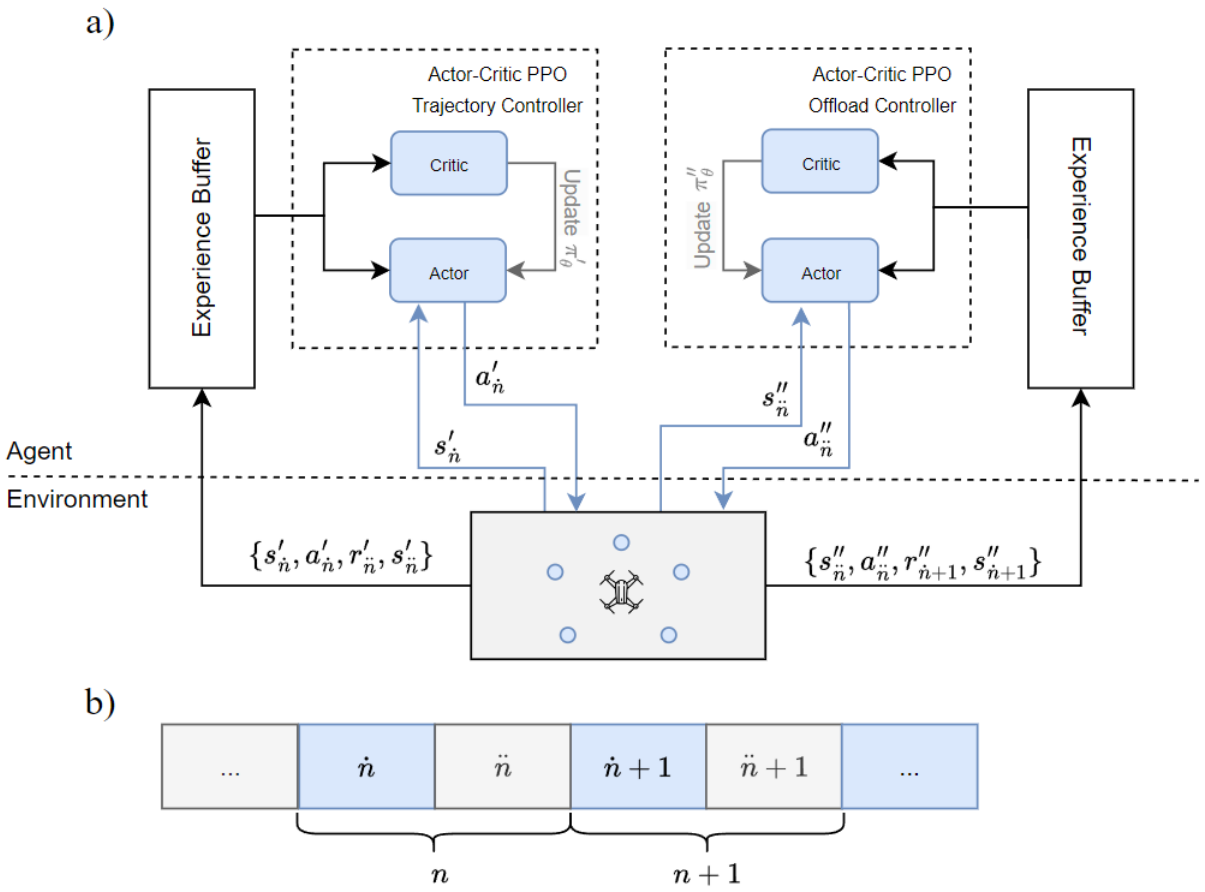


Figure 4.6: Hierarchical Actor-Critic PPO strategy. a) Diagram detailing the updates of states between trajectory and offloading controllers. b) Relation of sub-steps $\dot{n}$ and $\ddot{n}$ and the step n over time.

Action

Denoting the action space as the cluster assignment decision, we have the action space for the agent defined as:

$$\mathbf{a}'_n = \{\rho^n\}. \tag{4.24}$$

Algorithm 2 Hierachical-based PPO solution

Initialize actor and critic network parameters of trajectory controller θ_0^a and θ_0^c
Initialize actor and critic network parameters of offload controller ϕ_0^a and ϕ_0^c
for episodes $= 1, \ldots, E$ **do**
    Initialize buffers $\mathcal{B}^T$ and $\mathcal{B}^O$
    for timestep $n = 0, \ldots, N-1$ **do**
        Observe state s'_n
        Using the current policy $\pi_{\theta^a}(a_n|s_n)$, select action $a'_{\dot{n}}$
        Run $a'_{\dot{n}}$, go to next state $s'_{\ddot{n}}$ and receive reward $r'_{\ddot{n}}$
        Store $a'_{\dot{n}}$, $s'_{\dot{n}}$, $s'_{\ddot{n}}$ and $r'_{\ddot{n}}$ in buffer $\mathcal{B}^T$
        Observe updated state $s''_{\ddot{n}}$
        Using the current policy $\pi_{\phi^a}(a_n|s_n)$, select action $a''_{\ddot{n}}$
        Run $a''_{\dot{n}}$, go to next state $s''_{\dot{n}+1}$ and receive reward $r''_{\dot{+}1}$
        Store $a''_{\dot{n}}$, $s''_{\dot{n}}$, $s''_{\dot{n}+1}$ and $r''_{\dot{n}+1}$ in buffer $\mathcal{B}^O$
    end for
    Compute advantage estimates $\hat{A}_n^T$ using GAE [53]
    Update θ^a according to 2.17
    Update θ^c according to 2.18
    Clear buffer $\mathcal{B}^T$
    Compute advantage estimates $\hat{A}_n^O$ using GAE [53]
    Update ϕ^a according to 2.17
    Update ϕ^c according to 2.18
    Clear buffer $\mathcal{B}^O$
end for

Reward

Aiming to optimize problem **SP1**, we take the immediate reward as:

$$\mathbf{r}'_n = \Psi_k^n - \Psi_k^{n-1}. \tag{4.25}$$

4.4.4 MDP Formulation - Offloading Controller

State

It is defined as the average AoL of each served IoTD, the generation time of each collected task and the number of packets collected in the step:

$$\mathbf{s}_n'' = \{\Delta^n_{i\in\mathcal{I}^n}, \psi^n_{i\in\mathcal{I}}, b^n_{i\in\mathcal{I}^n}\}. \tag{4.26}$$

Action

We denote the action space as the offloading decision:

$$\mathbf{a}_n'' = \{\alpha^n\}. \tag{4.27}$$

Reward

In order to maximize the system cost we define the instantaneous reward according to **SP2**, or

$$\mathbf{r}_n'' = -\left[\frac{\vartheta}{\Theta^A}\bar{\Delta}[n] + \frac{(1-\vartheta)}{\Theta^T}\bar{\Omega}[n]\right]. \tag{4.28}$$

Chapter 5

Simulation and Results

In this chapter, we detail the simulation parameters and scenario. With the results of extensive simulations, we analyze the proposed solution for problem **P** with different approaches, analyzing the advantages and cons of each and examining the trade-off between AoL and TAT.

5.1 Simulation Platform

To simulate the UAV-aided MEC smart farm system, we developed the system model detailed in Chapter 4 using an OpenAI gym module [87]. In addition, the package PyTorch [88] was used to build the neural network solutions for our algorithms. The hyperparameter tuning for the machine learning model was performed with the Optuna framework [89]. All the code produced for this work was written using Python.

5.2 Simulation Settings

5.2.1 PPO Architecture Implementation

The controller architectures were implemented based on the parameters listed in Table 5.1. Both the PPO and HPPO algorithms used the same parameters. The key difference, as shown in Figures 4.4 and 4.6, is that the PPO-based algorithm uses a single controller, while the HPPO-based solution uses two controllers. From this point on, we refer to the

models as ML-PPO and ML-HPPO in order to differentiate them from the heuristic models detailed in the following sessions.

Table 5.1: Algorithm Main Hyperparameters

Neural Networks Parameters	**Value**
Actor hidden dimension	128
Actor hidden layers	3
Actor activation function	Tanh
Actor optimizer	Adam
Actor learning rate	$3 \cdot 10^{-5}$
Critic hidden dimension	128
Critic hidden layers	3
Critic activation function	ReLU
Critic optimizer	Adam
Critic learning rate	$9 \cdot 10^{-5}$
PPO Parameters	**Value**
Number of episodes (E)	15000
Trajectory length (N)	32
Mini-batches (M)	4
Epochs (K)	10
Clipping coefficient (ϵ)	0.2
Entropy coefficient (c)	$5 \cdot 10^{-3}$

5.2.2 Environment Configuration

Learning Phase

Both ML-PPO and ML-HPPO models were trained using a simulated environment with parameters summarized in Table 5.2. Specifically, there are 5 clusters with 6 IoTDs each. The size of each packet varies randomly between 256 and 512 Kb and, as pointed in [73], the amount of CPU cycles to process each packet is $C_i = 1300$ x ϕ_i. The CPU-cycle frequency of the UAV and at the MEC server is $f_{\text{UAV}} = 2$ GHz and $f_{\text{MEC}} = 20$ GHz, respectively, as suggested in [75], and the UAV speed is $s_u = 30$ m/s [71]. The packet arrival rate at each

IoTD depends on the cluster it is at, which is represented as $\lambda_{k \in \mathcal{K}} = \gamma/(10 + 30k)$, where $\gamma = 5$ is the arrival rate multiplier used for model training.

In addition, we aimed to simulate various network conditions by using different power spectral densities of AWGN to train both models. These varying network conditions influence mainly the UAV-MEC offloading decisions. Better network conditions lead to more optimal decisions when offloading to the MEC server since the transmission delay is reduced. Specifically, we employ $N_o \in \{-167, -171, -175, -179, -183\}$ dBm/Hz.

Finally, the weight and normalization values ϑ, Θ^A and Θ^T were chosen through extensive simulations aiming the best balance between average AoL and average TAT. Their values are also used during the testing phase.

Table 5.2: Training Environment Parameters

Variable	**Value**	**Description**
K	5	Number of clusters
I_k	6	Number of IoTDs per cluster
I	30	Total number of IoTDs
s_u	30 m/s	Speed of UAV
h_u	5 m	Height of UAV
f_{UAV}	2 GHz	CPU-cycle frequency at UAV
f_{MEC}	20 GHz	CPU-cycle frequency at MEC server
γ	5	Data arrival rate multiplier
ϕ_i	[256, 512] Kb	Packet size of IoTD i
C_i	1300 x ϕ_i	CPU-cycles necessary to process each packet
Θ^A	50	Normalization factor for AoL
Θ^T	0.1	Normalization factor for TAT
ϑ	0.96	Weight balance for AoL and TAT

Test Sets

In order to test the trained models, a similar configuration was used for testing, except for varying γ, f_{MEC} and C_i. A summary of the values used in the tests is presented next, in Table 5.3.

The objective of testing with different values of γ is to simulate an increase in the data collection rate and analyze the resulting average AoL and average TAT. By varying

Table 5.3: List of Test Sets

	γ	$f_{\mathbf{MEC}}$ **(GHz)**	C_i **(x** ϕ_i**)**
Dynamic Data Generation Rate	[10, 15, 20, 25]	20	1300
Variation in Computational Capability at MEC	15	[5, 10, 20, 30, 40]	1300
Variation in Packet Workload	15	20	[325, 650, 1300, 2600]

γ, we can observe how changes in data collection frequency affect system performance metrics. Higher values of γ represent more frequent data collection in the IoTDs and a potential reduction in AoL, but may also increase the load on the system, potentially affecting processing and offloading efficiency, especially considering that the packets will accumulate in the origin node until being collected by the UAV.

In addition, the goal of varying f_{MEC} is to simulate different levels of computational resource availability. As aforementioned, we assume that the MEC server is responsible for processing data from various sources, not just interacting with the UAV. While the processing of the data collected by the UAV is given the highest priority in this system, the availability of computational resources can fluctuate based on the demand from other devices and applications, and these fluctuations are unknown by the RL agent, located at the UAV. By adjusting f_{MEC}, we can model scenarios where computational resources are either abundant or scarce. When resources are abundant, the MEC server can allocate more power to processing data sent from the UAV, leading to faster processing times and potentially lowering the average AoL and TAT. Conversely, during periods of high demand and processing congestion, the resources available may decrease, which can negatively impact these performance metrics.

Finally, as pointed out by Miettinen and Nurminen in [90], the amount of CPU cycles required to process a task depends significantly on the specific application. This variability highlights the need to evaluate how our solutions handle different workloads and applications with varying values of C_i. For instance, image processing tasks might require more CPU cycles compared to simple data aggregation tasks. Evaluating our solutions under different workloads is necessary for understanding their robustness and efficiency across a

wide range of applications without the need to retrain the ML agent.

5.2.3 Baselines

As heuristic baselines for the simulations, we consider the UAV trajectory to follow the shortest path using a round-robin policy, where each cluster is visited in sequence. The UAV-MEC offloading strategy varies and, based on that, we define the heuristic baselines:

- **Full local processing (H-SPL):** In this strategy, all data processing tasks are handled locally by the UAV, without offloading any tasks to the MEC server. This approach tests the system's performance when relying solely on the UAV's computational resources.
- **Full MEC processing (H-SPM):** Here, all data processing tasks are offloaded to the MEC server. This strategy assesses the system's performance when the UAV is employed solely as a relay and fully leveraging the MEC server's computational capabilities.
- **Offloading half of the tasks to MEC (H-SPH):** This mixed approach offloads half of the data processing tasks to the MEC server while processing the other half locally on the UAV. It aims to balance the computational load between the UAV and the MEC server.

Finally, as aforementioned, we name the optimization with a simultaneous decision over ρ and α as ML-PPO and the hierarchical optimization as ML-HPPO.

5.3 Simulation Results

5.3.1 Convergence Analysis

Initially, we present the reward convergence curve in the training scenario, illustrated in Fig. 5.1. To produce these results, we ran five simulations with 15,000 episodes each, where each episode simulated 500 seconds of the environment. The curves are the average reward among these five simulation runs, and the shaded region indicates the confidence interval for each point across 50 episodes. The reward is calculated according to 4.20 and 4.28 for ML-PPO and ML-HPPO, respectively.

While ML-HPPO's convergence is slower than that of ML-PPO, it is smoother and exhibits less variability across different simulation runs while achieving similar average convergence values. This indicates that the hierarchical strategy is more robust to variations in network conditions. This can be explained by the fact that dividing the decision-making process reduces its complexity, leading to smoother learning.

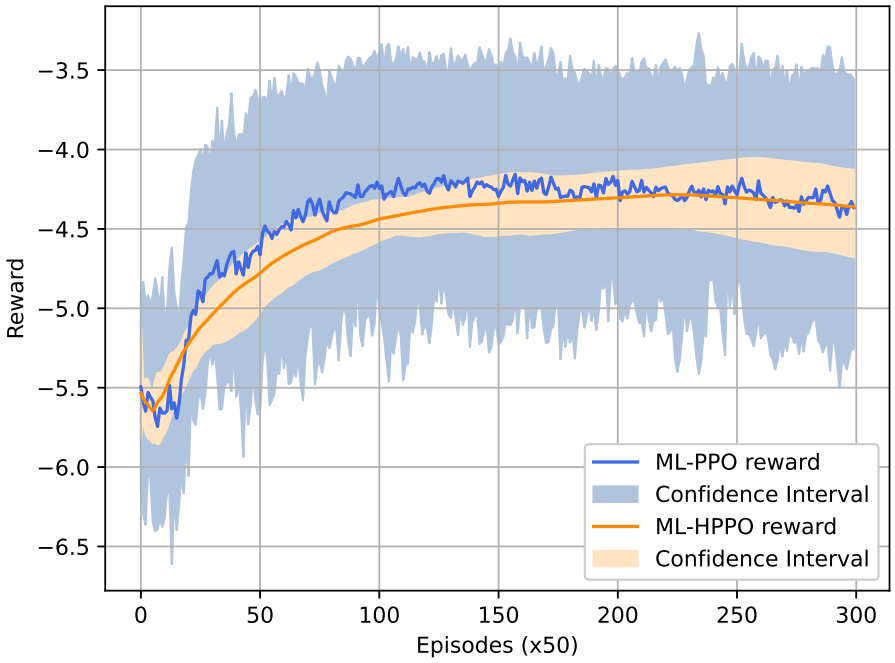


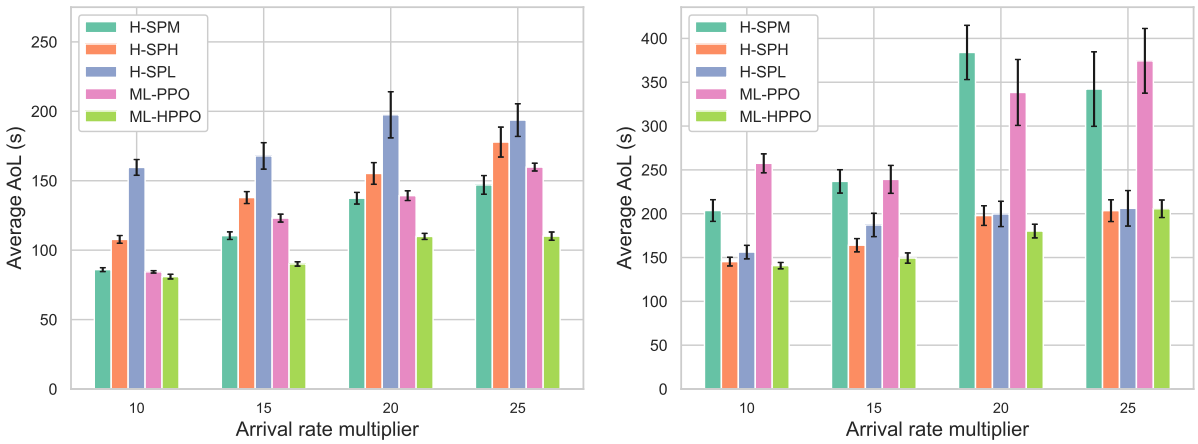
Figure 5.1: Reward convergence curve of PPO and HPPO.

5.3.2 Dynamic Data Generation Rate

Average AoL

In Figure 5.2a and Figure 5.2b, it is depicted the average AoL results for different network conditions when varying the IoTDs' arrival rate multiplier (γ). Specifically, we use the best and worst trained case scenarios, $N_o = -183$ dBm/Hz and $N_o = -167$ dBm/Hz, respectively.

1) Overall analysis over different network conditions



(a) Average AoL, $N_o = -183$ dBm/Hz.

(b) Average AoL, $N_o = -167$ dBm/Hz.

Figure 5.2: Results for average AoL varying the IoTDs' arrival rate.

Comparing the results obtained for $N_o = -183$ dBm/Hz with those for $N_o = -167$ dBm/Hz, we notice that for all test cases, offloading tasks to the faster processor in the MEC server outcomes in higher average AoL when the network noise is higher. Overall, it is a disadvantage to do so when the network conditions are poor (i.e., N_o is higher) because the transmission delay is higher, affecting the time necessary for the packets to arrive at the border of the farm and the faster data processing do not counterbalance the higher delay. For instance, when comparing the heuristic approaches, H-SPM achieves a lower average AoL in better network conditions ($N_o = -183$ dBm/Hz). In this case, offloading all collected tasks to the faster processor at the MEC is the best action because the much lower processing delay compensates the relatively low transmission delay. In contrast, H-SPH (i.e., sending half of the collected tasks to the MEC) achieves better AoL performance in a more noisy environment as illustrated in Figure 5.2b. Notably, its performance is even better than the one obtained with H-SPL, when all tasks are processed at the UAV, highlighting the need to split the tasks among the available processors. In other words, it indicates that a balance between local processing at the UAV and at the MEC server is an efficient strategy to improve the freshness of information.

2) ML techniques analysis

In Figure 5.2a, we see how ML-PPO shows that an adequate trajectory decision is an efficient strategy to decrease the average AoL. The algorithm's performance is efficient, being better than the heuristic baselines with $\gamma = 10$. However, when γ increases, the ML-PPO performance decreases, being worse than H-SPM. It shows that ML-PPO is not

optimized for increased arrival rates, being optimized for lower rates. In addition, ML-PPO presents higher values of AoL under poorer network conditions, indicating a preference to offload packets to the MEC server, even when there is a risk of a higher transmission delay and, hence, a higher average AoL. Therefore, its trajectory and offloading strategy are not optimized for poorer network conditions.

ML-HPPO, with more robust and smooth learning, presents an appropriate solution for any network condition. Noticeably, our hierarchical approach achieved the lowest AoL over all the tested methods. Remarkably, we notice that ML-HPPO achieved an average of 23% ($N_o = -183$ dBm/Hz) and 44% ($N_o = -167$ dBm/Hz) lower AoL with respect to ML-PPO. Moreover, it enabled an average of 17% AoL reduction compared to H-SPM in $N_o = -183$ dBm/Hz scenario and a 5% AoL reduction with respect to H-SPH in the $N_o = -167$ dBm/Hz scenario.

Essentially, dividing each step into two parts allows for the customization of a more specific reward function for the trajectory controller and establishing a more precise state space for the offloading controller. This configuration accommodates the trajectory definition with an indirect objective for AoL reduction: collect fresher data. As the arrival rate increases proportionally to the trained arrival rate, the agent can collect even fresher packets at each visit. Nevertheless, with the knowledge about the collected packets and the updated position of the UAV, the agent defines an efficient UAV-MEC offloading strategy, thus further decreasing AoL.

Average TAT

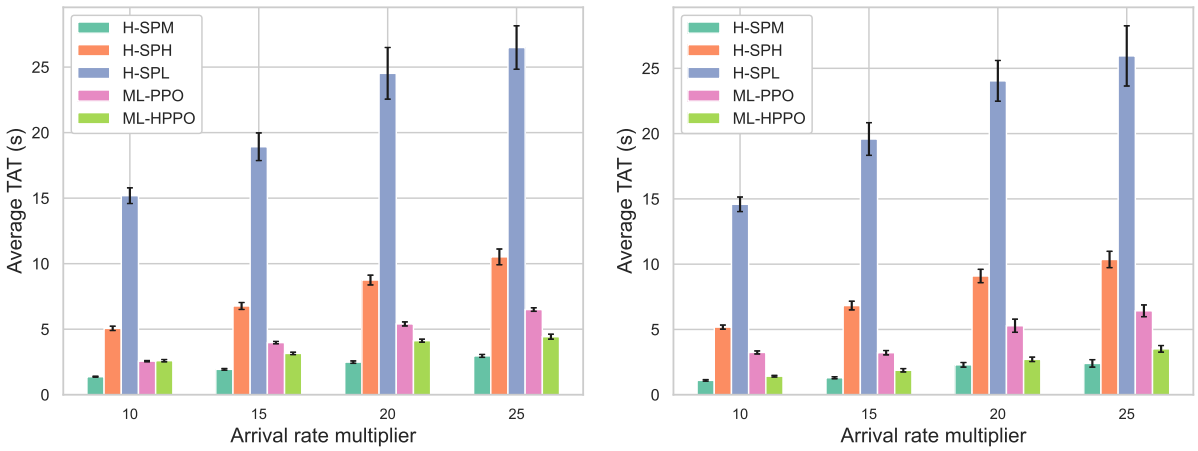
In Figure 5.3a and Figure 5.3b, we investigate the TAT when varying the arrival rate.

1) Overall analysis over different network conditions

In this case, f_{MEC} is ten times higher than f_{UAV} and, because the TAT only considers the CPU waiting and processing delays, H-SPM (when all tasks are offloaded to the MEC) shows the lowest TAT for all the analyzed cases. This is because the MEC server, with its significantly higher computational power, can process tasks much more quickly than the local CPU on the UAV.

Additionally, TAT increases with γ because there are more packets to process per step. Higher values of γ result in a greater volume of data being collected and processed within each time step. This increase in workload naturally leads to longer processing times, as the system has to handle a larger number of packets.

It is also worth mentioning that the TAT for H-SPH is closer to H-SPM than to H-SPL, indicating that splitting the tasks among available processors can effectively reduce TAT.



(a) Average TAT, $N_o = -183$ dBm/Hz.

(b) Average TAT, $N_o = -167$ dBm/Hz.

Figure 5.3: Results for average TAT varying the IoTDs' arrival rate.

This finding is consistent with our observations in the AoL case, where task division also proved beneficial.

Finally, the trade-off between AoL and TAT becomes more evident in the case of $N_o = -167$ dBm/Hz. In this scenario, H-SPM results in the highest AoL but the lowest TAT. Conversely, H-SPL and H-SPH show lower AoL values but higher TAT values. Therefore, we conclude that reducing one variable does not necessarily lead to a reduction in the other.

2) ML techniques analysis

For the reasons previously discussed, H-SPM is the best case, where TAT is the lowest. However, ML-PPO outperforms both H-SPH and H-SPL. On average, TAT with ML-PPO strategy is 42% better than H-SPH and 78% better than H-SPL in both network scenarios.

Although the average AoL is similar for ML-HPPO, H-SPH and H-SPL in specific cases (especially for $N_o = -167$ dBm/Hz), upon analyzing the TAT results, ML-HPPO consistently outperforms H-SPH and H-SPL, demonstrating a favourable trade-off between TAT and AoL. Specifically, for $N_o = -183$ dBm/Hz case, ML-HPPO enabled a TAT reduction of 53% and 83% with respect to H-SPH and H-SPL, respectively on average. In $N_o = -167$ dBm/Hz scenario, the decrease was 70% and 89%.

Moreover, ML-HPPO enables a 19% and 48% lower TAT compared to ML-PPO, demonstrating that the first provides a more efficient solution. A more effective trajectory strategy, learned to optimize fresher packet collection, efficiently reduces both AoL and TAT

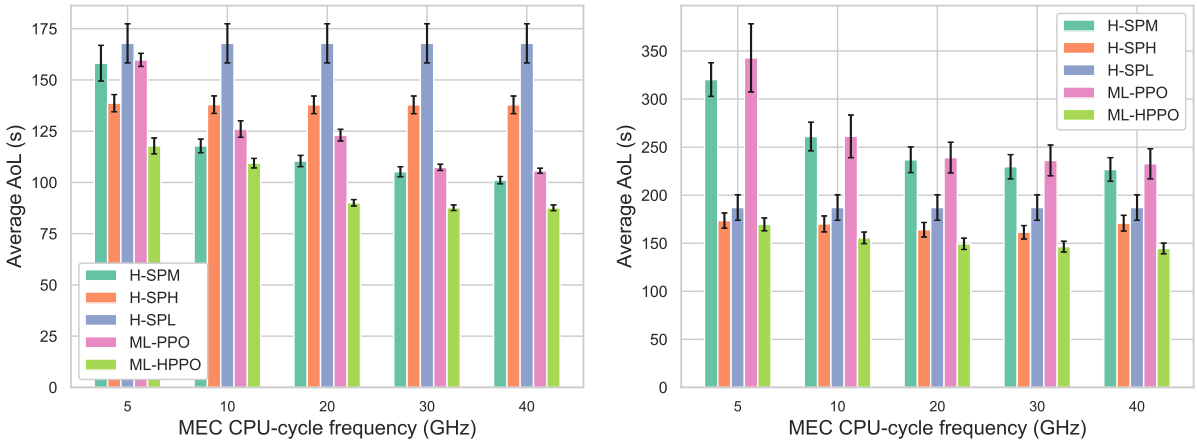
because the UAV collects fewer and smaller tasks. To elaborate, a shorter time between visits results in less amount of collected packets at the IoTDs, thus reducing the waiting and processing times and, therefore, the TAT.

5.3.3 Variation in Computational Capability at MEC

We considered a scenario where the CPU-cycle frequency at the MEC server changes, resulting in variable resources to the UAV-collected packet processing, with a simulated arrival rate multiplier of $\gamma = 15$ and a workload per packet of $C_i = 1300$ x ϕ_i $\forall i \in \mathcal{I}$.

Average AoL

In Figure 5.4a and Figure 5.4b, we evaluate the AoL results for $N_o = -183$ dBm/Hz and $N_o = -167$ dBm/Hz, respectively.



(a) Average AoL, $N_o = -183$ dBm/Hz.

(b) Average AoL, $N_o = -167$ dBm/Hz.

Figure 5.4: Results for average AoL under varying MEC CPU-cycle frequency.

1) Overall analysis over different network conditions

First, it is evident that the values for H-SPL remain constant across all test cases since the parameters of the MEC server do not influence the UAV's processing performance. For the other cases, there is a noticeable overall reduction in the average AoL as computational capability at the MEC server increases. This is an expected result, because the tasks can

be processed faster, reducing the time necessary to generate the resulting command and hence improving the information freshness at the origin nodes.

In the case of $N_o = -183$ dBm/Hz, among the heuristic baselines, H-SPM generally achieves the lowest AoL values, except when $f_{\text{MEC}} = 5$ GHz, where H-SPH (which divides tasks between the MEC server and UAV processors) performs best. This indicates that under better network conditions, offloading all tasks to the MEC server efficiently reduces AoL when the MEC's processing capabilities are approximately five times higher than those of the UAV. The higher processing speed at the MEC compensates for the transmission delay. However, when the MEC's processing capabilities are closer to those of the UAV ($f_{\text{UAV}} = 2$ GHz), it is more effective to split the tasks among the available processors (H-SPH), resulting in the lowest AoL among the heuristic baselines.

In the case of $N_o = -167$ dBm/Hz, H-SPM's performance is significantly higher than other baselines due to the higher transmission delay, indicating that not even a faster processor can counterbalance it in order to decrease the average AoL. In this scenario, the achieved average AoL for H-SPL is around 10% lower when compared to H-SPL.

2) ML techniques analysis

ML-PPO significantly benefits from increased processing capability at the MEC, reducing the average AoL by 33% for $N_o = -183$ dBm/Hz and by 32% for $N_o = -167$ dBm/Hz when comparing the $f_{\text{MEC}} = 5$ GHz and $f_{\text{MEC}} = 40$ GHz scenarios. However, the AoL is noticeably high when f_{MEC} is lower than the trained case of $f_{\text{MEC}} = 20$ GHz, indicating poor adaptability to conditions that differ greatly from the training scenario.

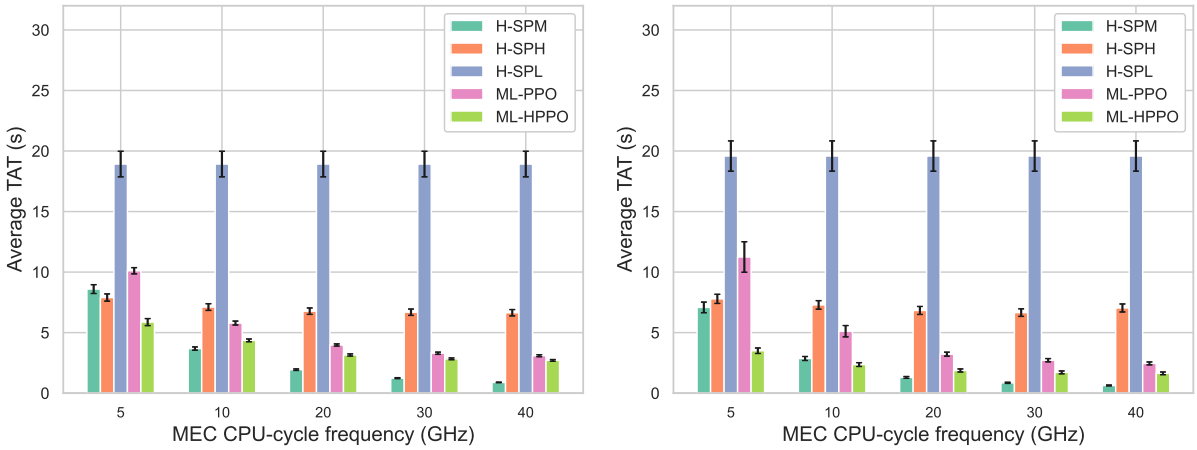
The same issue observed in the first test (varying data generation rate) recurs here for ML-PPO. The AoL is very high for the $N_o = -167$ dBm/Hz case, achieving similar results to H-SPM, suggesting that ML-PPO struggles under these conditions.

Nevertheless, ML-HPPO achieves the lowest AoL in all cases, with an average of 31% and 46% reduction AoL compared to ML-PPO according to Figures 5.4a and 5.4b, respectively. Moreover, compared to the second best scenario (H-SPM for -183 dBm/Hz; H-SPH for -167 dBm/Hz), the improvement on AoL was 25% and 9%.

These results corroborate the idea that ML-HPPO has a better generalization capability than ML-PPO, making better use of the available resources to increase information freshness.

Average TAT

In Figures 5.5a and 5.5b we see the TAT results for the case when we are varying the computational capability at the MEC server.



(a) Average TAT, $N_o = -183$ dBm/Hz.

(b) Average TAT, $N_o = -167$ dBm/Hz.

Figure 5.5: Results for average TAT under varying MEC CPU-cycle frequency.

1) Overall analysis over different network conditions

The baseline results for TAT are largely consistent with those observed for AoL. H-SPL shows the same TAT across all test scenarios, and starting from $f_{\text{MEC}} = 10$ GHz in the $N_o = -183$ dBm/Hz case, H-SPM achieves the best results. With this, we can observe that the main differences in AoL are driven by variations in TAT.

It is also noticeable that for H-SPM, TAT is reduced by 36% when moving from $N_o = -183$ dBm/Hz to $N_o = -167$ dBm/Hz. This is because the higher transmission delay in poorer network conditions increases the likelihood of reducing processing waiting time, thus decreasing TAT. This relevant difference between network conditions is not observed for H-SPL and H-SPH.

2) ML techniques analysis

In Figures 5.5a and 5.5b, we observe that ML-HPPO achieves an average of 23% and 47% lower TAT compared to ML-PPO, respectively.

Additionally, ML-HPPO reduces TAT by 32% ($N_o = -183$ dBm/Hz, $f_{\text{MEC}} = 5$ GHz) and 51% and 18% ($N_o = -167$ dBm/Hz, $f_{\text{MEC}} = 5$ and $f_{\text{MEC}} = 10$ GHz) compared to H-SPM. This demonstrates that the ML-HPPO solution optimizes the use of computational

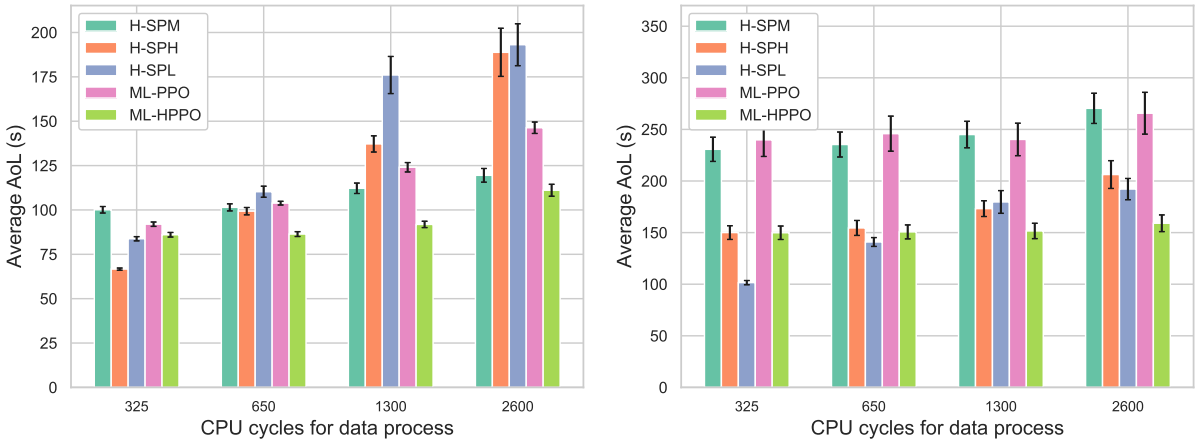
resources in highly competitive scenarios, where fewer computational resources are available at the MEC (i.e., when f_{MEC} is lower). Overall, ML-HPPO presents an excellent trade-off between AoL and TAT.

5.3.4 Variation in Packet Workload

Lastly, we analyze the performance of the proposed solutions when there are different packet workloads, i.e., when we change the value of C_i $\forall i \in \mathcal{I}$. We consider the data arrival rate as $\gamma = 15$ and the processing speed at MEC server as $f_{\text{MEC}} = 20$ GHz.

Average AoL

Figures 5.6a and 5.6b illustrate the results for AoL obtained for $N_o = -183$ dBm/Hz and $N_o = -167$ dBm/Hz, respectively.



(a) Average AoL, $N_o = -183$ dBm/Hz. (b) Average AoL, $N_o = -167$ dBm/Hz.

Figure 5.6: Results for average AoL when varying the packet CPU-cycle frequency.

1) Overall analysis over different network conditions

For $N_o = -183$ dBm/Hz, we see that the H-SPM strategy is inefficient in obtaining fresher commands (i.e., smaller AoL) for small values of C_i. Despite the faster processing at the MEC, the faster processing times mean that the higher transmission delay is not justified. This changes when C_i increases and offloading more tasks to the MEC server

becomes better since the faster processing time compensates for the higher transmission delay.

For $N_o = -167$ dBm/Hz, H-SPL achieves lower AoL, especially for smaller values of C_i. This is because the UAV can handle simpler packets (i.e., with smaller C_i) fast enough with reduced transmission delay. Previous cases show that offloading to the MEC server is not ideal under poor network conditions. Instead, processing all packets at the UAV when the tasks are the simplest ($C_i = 325$) or offloading half to the MEC (in other cases) is the best approach to achieve the lowest average AoL under poor network conditions.

2) ML techniques analysis

For $N_o = -183$ dBm/Hz case, the AoL is on average 20% lower for ML-HPPO with respect to ML-PPO. For $N_o = -167$ dBm/Hz, the AoL reduction was, on average, 38%, corroborating with the idea that the ML-PPO strategy does not work well under poor network conditions. More specifically, the solution for UAV trajectory and UAV-MEC offloading achieved with ML-HPPO is more robust and applicable to different scenarios when compared to ML-PPO.

Average TAT

In Figure 5.7a and Figure 5.7b, we investigate the TAT when varying C_i for $N_o = -183$ dBm/Hz and $N_o = -167$ dBm/Hz, respectively.

1) Overall analysis over different network conditions

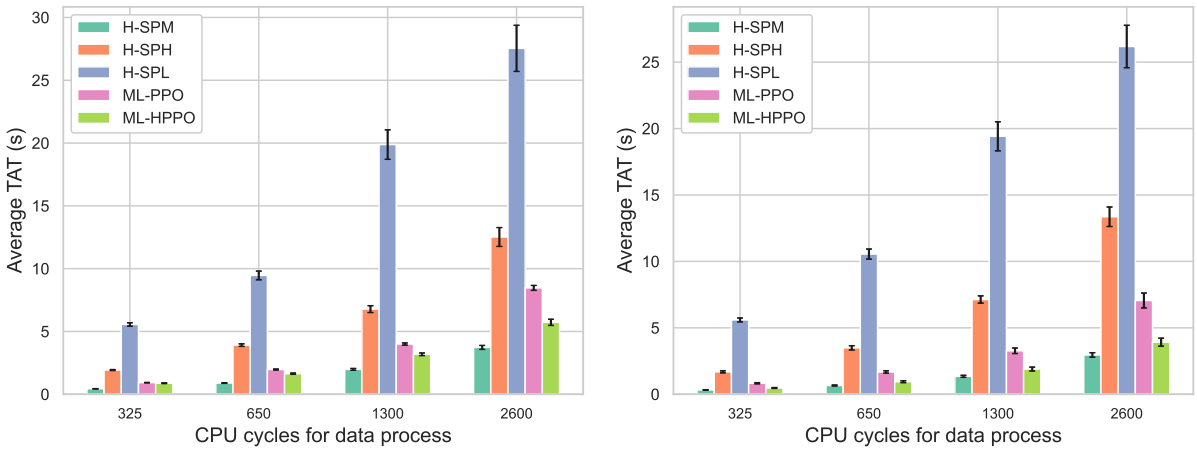
As expected and as seen in previous test cases, the average TAT is the smallest one for H-SPM, being higher for H-SPH and even higher for H-SPL. In addition, it grows with C_i because the task is more complex and more time is necessary to process it, which also increases the CPU waiting time. Finally, although the H-SPM strategy presents high values of AoL, it is always the smallest value of TAT, exemplifying the trade-off between those two metrics.

2) ML techniques analysis

ML-PPO and ML-HPPO have TAT close to H-SPM solution, showing a better trade-off between AoL and TAT when compared to H-SPH and H-SPL.

As aforementioned, H-SPH and H-SPL have an overall low AoL, especially for smaller values of C_i. However, even in these cases, the TAT is higher because it depends on the processing capability only, which is lower for the UAV.

To conclude, ML-HPPO resulted in a reduction of TAT in 25% for $N_o = -183$ dBm/Hz and 43% for $N_o = -167$ dBm/Hz when compared to ML-PPO, showing a robust and adaptable solution.



(a) Average TAT, $N_o = -183$ dBm/Hz.

(b) Average TAT, $N_o = -167$ dBm/Hz.

Figure 5.7: Results for average TAT when varying the packet CPU-cycle frequency.

5.4 Overview

From the analyzed results, we observed that ML-HPPO demonstrated excellent performance in scenarios that were not part of the training set. In contrast, ML-PPO performed well only in specific scenarios with lower arrival rates, better network conditions, higher CPU processing power at the MEC server and more complex tasks. Its performance performance was significantly compromised in poor network conditions.

The superior results for ML-HPPO can be mainly attributed to its adequate trajectory strategy, which ensures constant visits to all IoTD clusters and a shorter time between visits. Because the computational tasks are composed of the accumulated packets in each IoTD, this UAV trajectory approach results in smaller tasks and consequently lower average AoL and TAT. Additionally, its effective UAV-MEC offloading strategy enables a balanced use of the computational capabilities in both UAV and MEC server, counterbalancing AoL and TAT across various network conditions, computational capabilities and task complexities.

The results provide insight into the scalability of the solution. When dealing with large volumes of data, as shown in the arrival rate multiplier test case, the average AoL in the ML-PPO approach tends to degrade under poor network conditions, making other heuristic solutions, such as H-SPL and H-SPM, more effective, as illustrated in Figure 5.2b. However, even in this scenario, the average TAT is expected to remain low and comparable to H-SPM. Moreover, when the data becomes more complex, as demonstrated in the packet workload test case, the ML-HPPO solution proves advantageous for reducing average AoL due to its efficient offloading strategy. Compared to the baseline approaches, it remains an effective solution for maintaining low average TAT as well.

This demonstrates that ML-HPPO provides a robust and adaptable solution for UAV trajectory and UAV-MEC offloading, enabling efficient use of the computational resources while minimizing AoL and TAT across different scenarios.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In precision farming, timely and fresh information is vital for effective decision-making in irrigation and pest control processes. UAVs and MEC servers optimize data collection from ground sensors and data processing. However, integrating sensors, controllers, and actuators complicates data freshness in closed-loop communication. Computational resource optimization is also crucial, especially in energy-constrained equipment and competitive resource scenarios.

This work investigates the optimization of information freshness in terms of AoL and computational resource usage as TAT in a UAV-aided MEC system with a focus on smart agriculture applications. Aiming to design an optimal UAV trajectory for data gathering and UAV-MEC offloading strategy for the data processing, we formulate an optimization problem, discuss its hierarchical nature and decompose it in two sub-problems. Finally, we define one separated MDP for each sub-problem and solve them with a hierarchical PPO approach, referred to as ML-HPPO.

Our findings indicate that considering both UAV trajectory and offloading decisions simultaneously complicates the learning process, limiting the agent's ability to make optimal decisions due to the constrained information available at each step. In contrast, decoupling the problem and using a hierarchical DRL solution enables the agent to receive updated and specific information about the environment for each individual decision. This approach allows the agent to focus on and learn the best policies for trajectory and offloading decisions individually, thus optimizing its decision-making process. This high-

lights the importance of real-time, specific information about the environment for robust and generalizable decision-making.

As a result, with ML-HPPO, we achieved a robust and stable solution for both UAV trajectory and UAV-MEC offloading, enabling efficient use of computational resources while minimizing AoL and TAT across different scenarios. Our hierarchical DRL solution demonstrated superior and adaptable results compared to traditional approaches, particularly in new and dynamic scenarios.

6.2 Future Works

Despite the promising results, several challenges and open issues remain. One significant challenge is the energy consumption evaluation of the UAV. The energy required for communication, flight and computational processing must be carefully managed to ensure the longevity and efficiency of the equipment. Another open issue is the limited scalability of the current solution in multi-agent scenarios. The variability in network conditions and the computational demands of different tasks also add layers of complexity that need to be addressed to improve the adaptability and robustness of the system.

Future research can expand on this work by investigating the above open challenges and issues. Regarding the consideration of energy consumption of UAVs, more specific and complex decisions regarding the trajectory, offloading and computational processing can be made. For instance, the UAV path can be optimized regarding not only which IoT cluster to serve but also finding its optimal gathering point, while the UAV speed can also be optimized according to the current state of the environment. Additionally, offloading tasks to the MEC server needs to consider the energy the UAV spends for the transmission. We also recognize the limitations of the current UAV mobility model. Future works should consider the acceleration and deceleration stages for a more realistic representation. Finally, variable CPU-cycle frequency in both UAV and MEC server can be included as a decision variable to this problem. A higher computational speed results in an increased computational energy consumption while helping to reduce both AoL and TAT.

Another promising direction is the exploration of multi-agent scenarios involving multiple UAVs, which could enhance the system's efficiency and robustness. Handling multiple UAVs, coordinating their tasks, and offloading strategies presents a complex challenge requiring coordination algorithms and consideration of interference and variable signal quality. Moreover, comparing our hierarchical PPO approach with other DRL methodologies

also yields valuable insights and potentially leads to the development of even more efficient solutions.

Finally, integrating more sophisticated system models for network conditions and considering the variability in computational demands of different tasks could enhance the adaptability and scalability of the proposed system. Addressing these challenges will contribute to the advancement of UAV-MEC systems, making them more efficient and versatile in real-world applications.

References

[1] B. B. Sinha and R. Dhanalakshmi, "Recent advancements and challenges of internet of things in smart agriculture: A survey," *Future Generation Computer Systems*, vol. 126, pp. 169–184, 2022.

[2] Revanth, "Towards future farming: How artificial intelligence is transforming the agriculture industry," 2019. Available from https://www.wipro.com/holmes/towards-future-farming-how-artificial-intelligence-is-transforming-the-agriculture-industry/. Accessed on March 25, 2023.

[3] P. Gralla, "Precision agriculture yields higher profits, lower risks," 2018. Available from https://www.hpe.com/us/en/insights/articles/precision-agriculture-yields-higher-profits-lower-risks-1806.html. Accessed on March 25, 2023.

[4] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *2015 International Conference on Intelligent Environments*, pp. 140–143, 2015.

[5] B. S. Faiçal, H. Freitas, P. H. Gomes, L. Y. Mano, G. Pessin, A. C. de Carvalho, B. Krishnamachari, and J. Ueyama, "An adaptive approach for UAV-based pesticide spraying in dynamic environments," *Computers and Electronics in Agriculture*, vol. 138, pp. 210–223, 2017.

[6] C. W. Bac, E. J. Van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of field robotics*, vol. 31, no. 6, pp. 888–911, 2014.

[7] A. A. Araby, M. M. Abd Elhameed, N. M. Magdy, L. A. Said, N. Abdelaal, Y. T. Abd Allah, M. S. Darweesh, M. A. Fahim, and H. Mostafa, "Smart IoT monitoring system for agriculture with predictive analysis," in *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–4, 2019.

[8] T. Pamuklu, A. C. Nguyen, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "IoT-aerial base station task offloading with risk-sensitive reinforcement learning for smart agriculture," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 171–182, 2023.

[9] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2019.

[10] International Telecommunication Union, "Recommendation ITU-R M. 2083-0 (09/2015), IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond," tech. rep., International Telecommunication Union (ITU) Recommendation (ITU-R) M.2083-0, 2015.

[11] J. T. J. Penttinen, "On 6G visions and requirements," *Journal of ICT Standardization*, vol. 9, no. 3, pp. 311–325, 2021.

[12] C. Ge, N. Wang, I. Selinis, J. Cahill, M. Kavanagh, K. Liolis, C. Politis, J. Nunes, B. Evans, Y. Rahulan, N. Nouvel, M. Boutin, J. Desmauts, F. Arnal, S. Watts, and G. Poziopoulou, "QoE-assured live streaming via satellite backhaul in 5G networks," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 381–391, 2019.

[13] G. Minopoulos and K. E. Psannis, "Opportunities and challenges of tangible XR applications for 5G networks and beyond," *IEEE Consumer Electronics Magazine*, vol. 12, no. 6, pp. 9–19, 2023.

[14] F. Hamidi-Sepehr, M. Sajadieh, S. Panteleev, T. Islam, I. Karls, D. Chatterjee, and J. Ansari, "5G URLLC: Evolution of high-performance wireless networking for industrial automation," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 132–140, 2021.

[15] P. I. Tebe, G. Wen, J. Li, Y. Yang, W. Tian, J. Chong, and W. Zhang, "5G-enabled medical data transmission in mobile hospital systems," *IEEE Internet of Things Journal*, vol. 9, no. 15, pp. 13679–13693, 2022.

[16] H. Bagheri, M. Noor-A-Rahim, Z. Liu, H. Lee, D. Pesch, K. Moessner, and P. Xiao, "5G NR-V2X: Toward connected and cooperative autonomous driving," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, 2021.

[17] S. B. Damsgaard, N. J. Hernández Marcano, M. Nørremark, R. H. Jacobsen, I. Rodriguez, and P. Mogensen, "Wireless communications for internet of farming: An early 5G measurement study," *IEEE Access*, vol. 10, pp. 105263–105277, 2022.

[18] F. Nizzi, T. Pecorella, M. Bertini, R. Fantacci, M. Bastianini, C. Cerboni, A. Buzzigoli, M. Gattoni, and A. Fratini, "Evaluation of IoT and videosurveillance applications in a 5G smart city: the italian 5G experimentation in prato," in *2018 AEIT International Annual Conference*, pp. 1–6, 2018.

[19] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[20] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[21] F. Hussain, S. A. Hassan, R. Hussain, and E. Hossain, "Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1251–1275, 2020.

[22] M. Wang, Y. Sun, H. Sun, and B. Zhang, "Security issues on industrial internet of things: Overview and challenges," *Computers*, vol. 12, no. 12, p. 256, 2023.

[23] S. N. K. Veedu, M. Mozaffari, A. Höglund, E. A. Yavuz, T. Tirronen, J. Bergman, and Y.-P. E. Wang, "Toward smaller and lower-cost 5G devices with longer battery life: An overview of 3GPP release 17 RedCap," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 84–90, 2022.

[24] W.-L. Chen, Y.-B. Lin, Y.-W. Lin, R. Chen, J.-K. Liao, F.-L. Ng, Y.-Y. Chan, Y.-C. Liu, C.-C. Wang, C.-H. Chiu, and T.-H. Yen, "Agritalk: IoT for precision soil farming of turmeric cultivation," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5209–5223, 2019.

[25] Y. Wang, S. Wu, C. Lei, J. Jiao, and Q. Zhang, "A review on wireless networked control system: The communication perspective," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 01 2023.

[26] Y. Tang, S. Dananjayan, C. Hou, Q. Guo, S. Luo, and Y. He, "A survey on the 5G network and its impact on agriculture: Challenges and opportunities," *Computers and Electronics in Agriculture*, vol. 180, p. 105895, 2021.

[27] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[28] Y. Yu, "Mobile edge computing towards 5G: Vision, recent progress, and open challenges," *China Communications*, vol. 13, no. Supplement2, pp. 89–99, 2016.

[29] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[30] N. Cheng, W. Xu, W. Shi, Y. Zhou, N. Lu, H. Zhou, and X. Shen, "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 26–32, 2018.

[31] C. Rottondi, F. Malandrino, A. Bianco, C. F. Chiasserini, and I. Stavrakakis, "Scheduling of emergency tasks for multiservice uavs in post-disaster scenarios," *Computer Networks*, vol. 184, p. 107644, 2021.

[32] N. Hossein Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[33] A. Silberschatz and P. B. Galvin, *Operatings System Concepts*. Wiley, 2018.

[34] A. C. Nguyen, T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Reinforcement learning-based deadline and battery-aware offloading in smart farm IoT-UAV networks," in *ICC 2022 - IEEE International Conference on Communications*, pp. 189–194, 2022.

[35] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile networks and Applications*, vol. 18, pp. 129–140, 2013.

[36] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synthesis Lectures on Communication Networks*, vol. 12, no. 2, pp. 1–224, 2019.

[37] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 350–358, 2011.

[38] P. Popovski, F. Chiariotti, K. Huang, A. E. Kalør, M. Kountouris, N. Pappas, and B. Soret, "A perspective on time toward wireless 6G," *Proceedings of the IEEE*, vol. 110, no. 8, pp. 1116–1146, 2022.

[39] Z. Li, P. Tong, J. Liu, X. Wang, L. Xie, and H. Dai, "Learning-based data gathering for information freshness in UAV-assisted IoT networks," *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2557–2573, 2023.

[40] Q. Dang, Q. Cui, Z. Gong, X. Zhang, X. Huang, and X. Tao, "AoI oriented UAV trajectory planning in wireless powered IoT networks," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 884–889, 2022.

[41] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multiagent actor–critic learning for age sensitive mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1053–1067, 2022.

[42] Y. Yang, J. Yang, H. Xu, J. Hu, and T. Song, "Trajectory and offloading policy optimization in age-of-information-aware UAV-assisted MEC systems," in *2023 International Conference on Networking and Network Applications (NaNA)*, pp. 175–180, 2023.

[43] P. M. de Sant Ana, N. Marchenko, P. Popovski, and B. Soret, "Age of loop for wireless networked control systems optimization," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–7, 2021.

[44] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[45] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[46] R. Bellman, "Dynamic programming," *science*, vol. 153, no. 3731, pp. 34–37, 1966.

[47] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, pp. 9–44, 1988.

[48] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[50] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, pp. 1995–2003, PMLR, 2016.

[51] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[52] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[53] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[54] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[56] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for UAV-assisted data collection," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 553–558, 2018.

[57] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "UAV-enabled age-optimal data collection in wireless sensor networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, 2019.

[58] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1211–1223, 2021.

[59] Y. Long, W. Zhang, S. Gong, X. Luo, and D. Niyato, "AoI-aware scheduling and trajectory optimization for multi-UAV-assisted wireless networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pp. 2163–2168, 2022.

[60] 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), pp. 1–6, 2019.

[61] 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)*, pp. 1–6, 2020.

[62] H. Sun, Y. Zhou, J. Tang, Z. Kang, X. Wang, and T. Q. S. Quek, "Average AoI-minimal trajectory design for UAV-assisted IoT data collection system: A safe-TD3 approach," *IEEE Wireless Communications Letters*, vol. 13, no. 2, pp. 530–534, 2024.

[63] M. Wang, L. Li, W. Lin, B. Wei, W. Chen, and Z. Han, "UAV position optimization based on information freshness: A mean field game approach," in *2021 13th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–5, 2021.

[64] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and Z. Gao, "UAV trajectory planning for AoI-minimal data collection in UAV-aided IoT networks by transformer," *IEEE Transactions on Wireless Communications*, vol. 22, no. 2, pp. 1343–1358, 2023.

[65] K. Chi, F. Li, F. Zhang, M. Wu, and C. Xu, "AoI optimal trajectory planning for cooperative UAVs: A multi-agent deep reinforcement learning approach," in *2022 IEEE 5th International Conference on Electronic Information and Communication Technology (ICEICT)*, pp. 57–62, 2022.

[66] M. Akbari, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "AoI-aware energy-efficient SFC in uav-aided smart agriculture using asynchronous federated learning," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1222–1242, 2024.

[67] Y. Wang, S. Fu, C. Yao, H. Zhang, and F. R. Yu, "Caching placement optimization in UAV-assisted cellular networks: A deep reinforcement learning based framework," *IEEE Wireless Communications Letters*, pp. 1–1, 2023.

[68] J. Hu, H. Zhang, L. Song, R. Schober, and H. V. Poor, "Cooperative internet of UAVs: Distributed trajectory design by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 6807–6821, 2020.

[69] M. Samir, C. Assi, S. Sharafeddine, D. Ebrahimi, and A. Ghrayeb, "Age of information aware trajectory planning of UAVs in intelligent transportation systems: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 11, pp. 12382–12395, 2020.

[70] M. N. Ndiaye, E. H. Bergou, and H. El Hammouti, "Muti-agent proximal policy optimization for data freshness in uav-assisted networks," in *2023 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1920–1925, 2023.

[71] M. Sun, X. Xu, X. Qin, and P. Zhang, "AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17275–17289, 2021.

[72] F. Wu, H. Zhang, J. Wu, Z. Han, H. V. Poor, and L. Song, "UAV-to-device underlay communications: Age of information minimization by multi-agent deep reinforcement learning," *IEEE Transactions on Communications*, vol. 69, no. 7, pp. 4461–4475, 2021.

[73] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE international symposium on information theory (ISIT)*, pp. 1451–1455, IEEE, 2016.

[74] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[75] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[76] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "Joint trajectory and computation offloading optimization for UAV-assisted MEC with NOMA," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, 2019.

[77] Y. Wang, Y. Liu, J. Zhang, and B. Liu, "Joint trajectory optimization and task offloading for UAV-assisted mobile edge computing," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2023.

[78] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4353–4366, 2020.

[79] M. Li, J. Gao, L. Zhao, and X. Shen, "Adaptive computing scheduling for edge-assisted autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5318–5331, 2021.

[80] X. Chen, C. Wu, T. Chen, Z. Liu, H. Zhang, M. Bennis, H. Liu, and Y. Ji, "Information freshness-aware task offloading in air-ground integrated edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 243–258, 2022.

[81] A. Alabbasi and V. Aggarwal, "Joint information freshness and completion time optimization for vehicular networks," *IEEE Transactions on Services Computing*, vol. 15, no. 2, pp. 1118–1129, 2022.

[82] T. Ren, J. Niu, B. Dai, X. Liu, Z. Hu, M. Xu, and M. Guizani, "Enabling efficient scheduling in large-scale UAV-assisted mobile-edge computing via hierarchical reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7095–7109, 2022.

[83] S. Gong, L. Cui, B. Gu, B. Lyu, D. T. Hoang, and D. Niyato, "Hierarchical deep reinforcement learning for age-of-information minimization in IRS-aided and wireless-powered wireless networks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 8114–8127, 2023.

[84] S. Wan, J. Lu, P. Fan, and K. B. Letaief, "Toward big data processing in IoT: Path planning and resource management of UAV base stations in mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5995–6009, 2020.

[85] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[86] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[87] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[88] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[89] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[90] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *2nd USENIX workshop on hot topics in cloud computing (HotCloud 10)*, 2010.