Politecnico di Torino

Master's Degree in Mechatronic Engineering

A.a 2023/2024

July 2024

Optimizing Data Collection in IoT Networks with LoRa Equipped Drone: Minimizing Age of Information through Reinforcement Learning

Supervisor:

Prof. Marchetto Guido

Prof. Sacco Alessio

Prof. Silvestri Simone

Prof. Campoy Pascual

Candidate:

Bassino Stefano

Politecnico di Torino

1859

UNIVERSIDAD POLITÉCNICA DE MADRID

POLITÉCNICA

University of Kentucky

Abstract

This thesis considers an UAV-assisted wireless network, where an unmanned aerial vehicle (UAV) is deployed to collect status update data from various heterogeneous sensors monitoring physical processes. The UAV then transmits the collected data to a base station using LoRa (Long Range) communication, which is known for its low-power, low-cost, and long-range capabilities. However, in urban environments, the effectiveness of LoRa can be significantly compromised due to diverse physical settings, signal interference, and obstructions, making it difficult to determine the optimal locations from which transferring the collected data to the base station.

The primary objective of this thesis is to develop a comprehensive system that leverages a UAV to optimize data collection from sensor networks within an unknown environment, where the LoRa signal qualities of different locations are not predetermined. Specifically, this work aims to minimize the Age of Information (AoI) to ensure that the data received at the base station is as fresh as possible, thereby enhancing the timeliness and relevance of the information. A critical challenge in achieving this objective is the joint optimization of the UAV's trajectory and the selection of data transmission locations to minimize the AoI. We formulate this as a new optimization problem and demonstrate that it is NP-Hard.

First, to understand how the real environment affects the LoRa connection, we developed a testbed with a LoRa transmitter and receiver to collect real-world transmission data in a 1 km^2 urban area. Using this collected data, an initial solution was implemented with Gurobi, based on the assumption of a perfectly known data rate distribution. Next, we considered a real-world scenario where the LoRa signal quality is unknown at different locations. Given the high computational cost of the Gurobi solution, we propose a heuristic algorithm that leverages reinforcement learning to learn the environment and plan the UAV trajectory with the objective of minimizing the AoI. Additionally, we performed extensive experiments to assess the performance of the proposed solution against existing approaches.

In conclusion, this thesis develops an efficient UAV-aided data collection system with LoRa communication capabilities for base stations in urban environments. By jointly optimizing transmission locations and the UAV's trajectory, the system effectively reduces the Age of Information (AoI) and enhances data relevance.

Contents

List of Acronyms

AoI	Age of Information
UAV	Unmanned Aerial Vehicle
UCB	Upper Confidence Bound
IoT	Internet of Things
LoRa	Long Range Wide Area Network
SF	Spreading Factor
BW	Bandwidth
ADR	Adaotice Data Rate
CR	Coding Rate
DR	Data Rate
BS	Base Station
TA	Time Arrival
TD	Time Departure
TU	Time Upload
CS	Check Sum
IoTD	Internet of Things Device

List of Figures

List of Algorithms

Chapter 1

Introduction

The rapid development of the Internet of Things (IoT) has driven advancements in applications like traffic control, autonomous driving, infrastructure inspection, disaster management, and industrial control. However, the increasing data volume and demand for timely delivery pose significant challenges, especially in energy-constrained IoT systems and areas with limited internet coverage and sparse sensor deployment. Unmanned Aerial Vehicles (UAVs) offer a solution by extending network connectivity and assisting in data collection as mobile relay nodes. Unlike multi-hop transmission, UAVs can operate near IoT devices, leveraging line-of-sight (LOS) communication to reduce transmission energy, improve reliability, and enhance throughput.

Meanwhile, Long Range Wide Area Network (LoRa) [1], a low-power, wide-area network (LPWAN) protocol, has emerged as a promising network solution for wirelessly connecting battery-operated devices. Its exceptional capabilities in providing extensive communication range and cost-efficient deployment make it widely adopted for large-scale IoT applications such as smart cities, manufacturing, and smart agriculture. Existing works related to LoRa focus on optimizing parameter settings and resource allocation to improve network performance metrics such as throughput, packet loss rate, and retransmission time. However, these approaches primarily consider static LoRa nodes, i.e, the distance between transmitter node and the receiver node are fixed. LoRa communication quality can be significantly impacted by heterogeneous physical environments, i.e., trees, buildings, obstacles, and signal interference. Consequently, nodes placed far from the receiver or surrounded by buildings and trees, as in urban environments, can experience prolonged transmission times and high packet loss rates.

This work utilizes a UAV-assisted mobile LoRa node equipped with a LoRa transceiver to collect data from ground IoT devices and transmit it to a base station. The UAV's mobility allows it to find optimal positions for good communication, overcoming environmental and signal interference challenges.

Timeliness of data is crucial, especially in delay-sensitive applications like safety control. Age of Information (AoI) quantifies data freshness at the receiver. Existing AoI optimization in UAV-assisted IoT networks generally falls into two categories: one neglects UAV-to-core network transmission, which is impractical due to unlicensed spectrum delays; the other considers UAVs as mobile nodes connected to ground base stations. Some approaches have UAVs deliver all collected data after returning to the base station, which is inefficient. Others transmit data during flight but assume deterministic transmission rates, which is unrealistic due to environmental changes and interference.

To understand the impact of the physical environment on LoRa communication quality, we conducted extensive LoRa transmissions within a 1 km^2 urban area divided into 100 grids. With the LoRa receiver at the center, we moved the transmitter across all grids and measured transmission data rates. Areas farther from the receiver or obstructed by buildings and trees had lower rates, with most transmissions failing in non-line-of-sight (NLOS) conditions. Rates increased up to 300 bytes/s in LOS conditions. The standard deviation reached up to 90, with over 40% of grids showing a deviation greater than 40, likely due to traffic conditions and mobile device interference. This variability complicates determining optimal data collection trajectories and transmission decisions to minimize AoI.

Using the collected data, an initial solution was implemented using the Gurobi optimizer, assuming a perfectly known data rate distribution. Due to high computational cost, an heuristic algorithm has been proposed to replace Gurobi. Subsequently, a real-world scenario was considered where the LoRa signal quality is unknown at different locations. To address this problem a UCB algorithm has been proposed. This algorithm learns the environment and plans the UAV trajectory with the objective of minimizing AoI. Extensive experiments were conducted to assess the performance of the proposed solution against existing approaches.

The results demonstrate that our heuristic algorithm, even if produce a sub-optimal solution, significantly outperforms the Gurobi solver in terms of speed, enabling the solution to scale to a higher number of sensors. Additionally, the Upper Confidence Bound (UCB) algorithm consistently outperforms the other two approaches when applied to real-world scenarios. This indicates superior efficiency and effectiveness. The detailed comparison of the different algorithms provides valuable insights into their relative performance and scalability.

In conclusion, this thesis develops an efficient UAV-aided data collection system with LoRa communication capabilities for base stations in urban environments. By jointly optimizing transmission locations and the UAV's trajectory, the system effectively reduces the Age of Information (AoI) and enhances data relevance.

In conclusion, this thesis develops an efficient UAV-aided data collection system with LoRa communication capabilities for base stations in urban environments. By jointly optimizing transmission locations

and the UAV's trajectory, the system effectively reduces the Age of Information (AoI) and enhances data relevance, outperforming the actual solutions.

1.1 Organization of the thesis

This thesis is organized into 6 chapters, structured as follows:

- **Chapter 1: Introduction**

  Explores the rapid development of IoT applications and the associated challenges in data volume and timely delivery. Discusses the role of UAVs in extending network connectivity and assisting data collection. Introduces LoRa technology and the concept of Age of Information (AoI) to assess data freshness. This chapter sets the context for the research and outlines the motivation behind the implementation of this work.

- **Chapter 2: Background & Related Work**

  Discusses the key technologies used in this work, including LoRa architecture, the Gurobi optimization solver, and the UCB algorithm. Rapidly reviews recent research on optimizing UAV trajectories for efficient data collection, focusing on the AoI minimization.

- **Chapter 3: Methodology**

  Details the system model, communication model and the AoI formulation. Explains the algorithms implemented for the optimization, including the Gurobi solver, the heuristic algorithm, and the UCB algorithm, and describes their application to the problem. This chapter explains the theoretical and practical approaches used in designing and implementing the UAV-assisted data collection system.

- **Chapter 4: Experiments**

  Provides a comprehensive explanation of the experimental setup, including the components and configurations used. Discusses the data collection process and presents the analysis, highlighting the impact of environmental factors on signal quality and data transmission. The chapter also explains the tuning process for the LoRa communication parameters and the methods used to ensure reliable data collection in urban environments.

- **Chapter 5: Results**

  This chapter analyzes the performance of the implemented solutions. It begins by comparing the Gurobi solver with the heuristic algorithm and then presents the UCB solution. The findings are discussed, highlighting their implications for optimizing UAV trajectories to minimize AoI and enhance data collection efficiency. The chapter also presents the advantages and disadvantages of each implemented solution, providing a clear explanation of the reasoning behind each choice.

- **Chapter 6: Conclusion**

  Summarizes the key findings and contributions of the study. Suggests potential directions for future research, including further optimization techniques and increasing the level of complexity of the solution, adding, for example, more UAVs.

Chapter 2

Background & Related Work

This chapter explores key technologies and theories essential in this work. It starts with LoRa (Long Range Wide Area Network), a leading IoT communication protocol, and covers its architecture, features, and regulatory differences between the USA and Europe. Next, it examines the Gurobi optimization solver, highlighting its performance and applications in solving complex mathematical problems. The concept of Age of Information (AoI) is then introduced, explaining its importance in ensuring information freshness in real-time applications. The chapter also covers the Upper Confidence Bound (UCB) algorithm, which balances exploration and exploitation in reinforcement learning. Finally, it reviews recent research on optimizing UAV trajectories for efficient data collection, focusing on energy efficiency and AoI minimization. This sets the stage for understanding the current advancements and challenges in UAV-assisted IoT data collection.S

2.1 LoRa

In recent years, the Internet of Things (IoT) has witnessed exponential growth, necessitating the development of efficient and scalable wireless communication protocols. LoRa (Long Range Wide Area Network) has emerged as a leading protocol in this domain, offering long-range communication, low power consumption, and the ability to support a massive number of devices. This section provides an in-depth exploration of LoRa, its architecture, key features, applications, regulatory differences between the USA and Europe, and future prospects.

2.1.1 Architecture of LoRa

LoRa is a protocol designed to manage communication between low-power devices and a central network server. The architecture of LoRa can be broadly divided into four key components: end devices, gateways, network servers, and application servers.

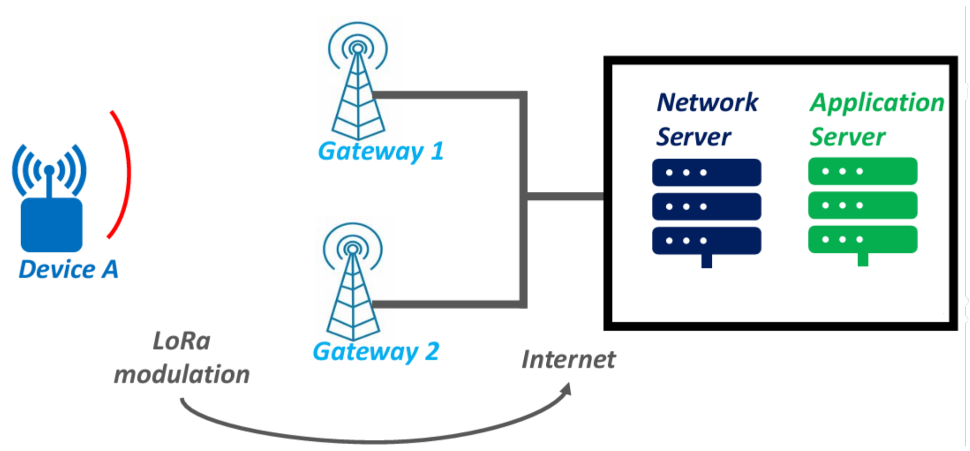


Figure 2.1: LoRa Infrastructure

- **End Devices**: These are the sensors or actuators deployed in the field. They communicate with the gateways using the LoRa (Long Range) modulation technique, which is robust against interference and capable of achieving long-range communication [11].
- **Gateways**: Gateways act as intermediaries between end devices and the network server. They receive data from end devices

and forward it to the network server over an IP backbone (e.g., Ethernet, cellular, Wi-Fi) [12].

- **Network Server**: The network server is responsible for managing the network, including tasks such as data de-duplication, security checks, and network management functions. It ensures the integrity and security of the data before it is sent to the application server [13].
- **Application Server**: This component processes and analyzes the data received from the network server, enabling end-user applications. It can trigger actions based on the analyzed data, providing real-time responses and insights [14].

2.1.2 Key Features

LoRa is distinguished by several features that make it ideal for IoT applications:

- **Long Range**: LoRa can achieve communication ranges up to 15 kilometers in rural areas This capability is critical for applications requiring widespread coverage [15].
- **Low Power Consumption**: End devices in a LoRa network are designed to operate for years on a single battery, thanks to low power consumption mechanisms. This makes it suitable for battery-powered IoT devices [11].
- **Scalability**: LoRa can support millions of devices within a single network, making it highly scalable. The use of spread spectrum modulation and adaptive data rate techniques helps in managing network capacity efficiently [12].

2.1.3 LoRa Parameters

Understanding the various parameters of LoRa is crucial for optimizing network performance and meeting specific application requirements. Key parameters include data rate, spreading factor, bandwidth, coding rate, and transmit power.

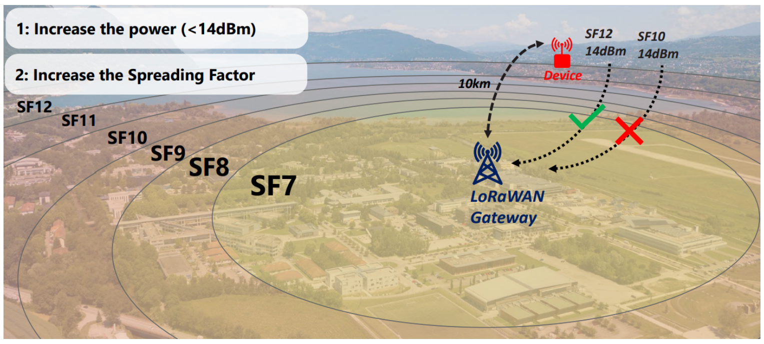


Figure 2.2: Visualization of the Spreading Factor

- **Data Rate**: The data rate in LoRa is a function of the spreading factor (SF) and the bandwidth (BW). LoRa supports a range of data rates from 30 bps to 1 kbps, allowing for flexibility in balancing range, power consumption, and data throughput [15]. The data rate is adjusted dynamically using Adaptive Data Rate (ADR) to optimize network performance [11].
- **Spreading Factor (SF)**: The spreading factor determines the duration of the chirp signal used in LoRa modulation. It ranges from SF7 to SF12, with higher spreading factors resulting in longer chirps [12]. Higher spreading factors increase the link budget, enabling longer communication ranges at the cost of lower data rates and higher airtime [14].
- **Bandwidth (BW)**: LoRa supports multiple bandwidth options, including 125 kHz, 250 kHz, and 500 kHz. The bandwidth affects the data rate and the robustness of the signal [13]. Narrower

bandwidths (e.g., 125 kHz) provide better sensitivity and longer range, while wider bandwidths (e.g., 500 kHz) support higher data rates [15].

- **Coding Rate (CR)**: The coding rate is a measure of error correction applied to the transmitted data. LoRa uses forward error correction (FEC) to improve communication reliability [11]. Coding rates range from 4/5 to 4/8, with higher coding rates providing better error correction at the expense of reduced effective data rate [12].
- **Transmit Power**: Transmit power in LoRa can be adjusted to balance range and power consumption. Higher transmit power increases the communication range but also consumes more battery [14]. The maximum allowable transmit power is regulated and varies by region (e.g., up to 1 Watt in the USA and 25 mW in Europe) [16, 17].
- **Channel Plan**: The channel plan specifies the frequencies and channel spacing used for communication. It is region-specific to comply with local regulatory requirements [13]. In the USA, the channel plan includes 64 channels in the 902-928 MHz band, while in Europe, the plan includes 8 channels in the 863-870 MHz band [18].
- **Duty Cycle**: The duty cycle is the fraction of time a device is allowed to transmit in a given period. It is regulated to prevent spectrum congestion and ensure fair access [17]. In Europe, the duty cycle is strictly limited (e.g., 1% or 0.1% depending on the sub-band), while in the USA, duty cycle restrictions are more relaxed [16].
- **Adaptive Data Rate (ADR)**: ADR is a mechanism in LoRaWAN that dynamically adjusts the data rate, spreading fac-

tor, and transmit power of end devices based on network conditions [11]. ADR optimizes network capacity, battery life, and overall performance by adapting to the changing environment and device locations [12].

2.1.4 Regulatory Differences between the USA and Europe

LoRa operates in the unlicensed Industrial, Scientific, and Medical (ISM) radio bands, which are regulated differently in the USA and Europe. These differences affect the deployment and operation of LoRa networks in these regions.

- **Frequency Bands**:
  - **USA**: In the United States, LoRa operates primarily in the 902-928 MHz ISM band. This band is regulated by the Federal Communications Commission (FCC) and is subject to specific rules regarding output power and duty cycle [16].
  - **Europe**: In Europe, LoRa operates in the 863-870 MHz ISM band, regulated by the European Telecommunications Standards Institute (ETSI). The regulations here are more stringent, with tighter restrictions on duty cycle and transmission power [17].
- **Duty Cycle and Transmission Power**:
  - **USA**: The FCC regulations in the USA allow for higher transmission power up to 1 Watt (30 dBm) and do not impose strict duty cycle limitations, making it possible to achieve longer ranges and higher data rates [16].
  - **Europe**: ETSI regulations in Europe limit the transmission power to 25 mW (14 dBm) for most applications and enforce a duty cycle limit of 1% or lower, depending on the specific

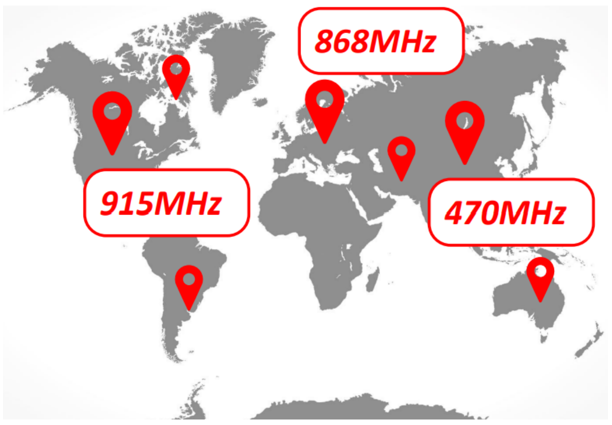


Figure 2.3: Frequency band regulation in different continents

sub-band. These restrictions are designed to minimize interference and ensure fair usage of the spectrum [17].

- **Channel Plan**:
  - **USA**: The LoRa channel plan in the USA typically includes a larger number of channels due to the wider available spectrum (902-928 MHz). This allows for greater flexibility and capacity in network design [18].
  - **Europe**: In Europe, the available spectrum (863-870 MHz) is narrower, resulting in fewer channels and potentially more congestion. The channel plan must be carefully managed to optimize performance and compliance with duty cycle regulations [17].
- **Regional Adaptations**:
  - **USA**: The flexible regulations in the USA support a broader range of applications and facilitate rapid deployment and innovation in LoRaWAN technologies [16].
  - **Europe**: The stricter European regulations ensure that the spectrum is used efficiently and fairly, which can lead to more

predictable network performance and reduced interference [17].

2.1.5 Conclusion

LoRa has established itself as a pivotal technology in the IoT landscape, offering a robust solution for long-range, low-power, and scalable communication. Its versatile applications across multiple domains underscore its significance and potential in driving the future of connected devices. The regulatory differences between the USA and Europe present both challenges and opportunities for optimizing LoRa deployments to meet regional needs and compliance requirements [12].

2.2 Gurobi

Gurobi is a state-of-the-art optimization solver designed for solving complex mathematical problems. It is widely used in academia and industry for tackling a variety of optimization problems, including linear programming (LP), mixed-integer programming (MIP), quadratic programming (QP), and mixed-integer quadratic programming (MIQP).

2.2.1 Key Features of Gurobi

- **High Performance**: Gurobi is known for its speed and efficiency. It employs advanced algorithms and parallel processing techniques to solve large and complex problems quickly.
- **Wide Range of Problem Types**: Gurobi supports a diverse set of problem types:
  - **Linear Programming (LP)**: Optimization of a linear objective function, subject to linear equality and inequality

constraints.

  - **Mixed-Integer Programming (MIP)**: Extension of LP with the inclusion of integer variables, making it suitable for problems requiring discrete decisions.
  - **Quadratic Programming (QP)**: Optimization involving a quadratic objective function with linear constraints.
  - **Mixed-Integer Quadratic Programming (MIQP)**: Combines the elements of MIP and QP, allowing for quadratic objective functions and integer variables.
  - **Quadratically Constrained Programming (QCP)** and **Mixed-Integer Quadratically Constrained Programming (MIQCP)**: For problems with quadratic constraints.

- **User-Friendly Interfaces**: Gurobi offers a variety of interfaces to suit the main programming environments:
  - **Python, C, C++, Java**: Compatibility with the main programming languages.
  - **MATLAB and R**.
  - **AMPL, GAMS, and MPL**: Support for these algebraic modeling languages.

2.2.2 Applications of Gurobi

The versatility of Gurobi makes it suitable for a wide range of applications, including but not limited to:

- **Supply Chain Optimization**: Designing efficient supply chains, optimizing inventory levels, and improving logistics.
- **Financial Modeling**: Portfolio optimization, risk management, and financial planning.

- **Scheduling**: Workforce scheduling, production planning, and project management.
- **Energy Systems**: Optimizing energy production, distribution, and consumption.
- **Transportation**: Route planning, vehicle scheduling, and traffic management.

2.2.3 Conclusion

Gurobi is a powerful and flexible optimization solver that has become a cornerstone in the field of mathematical optimization. Its ability to handle a wide range of problem types, combined with its high performance and user-friendly interfaces, makes it an invaluable tool for researchers and practitioners alike.

2.3 Age of Information

The Age of Information (AoI) is a metric that quantifies the freshness of information in a network. Unlike traditional metrics such as delay or latency, which measure the time taken for a packet to travel from the source to the destination, AoI captures the age of the most recently received update at the destination. This age is crucial in applications where the most up-to-date information is needed to make decisions or take actions.

Mathematically, AoI is defined as follows:

$$\Delta(t) = t - u(t)$$

where:

- $\Delta(t)$ is the Age of Information at time t.

- t is the current time.
- $u(t)$ is the generation time of the last received update at the destination.

For example, if the current time is 10 seconds and the last received update was generated at 7 seconds, the AoI is 3 seconds. This means the information available at the receiver is 3 seconds old.

2.3.1 Importance

The importance of AoI stems from its ability to provide a direct measure of information freshness, which is critical in various real-time and latency-sensitive applications. Here are some key reasons why AoI is important:

1. **Performance Optimization:**
    - Traditional metrics like throughput and delay do not adequately capture the performance of systems that rely on timely updates. AoI provides a more relevant performance measure for optimizing such systems [20].
    - For instance, in wireless sensor networks used for environmental monitoring, ensuring that the data is fresh (low AoI) can significantly improve the quality of monitoring and subsequent actions based on the sensor data [21].
2. **Network Efficiency:**
    - Understanding and managing AoI can lead to more efficient use of network resources. By optimizing update frequencies and scheduling policies, it is possible to minimize AoI without unnecessarily increasing the load on the network [24].

- This is particularly important in resource-constrained networks such as wireless sensor networks or Internet of Things (IoT) systems, where energy and bandwidth are limited [25].

2.3.2 Applications Highlighting the Importance of AoI

1. **Data Collection Optimization:**
   - In IoT sensor networks, timely data collection is crucial for accurate monitoring. High AoI can result in outdated or irrelevant information being processed, leading to poor decisions.
   - Optimizing UAV trajectories and scheduling data transmissions can significantly reduce AoI, ensuring that the collected data is as fresh and relevant as possible. .
2. **Autonomous Vehicles:**
   - Autonomous vehicles rely on real-time data from various sensors and communication with other vehicles and infrastructure. High AoI can result in outdated information about traffic conditions, road hazards, or the positions of other vehicles, leading to unsafe driving decisions.
   - Techniques to minimize AoI in vehicular networks can improve both safety and efficiency by ensuring that vehicles are operating on the most recent data [26].
3. **Industrial Automation:**
   - Industrial processes often involve real-time control systems where timely updates from sensors are critical for maintaining process stability and efficiency. High AoI can lead to delays in detecting and responding to process deviations, potentially causing defects or downtime.

- By managing AoI, industrial automation systems can ensure more reliable and efficient operations, reducing the risk of production losses and enhancing overall productivity [23].

2.3.3 Conclusion

The Age of Information is a crucial metric for evaluating and optimizing the timeliness of information in various systems. Its importance spans a wide range of applications, from autonomous vehicles to healthcare and industrial automation, where timely information is vital for making informed decisions, ensuring safety, and enhancing system performance. By focusing on AoI, researchers and engineers can develop more effective strategies to keep information fresh, ultimately leading to more reliable and efficient systems.

2.4 UCB algorithm

The Upper Confidence Bound (UCB) algorithm is designed to manage the exploration-exploitation trade-off in reinforcement learning. In multi-armed bandit problems[1], a decision-maker must choose between multiple actions (arms) to maximize the cumulative reward. Each action has an unknown reward distribution, and the decision-maker must explore different actions to learn their rewards while exploiting the actions that yield the highest known rewards.

Mathematically, the UCB algorithm selects the action a at time t that maximizes the following expression:

$$a_t = \arg\max_{a \in A} \left(\hat{\mu}_a(t) + c\sqrt{\frac{\ln t}{N_a(t)}} \right)$$

[1]In this type of problem, a decision-maker (or gambler) is faced with several options, each with an unknown probability distribution of rewards. These options are metaphorically referred to as "arms" of a slot machine or "bandit," hence the name "multi-armed bandit."

where:

- $\hat{\mu}_a(t)$ is the estimated mean reward of action a at time t.
- $N_a(t)$ is the number of times action a has been selected up to time t.
- c is a positive parameter that controls the level of exploration.
- A is the set of all possible actions.

The term $\sqrt{\frac{\ln t}{N_a(t)}}$ is the upper confidence bound, which decreases as the number of times action a is chosen increases. This encourages exploration of less frequently chosen actions.

2.4.1 Importance

The importance of the UCB algorithm in reinforcement learning arises from its effectiveness in balancing exploration and exploitation. Here are some key reasons why the UCB algorithm is important:

1. **Optimal Exploration-Exploitation Trade-off:**
   - The UCB algorithm provides a theoretically sound approach to balancing exploration and exploitation, which is a fundamental challenge in reinforcement learning [27]. By considering both the estimated reward and the uncertainty of the reward, UCB ensures that actions with high potential rewards are explored sufficiently.
2. **Theoretical Guarantees:**
   - The UCB algorithm has strong theoretical guarantees on its performance. It is proven to achieve logarithmic regret, meaning that the difference between the reward obtained by the algorithm and the reward obtained by always choosing the best action grows logarithmically with time [27].

3. **Simplicity and Efficiency:**
   - The UCB algorithm is simple to implement and computationally efficient. It requires maintaining estimates of the mean rewards and counts of the actions, which can be updated incrementally [28].

2.4.2 Applications Highlighting the Importance of UCB

The field of usage of this type of algorithm is really wide:

1. **Online Advertising:** In online advertising, UCB algorithms are used to select which ads to display to users in order to maximize click-through rates. By balancing exploration of new ads and exploitation of known successful ads, UCB helps in efficiently identifying the most effective advertisements [30].
2. **Clinical Trials:** In clinical trials, UCB algorithms can be used to dynamically allocate treatments to patients. This ensures that patients receive treatments that are most likely to be effective while still gathering enough data to learn about all treatment options [31].
3. **Adaptive Routing:** In networking, UCB algorithms are employed to optimize routing decisions. By exploring different routes and exploiting the best-known routes, UCB helps in minimizing latency and improving overall network performance [32].

2.4.3 Conclusion

The Upper Confidence Bound algorithm is a powerful tool in reinforcement learning, offering an effective solution to the exploration-exploitation trade-off. Its theoretical guarantees, simplicity, and wide applicability make it a valuable approach in many domains. By ensuring a balanced exploration of all actions, UCB algorithms help in maximizing cumulative rewards and have demonstrated success in numerous practical applications.

2.5 Related Work

Numerous studies have investigated the deployment of UAVs for data collection from distributed sensor networks. In [33], the authors explored the optimization of UAV trajectories to maximize data collection efficiency while considering energy constraints. The study presented a framework where UAVs are utilized to gather data from ground sensors and then transmit this data to a base station. The authors formulated an optimization problem aiming to determine the optimal UAV trajectory that maximizes the total amount of collected data within a given flight time, subject to energy limitations. Their results indicated that significant improvements in data collection efficiency could be achieved through optimal path planning, especially in scenarios with sparse sensor distributions.

Similarly, [34] presented an energy-efficient UAV trajectory design that balances the trade-off between data collection and UAV battery life. This work proposed an optimization framework that simultaneously considers the UAV's flight energy consumption and the data collection performance. By optimizing the UAV's flight path, the authors demonstrated that it is possible to extend the operational time of the UAV while ensuring effective data collection from sensor networks. The study highlighted the importance of energy-aware trajectory planning, particularly in applications where UAVs have limited battery capacities.

The optimization of UAV trajectories for data collection is critically linked to the concept of Age of Information (AoI). AoI is a metric that quantifies the freshness of the information received at the base station, defined as the time elapsed since the generation of the latest received update. In scenarios involving UAV-assisted data collection, minimizing AoI is essential to ensure that the base station has the most current information about the monitored environment.

Recent research has focused on integrating AoI considerations into the trajectory optimization problem. For instance, [35] investigated scheduling policies that minimize AoI in wireless networks, and their findings are applicable to UAV-assisted scenarios. The authors developed algorithms that strategically determine the timing of data transmissions to keep the AoI as low as possible. This work underscores the importance of timely data collection and transmission, which can be directly influenced by the UAV's flight path and scheduling decisions.

By jointly optimizing the UAV's trajectory and the data transmission schedule, it is possible to achieve a balance between energy efficiency and AoI minimization. [36] introduced a framework for optimizing AoI in wireless networks through adaptive scheduling policies. Although their work primarily focused on static base stations, the principles can be extended to dynamic UAV scenarios. The study demonstrated that by carefully planning the UAV's route and data transmission times, the AoI can be significantly reduced, leading to more timely and relevant data being available at the base station.

In summary, the optimization of UAV trajectories for data collection involves a complex interplay path planning and AoI minimization. By considering both factors, it is possible to design UAV-assisted systems that provide timely, fresh data while operating within the energy constraints of the UAV. This integrated approach is crucial for enhancing the performance and effectiveness of UAV-assisted wireless networks, particularly in urban environments where signal propagation challenges and dynamic conditions are prevalent.

Chapter 3

Methodology

In this chapter, we dig into the core aspects of the system and provide a comprehensive formulation of the problem. We start by presenting an overview of the system architecture, detailing the components and their interactions. Following this, we articulate the specific problem we aim to address.

To address the problem effectively, we introduce and examine various proposed solutions. Each solution is meticulously analyzed, including a detailed explanation of the pseudocode, offering insights into the logic and structure of the algorithms.

The chapter aims to provide a solid foundation for understanding the complexities of the system and the challenges involved in optimizing UAV trajectories for enhanced data collection and minimized Age of Information (AoI) considering LoRa.

3.1 System Presentation

Consider a set $\mathcal{S} = \{S_1, S_2, ..., S_M\}$ of M heterogeneous ground Internet of Things devices (IoTDs) that are sparsely distributed over a specific geographical area. These devices are responsible for monitoring various physical processes and periodically collecting data such as temperature, humidity, light intensity, and other environmental

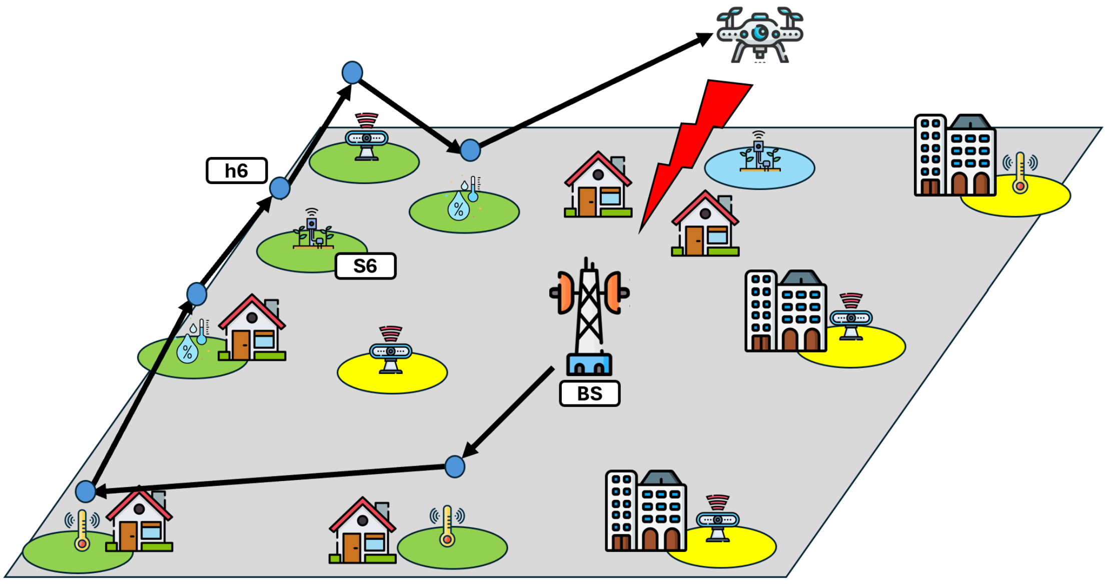


Figure 3.1: Schematic of the system described. The blu dots are the hovering points and the circles on the ground are their projections. The Green circle means the sensor has already been visited, light blue means the UAV is in that location and yellow circle means the sensor has not been covered yet. The red lighting means the drone is transmitting data to the BS

metrics, or capturing images.

Due to the large urban deployment of the network, the sensors cannot communicate directly with the base station. To facilitate data collection and transmission, an Unmanned Aerial Vehicle (UAV) is deployed to gather data from these ground IoTDs and relay it to the Base Station (BS) for further processing. This relay is performed using LoRa (Long Range) communication. The effective communication range of LoRa varies with the environment, achieving a maximum range of 6 kilometers in rural areas and only 800 meters in urban environments with the implemented set-up.

The UAV departs from the BS S_0, follows a predetermined trajectory $\mathcal{T}$, hovers above the ground sensors to collect the generated data, and transmits the collected data to the BS from strategically chosen hovering positions to ensure the freshness of information. After completing its data collection mission, the UAV returns to the BS. For simplicity, we assume that the UAV operates at a fixed altitude H and maintains a constant speed V.

The 2D location of IoTD $S_i \in \mathcal{S}$ is denoted as $l_i = (x_i, y_i)$, while the location of the BS S_0 is denoted as $l_0 = (x_0, y_0)$. The set of hovering points where the UAV stays for data collection from ground devices and for data transmission to the BS is represented as $\mathcal{H} = \{h_1, h_2, ..., h_M\}$. Each hovering point $h_i = (x_i, y_i, H)$ is the projection of l_i at altitude H. We assume that the UAV only collects data from device S_i when it is hovering at h_i.

After collecting data from ground device $S_i \in \mathcal{S}$, the UAV has two operational choices:

1. Transmitting all or part of the currently collected data to the BS if the LoRa communication quality is adequate at the current hovering position.

2. Retaining the collected data and flying to the next hovering position if the communication quality to the BS is poor.

The UAV offloads all the remaining data not transmitted once arrived to the BS.

3.1.1 LoRa Communication Model

As mentioned in the previous chapter, the communication quality of LoRa can be significantly influenced by various factors such as buildings, trees, and other obstacles (both static and dynamic). These factors cause variability in the quality of LoRa communication at different times, even at the same location, as depicted in Figure 3.2. To model this variability and achieve a more accurate representation, the data rate of LoRa communication to the BS at each hovering position h_k has been modeled as a random variable X_k. At any given time t, the data rate at h_k is represented by the realization X_k^t of the random variable X_k.

We assume that the distribution of X_k remains stationary over a sufficiently extended period. This implies that the type of distribution and the expected value of X_k do not change within this period. Once real data is obtained (as will be explained in the next chapter), the distribution can be computed. To model the variability of the data rate, a Gaussian distribution has been used.

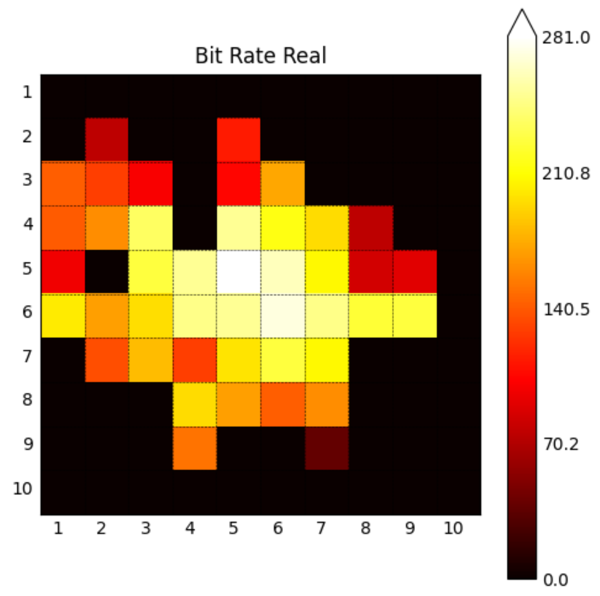


Figure 3.2: Real distribution of the data rate

3.1.2 Age of Information (AoI)

To measure the freshness of the information collected from the IoTDs, we employ the concept of Age of Information (AoI). AoI is defined as the time elapsed since the instant at which the latest received status update packet at the base station was generated. We denote the timestamp of data generated at S_i as τ_i and the data size of the information stored in S_i as D_i.

The timestamp of the UAV taking off from the BS is denoted as τ_{BS}. All data are generated before the UAV departs on the current tour, such that $\tau_{BS} > \tau_i$ for all $i \in \{1, ..., M\}$. We also denote TA_i as the arrival time of the UAV at hovering position h_i and TD_i as the departure time of the UAV from h_i. Let $z_{ij} \in \{0, 1\}$ be a decision variable equal to 1 if the UAV transmits the data of S_i from the location of h_j, and 0 otherwise.

If S_i is the k-th sensor whose data is transmitted to the BS by the UAV when hovering at position h_j, it can also be labeled as v^j_k, i.e., $v^j_k = i$.

Let's define TA_j and TD_j respectively as the arrival time and the departure time of the UAV from the sensor $j - th$.

The UAV leaves the base station at $t = 0$, so

$$\mathrm{TD}_0 = 0$$

The arrival time of the UAV at sensor j is given by the time the UAV has left sensor i plus the travel time between the two sensors. Generalizing this, we can express it as:

$$\mathrm{TA}_j = \Sigma_{i=0}^{M} \left(\frac{||S_j - S_i||}{V} + TD_i \right) \cdot x_{ij}$$

Instead, the departure time from a sensor j can be computed by considering the arrival time at the sensor and the transmission time of all the data and it can be expressed as follows:

$$\mathrm{TD}_j = TA_j + \Sigma_{i=1}^{M} z_{ij} \cdot \frac{D_i}{B_j}$$

Here, D is an array containing the data size of every sensor and B is an array containing the LoRa data rate from each sensor's location to the base station.

Now that the various terms have been explained it is possible to formulate the AoI.

Denote τ_i as the data generation time for the IoT sensor $i \in M$. As mentioned previously, we assume this time is the same for all sensors, corresponding to the time when the UAV leaves the base station. Thus, the AoI of the sensor S_i is defined as follows:

$$\text{AoI}_i = TU_i - \tau_i$$

where

$$\text{TU}_i = \sum_{j=0}^{M} \left(z_{ij} \cdot TA_i + z_{ij} \cdot \sum_{k=1}^{i} z_{kj} \cdot \frac{D_k}{B_j} \right)$$

This term refers to the uploading time of a data from the UAV to the base station. Moreover this formulation has been used in order to consider the transmission of multiple data from the same location. In fact the LoRa connection doesn't allow the transmission of multiple data in the same time. So in the computation of the AoI is essential to consider the order of transmission because it affects the AoI of the data.

If the UAV from a certain location i decide to transmitt two data, D_1 and D_2, the AoI of the two data will be:

$$AoI_1 = TA_i + \frac{D_1}{B_i}$$
$$AoI_2 = TA_i + \frac{D_1}{B_i} + \frac{D_2}{B_i}$$

As it is possible to see, the AoI of the sensor 2 includes also the transmission time of the data 1.

The metric used to evaluate the performance of the solution is the sum of the AoI of all the sensors:

$$AoI = \sum_{i=1}^{M} AoI_i$$

3.2 Problem Formulation

Now that the system model has been explained, the problem to be addressed can be articulated.

As mentioned in the introduction, the objective of this research is to visit all sensors while maximizing the freshness of the data collected. The terms that express the freshness of the data is the AoI, so the objective function can be expressed as

$$min \textstyle\sum_{i=1}^{M} AoI_i$$

so minimizing the total sum of the AoI of every sensor.

There are some restrictions in the formulation of this problem that have to be presented:

- The UAV has to visit all the sensor only once
- The drone has to start and coming back to the initial point

In the next section the solutions proposed are analyzed

3.3 Proposed Solution

In this section is present the proposed solution to the problem outlined in the previous section.

The final solution has been developed through several iterations, starting with simpler approaches and gradually addressing the complete problem. The subsequent subsections will detail the entire process undertaken to arrive at the final, comprehensive solution.

3.3.1 Gurobi Optimizer

The first step has been to solve the problem with Gurobi(2.2). The problem, as it is formulated, is a quadratic programming(QP) problem, since a lot of constraint are expressed through quadratic functions.

Since Gurobi is an optimizer and does not account for physical constraints, a series of constraints must be introduced to ensure an accurate description and effective optimization.

Before listing all the constraints, it is essential to explain two variables to enhance comprehension:

- The variable $x_{i,j} \in 0, 1$ is a decision variable equal to 1 if the drone travel from sensor i to j with $i, j \in M$
- The variable $z_{i,j} \in 0, 1$ is a decision variable equal to 1 if the drone transmit the data i from the sensor position j with $i, j \in M$

These first two constraints are essential to ensure that the UAV visits all the sensors available.

$$\Sigma_{j=1}^{M} x_{ij} = 1 \quad \forall i \in M$$

$$\Sigma_{i=1}^{M} x_{ij} = 1 \quad \forall j \in M$$

The next one ensure that the drone starts and ends from the same position

$$\Sigma_{i=1}^{M} x_{i0} = \Sigma_{j=1}^{M} x_{0j} = 1$$

The three sub-tour elimination constraints are used in order to avoid that sensors are visited multiple times and they work by labeling the order in which the nodes are visited through u_i . The formulation used is the Miller-Tucker-Zemlin (MTZ)[1]

$$\begin{gathered} u_0 = M + 1 \\ 1 \leq u_i \leq M + 1 \quad \forall i \in [0, M] \\ u_i - u_j + 1 \leq M(1 - x_{ij}) \quad \forall i \in [1, M], \forall j \in [0, M] \end{gathered}$$

[1]There are several formulations for the sub-tour problem

The final set of constraints is necessary to optimize the transmission positions accurately. These constraints ensure that data is transmitted only after it has been collected and that all data is delivered to the base station.

$$z_{ij}(u_j - u_i) \geq 0 \quad \forall i \in [1, M], \forall j \in [0, M]$$
$$\Sigma_{j=0}^{M} z_{ij} = 1 \quad \forall i \in M$$

As it is possible to notice most of the constraints are quadratic and even though Gurobi is able to solve QP problem, in order to speed up the solver, the constraints have been linearized.

Using this approach, the optimal solution is guaranteed to be found thanks to Gurobi. However, the primary challenge with this approach is its scalability. Specifically, the constraints used to prevent subtours grow exponentially (at a rate of n^n, where n is the number of sensors) as the number of sensors increases. This exponential growth in constraints can lead to significant computational challenges. Even with a small number of sensor the solving time was really high.

3.3.2 Heuristic Algorithm

For the reasons mentioned above, while the Gurobi solution is optimal, it is not scalable. Therefore, a heuristic algorithm has been implemented to achieve scalability. Although the performance is inferior compared to the Gurobi solution (a performance comparison will be presented in Chapter 5), the scalability introduces numerous advantages.

The pseudocode 1 describes the heuristic algorithm implemented.

First, the UAV's tour is initialized at the base station s_0 with empty sets for transmission points and collected data. The set $\mathcal{Y}$ is initialized with all possible hovering points, and $\mathcal{D}$ is initially empty.

Algorithm 1: Heuristic algorithm

```
1  Unmanned Aerial Vehicle (UAV) hovers at $h_j$, collects data $D_j$ from Internet of Things
     Device (IoTD) $S_j$, and
2  $\mathcal{T}^{tour} = \{s_0\}$;
3  $\mathcal{T}^{transmission} = \{\}$;
4  $\mathcal{Y} = \mathcal{H}$;
5  $\mathcal{D} = \{\}$;
   /* Determine the UAV Trajectory                                                     */
6  while $\mathcal{Y} \neq \emptyset$ do
7      for each $h_j \in \mathcal{Y}$ do
8          $w_j = \frac{\log_{10} DR_j}{||\mathcal{T}^{tour}[-1] - h_j||}$;
9      $h^* = arg\max_{h_l \in \mathcal{Y}} w_l$;
10     $\mathcal{T}^{tour} = \mathcal{T}^{tour} \cup h^*$;
11     $\mathcal{Y} = \mathcal{Y} \backslash h^*$
   /* The last point of the tour is the base station                                   */
12 $\mathcal{T}^{tour} = \mathcal{T}^{tour} \cup s_0$
   /* Let's initialize $h_j$ at the third value of the set $T^{tour}$                    */
13 $h_j = \mathcal{T}^{tour}[1]$
14 $AoI = 0$;
15 $t = 0$;
   /* Let's compute the transmission positions                                         */
16 for each $h_n \in \mathcal{T}^{tour}[2 : |\mathcal{T}^{tour}| - 1]$ do
17     $\mathcal{D} = \mathcal{D} \cup \{D_j\}$;
       /* Determine whether to transmit now or from next position.                     */
18     Rank the data in $\mathcal{D}$ in acceding order according to their data size.
19     Generate the action set $\mathcal{A}^j$ size of $2^{|\mathcal{D}|}$ , each action $k$ is a binary vector $a_k^j \in \mathcal{A}^j$
         size of $\mathcal{D}$;                                                          */
20     /* /* Evaluate the AoI of all possible transmission combination                 */
21     $AoI_{set} = \{\}$;
22     for each $a_k^j \in A^j$ do
23         $AoI_k = 0$;                                                                  */
24         /* /* Loop through all the actions of the current combination              */
25         for each $a_{k,l}^j \in a_k^j$ do
26             if $a_{k,l}^j = 1$ then
                   /*  transmit the data at the current hovering point $h_j$            */
27                 $AoI_l = \frac{\|D_l\|}{DR_j}$;
28             else
                   /*  transmit the data at the next hovering point $h_n$               */
29                 $AoI_l = \frac{||h_j - h_n||}{V} + \frac{\|D_l\|}{DR_n}$;
30             $AoI_k = AoI_k + AoI_l$;
31         $AoI_{set} = AoI_{set} \cup AoI_k$
       /* Select the optimal combination that minimize the AoI                         */
32     $a_*^j = argmin_{a_k^j \in A^j}(AoI_{set})$;
33     for each $a_{*,l}^j \in a_*^j$ do
34         if $a_{*,l}^j = 1$ then
               /*  transmit the data at the current hovering point $h_j$                */
35             $t = t + \frac{\|D_l\|}{DR_j}$;
36             $AoI = AoI + t$;
37             $\mathcal{D} = \mathcal{D} \backslash \{D_l\}$;
38             $\mathcal{T}^{transmission}[j] = \mathcal{T}^{transmission}[j] \cup l$
       /* Update the value of $h_j$                                                      */
39     $h_j = h_n$
40 Output $\mathcal{T}^{tour}$, $\mathcal{T}^{transmission}$;
```

To determine the UAV trajectory, the algorithm iterates while all the hovering points in $\mathcal{Y}$ are visited. Every iteration, for each point h_j in $\mathcal{Y}$, a weight w_j is calculated based on the logarithm if the data rate (DR_j) and the distance from the current tour endpoint. The data rate dampens the effect of data rate. The point h^*, with the maximum weight is selected as next hovering point to be visited and added to the tour $\mathcal{T}^{tour}$. The final point in the tour is set to be the base station.

Next, the transmission pattern is computed.

Let's define two variables used in the code:

- h_j is the current hovering point
- h_n is the next hovering point

The transmission process is initialized setting as first hovering point h_j the first sensor in the tour, excluding the base station. The AoI and time are both set to 0.

For each hovering point h_j, the collected data D_j is added to $\mathcal{D}$(represent the data collected and not yet transmitted) and the data is ranked by size. All possible transmission actions are generated for the data as binary vector[2]

The Age of Information (AoI) for each action is evaluated by considering whether to transmit at the current (1) or next (0) hovering point. For example, if the combination is 101, the AoI is computed as follows: data 1 and 3 are transmitted from h_j while data 2 is transmitted from h_n.

The action that minimizes the AoI is selected and the optimal transmission actions are executed. The transmission time and AoI are updated, and the transmitted data is removed from $\mathcal{D}$ the set. The current hovering point is then updated to the next in the tour.

[2]E.g. if $\mathcal{D}$ has three data the possible combinations are 9:(000, 001, 010,..., 111).

Finally, the algorithm outputs the final UAV tour and transmission points. This heuristic algorithm efficiently plans the UAV's trajectory and transmission strategy to minimize the AoI for data collected from IoT devices.

3.3.3 UCB algorithm

The previously discussed solutions perform effectively when the data rate is constant and known. However, their performance deteriorates significantly when these conditions are not met.

To address this issue, the Upper Confidence Bound (UCB) algorithm has been implemented. This reinforcement learning technique is particularly suited for tasks requiring the discovery of distributions and selecting actions that maximize "random" rewards. The theoretical foundation of the UCB algorithm is detailed in Section 2.4.

In our specific application, the goal is to minimize the Age of Information (AoI). Thus, the UCB formula has been adapted accordingly. The standard UCB action selection formula:

$$a_t = \arg\max_{a \in A} \left(\hat{\mu}_a(t) + c\sqrt{\frac{\ln t}{N_a(t)}} \right)$$

has been modified to:

$$a_t = \arg\max_{a \in A} \left(\hat{\mu}_a(t) - c\sqrt{\frac{\ln t}{N_a(t)}} \right)$$

In this revised formula, the "+" sign is replaced with a "-" sign to reflect the objective of minimizing the AoI rather than maximizing the reward. This version of the algorithm is called the Lower Confidence Bound (LCB), but to avoid confusion, we will continue to refer to it as UCB.

The pseudo-code of the UCB algorithm is shown in 2

Algorithm 2: UCB algorithm

```
    /* Initialization phase                                                   */
1   $limit = X$
2   $tour = 0$;
3   $\hat{\mathbf{b}} = [0]_{1\times|M|}$;
4   $\mathbf{d} = [0]_{1\times|M|}$;
5   Calculate the UAV trajectory $\mathcal{T}^r$ with Nearest Neighbor [] and UAV starts data collection following $\mathcal{T}^r$.
6   for $each\ h_j \in \mathcal{T}^{tour}$ do
7       UAV hovers at $h_j$, collects data $D_j$ from IoTD $S_j$, and immediately transmits the collected data to the Base
          Station (BS);
8       Observe the total transmission time $T_i$ at the BS and calculate the data rate $b_j^{tour} = \frac{|D_j|}{T_j}$;
9       Update $\hat{b_i} = b_j^{tour}$ and $d_j = d_j + 1$;
    /* Learning phase                                                         */
10  while $tour < limit$ do
11      $tour = tour + 1$;
12      $\mathcal{T}^{tour} = \{s_0\}$;
13      $\mathcal{Y} = \mathcal{H}$;
        /* Recompute the UAV Trajectory                                       */
14      while $\mathcal{Y} \neq \emptyset$ do
15          for $each\ h_j \in \mathcal{Y}$ do
16              $w_j = \frac{\log_{10}\hat{b_j}}{||\mathcal{T}^{tour}[|\mathcal{T}^{tour}|-1]-h_j||}$;
17          $h^* = arg\max_{h_l\in\mathcal{Y}} w_l$;
18          $\mathcal{T}^{tour} = \mathcal{T}^{tour} \cup h^*$;
19          $\mathcal{Y} = \mathcal{Y} \backslash h^*$
        /* The last point of the tour is the base station                     */
20      $\mathcal{T}^{tour} = \mathcal{T}^{tour} \cup s_0$
21      $ChangePath = False$
22      while $ChangePath = False$ do
            /* Let's initialize $h_j$ at the first value of the set $\mathcal{T}^{tour}$    */
23          $h_j = \mathcal{T}^{tour}[1]$;
24          $\mathcal{T}^{transmission} = \{\}$;
25          $\mathcal{D} = \{\}$;
26          $AoI = 0$;
27          $t = 0$;
            /* Let's compute the transmission positions                       */
28          for $each\ h_n \in \mathcal{T}^{tour}[2 : |\mathcal{T}^{tour}| - 1]$ do
29              $\mathcal{D} = \mathcal{D} \cup \{D_j\}$;
                /* Determine whether to transmit now or from next position.   */
30              Rank the data in $\mathcal{D}$ in acceding order according to their data size.
31              Generate the action set $\mathcal{A}^j$ size of $2^{|\mathcal{D}|}$, each action $k$ is a binary vector $\mathbf{a}_k^j \in \mathcal{A}^j$ size of $\mathcal{D}$;
                /* Evaluate the AoI of all possible transmission combination  */
32              $AoI_{set} = \{\}$;
33              for $each\ a_k^j \in A^j$ do
34                  $AoI_k = 0$;
                    /* Loop through all the actions of the current combination */
35                  for $each\ \boldsymbol{a}_{k,l}^j \in a_k^j$ do
36                      if $\boldsymbol{a}_{k,l}^j = 1$ then
                            /*  transmit the data at the current hovering point $h_j$  */
37                          $AoI_l = \frac{\|D_l\|}{\hat{b}_n - c\cdot\frac{\sqrt{ln(tour)}}{d[j]}}$;
38                      else
                            /*  transmit the data at the next hovering point $h_n$     */
39                          $AoI_l = \frac{||h_j - h_n||}{V} + \frac{\|D_l\|}{\hat{b}_n - c\cdot\frac{\sqrt{ln(tour)}}{d[n]}}$;
40                      $AoI_k = AoI_k + AoI_l$;
41                  $AoI_{set} = AoI_{set} \cup AoI_k$
                /* Select the optimal combination that minimize the AoI      */
42              $a_*^j = argmin_{a_k^j \in A^j}(AoI_{set})$;
43              for $each\ \boldsymbol{a}_{*,l}^j \in a_*^j$ do
44                  if $\boldsymbol{a}_{*,l}^j = 1$ then
                        /*  transmit the data at the current hovering point $h_j$      */
45                      $t = t + \frac{\|D_l\|}{DR_j}$;
46                      $AoI = AoI + t$;
47                      $\mathcal{D} = \mathcal{D}\backslash\{D_l\}$;
48                      $\mathcal{T}^{transmission}[j] = \mathcal{T}^{transmission}[j] \cup l$
49                      $z_{lj} = 1$
                /* Update the value of $h_j$                                  */
50              $h_j = h_n$
51          for $each\ h_j \in \mathcal{H}$ do
52              $b_j^{tour} = \frac{\sum_i |D_i|\cdot z_{ij}}{T_j}$;
53              if $\sum_i z_{ij} \geq 1$ then
54                  Update $\hat{b}_j = \frac{\hat{b}_j\cdot d[j] + b_j^{tour}}{d[j]+1}$ and
55                  $d[j] = d[j] + 1$;
56          If the criteria to recompute the path is met, $ChangePath = True$
57  Output $\mathcal{T}^{tour}$,$\mathcal{T}^{transmission}$, $\hat{b}$;
```

Let's analyze the code in order to understand how it works.

Initialization Phase

1. **Initialization of the Variables:**
   - *limit* $= X$: Set the maximum number of tours.
   - *tour* $= 0$: Initialize the tour counter.
   - $\hat{\mathbf{b}} = [0]_{1\times|M|}$: Initialize the estimated data rate for each hovering point to zero.
   - $\mathbf{d} = [0]_{1\times|M|}$: Initialize the number of times a transmission has been performed from a certain location.
2. **Compute Initial UAV Trajectory:**
   - An initial tour is conducted to gain preliminary data on the datarate across different locations. During this first tour, transmissions are attempted from each location. If a location lacks a connection and the transmission fails, the UAV will wait for a time-out period before moving to a new location. The data collected from the location with no connection will be transmitted back to the base station. Although this initial tour will result in a high Age of Information (AoI), it is essential for initializing the UCB algorithm.

Learning Phase

Now that the initialization phase has been completed the learning phase can start

(a) **Computation of the tour**
   - The tour is determined using the Heuristic algorithm described in the previous section. The data rate value used

to compute the path is denoted as $\hat{b}$. Initially, $\hat{b}$ does not accurately represent the actual data rate. However, as the UAV iterates through the process, the value $\hat{b}$ converges to the true data rate. Consequently, the path computed using $\hat{b}$ will eventually become optimal.

- The path is computed and stored in T^{tour}

3. **Data Transmission Optimization:**
    - Set the last point of the tour as the base station.
    - Initialize $ChangePath$ to $False$.
    - While $ChangePath$ is $False$(this variable is used to communicate that is time to recompute the path), the transmission location optimization is run. It works exactly as in the heuristic algorithm, the only difference is the data rate value used to compute the transmission path. In fact, in order to explore the and find the best solution the data rate used isn't the learned one $\hat{b}$, but it is modified with the UCB formula. The data rate value used to compute the the transmission order is $\hat{b}_n - c \cdot \frac{\sqrt{ln(tour)}}{d[j]}$, that is the learned value decrease of a certain factor. This factor allows to explore more at the beginning, since $\hat{b}$ is completely wrong, and to exploit more and more the number of tour grows and the $\hat{b}$ converge to the expected value.

      After determining the transmission order, the actual transmission uses the data rate value available at the location at that specific time.

4. **Update Data Rates:**
    - For each hovering point h_j:
        - Calculate the new data rate b_j^{tour}.

  - Update the estimated data rate $\hat{b}_j$ if data was transmitted.
  - Increment the data collection count $d[j]$.

5. **Path Recalculation Criteria:**

   - Set $ChangePath$ to $True$ if criteria are met to recompute the path. If the criteria are not met, the UAV does another transmission tour with the same path, but maybe changing the transmission order.

In summary, the algorithm iteratively improves the UAV's trajectory and transmission strategy to optimize data collection and transmission efficiency by leveraging the UCB approach to balance exploration and exploitation in learning the best data rates at different hovering points.

Chapter 4

Experiments

This chapter provides a comprehensive explanation of how the experimental setup has been implemented. It details the components that make up the setup and describes how the software operates. Furthermore, the chapter gives a deep explanation of the data collection process, which is fundamental to the project. Finally, it presents the results obtained from the data collection, offering an brief analysis and discussion of the findings.

4.1 LoRa Testbed Implementation

To test the performance of the LoRa connection and the data collection capabilities, two LoRa nodes were constructed: one for transmitting data and the other serving as a base station.

The configuration of the two nodes is as follows:

- Raspberry Pi 3 Model B
- Heltec WiFi LoRa 32 V2 Module
- Bendable Antenna TX915-JKD-20

Additionally, on the transmitter side, to enhance user interaction with the system, a 3-inch touch screen was integrated. This touch

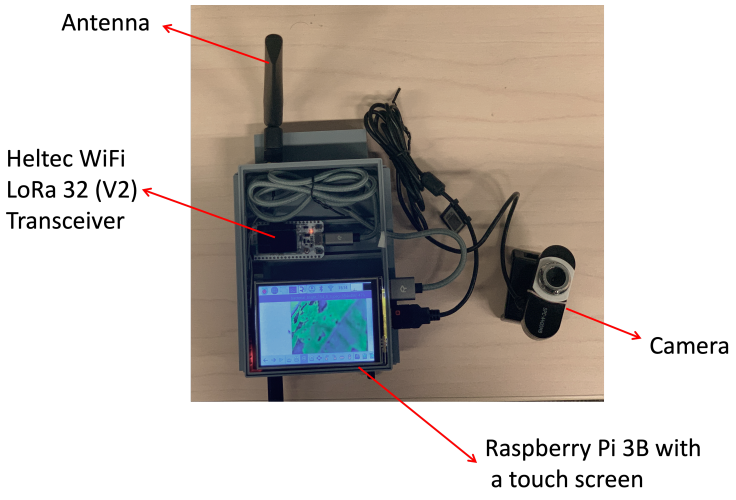

Antenna
Heltec WiFi LoRa 32 (V2) Transceiver
Camera
Raspberry Pi 3B with a touch screen

Figure 4.1: Test bed implemented. This is the transmission node. The camera has not been used in our test

screen facilitates easier control and monitoring of the data transmission process.

Overall, by combining the Raspberry Pi 3 Model B, touch screen, RGB camera, Heltec WiFi LoRa 32 V2 transceiver, and high-gain external antenna, I created a robust and versatile LoRa node. This setup supports data collection and transmission and enhances user interaction and visualization, making it suitable for a wide range of IoT applications.

The following subsections will detail the specific components and configurations used in the testing setup.

4.1.1 Heltec WiFi LoRa 32 V2 Module

The Heltec WiFi LoRa 32 V2 is a versatile development board that combines WiFi, Bluetooth, and LoRa (Long Range) communication capabilities. It is based on the ESP32 microcontroller, which integrates a dual-core processor with robust processing power and a rich set of features suitable for a wide range of IoT applications. The board includes an SX1276 LoRa transceiver chip, which operates in the 915

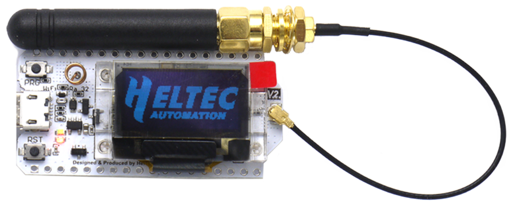


Figure 4.2: Heltec WiFi LoRa 32 V2

MHz frequency band commonly used in North America. LoRa technology is known for its long-range, low-power communication, making it ideal for IoT applications that require devices to communicate over long distances. Additionally, the Heltec module features an onboard 0.96-inch OLED display with a resolution of 128x64 pixels, which is useful for debugging, monitoring, and displaying information without needing an external screen.

The ESP32 microcontroller provides built-in WiFi and Bluetooth connectivity, allowing the device to connect to the internet, local networks, or other Bluetooth devices. This dual connectivity makes it suitable for a variety of applications, including remote monitoring, data logging, and control systems. The Heltec WiFi LoRa 32 V2 supports a wide range of peripherals and interfaces, including multiple GPIO pins, UART, I2C, SPI, and ADC interfaces, enabling it to interface with a wide range of sensors, actuators, and other devices.

Power management is another crucial feature of the Heltec WiFi LoRa 32 V2. It can be powered via a micro-USB connector or an external power source, and it includes a battery connector for portable and remote applications. The board has built-in power management features to optimize power consumption, which is critical for battery-operated devices.

This module provides a ESP32 + LoRaWAN protocol Arduino library, this is a standard LoRaWAN protocol that can communicate with any LoRa gateway running the LoRaWAN protocol.

In the experiments the module was connected to a Raspberry Pi 3B, the gateway, via serial port. This solution was adopted in order to provide the necessary processing power and flexibility for handling data collection and transmission tasks. Both on the raspberry Pi and the Heltec module run a custom code, that will be briefly explained later

4.1.2 Bendable Antenna TX915-JKD-20

To get the best performance from the LoRa communication, a high-gain external antenna has been used. This antenna operates at the 915 MHz frequency band, boosting the signal strength and range of the LoRa transceiver. Here are some key features of the antenna:

- Frequency: 915 MHz
- Bandwidth: 900-931 MHZ
- Gain: 5dbi
- Radiation Direction: Omnidirectional
- Polarization: Vertical
- Power Capacity: 20W

A higher gain antenna could have been utilized to achieve a longer communication range; however, a trade-off between range and maneuverability was considered and ultimately adopted.

4.1.3 Software Implementation

An important part that needs to be deepened is the algorithm implemented on the Raspberry Pi, used to manage the communication between the two nodes. The Raspberry Pi communicates with the LoRa module via serial communication.

Serial interfacing between a Raspberry Pi and a LoRa module involves a few critical steps to ensure efficient data transmission and reception. Initially, the Pi sends a stream of bytes up to a certain buffer limit and waits for the LoRa module to signal readiness for more data. The LoRa module automatically forwards any received data to the Raspberry Pi.

There are two processes we need to analyze:

- The transmission
- The reception

Transmission

The transmission starts with the sending of the header packet. Multiple header packets are sent to account for any potential packet loss. These header packets inform the receiver of the total number of bytes and the name of the file being transmitted.

After the receiver is notified with the communication information, the entire file is then sent packet by packet over the serial interface for LoRa transmission. After sending the file, the transmitter waits for a final handshake from the receiver confirming successful transmission without any missing packets. If packets are missing, the receiver sends back the IDs of the missed packets, which are then retransmitted by the transmitter. This process repeats until all packets are successfully received.

Reception

On the receiving end, the process begins by waiting for a header packet to determine the expected number of packets. The receiver then reads from the serial interface the packets. It keeps track of the number of the packet that it is receiving and if the expected packet ID is not received, it is added to a list of packets to request for retransmission. Once the final packet is detected or a timeout occurs (in case the final packet is lost during the transmission), the receiver requests any missing packets if necessary. If there are no missed packets, a confirmation is sent to the transmitter. Otherwise, the process of retransmission starts.

Packet Structure

The packet structure consists of 64 bytes. The first two bytes are used to represent the packet ID while the last two bytes contain the checksum to detect if there have been any problems in the serial communication. The remaining 60 bytes make up the data payload. If the payload is not fully utilized, the packet is padded with null data to maintain communication integrity.

Correcting the communication protocol ensures reliable and efficient data transfer between the Raspberry Pi and the LoRa module, handling potential issues like packet loss and ensuring data integrity through checksums.

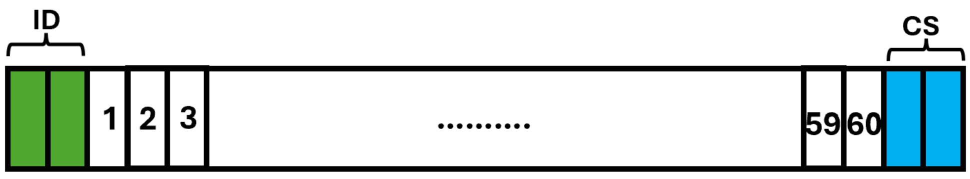


Figure 4.3: Structure of the packet

4.2 Data Collection

A fundamental part of this work has been the data collection process. However before conducting the final data collection the best setting for the various parameters had to be found.

The performance tests were conducted to better understand how the urban environment affects transmission range and to determine the optimal setup for parameters such as Bandwidth (BW) and Spreading Factor (SF).

4.2.1 Tuning the testbed

First, several tests were conducted to understand the capabilities of the testbed. The base station was fixed in the lab, and the transmitter was brought to two different locations, as shown in Figure 4.4.

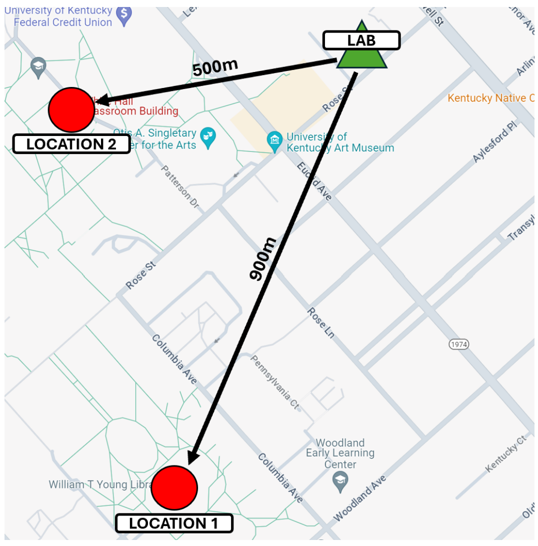


Figure 4.4: Map with the the two locations used for the preliminary tests

The first location was the campus library, situated 900 meters from the base station. After conducting several tests, it was observed that the urban environment significantly affected the transmission range.

To achieve successful transmission, very conservative parameters were required:

- SF > 10
- BW $= 125$ KHz

This parameter setup allowed communication between the two nodes, but the transmission was very slow (100 bytes/s) and unstable, leading to considerable data loss. The RSSI was very close to the connection limit for our module and antenna, which is -130 dBi. This demonstrated the significant impact of buildings on the signal. In an open field, with the same setup, stable communication can be achieved up to 5-6 km.

Having discovered the setup's limits, I then aimed to understand how various parameters affected the signal and to determine the best settings for this research. The second test was conducted in another campus building, located 500 meters from the laboratory.

The setup of the experiment was as follows:

- One PC[1] in the lab with the LoRa module connected(fig. 4.5)
- One PC on the receiving side (with me)
- Remote control of the lab PC using TeamViewer
- Re-uploading the code to the LoRa module using the Arduino IDE for both Tx and Rx sides whenever changing spreading factor and bandwidth values. The coding rate remains unchanged, as any value other than 5 worsened the connection.
- Sending the same image file over LoRaWAN each time
- Collecting the following data for each set of parameters:

[1]For this test the PC was used instead of the Raspberry-pi in order to have a more efficient experiments setup

  - Missing Packets Rate
  - Tx Time (No Re-Tx)
  - Bit Rate

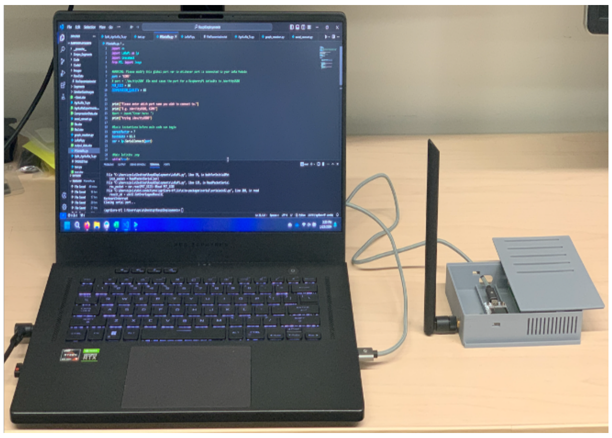


Figure 4.5: Receiver node

The Bandwidth values ranged from 125 to 500 kHz, while the spreading factor ranged between 7 and 10 (values of 11 and 12 were too high for our purposes). Two parameters have been fixed and the same values will be used for all the next transmissions:

- Transmission Power: 17dBm
- Coding factor : $\frac{4}{5}$

After analyzing the results, the chosen parameters for subsequent experiments were BW = 250 kHz, CR = 5, and SF = 9. These values were selected as they provide a good trade-off between speed and transmission robustness as it is possible to notice in 4.6.

These values will be used for the next experiments.

4.2.2 Data Collection

The data collection part has been fundamental in order to understand how the signal was spread in a urban environment. After the analysis

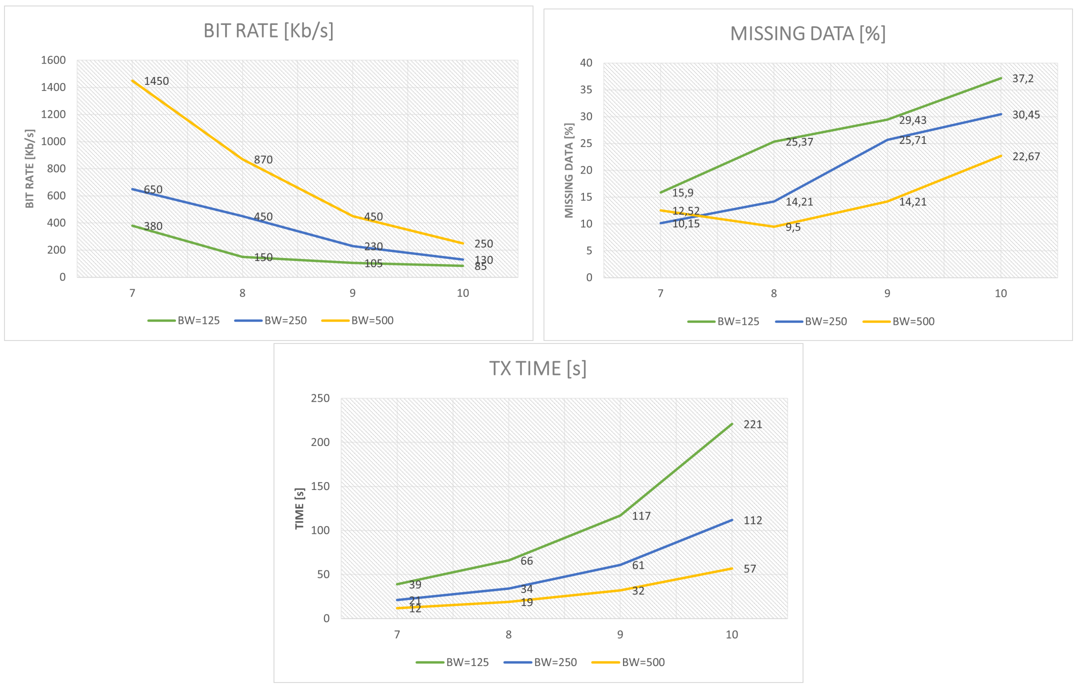


Figure 4.6: Data obtained during the tuning phase. The x-axis shows the values of the SF.

done in the previous section, the dimension of the area selected for the collection of the data is $1km^2$, with the base station placed in the middle. This dimension has been selected because the range with the parameters selected is about 600 meters in the best direction(with less buildings).

The area has been divided in 100 squares with dimension 100x100. This splitting has been selected since a bigger dimension of the squares would have lost some information about the spreading of the signal, while a smaller size would have capture more information but the collecting time would have increased drastically. So, for our setting, this resolution was the best one.

The data collection process requires to take 6/8 measurements from every cell of the grid. Considering the measurements of a cell, they had to be taken in two different hours and days, in order to highlight how, even within the same cell, in different moment, the signal was different. This was accomplished collecting data in February and May.

The grid that was placed on the real map, in order to find the best location. The lab has been selected as central point. The creation of the grid has been done with a python algorithm, that gives in input the coordinate of the central point, the number of rows and columns, the size of the cells and the orientation of the grid is able to plot on google maps the resulting grid, as it is possible to see in 4.9. In order to obtain a really accurate grid the position of the various points has been computed using the Vincenty algorithm[2]

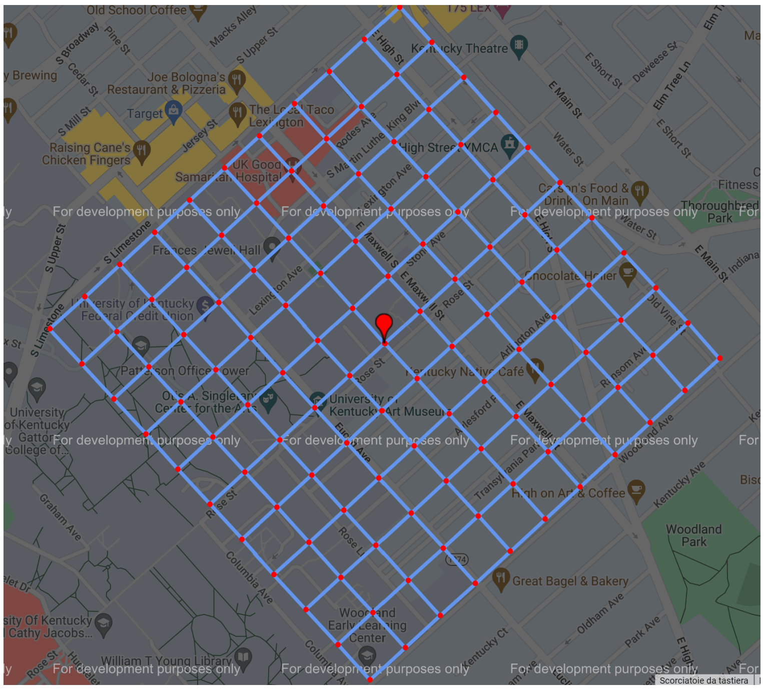


Figure 4.7: Grid used to collected the points. It is placed in the actual location where the points have been collected

Now that the collecting area has been defined it is possible to start to collect the points. The setup used for the points collection is explained in the section 4.1

[2]The Vincenty algorithm is a formula used to calculate the distance between two points on the Earth's surface. It accounts for the Earth's shape, providing a more accurate result compared to simpler distance calculations that assume a perfectly spherical Earth.

In order to speed up the collection, the gateway of the receiver has been modified so that it was able to communicate with telegram.

In fact given that the RX device must remain fixed in the lab, a Telegram bot to monitor the live reception of files has been implemented. The Telegram bot facilitates control over the main receiving program, enabling the user to launch it, check transmission parameters, and receive live updates on their phone. Upon successful transmission, the bot sends detailed transmission parameters (bit rate, time, RSSI) and saves them into a file for historical reference. Additionally, it prompts the user to send the transmission coordinates to autonomously plot the grid points. In the event of transmission failure, the bot requests a new transmission attempt. This system is really robust and it allows a easier points collection.

Measurements were taken at all grid points to provide an overview of the LoRa signal distribution. The antenna was placed in four different locations, one on each side of the lab. This was done because the antenna couldn't be placed on the roof, and if the antenna was left in the middle of the lab, the signal would have been strongly affected by the walls. To simulate the placement on the roof, I physically moved the antenna close to four windows (less interference) on the four sides of the lab. The error introduced is negligible.

As mention before, 6/8 points has been collected in every useful grid: the cells too far away, where the signal is strongly unstable or it doesn't exist, didn't gave any information, so I didn't perform any measurements.

The points within a cell were collected trying to cover all the part of the cell so that the data collected best reflects the nature of the signal.

4.3 Data analysis

After the collection process was finalised, the data collected has been processed.

The collected data has been filtered and plotted on 2D and 3D graphs to facilitate analysis. The metrics used to analyse the range distribution are:

- RSSI
- Total transmission time(including re-transmission of the lost package)
- Data Rate
- Missing packets

It is crucial to consider the total transmission time rather than just the transmission time without the re-transmission, as the latter is relatively constant across points, but it doesn't express the quality of the signal. In fact, where the RSSI is lower and so the transmission more unstable the number of missing packets increase, affecting retransmission time.

As seen in Figure 4, the data are not uniformly distributed. All four graphs are clearly correlated with each other.

One interesting observation is that distance is not the only factor affecting signal quality; obstacles present in the environment also play a significant role. This can be observed in cell [5;2], which has no connection despite the surrounding cells having a connection.

The points collected were plotted on the real map in order to show how the distribution of the collected points and where they have been collected. The map is showed in 4.9. The last analysis performed on the collected points involved computing the standard deviation to quantify how much the signal varies within a cell. The results are

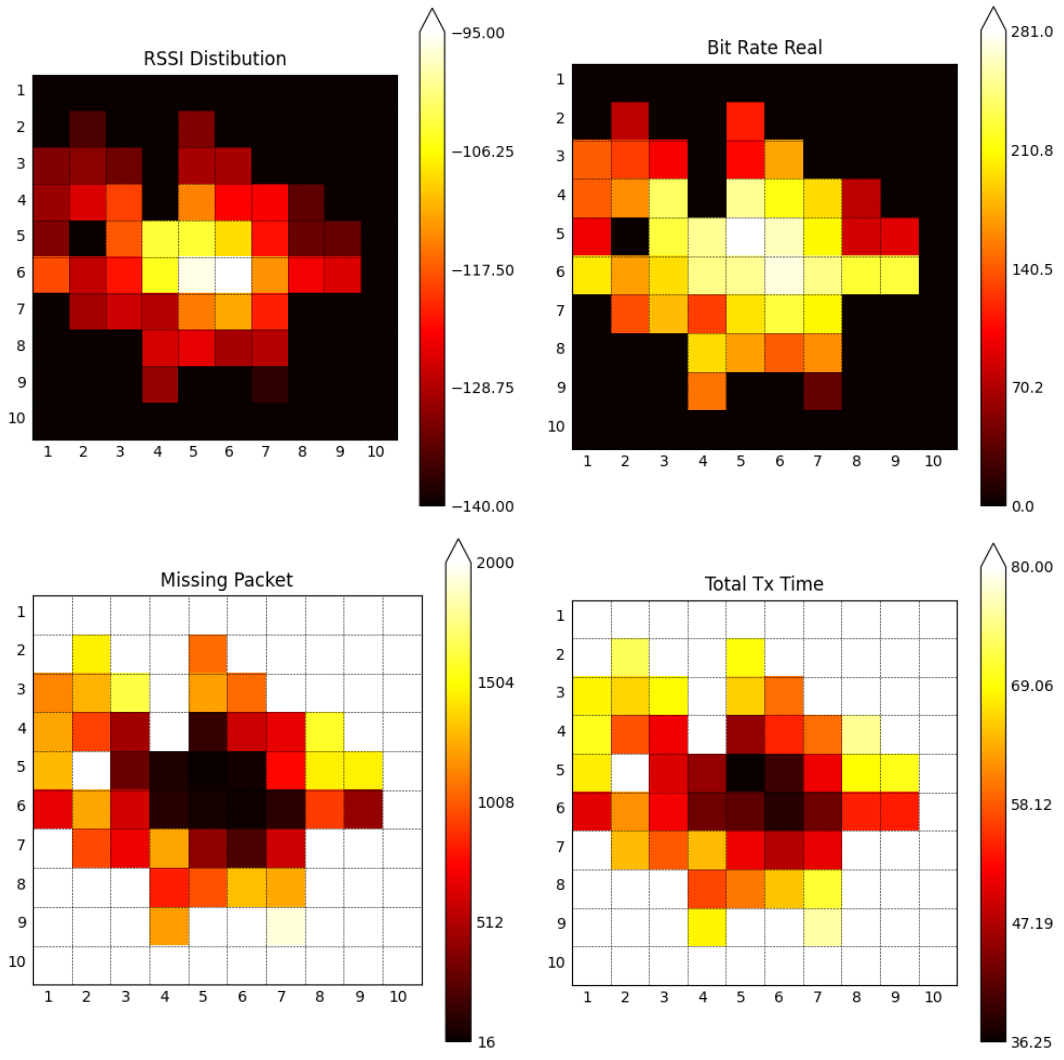


Figure 4.8: Plots with the different metrics used to evaluate the distribution of the signal

displayed in Figure 4.10. Only the cells with a connection are present in the graph, as the others don't have any standard deviation.

This values are then stored in a .txt file in order to be used in the next chapter.

The results indicate heterogeneity, with data not uniformly distributed due to the presence of buildings, cars, and trees affecting signal distribution. This finding supports the implementation of an

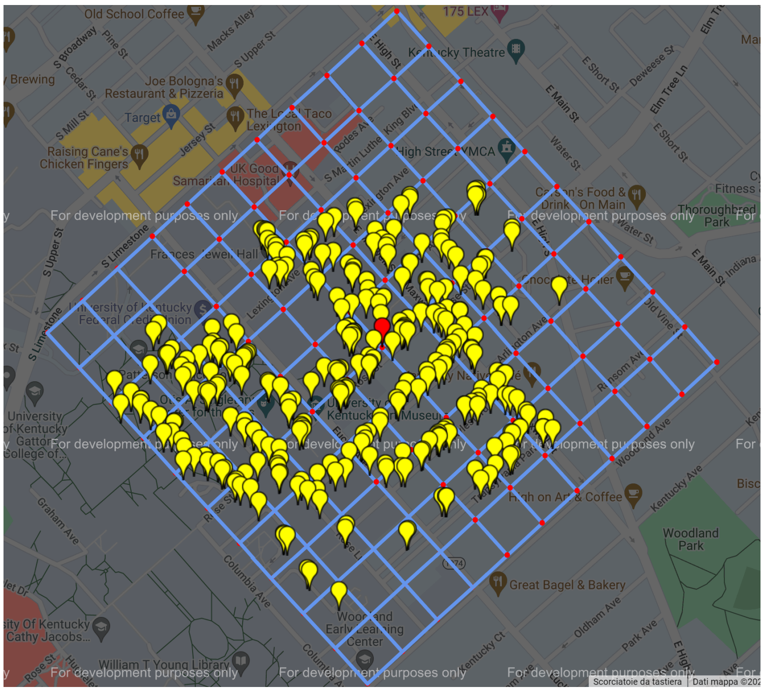


Figure 4.9: Grid showing the points collected. Every yellow marker contains the information about its own transmission

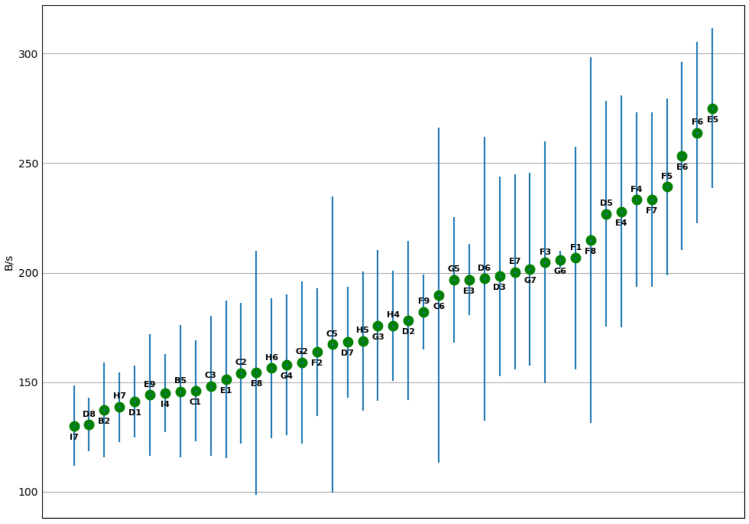


Figure 4.10: Average DR values(green dots) and their own variances(blue lines). Here are displayed only the cells which values are different from zero

algorithm to identify the best transmission spot in order to minimize the AoI and the trajectory. The 3D graph4.11 shows perfectly this

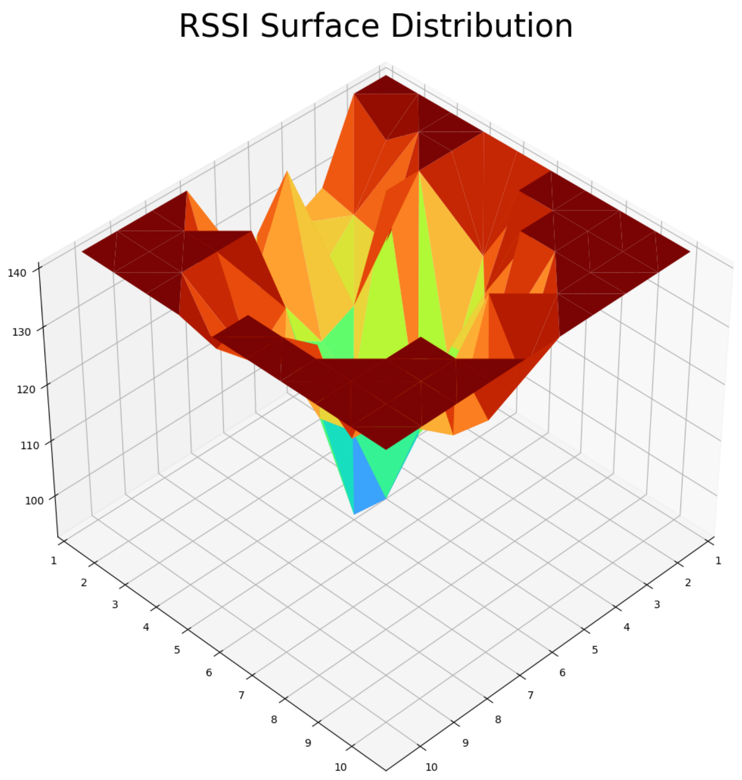


Figure 4.11: 3D representation of the RSSI distribution

heterogeneity, using the RSSI as metrics.

Chapter 5

Results

After an exhaustive presentation of the implemented solution and the data collection process, we can now analyse the results obtained. As explained in Chapter 3, three main solutions have been implemented:

- Gurobi Solver
- Heuristic Algorithm
- UCB Algorithm

As will be analyzed in this chapter, the evolution of the solution has been necessary as the complexity of the problem increased, requiring the solutions to evolve accordingly.

The first two solutions operate under the assumption that the data rate in every cell of the grid is constant, meaning it doesn't vary over time and is known beforehand. This is a flawed assumption, as in real scenarios, the data rate can vary even between two consecutive transmissions from the same point. We aimed to address the variability of the data rate and to implement a case where the distribution of the data rate was unknown, requiring the UAV to learn it. Therefore, we decided to utilize a reinforcement learning technique capable of exploring and learning the distribution of the data rate, and find the best path to minimize the AoI.

Before presenting the results, it is important to highlight that for all the subsequent experiments, the speed of the drone is fixed at $7\frac{m}{s}$, while the data generated by the sensors are considered to be of the same size, equal to 800 Bytes.

5.1 Gurobi VS Heuristic

Let's dig deeper into the results obtained with the first two solutions. Before starting to explain the results obtained, it's important to make a crucial premise: the data rates during the resolution of these solutions is considered **constant**

5.1.1 Gurobi

The first approach to solving the problem involves using Gurobi. The problem formulation is detailed in 3.3.1.

Gurobi, being an optimizer, provides the optimal solution, which is the best possible path and transmission schedule that can be achieved given a specific distribution of sensors. Every cell of the grid possesses a unique real value representing the data rate. For example, When the UAV collects data from a sensor located in cell B5, the data rate between the UAV and the base station will correspond to that of cell B5.

In Figure 5.1, a solution obtained using Gurobi is presented. Given 7 sensors randomly distributed within the grid, Gurobi computes the optimal solution to minimize the AoI. The map on the right illustrates the path followed by the UAV (depicted by the red arrow) and the actions performed at each sensor location:

- The green numbers above each sensor indicate the data transmitted back to the base station from that location.

- The black numbers indicate the duration for which the UAV remains at that location.

The map on the left visualizes the data rate at each sensor location.

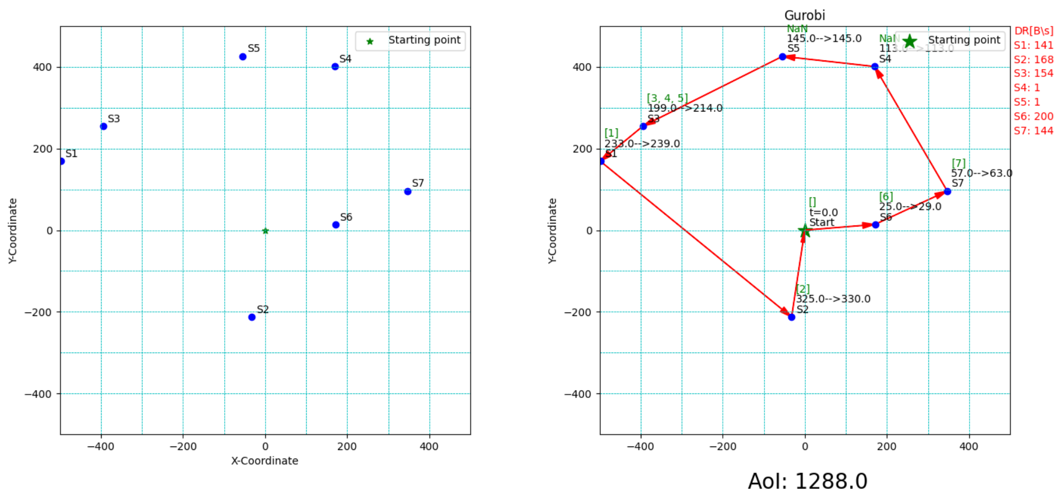


Figure 5.1: Joint optimization of the path and transmission using Gurobi.

5.1.2 Heuristic

The second approach used to solve this problem has been the implementation of an heuristic algorithm, detailed in 3.3.2.

Since the heuristic search is used, the solution obtained is sub-optimal. However, the results are promising and not too far from the optimal solution.In Figure 5.2 the solution generated by the heuristic method is shown. The sensor distribution is identical to that used in the previous section, allowing for an initial visual comparison between the two solutions. A more in-depth comparison between these two solutions will be provided in the next section.

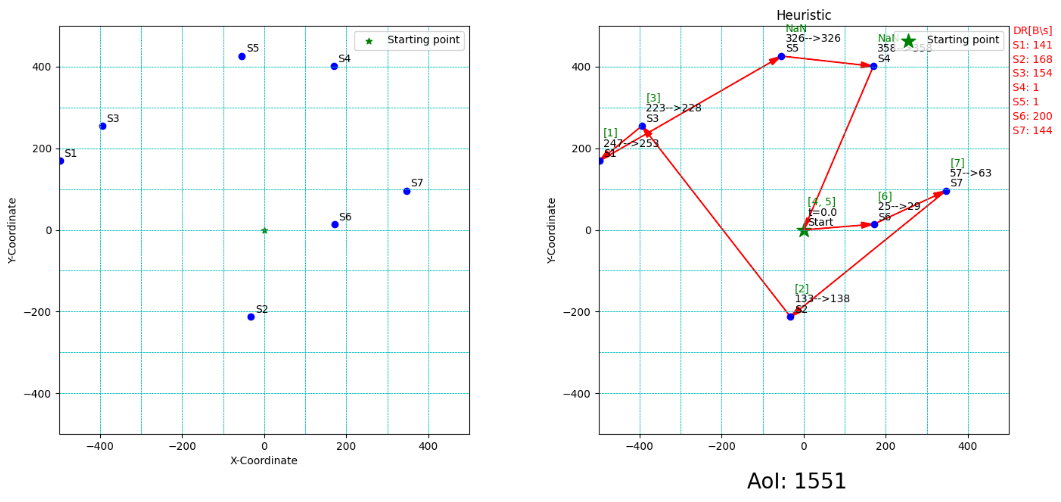


Figure 5.2: Joint optimization of the path and transmission using the Heuristic algorithm.

5.1.3 Performances comparison

Although Gurobi provides the optimal solution, it requires substantial computational power. Indeed, the problem presented in 3.3.1 is NP-hard, meaning that the complexity increases exponentially with the number of sensors.

This complexity precludes the exploration of scenarios with a higher number of sensors, and seven sensors are insufficient to address a real-world case. Therefore, we decided to implement the heuristic algorithm. Using this solution, the optimal outcome is not guaranteed. To better understand the performance of our algorithm relative to the optimal solution, several tests were conducted, and the results are shown in Figure 5.3.

The optimization was performed on 40 maps using Gurobi initially and then the heuristic algorithm. The average of the three most significant parameters is displayed. During these experiments, the number of sensors considered was 7.

The first bar illustrates the comparison of the total travel time: as observed, the heuristic solution computes a more greedy path solution (evident in the way the path is computed), resulting in a faster travel time. However, the greediest path solution for this type of problem is not optimal.

The third bar shows the average Age of Information (AoI): on average, the heuristic solution results in an AoI that is 18% higher. This is a reasonably good result when considering the computational time (second bar). To compute the solution with 7 sensors, Gurobi takes 70 seconds per map, while the heuristic algorithm only requires 1 second. This difference in computation time increases exponentially with the number of sensors, but this issue will be addressed in the next section.

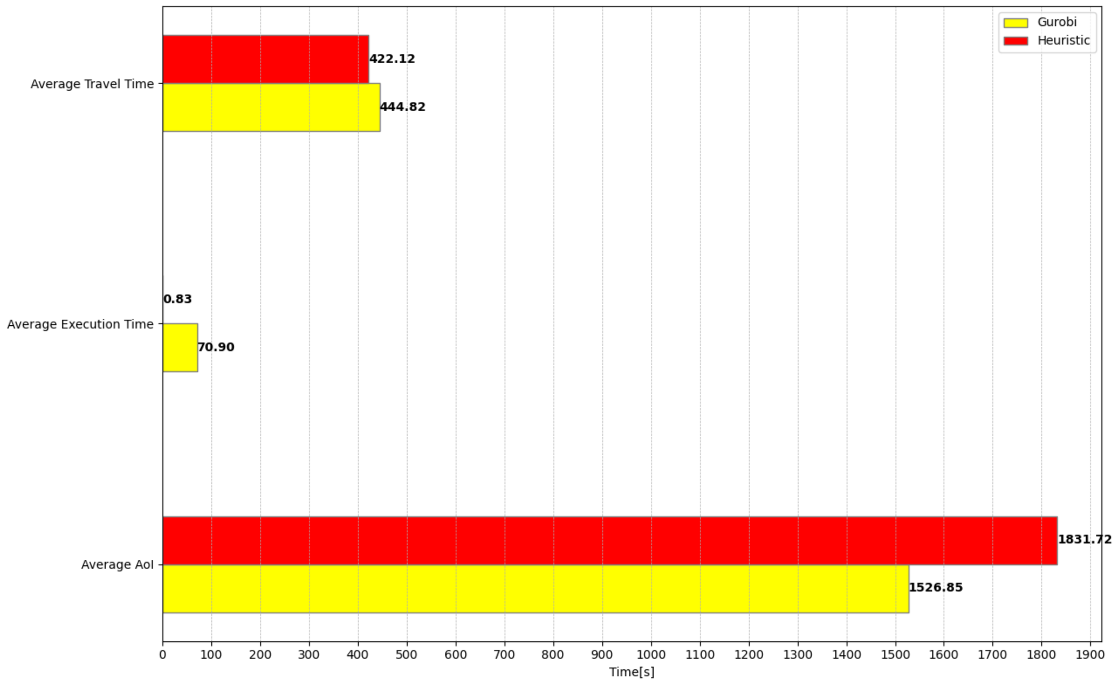


Figure 5.3: This bar graph compares the performances of the Gurobi optimization solver and the Heuristic algorithm. The performance metrics are plotted on the Y-axis, the time in [s] on the X-axis

Let's now compare two solution obtained with the two methods.

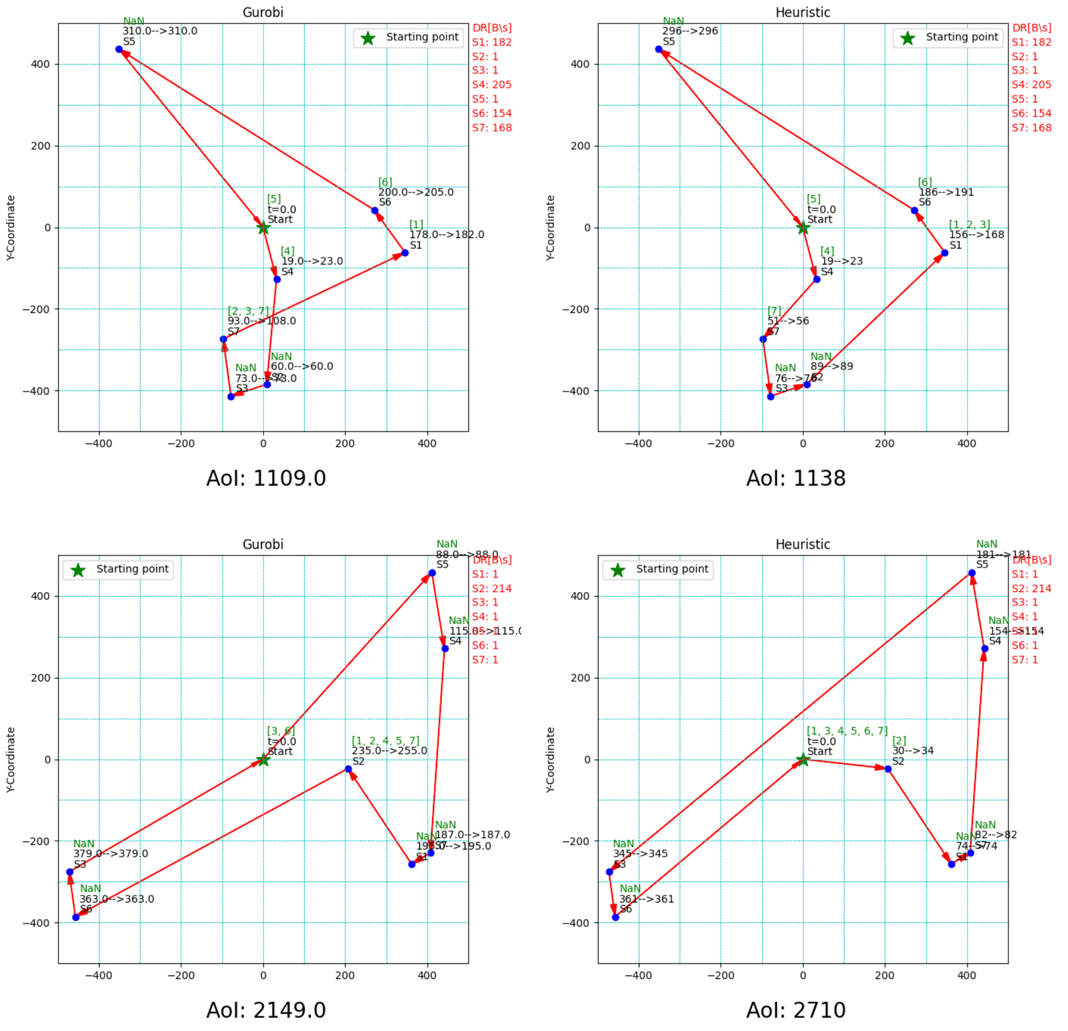


Figure 5.4: Comparison of joint optimization of path and transmission using the two implemented solutions in two different maps. Top: Case where the two algorithms almost coincide. Bottom: Case where the heuristic completely diverges respect to Gurobi.

In Figure 5.4, two cases are illustrated: the top one shows close results between the heuristic and Gurobi, while in the bottom case the heuristic solution is significantly worse than the optimal one.

In the first case, the computed path is almost the same except for the visit to three sensors. Since the three sensors are really close to one another, the metric that decides the path in the heuristic is the

Data Rate (S7 is visited earlier even if it isn't the optimal one because it has higher Data Rate). The transmission positions, however, are the optimal ones.

Looking at the second case, it is possible to notice that the path is completely different. After conducting several tests, it has been discovered that the heuristic algorithm performs worse when there are a lot of sensors with no connection, such as in the second case shown. In that situation, the algorithm tends to visit the few sensors with some coverage first, regardless of the distance, and then visits all the remaining sensors (without connection) at the end, planning to directly bring back data to the base station. This increase a lot the AoI of all the sensors which data are not transmitted.

5.1.4 Scalability of the solution

As previously discussed, although Gurobi guarantees the optimal result, the computational cost of finding it increases exponentially as the number of sensors increases. This phenomenon is illustrated in Figure 5.5, which analyzes the computational time required for different numbers of sensors.

From the data presented, it is evident that the Gurobi solution becomes impractical beyond eight sensors. Specifically, the computational time required for nine sensors reaches approximately 37,000 seconds (roughly 10 hours). For ten sensors, the estimated computational time soars to around 194 hours. It is important to note that these latter values were obtained by fitting the available data, extrapolating the observed trend to predict the computational cost for larger numbers of sensors, considering that the problem formulated is NP-hard

In stark contrast, the heuristic solution exhibits a markedly different performance profile. Although the computational time still grows exponentially, the rate of growth is significantly slower compared to Gurobi. For instance, the heuristic approach requires only 10 seconds to compute the solution when deploying 40 sensors. This substantial difference in computational efficiency underscores the heuristic method's practicality for scenarios involving a larger number of sensors.

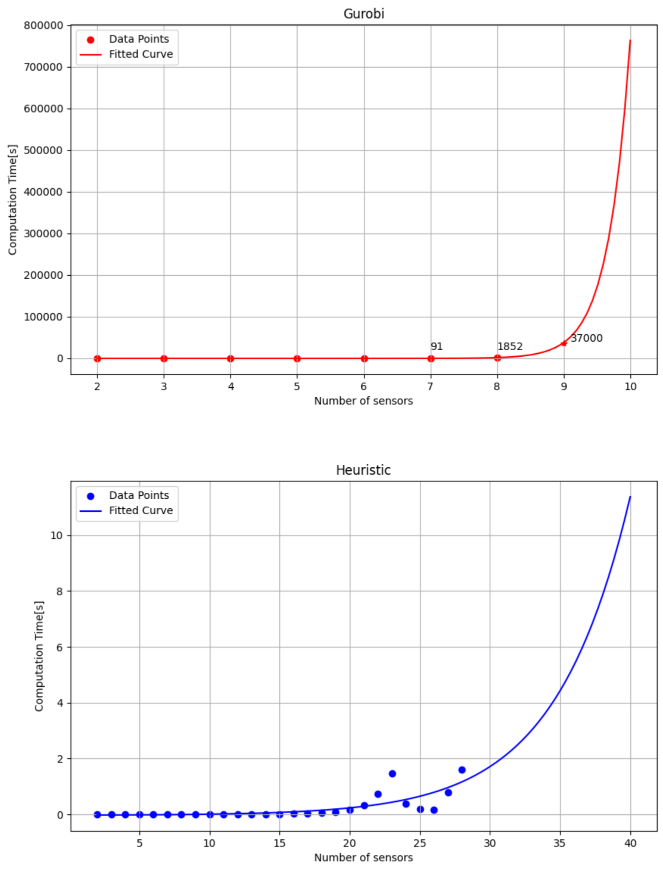


Figure 5.5: Comparison of Computational Time: Gurobi vs. Heuristic Methods

In summary, while Gurobi provides guaranteed optimal solutions, its scalability is severely limited by the exponential growth in computational time. On the other hand, the heuristic solution, despite also exhibiting exponential growth, maintains a much more manageable computational cost, making it a viable alternative for larger-scale sensor deployment scenarios.

5.2 UCB algorithm

The results previously presented are applicable under the assumption that the data rate is known and constant. Unfortunately, in real-world scenarios, the data rate is not constant but varies according to a certain distribution. Indeed, by examining the actual data obtained, it is evident that even at the same location, the data rate between successive transmissions can differ significantly.

Moreover, we are considering a scenario where the UAV is deployed in an area with an unknown data rate distribution, rendering traditional solution methods impractical.

To address this challenge, which generalizes the previous problem, we have decided to adopt the solution outlined in Section 3.3.3. This reinforcement learning approach is well-suited for problems of this nature, where it is necessary to explore and learn the distribution of the data rate at each sensor location.

It is important to highlight that for the next results we will use 20 sensors and a c=20.In fact, to appreciate the capabilities of this algorithm is fundamental to use a high number of sensors.

5.2.1 Data Distribution

To model the distribution of the data and simulate the learning process, we decided to use a Gaussian distribution. In the 4.3 part, it has been explained that for every grid cell, the mean value and the standard deviation have been computed. This values are used to compute the distribution. Two example are shown in 5.6

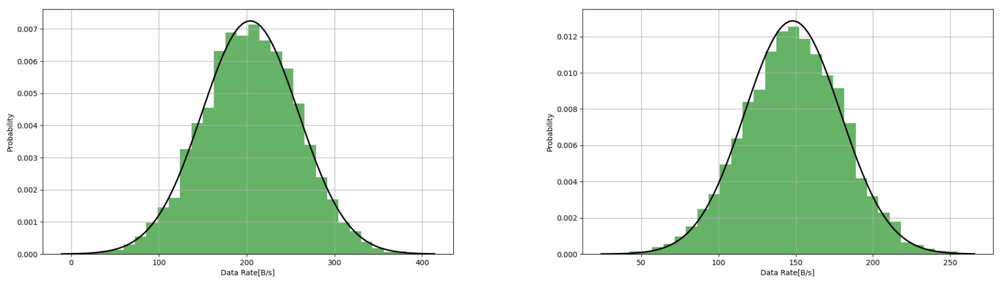


Figure 5.6: Example of the distribution of two different cells

5.2.2 Performances

The main goal of this Reinforcement Learning algorithm is to learn the distribution, specifically the mean value, in order to take the right decisions during optimization.

The value $\hat{b}$, which the UAV is learning, represents its current knowledge about the mean value of the data rate for a certain grid cell. In 5.7, the blue line represents $\hat{b}$, while the red line represents the expected mean value. It is evident that the algorithm is able to learn the correct mean value of the distribution. In green, the data rate values for each iterations are also represented. Since these values are highly spread out, using one of the previous algorithms that did not consider the mean value would have resulted in very poor performance.

After 200 iterations the $\hat{b}$ and the expected value coincide.

To better show the learning capabilities, Figure 5.8 displays the cumulative average error over all the sensors during the learning process. It is clear that the error decrease until it converge.

One of the last two metrics used to evaluate the performances of this algorithm is the regrets, define as:

$$Regret_i = AoI_{optimal} - AoI_i$$

For iteration i, the regret is computed as the difference between

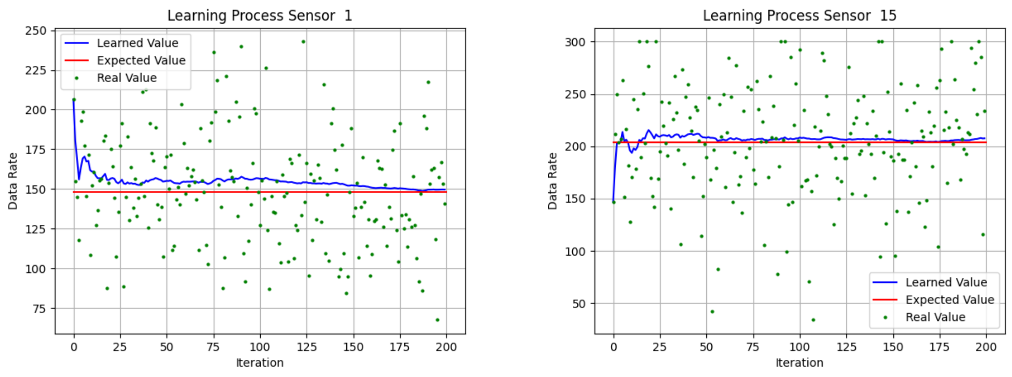


Figure 5.7: Learning Curve for two different sensors

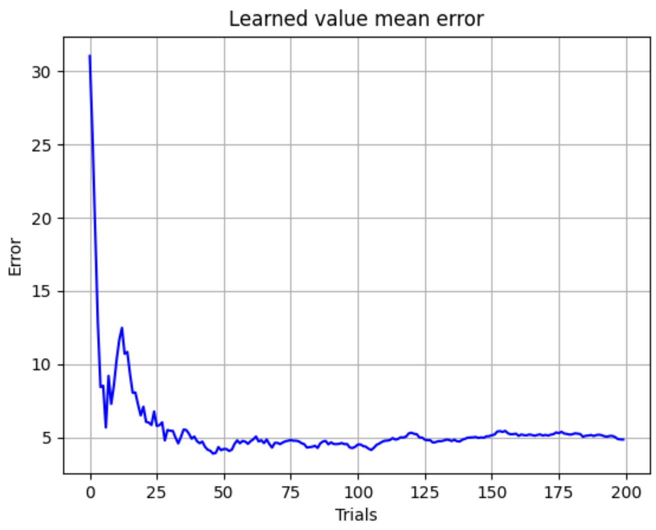


Figure 5.8: This graph shows the convergence of the cumulative average of the data rate learned value of all the sensors

AoI$_{\text{Optimal}}$, which is the AoI obtained using the real mean value of the data rate (DR), and the AoI obtained at that iteration with the random DR values. The cumulative average of the instant values of the regret then has been computed.

The other metric is the average of cumulative AoI.

By examining Figure 5.9, it is evident that both graphs converge to the correct value(Regret to 0 and the AoI to $AoI_{optimal}$). This

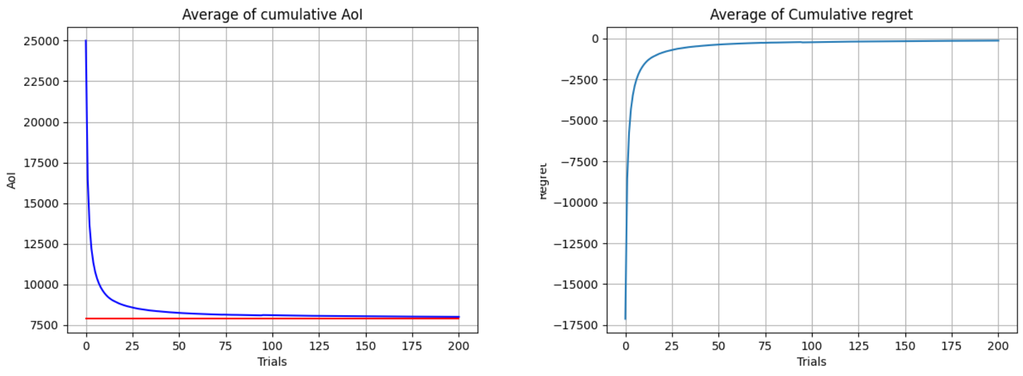
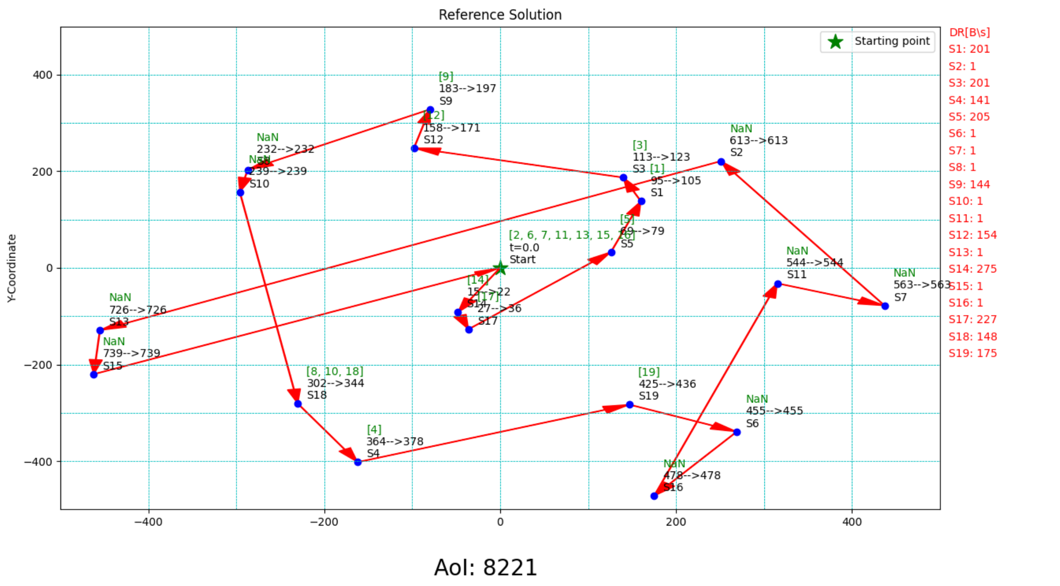


Figure 5.9: Average of cumulative AoI(Left) and Regret(Right)over 200 iterations

indicates that the algorithm is capable of correctly learning the distribution and compute the correct path.

The first value is really high because, as shown in Algorithm 2, in the first iteration, the drone attempts to transmit from every position to "explore" all the the data rate of the sensors. Even if there is no connection, the UAV waits for the end of the transmission, which obviously won't occur. After a time-out period, the drone will move in any case and the data will be brought back to the base station.
In fig. 5.10 it is possible to see that the algorithm propose works. Indeed at the last iteration, the path computed with the learned values is exactly the same respect to the one computed with the expected values. Also the transmission patterns match, showing a very good convergence between the two solutions. The AoI computed at the 200th iteration is a bit lower because in that iteration some sensors had the data rate higher than the mean value, resulting in a lower AoI.



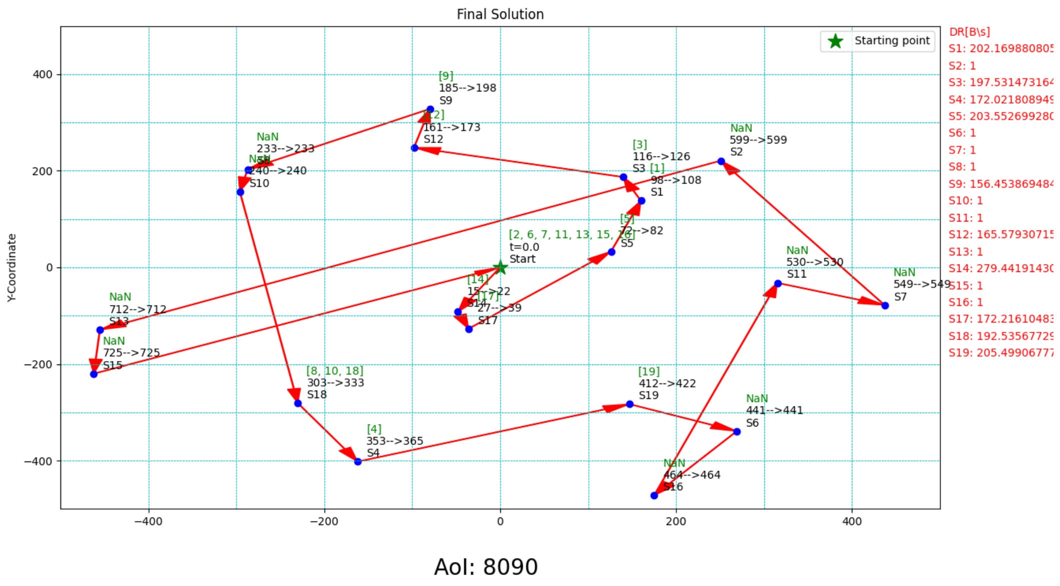


Figure 5.10: Optimization problem computed using the expected value(UP) and optimization problem computed with the learned value at the 200th tour(BOTTOM)

Chapter 6

Conclusion

In conclusion, this thesis presents an innovative UAV-aided data collection system using LoRa communication for base stations in urban environments. By integrating the optimization of transmission locations and the UAV's trajectory, the system significantly reduces the Age of Information (AoI), ensuring that the data collected is timely and relevant. The primary contributions of this work include the development of a heuristic algorithm that leverages reinforcement learning to adapt to the dynamic urban environment and optimize the UAV's path.

A lot of experiments and data analysis validate the performance of the proposed system. The heuristic algorithm demonstrates notable improvements in speed and scalability compared to traditional optimization methods, while the Upper Confidence Bound (UCB) algorithm excels in real-world scenarios where data rate distributions are unknown and variable. This robust performance across different conditions highlights the system's practical applicability and effectiveness in maintaining low AoI, thereby enhancing the quality and freshness of the collected data.

Future research aims to further deepen this solution by improving the performance of the heuristic algorithm and modifying the UCB to achieve better performance and this could involve utilize other reinforcement learning techniques. Exploring additional optimization techniques and considering new parameters may expand the system's capabilities and efficiency. Moreover, introducing multiple UAVs and enlarging the map could present new challenges but more possibilities.

Another crucial aspect to investigate is the introduction of battery limitations. Currently, it is assumed that the drone has sufficient charge to visit all sensors, which is feasible due to the low number of sensors and the small map size. Incorporating battery limitations introduces significant challenges, such as managing the Age of Information (AoI) for sensors that are not visited due to battery constraints.

Bibliography

[1] Shilpa Devalal and A. Karthikeyan. LoRa technology-an overview. In *International Conference on Electronics, Communication and Aerospace Technology*, pages 284–290. IEEE, 2018.

[2] Yingmeng Hu, Yan Liu, Aryan Kaushik, Christos Masouros, and John S. Thompson. Timely data collection for UAV-based IOT Networks: A deep reinforcement learning approach. *IEEE Sensors Journal*, 23(11):12295–12308, 2023.

[3] Mengjie Yi, Xijun Wang, Juan Liu, Yan Zhang, and Ronghui Hou. Deep reinforcement learning for energy-efficient fresh data collection in rechargeable UAV-assisted IoT networks. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2023.

[4] Mengying Sun, Xiaodong Xu, Xiaoqi Qin, and Ping Zhang. AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method. *IEEE Internet of Things Journal*, 8(24):17275–17289, 2021.

[5] Juan Liu, Peng Tong, Xijun Wang, Bo Bai, and Huaiyu Dai. UAV-aided data collection for information freshness in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 20(4):2368–2382, 2020.

[6] Aidin Ferdowsi, Mohamed A. Abd-Elmagid, Walid Saad, and Harpreet S. Dhillon. Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks. *IEEE Journal on Selected Areas in Communications*, 39(5):1250–1265, 2021.

[7] Mohamed A. Abd-Elmagid, Aidin Ferdowsi, Harpreet S. Dhillon, and Walid Saad. Deep Reinforcement Learning for Minimizing Age-of-Information in UAV-Assisted Networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019.

[8] Mengjie Yi, Xijun Wang, Juan Liu, Yan Zhang, and Bo Bai. Deep Reinforcement Learning for Fresh Data Collection in UAV-assisted IoT Networks. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020.

[9] Aidin Ferdowsi, Mohamed A. Abd-Elmagid, Walid Saad, and Harpreet S. Dhillon. Neural Combinatorial Deep Reinforcement Learning for Age-Optimal Joint Trajectory and Scheduling Design in UAV-Assisted Networks. *IEEE Journal on Selected Areas in Communications*, 2021.

[10] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? In *IEEE International Conference on Computer Communications (INFOCOM)*, 2012.

[11] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, 55(9):34–40, 2017.

[12] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors*, 16(9):1466, 2016.

[13] LoRa Alliance. LoRaWAN 1.0.4 Specification. Retrieved from `https://lora-alliance.org/resource_hub/lorawan-specification-104/`, 2020.

[14] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi. Long-Range Communications in Unlicensed Bands: The Rising Stars in the IoT and Smart City Scenarios. *IEEE Wireless Communications*, 23(5):60–67, 2016.

[15] B. Reynders and S. Pollin. Chirp Spread Spectrum as a Modulation Technique for Long Range Communication. In *Symposium on Communications and Vehicular Technologies (SCVT)*, pages 1–5, 2016.

[16] Federal Communications Commission (FCC). FCC Part 15 Rules. Retrieved from `https://www.fcc.gov/general/rules-regulations-title-47`, 2018.

[17] European Telecommunications Standards Institute (ETSI). EN 300 220-2 V3.2.1 – Short Range Devices (SRD). Retrieved from `https://www.etsi.org/deliver/etsi_en/300220_300239/30022002/03.02.01_60/en_30022002v030201p.pdf`, 2019.

[18] The Things Network. LoRaWAN Regional Parameters. Retrieved from `https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/`, 2021.

[19] S. Kaul, R. Yates, and M. Gruteser. Real-time status: How often should one update? In *IEEE INFOCOM*, 2012.

[20] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff. Update or wait: How to keep your data fresh. *IEEE Transactions on Information Theory*, 63(11):7492–7508, Nov. 2017.

[21] A. M. Bedewy, Y. Sun, and N. B. Shroff. Minimizing the age of information through queues. In *IEEE Annual Conference on Computer Communications (INFOCOM)*, 2016.

[22] Q. He, H. Zhang, and X. Liang. Age of information in healthcare monitoring systems. *IEEE Transactions on Information Technology in Biomedicine*, 21(5):1014–1023, Sept. 2017.

[23] C. Kam, S. Kompella, and A. Ephremides. Age of information under random updates. In *IEEE International Symposium on Information Theory (ISIT)*, 2016.

[24] R. D. Yates and S. K. Kaul. Real-time status updating: Multiple sources and multiple servers. *IEEE Transactions on Information Theory*, 65(3):1807–1827, Mar. 2012.

[25] Y. Sun and T. Başar. Energy-aware age of information management in wireless sensor networks. *IEEE Transactions on Network Science and Engineering*, 6(3):455–468, July 2019.

[26] P. Zou, Y. Chen, and H. H. Yang. AoI-aware communication for vehicular networks. *IEEE Transactions on Vehicular Technology*, 68(6):4911–4924, June 2019.

[27] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2-3, pp. 235-256, May 2002.

[28] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1-122, 2012.

[29] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommenda-

tion," *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[30] O. Chapelle and L. Li, "An Empirical Evaluation of Thompson Sampling," *Advances in Neural Information Processing Systems*, 2011.

[31] S. S. Villar, J. Bowden, and J. Wason, "Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges," *Statistical Science*, vol. 30, no. 2, pp. 199-215, May 2015.

[32] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial Network Optimization with Unknown Variables: Multi-Armed Bandits with Linear Rewards and Individual Observations," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1466-1478, Oct. 2012.

[33] Mozaffari, M., Saad, W., Bennis, M., & Debbah, M. (2017). Wireless communication using unmanned aerial vehicles (UAVs): Optimal transport theory for hover time optimization. *IEEE Transactions on Wireless Communications*, 16(12), 8052–8066.

[34] Zhan, C., Zeng, Y., & Zhang, R. (2018). Energy-efficient UAV trajectory design for data collection. *IEEE Wireless Communications Letters*, 7(3), 328–331.

[35] Maatouk, A., Assaad, M., Tabet, I. (2022). Minimizing Age of Information: A New Approach to Scheduling Problems. *IEEE Transactions on Communications*, 70(2), 896–909.

[36] Zhou, Y., Yu, W., Li, X., Yang, Y. (2021). UAV-Enabled Age-Minimal Trajectory Planning for Data Collection Systems. *IEEE Internet of Things Journal*, 8(6), 4316–4329.