Rodrigo Zuolo Carvalho

Mobility Planning in Edge Assisted Low Power Wide Area Networks

UAlg
UNIVERSIDADE DO ALGARVE

Rodrigo Zuolo Carvalho

Mobility Planning in Edge Assisted Low Power Wide Area Networks

Ph.D. Thesis in Computer Science

Work done under the supervision of:
Profª Doutora Noélia Susana Costa Correia
Doutor Faroq Al-Tam

2024

Statement of Originality

Mobility Planning in Edge Assisted Low Power Wide Area Networks

Statement of authorship: The work presented in this thesis is, to the best of my knowledge and belief, original, except as acknowledged in the text. The material has not been submitted, either in whole or in part, for a degree at this or any other university.

Candidate:

_____

(Rodrigo Zuolo Carvalho)

NETWORKING

Work done at Research Center of Electronics Optoelectronics and Telecommunications (CEOT)

Acknowledgements

I want to express my gratitude to my supervisors, profª Drª Noélia Correia and Dr. Faroq Al-Tam, who provided their invaluable knowledge and guided me through the inherent academic challenges of this work. I also recognize my PhD colleagues' crucial support for this project's success. I could not have enough words to thank my family: my sister Renata; my parents, Roberto and Regina; and my beloved wife, Ângela, who never ceased supporting me, even in the harshest moments. Lastly, I dedicate this to my little ones, Romeu and Rael, who inspire and reinvigorate my power of will at every rising sun.

This work is supported by Fundação para a ciência e Tecnologia within CEOT (Center for Electronic, Optoelectronic and Telecommunications) and the UID/MULTI/00631/2020 project.

Abstract

Edge computing infrastructures are being integrated with Internet of Things (IoT) systems to facilitate time-critical applications. These systems often require data to be processed within a specific time window, so the edge becomes vital in developing reactive IoT applications with time restriction requirements. Although different architectural designs will always have advantages and disadvantages, mobile gateways are particularly relevant in enabling this integration with the edge, particularly in the context of wide-area networks with occasional data generation. In these scenarios, mobility planning is necessary, as aspects of the technology need to be aligned with the temporal needs of an application. This dissertation intersects machine learning techniques and mathematical models to establish a framework that solves the problem of mobility planning for LoRaWAN gateways when cooperating with edge system architectures. Throughout the pipeline for attaining this objective, some sideline contributions are yielded, such as machine learning agents to improve the Adaptive Data Rate (ADR) mechanism, mathematical models to estimate the gateways' journey time, and machine learning agents that meet the constraints on valid data collection and delivery to edge systems. The nature of the problem at hand makes cutting-edge Deep Reinforcement Learning (DRL) techniques helpful in solving inherited issues, such as dealing with multiple dimensions in the action space while aiming for optimum system performance. This dissertation culminates in a novel scalable DRL model incorporating a pointer network (Ptr-Net) and an Actor-Critic (AC) algorithm to handle complex action spaces. The model synchronously determines the gateway location and visit time. Ultimately, the gateways can achieve a trajectory planning fulfilling all requirements while reducing latency and energy waste.

Keywords: Action branching, Internet-of-Things, edge systems, long-range wide-area network, LoRaWAN, mobility, pointer networks, reinforcement learning.

Resumo

As infraestruturas de computação de borda começam a ser integradas nos sistemas de Internet das Coisas (IoT) para facilitar aplicações de baixa latência. Esses sistemas geralmente requerem que os dados sejam processados numa janela limitada de tempo, de modo que a computação de borda se torna vital no desenvolvimento e integração dessas aplicações de tempo real. Embora diferentes projetos arquitetónicos tenham diferentes vantagens e desvantagens, os gateways móveis são particularmente relevantes para permitir esta integração com a borda, particularmente no contexto de redes de longa distância com geração ocasional de dados. Nestes cenários, o planeamento da mobilidade é necessário pois garantirá que todos os requisitos estejam alinhados com as necessidades temporais da aplicação. Esta tese interliga técnicas de aprendizagem de máquina e modelos matemáticos para estabelecer uma *framework* que resolva o problema de planeamento de mobilidade para gateways LoRaWAN que cooperam com arquiteturas de sistemas de borda. Ao longo das etapas de trabalho para atingir esse objetivo, algumas contribuições adicionais são geradas, como agentes de aprendizagem de máquina para melhorar o mecanismo do *Adaptive Data Rate* (ADR), modelos matemáticos para estimar o tempo de viagem das *gateways* móveis e agentes de aprendizagem de máquina que atendem às restrições da validade dos dados que são coletados nos sensores e entregues aos sistemas de borda. A natureza do problema em questão faz com que técnicas de *Deep Reinforcement Learning* (DRL) sejam úteis na resolução dos problemas inerentes ao cenário dado, como por exemplo lidar com múltiplas dimensões no espaço de ação enquanto se tenta manter o desempenho ideal desse mesmo sistema. Esta tese culmina num novo modelo DRL escalável que incorpora uma *Pointer Netowrk* (Ptr-Net) e um algoritmo *Actor-Critic* (AC) para lidar com espaços de ação complexos. O modelo determina de forma síncrona, a localização da *gateway* e o tempo utilizado na visita aos nós e sistemas de borda. Em última análise, as *gateways* podem alcançar um planeamento de trajetória que satisfaça os requisitos do problema, enquanto reduzem a latência e consumo de energia.

Termos chave: Action branching, Internet das Coisas, sistemas de borda, long-range wide-area network, LoRaWAN, mobilidade, pointer networks, aprendizagem por reforço.

Contents

Statement of Originality i

Acknowledgements iv

Abstract v

Resumo vi

Nomenclature xii

List of Publications xiv

1 Introduction 1

1.1 The Era of the Internet of Things 1

1.2 The Need for Edge-assisted Large-scale IoT Systems Management . . . 1

1.2.1 LoRaWAN Networks in the IoT Context 2

1.2.2 The Edge Infrastructure Strategy 2

1.2.3 Mathematical Optimization and Reinforcement Learning Tools . 3

1.3 Focus of the Research Conducted in this Thesis 4

1.4 Thesis Outline 5

2 Edge-assisted Large-scale LoRaWAN 6

2.1 LoRaWAN: Protocol and Architecture 6

2.2 Edge-assisted LoRaWAN Systems 8

2.3 LoRaWAN and Gateway Mobility 9

2.4 Literature Review 9

2.4.1 LoRaWAN ADR and Transmission Parameters Optimization . . 10

2.4.2 LoRaWAN UAV and Mobile Gateways 11

2.4.3 LoRaWAN and Edge Systems Integration 12

2.4.4 DRL-Based Solutions 12

2.5 Adopted Research Framework 13

3 ADR Mechanism Improvement **16**
3.1 Introduction . . . 16
3.2 The RL-ADR proposal . . . 17
3.2.1 Q-learning ϵ-greedy . . . 17
3.2.2 State Space . . . 19
3.2.3 Action Space . . . 19
3.2.4 Reward function . . . 19
3.2.5 Implementation and Deployment . . . 20
3.3 Performance Evaluation . . . 20
3.3.1 Simulation Setup . . . 20
3.3.2 Analysis of Results . . . 21
3.3.3 Summary of Results . . . 24
3.4 RL-ADR Conclusions . . . 25

4 GMwES Agent as a Proof of Concept **26**
4.1 Introduction . . . 26
4.2 A Mixed Integer Linear Programming Approach . . . 26
4.2.1 LoRaWAN Premisses . . . 26
4.2.2 Problem Definition . . . 27
4.2.3 Mathematical Formulation . . . 29
4.3 The Reinforcement Learning Approach . . . 34
4.3.1 Background and Assumptions . . . 35
4.3.2 GMwES Agent Design . . . 35
4.3.3 Implementation Considerations . . . 36
4.4 Performance Evaluation . . . 38
4.4.1 Framework Pipeline Setup . . . 38
4.4.2 Analysis of Results . . . 39
4.4.3 Summary of Results . . . 40
4.5 GMwES Conclusions . . . 43

5 A2PC Agent **45**
5.1 Introduction . . . 45
5.1.1 A Mixed Integer Linear Programming Upper Bound . . . 46
5.1.2 The Augmented Advantage Pointer Critic Model (A2PC) . . . 51
5.1.3 A2PC Architecture . . . 54
5.1.4 The Action-Branching (Augmented) Model . . . 55
5.1.5 Loss Functions . . . 56
5.1.6 A2PC Training . . . 57
5.2 Performance Evaluation . . . 57

5.2.1 Scenarios Setup 59
5.2.2 Analysis of Results 61
5.2.3 Summary of Results 62
5.3 A2PC Conclusions 65

6 Comparison of A2PC and GMwES Agents **67**
6.1 Introduction 67
6.2 Assessment of the Agents 67
6.2.1 Scenarios Setup 68
6.2.2 Analysis of Results 68
6.2.3 Summary of Results 68
6.3 A2PC and GMwES Assessment Conclusions 70

7 Conclusions **71**
7.1 Framework Results 71
7.2 Future Work 72

List of Tables

2.1 Spreading factors and data rates for the European region. 7

3.1 Hyperparamaters for the proposed ADR agent. 21
3.2 ADR agent experiment scenarios. 21
3.3 ADR agent: goodput and energy results. 24

4.1 Hyperparameters of the GMWES agent. 37
4.2 GMwES deployment scenarios. 38
4.3 GMwES simulation setup information. 39
4.4 GMwES agent average gateway final arrival time (τ^{MAX}). 41
4.5 GMwES agent average throughput (episode duration = τ^{MAX}). 42
4.6 GMwES agent average throughput (episode duration = τ^{MAX} plus margin). 43

5.1 A2PC simulation setup information. 59
5.2 A2PC agent hyperparameters. 62
5.3 A2PC total data processed. 63
5.4 A2PC agent average data processed. 64

6.1 Average performance improvement from A2PC in relation to GMwES. . 69

Nomenclature

Abbreviations

AC	Actor-Critic
A2C	Advantage Actor-Critic
ADR	Adaptive Data Rate
ARM	Advanced RISC Machines
ADTW	Arrival-Departure Time Window
AI	Artificial Intelligence
A2PC	Augmented Advantage Pointer Critic
BW	Bandwidth
BDQ	Branching Dueling Q-Network
CSS	Chirp Spread Spectrum
CR	Code Rate
CDF	Cumulative Distribution Function
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DNS	Domain Name System
GMwES	Gateway Mobility with Edge Storage
GPS	Global Positioning System
IETF	Internet Engineering Task Force
IoT	Internet of Things
KPI	Key Performance Indicator
LoRaWAN	Long Range Wide Area Networks
LSTM	Long Short-Term Memory
LPWAN	Low Power Wide Area Networks
ML	Machine Learning
MDP	Markov Decision Processes
MIT	Massachusetts Institute of Technology
MIPOPT	Mathematical Optimization Model
MAC	Medium Access Control
MILP	Mixed Interger Linear Programming
PL	Payload Size
PRR	Packet Reception Ratio
Ptr-Net	Pointer Network

PoC	Proof of Concept
RL	Reinforcement Learning
RFC	Request for Comments
RFID	Radio Frequency Identification
SNR	Signal to Noise Ratio
SDN	Software Defined Networks
SF	Spreading Factor
SCHC	Static Context Header Compression
TD	Temporal Difference
TDC	Time Duty Cycle
ToA	Time on Air
TTL	Time to live
Tx	Transmit Power
UAV	Unmanned Aerial Vehicles

List of Publications

- Rodrigo Carvalho, Faroq AL-Tam and Noélia Correia. "Q-Learning ADR Agent for LoRaWAN Optimization", IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2021.
- Rodrigo Carvalho, Noélia Correia, and Faroq Al-Tam. "Mobility planning of LoRa gateways for edge storage of IoT data", Computer Networks, Vol. 221, 2023.
- Rodrigo Carvalho, Faroq AL-Tam and Noélia Correia. "Mobility Planning of Edge-Assisted IoT Systems Using DRL Action-Branching Approach", IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), 2023.
- Rodrigo Carvalho, Faroq Al-Tam, and Noélia Correia. "A2PC: Augmented Advantage Pointer-Critic Model for Low Latency on Mobile IoT with Edge Computing", IEEE Transactions on Machine Learning in Communications and Networking, Vol. 3, 2024.

CHAPTER 1

Introduction

1.1 The Era of the Internet of Things

The inter-communication of machines via Internet, and automation of all kinds are the gist of the Internet of Things (IoT). The term was first coined by Kevin Ashton, Massachusetts Institute of Technology (MIT) Executive Director of Auto-ID Labs, in 1999 [1]. At that moment, these systems focused on vending machines and Radio Frequency Identification (RFID) technology for controlling inventory and operations [2]. Since then, IoT has consolidated and propagated into other industries and fields. The most groundbreaking change occurred when IoT started enabling smarter environments [3], such as in applications for smart buildings, parking control, traffic control, environmental monitoring, and so on. IoT has permeated almost every industry that relies on automation and precision technology, ranging from the use of smart sensors in the manufacturing industry to healthcare. Lately, IoT has become a seamless part of human beings' lives, and the use of mobile IoT is a natural path that has been growing for more than five years already [4].

1.2 The Need for Edge-assisted Large-scale IoT Systems Management

Edge-assisted IoT is a strategy employed that yields various benefits by leveraging the ability to process data locally or store them near the source [5]. The composition of IoT devices with edge devices is a powerful solution to boost real-time system capabilities by simply keeping these two components physically closer. While this technique offers several benefits, it has not been thoroughly investigated in all families of IoT technologies. This is especially true for the Long Range Wide Area Network (LoRaWAN); an IoT stack still in the early stages of research and development regarding large-scale edge-assisted implementation [6].

In summary, this work revolves around implementing Machine Learning (ML) techniques to solve the trajectory planning of mobile LoRaWAN gateways operating in articulation with edge systems. This introduction briefly overviews the individual areas

within the broad scope of technical fields involved in this intersectional research. These backgrounds are presented below.

1.2.1 LoRaWAN Networks in the IoT Context

LoRaWAN is a protocol stack and is often referred to as a type of a Low Power Wide Area Network (LPWAN) that has garnered significant attention in recent years due to its widespread usage in the IoT field [7]. The architecture of the LoRaWAN protocol is widely utilized in numerous businesses and exhibits full integration with major cloud providers. Endpoint devices encompass a wide range of hardware, spanning from small micro-controllers to more comprehensive Arduino or Raspberry Advanced RISC Machines (ARM) devices. One of the outstanding characteristics of the LoRaWAN ecosystem is its capability for long-range communication, reaching distances of up to 16 kilometers when there is an unobstructed line of sight. Additionally, the battery features an extended lifespan of approximately 10 years. In order to enhance battery efficiency, one can employ either class A or class B mode for operations of devices, albeit resulting in down-link increased latency [7]. In the context of radio spectrum utilization, there are no mandatory licensing requirements or permissions. Nevertheless, it is crucial to acknowledge that compliance with specific regulations applicable to a given geographical area remains imperative. The system holds a comparatively low power output accompanied by a limited payload capacity that varies between 51 and 230 bytes, and data rate spanning from 0.3 Kbit/s to 11 Kbit/s. Another thing which is important to make reference is the Adaptive Data Rate (ADR): A mechanism that autonomously manages the tradeoff between energy consumption and data communication efficiency [8]. Such a mechanism enables a better adaptation of the transmission settings according to channel's communication conditions. Finally, the devices within this ecosystem are commonly recognized for their cost-effectiveness.

1.2.2 The Edge Infrastructure Strategy

Also known as edge computing, this approach emerges as a natural extension of cloud systems, and results in a distributed architecture known as the "edge" paradigm. This strategic placement of processing and storage units in proximity to the source of data can take advantage of caching and preprocessing tasks, as discussed in previous works [10]. The resultant system offloads the cloud system and the main upstream channel, facilitating the autonomy of the end node subsystem as well as reducing overall latency. Among the benefits, IoT edge-assisted solutions can [6]:

- Deliver better user experience by supporting more latency-sensitive applications;
- Enable faster system responses to local events;

- Identify localized patterns and enable appropriate reactions;
- Improve the resource distribution of complex ecosystems;
- Enable the scale of typical centralized infrastructure;
- Enhance cloud services in the IoT ends;
- Meet the increased demand for the ever-growing volume of data and devices in these distributed systems.

Because of all the above points, the IoT edge-assisted approach has become prevalent in different areas and is now crucial to autonomous vehicles and industrial manufacturing. In this work, the use of IoT and edge systems is focused on monitoring in vast landscape scenarios (e.g., agricultural), where mobility solutions can be a requirement to better meet the above advantages.

1.2.3 Mathematical Optimization and Reinforcement Learning Tools

Closer to a more theoretical realm, there is the mathematical optimization. This is a discipline that focuses on the resolution of problems by following an objective function that represents a desired target. This is typically done while considering predetermined constraints and conditions. The subject encompasses a wide range of applications, with its primary utilization observed in the domains of computer science and economics. Mathematical optimization is a valuable tool in the context of metrics or Key Performance Indicators (KPIs) [9]. The provided solutions to complex problems, by maximizing or minimizing objective functions, offer quantitative benchmarks that can be used to determine if results align with the anticipated objectives. A common used type of mathematical optimization is the Mixed Integer Linear Programming (MILP), which employs sets of linear equations with real variables, supported by additional variables and expressions that can result into boolean or integer values. Despite resolving numerous problems, models may become unfeasible at large scales as in most cases they hold NP-hard computational complexity.

Lastly, the field of ML, which is a branch of Artificial Intelligence (AI), has become prevalent across various scientific domains [11]. The machine reinforcement learning, or simply Reinforcement Learning (RL) is a ML model that learns its best values on a trial and error approach, as the sequence of its interactions evolve to reinforced improved results towards a goal. The RL models have demonstrated significant efficacy in providing solutions that are either optimal or very close to optimal for problems that were previously regarded as highly challenging to solve [12]. In light of its swift progression, ML with RL are unveiling a plethora of techniques that can facilitate the discovery of solutions to problems of configuration, management, and planning. Within the domain of computer networks, such issues are commonly characterized by a significant computational burden and occasionally necessitate prompt responses as they are subject to time constraints [13].

Taking all that into consideration, the present study focuses on the utilization of advanced and cutting-edge RL methodologies to address the challenge of interconnecting mobile LoRaWAN gateways with edge systems via an adequate trajectory planning. Under this circumstance, mathematical optimization aids and guides the ML agent's evaluation, by providing upper-bound values for assessment.

1.3 Focus of the Research Conducted in this Thesis

Sectors requiring monitoring, such as agriculture, are known to benefit significantly from the IoT and LoRaWAN technological advancements [14, 17]. The effective implementation of precision technology and the timely adaptation of relevant processes impact the business's operational efficiency. Consequently, it is anticipated that a growing number of constrained sensing devices and networks will demand remote and autonomous operation in the years to come. However, the success of this trend will be determined by the presence of practical and sustainable techniques. In this regard, deploying mobile gateways has proven valuable in contexts such as areas with limited connectivity or regions with sporadic data generation.

Furthermore, the growing demand for efficient data processing in remote area systems has resulted in the development of solutions that combine mobile gateways [15, 16] with edge storage technologies. This is due to the inherent constraints associated with these systems. Per this, edge computing mitigates latency and offloads work in cloud environments, enabling enhanced data management within the specified response times. To ensure the successful operation of this architecture, it is crucial to establish effective synchronization between the gateway and the storage system. On top of that, to optimize the timings and achieve the desired data liveliness, it is essential to use a common point-of-view when scheduling trajectory movements, collection, and delivery. The cohesion of this particular aspect is of utmost importance within the context of mobility planning of gateways.

In addition to what has just been mentioned, using mobile LoRaWAN gateways with edge support is currently not fully explored, suggesting the possibility of further improvements in LoRaWAN applications. This condition catalyzes the development of innovative solutions in mobile LoRaWANs, aided by edge systems.

In summary, the integration of edge systems with LoRaWAN has received limited attention in academic literature, and this observation also applies to research focused on mobile LoRaWAN gateways. Addressing these concerns encounters inevitable setbacks due to the characteristics of the LoRAWAN architecture, and most studies about mobile LoRaWAN primarily concentrate on the movement of end nodes [18]. The recent surge in the publishing of academic papers focusing on the use of ML techniques in the fields of systems management and network control [19] suggests them as potential candidates

for tackling challenges such as trajectory planning and LoRaWAN configuration management. These techniques have demonstrated efficacy in various scenarios that previously demanded high-cost computing and involved increasingly complex solutions.

With the above in mind, this work focuses on the resolution of the trajectory planning for LoRaWAN gateway assisted by the edge system. The intricate problem involves respecting temporal dimension limits and arranging movements that consider edge system positions. As a result, this work proposes an articulation of ML techniques that culminate in an Actor-Critic (AC) agent capable of handling multiple decision domains. This particular agent aims to enhance the efficiency of data collection at the nodes and delivery at the edge by ascertaining the optimal trajectory for gateways, along with the corresponding visit times and data freshness limits. The scenario involves arranging multiple devices into cluster regions, where the center of clusters represents the vertices of a complete graph. Each cluster consists of, either a collection of LoRaWAN devices transmiting data, or storage edge systems that collect data. The gateway is relocated to the central region of a cluster and remains in that location for a predetermined period. During that time, the gateway is actively collecting or transmitting data.

1.4 Thesis Outline

The thesis document is organized in the following manner: The second chapter clarifies the problem specification and a provides a comprehensive discussion of the relevant literature. The primary focus of investigation in the third chapter points to the RL-ADR proposed agent. That section provides explanations for the preliminary conclusions in the configuration improvement methods that serve as a base for the simulations that mimic real world scenarios. The fourth chapter describes and designs a Proof-of-Concept (PoC) Deep Reinforcement Learning (DRL) agent for trajectory planning. That also includes detailing a mathematical optimization model that is employed to parametrize an agent that yields the first findings of the ML applicability to the problem. Furthermore, the ultimate DRL agent is introduced in the fifth chapter, and its experiments are validated using another mathematical optimization model. In a similar vein, the subsequent chapter 6 brings both ML agents together for a performance comparison. Chapter 7 concludes the entire evaluation. That section provides a final interpretation of all obtained results and focuses on delivering closing remarks. This study's final chapter also includes a discussion of potential future research directions.

CHAPTER 2

Edge-assisted Large-scale LoRaWAN

This work focuses on implementing LoRaWAN systems on a large scale in remote and vast areas. The objective is to provide a strategy that capitalizes on the advantages of edge systems and gateway mobility to enhance the efficiency of resource allocation and data processing. Consequently, the fundamental LoRaWAN concepts are succinctly outlined in separate sections below.

2.1 LoRaWAN: Protocol and Architecture

LoRa is a wireless modulation technique, developed from the Chirp Spread Spectrum (CSS) technology that encodes radio waves by mapping them into chirp pulses. This ensures a robust signal, even when transmitted through long distances. On top of such LoRa modulation there is a Medium Access Control (MAC) layer protocol implementation known as LoRaWAN. Sometimes, the LoRaWAN term is merely used to refer to the network solution stacks that employ LoRaWAN. Ultimately, the protocol and solutions are officially developed and maintained by the LoRa Alliance; an open and nonprofit association [20].

The architecture of LoRaWAN networks is typically composed of a star of star networks, and the topology consists of end nodes, gateway, network, and application servers. The nodes communicate with the concentrator or gateway using messages that are described in the LoRaWAN specification [20]. The gateway streamlines communication with the network server, which validates, authorizes, and forwards the data to the final application server. In this final stage, the data is processed and consumed by the end users. A simple diagram in Figure 2.1 depicts this baseline topology.

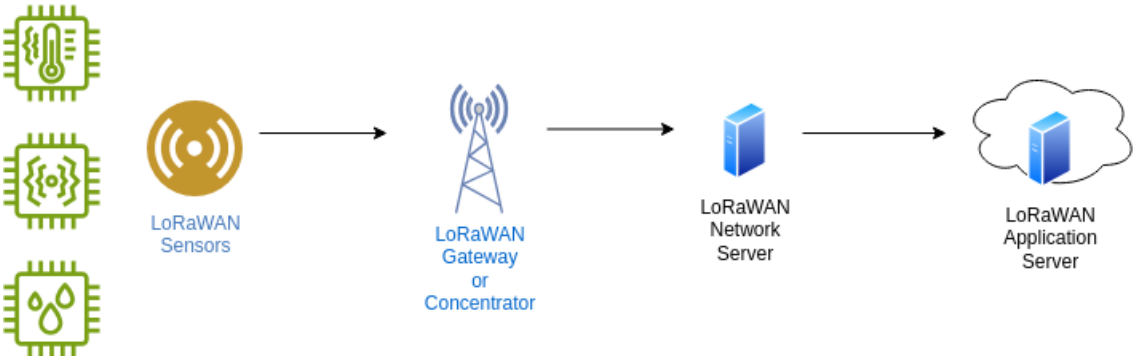

Figure 2.1: High level scheme LoRaWAN network.

Regarding regulations and spectrum usage, LoRaWAN networks operate under the unlicensed spectrum. LoRa Alliance has specified bands for the different regions of the world, so the local frequency availability does not impose a problem. One important characteristic that LoRaWAN network devices must follow is the maximum duty cycle. This parameter is often regulated by governments and tends to be 1% of the total time available of the allocated channel.

Given that LoRa utilizes CSS, the Spreading Factor (SF) mechanism is also employed to control the transmission speed and emission power. By tweaking the SF, the LoRaWAN device may increase the data rate or quality of the signal. This trade-off usually concerns distance, battery, bandwidth, and amount of data transmitted. The lower the SF, the higher the data rate with less range. On the other hand, the higher the SF, the lower the data rate, with more Time-on-Air (ToA) and more range. For the European region, the specification determines 7 different data rate levels, and they can be visualized in Table 2.1.

Data Rate (bit/s)	Configuration	Maximum Payload (bytes)
250	SF12/125kHz	51
440	SF11/125kHz	51
980	SF10/125kHz	51
1760	SF9/125kHz	115
3125	SF8/125kHz	222
5470	SF7/125kHz	222
11000	SF7/250kHz	222

Table 2.1: Spreading factors and data rates for the European region.

Because SF is an effective mechanism to control the data rates and transmission quality, it is also at the core of the ADR specification [22]. With such an automatic SF configuration in place, communications between LoRaWAN radios can self-regulate and stabilize their communications parameters to optimal values according to the moment conditions.

For all the characteristics described above, LoRaWAN networks are suitable for sensor networks that fall into the long-range, low-cost, and low-power use cases. Conversely, real-time applications are not strong candidates of use case. Nevertheless, hybrid architectures can help deliver less latency and potentially enable some subset of real-time applications. This is possible through combination of techniques and strategies that are presented in the coming sections.

2.2 Edge-assisted LoRaWAN Systems

Edge-assisted architecture has become valuable and crucial to integrating distributed systems, especially in the IoT domains. This is supported by the finding in the survey [21] that has explained how LoRaWAN systems can cooperate smoothly with edge architectures. Edge-assisted architecture not only improves indicators of LoRaWAN networks but also allows the implementation at scale, given that it can be strategically placed to fulfill local requirements or boost capacity in specific segments of the network. An illustration of the edge-assisted architecture is depicted in Figure 2.2.

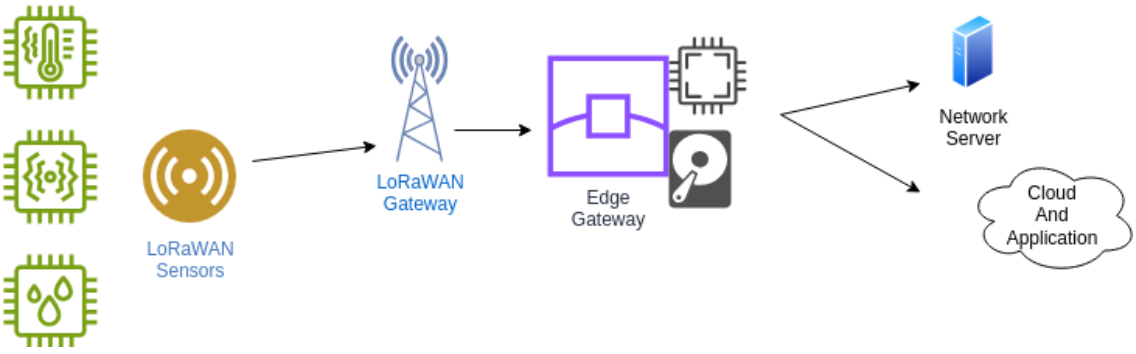

Figure 2.2: Edge-assisted LoRaWAN network diagram.

The typical edge-assisted architecture uses edge gateways positioned near the sensors or LoRaWAN nodes to perform preparation tasks such as compression, encryption, pre-

processing, and analysis. Those preliminary actions offload the cloud and save bandwidth, enabling more focused and lower-latency LoRaWAN applications. Examples mentioned in [21] highlight the extensive use of edge gateways in environment monitoring, animal tracking, farming, and the typical smart city scenario. Similarly, the scope of this thesis may also apply to farming use cases. However, the goal is to leverage novel gateways and edge systems approaches to deliver further improvements.

2.3 LoRaWAN and Gateway Mobility

The use of mobile gateways in low-power and long-range communication scenarios, like LoRaWAN, has proven to be a cost-efficient and smart alternative to the inefficient method of deploying stationary gateways to cover large areas [16]. The static-oriented approach requires the examination of multiple factors at different stages, including the positioning of gateways, assessment of signal overlap, battery consumption, and other relevant considerations [14].

The concerns above can be effectively addressed by enabling gateways to relocate, considering both environmental factors and their performance statuses. The architecture shown in Figure 2.3 turns out to be more practical and cost-effective by leveraging a strategic arrangement of the edge infrastructure. Overall, this enables expedited data storage and processing. Therefore, the gateway trajectory is the critical piece of design that can deliver efficiency compared to the static-oriented approach. Additionally, this approach offers the advantage of adaptability in response to changing conditions, rendering it applicable in various scenarios.

2.4 Literature Review

The interest in mobile LoRaWAN is continuously building up as researchers strive to find solutions for the inherent limitations associated with this architecture. Certain advancements in technology are derived from edge systems paradigms, which enable more effective identification of service locations and improved distribution of workloads. Understanding these subjects, alongside the LoRa ADR and the utilization of ML in the context of network management, holds significant importance within the scope of this thesis. Therefore, the subsequent subsections serve to highlight pertinent literature that is related to all these topics.

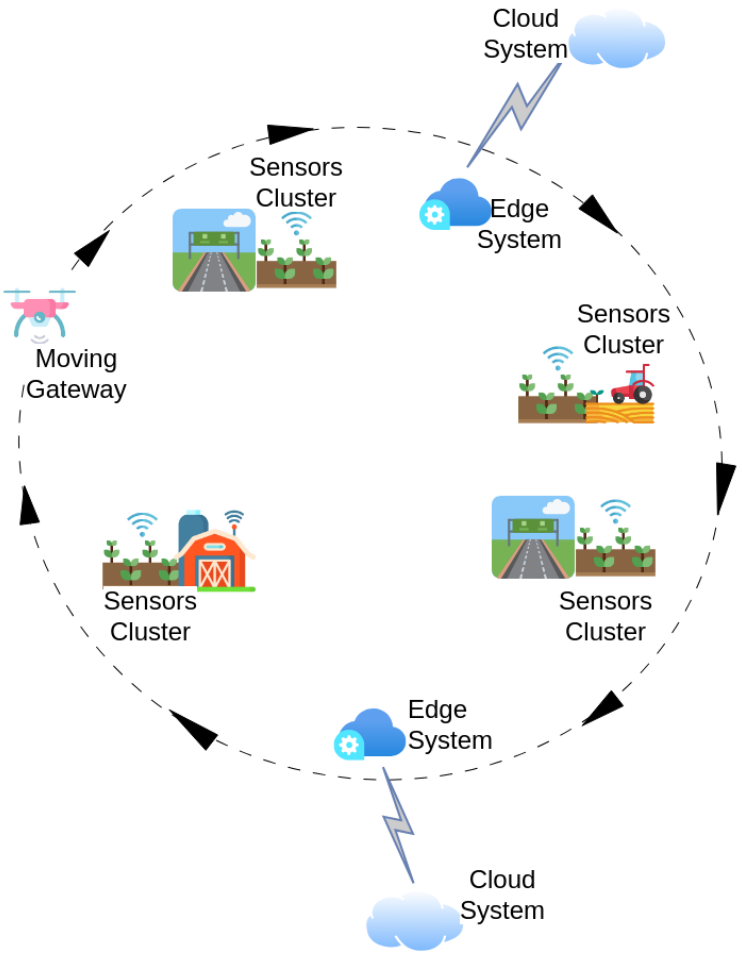

Figure 2.3: Illustration of the envisaged edge-supported architecture.

2.4.1 LoRaWAN ADR and Transmission Parameters Optimization

When configuring the transmission parameters in a LoRaWAN, the arrangements of the SF and Code Rate (CR) may heavily impact the performance indicators in the long run [24]. This happens due to a direct influence caused on ToA, which implies the number of collisions observed [23]. Lastly, another immediate effect is the power depletion speed per node. The latter is impacted by the receiving windows varying from the SF utilized.

Currently, many approaches [25], [26] focus on tweaking and controlling the ADR embedded in the MAC level. Its function enables gateways and end nodes to negotiate the best CR and transmission power over time. This mechanism is well suited for those environments where end nodes are static. If these nodes start to move around, they may eventually link to different gateways. In the event this transition occurs, the environment is considered to be non-stationary. Such a scenario brings additional complexity, given that nodes are roaming and messaging deduplication is required. Therefore, most ADR studies rely on stationary environments [25].

More recently, a couple of initiatives have started to address the LoRAWAN efficiency in non-stationary environments, such as the work in [27], which tracks the status of vehi-

cles in real-time. Nevertheless, there is still a long way to go before consolidating these experiments into practical architectures.

A survey in [8] has shed light on how ADR transmission settings can be improved, with a particular focus on ML. On the other hand, different surveys, like the one in [28], also points out the challenges and an increasing need for advancements in the area of ML and DRL applied to LPWAN management.

2.4.2 LoRaWAN UAV and Mobile Gateways

The utilization of mobile LoRaWAN gateway devices has been suggested to improve agricultural and livestock practices, specifically in scenarios where data collection or productivity enhancement is required [16, 29]. The findings of these studies demonstrate that using a sole mobile gateway can yield greater cost efficiency compared to deploying multiple static gateways. The primary subject of investigation in [30] pertains to planning mobility for a single gateway, and the authors propose using a genetic algorithm to ascertain the optimal settings of the nodes and the most suitable trajectory of the gateway. The findings indicate that the adopted methodology has a favorable influence on both energy consumption and data collection. In [31], the authors address the issue of mobility planning for multiple gateways. They approach the problem by initially formulating it as a traveling salesman problem, and subsequently employ a particle swarm optimization, to mitigate the computational complexity associated with the problem. The findings indicate that gateways can improve the data collection process.

Unmanned Aerial Vehicles (UAVs) are usually engineered with a control system to guarantee a specific mission. Otherwise, they can be designed to serve as the link between end users/devices and ground stations. In this thesis, the second category of UAVs is more relevant to the problem at hand. For UAVs to function as intermediary components connecting end users/devices and ground stations, they must possess the requisite hardware and communication protocols and sufficiently cover the designated area. Depending on the hardware and protocols used, UAVs can serve as relay nodes, flying base stations or collaborate to form self-organized networks. This has been discussed in various studies, such as [32, 33, 34, 35] for relay nodes, [36, 37, 38, 39] for flying base stations, and [40, 41] for self-organized networks. Those results can be used as inspiration despite not addressing particular conditions, such as integrating edge systems.

Regarding more particular issues, the authors in [42] have employed local data buffering in situations where gateways are not consistently accessible. Furthermore, the issue of numerous devices being required to transmit a significant quantity of packets, contingent upon the availability of the gateway, is also being tackled. To mitigate collisions and enhance the efficiency of data collection, a proposed solution involves the implementation of a time-slotted transmission scheduling mechanism.

In the study conducted in [43], the researchers create a solution that utilizes both LoRaWAN and WiFi technologies in UAVs. The LoRaWAN protocol is employed to gather positioning and sensor data from mobile firefighters. In contrast, WiFi technology establishes an ad hoc network, enabling communication with remote base stations. Similarly, the research conducted in [44] focuses on crises wherein conventional Global Positioning System (GPS) can offer limited coverage without cellular connectivity. A comprehensive assessment is conducted to evaluate the suitability of LoRa in urban and non-urban settings. Additionally, a multi-hop dissemination algorithm is suggested to maximize the likelihood of successfully delivering emergency requests while minimizing the need for message re-transmissions. The findings indicate that the proposed approach effectively reduces the time needed to address emergencies in comparison to alternative methods of information dissemination. Lastly, the work in [45] inspects the use of mobile end devices exceeding the coverage provided by their home operator. This study presents a proposed LoRaWAN scheme that facilitates inter-operator roaming by employing Domain Name System (DNS) resolution and end-device context migration across networks.

2.4.3 LoRaWAN and Edge Systems Integration

Prior investigations have examined the incorporation of edge computing in the LoRaWAN architecture; however, there are persistent issues regarding their efficient integration. As a result of this circumstance, new studies are consistently arising. The utilization of uplinks, as discussed in the study conducted in [46], enhances the LoRaWAN architecture by introducing additional hops that can be likened to edge points. Another relevant issue to consider is the development of appropriate data delivery protocols. The authors have discussed the topic in [47], presenting a lightweight, long-range protocol designed to enhance data delivery at the edge. This technology facilitates enhanced data transfer rates and reduced error occurrences while maintaining the advantageous features of LoRa. The survey conducted in [48] examines the feasibility of employing a mobile gateway for data collection and delivery, specifically focusing on minimizing travel time and distance. While the proposal encompasses trajectory planning, its primary objective is to provide data solely upon completion of circuit travels.

2.4.4 DRL-Based Solutions

Machine learning has emerged as a versatile tool and has gained extensive utilization across diverse domains. The RL and DRL agents have successfully addressed a wide range of problems. In the context of LoRaWAN, a few specific techniques could offer advantages to the mobile gateway planning resolution. First and foremost, it is vital to acknowledge the concept of dueling networks, as initially proposed in [49]. Using distinct

branches derived from a primary neural network enables the model to outperform the conventional Long Short-Term Memory (LSTM) in various scenarios within the domain of video game experiments. Additionally, the research in [50] introduces a new DRL agent that addresses challenges associated with discrete and high-dimensional action tasks. An additional illustration of this methodology can be observed in the work of [51], wherein a branching network addresses the challenge of a high-dimensional decision space. This approach effectively enhances the performance of a microgrid that incorporates distributed battery energy storage systems. The Branching Dueling Q-Network (BDQ) agent demonstrates effective scalability across diverse environments, making it a promising candidate for network management and planning implementation. Similarly, the approach described in [52] is utilized to configure network slices. This method has demonstrated efficacy in managing an extensive range of actions while ensuring minimal resource utilization. In terms of solving capacity, the DRL with Pointer Networks (Ptr-Net) demonstrates its success in trajectory planning and provides evidence of its achievements in various combinatorial problems [53]. In [54], AC and Ptr-Net techniques are employed within a hybrid architecture to ensure equitable coexistence among diverse nodes operating in unlicensed frequency bands.

2.5 Adopted Research Framework

The work in this thesis addresses the problem of trajectory planning of the gateways as they traverse clusters of nodes and edge systems. It suggests using an agent that effectively optimizes the data collection process at the nodes and delivery at the edge by determining the most efficient trajectory for the gateways, taking into account their respective visit times and data freshness at each stage. As illustrated in Figure 2.4, the given scenario encompasses a complete graph composed of device clusters. Each cluster comprises either LoRaWAN devices transmitting data or storage edge systems receiving data. The gateway relocates to the central region of a cluster and remains there for a designated duration, during which it performs data collection or upload.

The proposed solution consists of a framework that articulates ML agents responsible for determining the mobile gateway trajectory to achieve efficient data collection and delivery with freshness. The workflow is structured in a pipeline with three main machine-learning agents and two mathematical optimization models. First, an RL agent is used to determine the best transmission settings for the gateway when it is placed in the center of a cluster. A second agent is used as a PoC for the ML approach's efficacy in the context of the problem. It considers the best trajectory throughout the journey for multiple gateways and is referred to as Gateway Mobility with Edge Storage (GMwES). Lastly comes an ultimate DRL agent, the Augmented Actor Pointer-Critic (A2PC), which refactors the PoC and integrates state-of-the-art ML techniques to establish the consolidated solution.

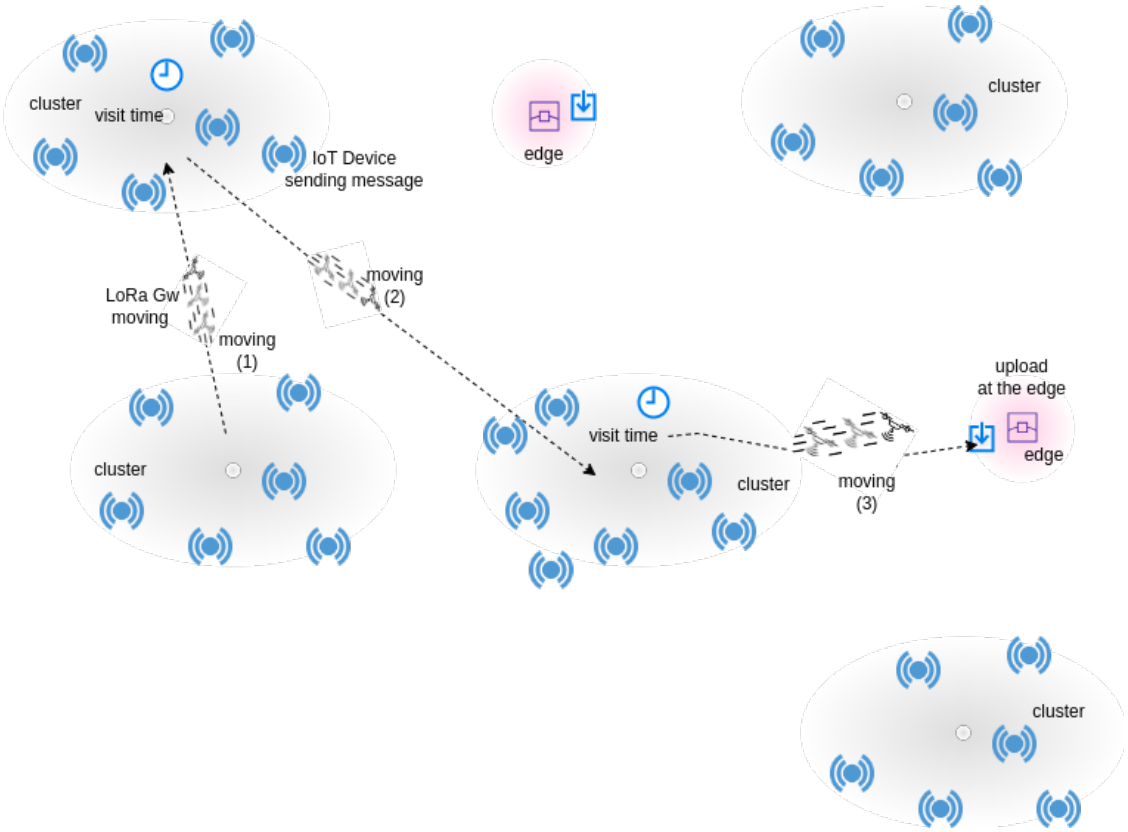

Figure 2.4: The LoRAWAN moving gateway edge assisted scenario overview.

The overall efficiency of the proposed DRL agents is evaluated by the system KPIs. Those are obtained from the respective mathematical optimization models, namely the Mixed Integer Linear Programming I (MILP-I) model for the GMwES and the MILP-II model for the A2PC. In particular, the MILP-I's objective seeks to maximize the travel time for a complete journey (i.e., all data collected and delivered). On the other hand, the main objective of the MILP-II is the amount of data that can be collected or delivered in an arbitrary period of time. Henceforth, in this document, the data obtained from LoRaWAN nodes and transmitted to the edge will be referred to as "processed data" for convenience.

They key contributions of the developed solutions can be outlined in the following:

- The integration of an RL agent with the NS3 LoRa simulator to enable an enhanced self-managed ADR.
- The MILP-I and MILP-II models that address the issue of LoRaWAN gateway mobility problem with edge system, considering data freshness and visit times.
- The GMwES agent, with a one-dimensional output. This agent leverages the Deep Q-Network (DQN) algorithm to attain the trajectory planning resolution and demonstrates the PoC for the DRL approach.

- A novel scalable DRL model, A2PC, that enhances the pointer-critic architecture by incorporating a new branch and loss functions to serve multiple objectives.

The evaluation workflow operates in a sequential manner and can be illustrated in the figure 2.5.

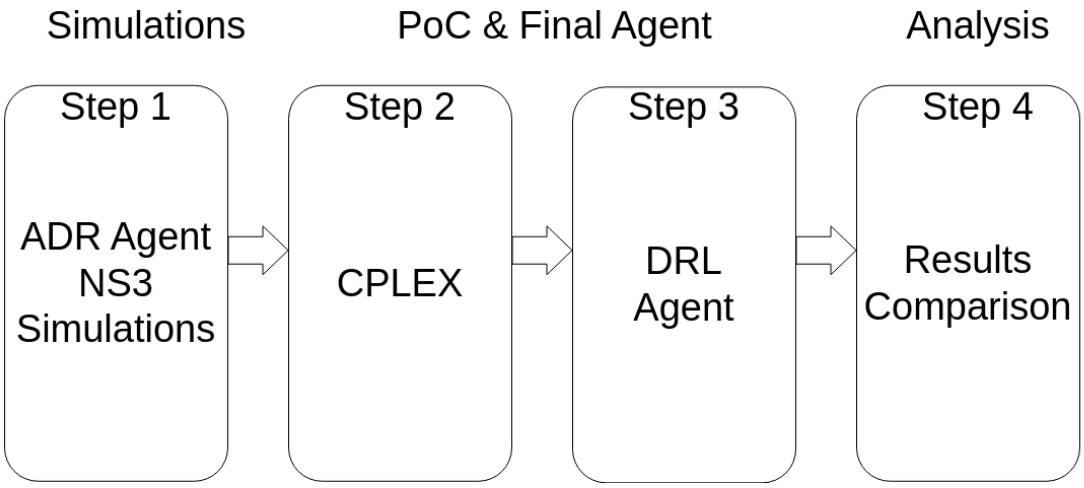

Figure 2.5: High level scheme of the framework workflow.

The workflow entails the generation of LoRaWAN data from NS3, followed by the agents executions and benchmarking of their respective output. This generates a novel approach as it allows DRL techniques to tackle the problem of LoRaWAN mobility and the temporal constraint imposed by the integration with edge systems. It is important to mention that vehicle routing and scheduling problems, such as the one at hand, are known to have NP-hard complexity [55]. Furthermore, it is important to acknowledge that existing mathematical models may not be sufficient in handling ecosystems at large scales. This is when ML methodologies may significantly impact in the issue resolution. With all that in mind, this study presents a novel methodology that uses DRL techniques to develop models capable of optimizing the multi-dimensional decision problem of LoRaWAN mobility planning. The main objective of the agent is to optimize the data collection and delivery during its trajectories, taking into account the advantages related to the waiting time at various locations throughout the journey.

CHAPTER 3

ADR Mechanism Improvement

3.1 Introduction

The utilization of the ADR mechanism in the LoRaWAN involves the dynamic configuration of transmission parameters to end nodes. Such a configuration can result in battery power conservation and ToA reduction. The technique is implemented through the MAC protocol, typically initiated by a request from an end node. While the LoRaWAN standards outline a default ADR mechanism, manufacturers and designers are free to choose any convenient one. This liberty allows the creation of alternative solutions to compete with Semtech's algorithm [56], which is widely recognized as the standard implementation.

Limitations and unresolved issues of the existing ADR options are emphasized in [8]. Solutions to these concerns often employ mathematical optimization methods that separately tackle energy efficiency and throughput [57]. There has been a notable surge in adopting ML techniques in the realm of IoT networks. As a result, some ML tools have been employed to investigate the management of LoRaWAN and the utilization of ADR in IoT networks [58]. Not rarely, these ML solutions impose some challenges, such as the requirement for compute power at terminal nodes and the presence of high-dimension prediction spaces [59]. Furthermore, using supervised ML techniques poses challenges in network management due to data collection and labeling impracticality in numerous scenarios. In contrast, unsupervised approaches, such as RL, are deemed more appropriate as they enable agents to acquire knowledge through active engagement with the environment and adjust their behavior based on the feedback they receive. Moreover, the implementation of LPWAN for IoT on a large scale presents numerous challenges due to the limited availability of radio resources and the complexities associated with their management in real-world networks.

Implementing procedures that ensure efficient resource management is imperative, particularly in devices with significant energy constraints. Nevertheless, executing these procedures in the LoRaWAN nodes may face considerable challenges. All these circumstances have led the research in self-driving networks to shift its focus towards RL

solutions. The primary objective is to acquire knowledge through learning, as opposed to traditional approaches that involve designing network management solutions [62].

RL emerges as a potential tool for an alternative implementation of the ADR mechanism. Its objective is to enhance the performance and prioritize a more agile and responsive ADR mechanism. Given the natural constraints of the LoRaWAN nodes, the ADR system based on RL is better if embedded into mobile gateways. This integration is achieved through simulations that accurately replicate real-world scenarios and corroborate the feasibility of this work. In the agent proposed here, the simulation outputs are utilized to evaluate the performance of the RL agent versus the traditional ADR implementations. The details and workflow of the standard ADR can be obtained from Semtech's implementations [56], and studies on the same, such as the one in [26]. Moreover, to enable a thorough comparison, the empirical scenarios have intentionally been diversified concerning the primary variable of interest in the ADR: the Signal-to-Noise-Ratio (SNR). Including signal loss attenuators, such as the separation between the gateway and the end nodes, physical barriers, and simultaneous peripheral devices, may lead to transmission settings variability. The forthcoming sections provide a more detailed explanation of this proposed agent.

Contributions

- Rodrigo Carvalho, Faroq AL-Tam and Noélia Correia. "Q-Learning ADR Agent for LoRaWAN Optimization", IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2021.

3.2 The RL-ADR proposal

The self-managed LoRa ADR mechanism is a fundamental piece in the design of the DRL agents that solve the LoRaWAN mobile gateway with edge systems. This is because the automated ADR mechanism is a key enabler of the simulations that produce data and parameters for the agents that will solve the problem at hand. Therefore, this chapter aims to detail the design and implementation of this crucial agent. The RL agent under consideration employs the Q-Learning algorithm with the $\epsilon-$greedy action selection strategy. Its design and components are described in the following subsections.

3.2.1 Q-learning ϵ-greedy

In RL, Markov Decision Processes (MDPs) can be used to model sequential decision problems. A MDP is a tuple $(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a probabilistic transition model (matrix), where each

element $p(s'|s,a)$ is the probability of moving from s to s', and $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function.

When an agent interacts with the environment at time t, it observes a state s_t from the environment and makes an action a_t. Accordingly, the environment state changes to s_{t+1} and the agent receives a reward $r(s_t, a_t)$. The objective of the agent is to maximize the collected reward (outcome):

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_0 = s_t\right] \tag{3.1}$$

where γ is a discount factor.

To determine how good it is to be in state s, the value function, $V(s)$ can be used:

$$V(s) = \mathbb{E}\left[G_t | s_t = s\right] \tag{3.2}$$

Similarly, to measure how good it is to be in state s and take action a, a quality function Q (aka action-value function), can also be used:

$$Q(s,a) = \mathbb{E}\left[G_t | s_t = s, a_t = a\right] \tag{3.3}$$

and the objective turns into obtaining the optimal policy from the optimal action-value function $Q^*(s,a)$ using the Bellman optimality equation

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a), \quad \forall s \in \mathcal{S} \tag{3.4}$$

An algorithm to solve this equation is the Q-learning algorithm [63]. Which is off-policy critic-only (compared to on-policy like SARSA and actor-critic algorithms like A2C). In this algorithm, Q is represented as a lookup table, which can be initialized by random guesses and gets updated in each iteration using the Bellman Equation:

$$Q(s,a) = Q(s,a) + \alpha[r(s,a) + \gamma \max_{a \in \mathcal{A}} Q(s',a) - Q(s,a)] \tag{3.5}$$

for some learning rate α.

The second term in (3.5) is known as the Temporal Difference (TD), and $r(s,a) + \gamma \max_{a \in \mathcal{A}} Q(s',a)$ is the TD target.

In order to balance exploration and exploitation, in Q-learning, the agent adapts an ϵ-greedy algorithm. At state s the agent selects an action $a = \arg\max_{a' \in \mathcal{A}} Q(s,a')$ with probability $1-\epsilon$, otherwise selects a random action with probability ϵ. This randomness controls the agent overconfidence and reduces the effect of the local minimum problem. During the training, ϵ is progressively updated via a decaying threshold δ_ϵ. With this annealing property of ϵ-greedy, in practice, an agent is expected to perform almost randomly

in the beginning and become more consistent with time. Next the agent is described in terms of state, action space, and reward function.

3.2.2 State Space

The state is represented by the latest arriving packet settings. This is comprised of the SNR ratio index, denoted by s^{SNR} and can be any value from the set of possible indexes $\{0, 1, 2, 3, \cdots, 13\}$, transmission power value (dBm), s^{TP}, which can be any value of $\{2, 4, 6, \cdots, 16\}$, and spreading factor s^{SF}, which can be any value of $\{7, 8, 9, 10, 11, 12\}$. Therefore, the number of valid states will be $14 \times 8 \times 6 = 672$.

At any time t, the state of the agent s_t will be the concatenation of these sub-states:

$$s_t = [s_t^{\text{SNR}}, s_t^{\text{TP}}, s_t^{\text{SF}}] \tag{3.6}$$

3.2.3 Action Space

The model employs an action space that reflects the combinations of the transmission power value (dBm), which can be any value of $\{2, 4, 6, \cdots, 16\}$, and the spreading factor, which can be any value of $\{7, 8, 9, 10, 11, 12\}$, thus yielding $8 \times 6 = 48$ possible different actions.

3.2.4 Reward function

The proposed reward function is:

$$R = \frac{ToA_{\text{min}}}{ToA} - \frac{\omega}{\omega_{\text{max}}} \times \frac{ToA_{\text{min}}}{ToA} \tag{3.7}$$

where ToA and ω are the time on-air (ms) and consumed energy (mJ), respectively.

In (3.7), the ToA and energy are key factors in the reward function. ToA is measured from the last transmission, while ToA_{min} represents the lowest value measured through all rounds so far. Alike, the energy ω is a measurement from the last transmission, which is divided by the maximum energy measured so far (ω_{max}).

In turn, ToA depends on the packet size and the spreading factor [60]. More precisely and as stated in (3.8), ToA is obtained by using the header implicit flag (H), spreading factor (SF), packet payload (PL), coding rate (CR), bandwidth (BW), and low data rate optimization flag (DE).

$$ToA = \frac{2^{SF}}{BW} \times [8 + \max(\left\lceil \frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)} \right\rceil \times (CR + 4), 0)] \tag{3.8}$$

For the sake of comparison with the classic ADR mechanism, this RL-based ADR agent runs in the LoRaWAN gateway. The overall idea is to prioritize successful transmission parameters consuming less energy.

3.2.5 Implementation and Deployment

This RL agent turns out to be a seamless implementation from the end node point of view. To mimic ADR, the RL agent starts running whenever the gateway receives a MAC `LinkADRReq` command (ADR kickoff request). One of the advantages of having such an RL agent is to eliminate the time for computing an accurate SNR step. Contrary to what the gateway does in the traditional ADR, the agent does not need to collect a certain number of packets before sending a configuration change in the RL approach. In fact, the very first arriving `LinkADRReq` packet triggers a new downlink configuration towards the end node.

In theory, a trained RL agent will respond faster to necessary changes in comparison to the *de facto* ADR implementation. According to Semtech's ADR implementations, it is advised for the gateway to carry out a collection of up to 20 packets before it can start dictating parameter changes to end nodes [56]. Therefore, the time for a gateway to computing its first set of parameters would be 20 times the elapsed ToA, from (3.8). That fact alone can play a remarkable difference between the two methods.

3.3 Performance Evaluation

3.3.1 Simulation Setup

The simulation tool selected to carry out the experiments is the LoRaWAN NS3 module. Its completeness and well-known records [64] have justified its election. Consequently, the RL-ADR has been developed in C++.

Three distinct experiment scenarios were designed based upon the SNR variation behavior. The first scenario with little attenuation, composed of a single end node, distant 2000 meters from the gateway. The second experiment holding moderate attenuation, with the same end node contained inside walls. The last scenario used 3 end nodes, where 2 of them are inside walls, all distant 2000 meters from the gateway, and distributed in 3 perpendicular axes. By introducing obstacles, it is expected that the SNR margin varies from one experiment to another, hence causing the ADR to react differently for each one of the cases. All the three experiments used a constant packet size of 14 bytes, following what is a trend in real-world ecosystems [65]. In a nutshell, the parameters adopted in the training are depicted in the table 3.1 while the scenarios are summarized in the table 3.2.

Parameter	Value	Description
γ	0.8	Discount Factor
α	0.4	Learning Rate
ϵ	0.8	E-Greedy Decay

Table 3.1: Hyperparamaters for the proposed ADR agent.

Scenario Name	Number of Nodes	Distance	Obstacles
Low Attenuation	1	2000 m	None
Moderate Attenuation	1	2000 m	1
High Attenuation	3	2000 m	2

Table 3.2: ADR agent experiment scenarios.

3.3.2 Analysis of Results

Across all three scenarios, the RL-based ADR agent (RL-ADR) was compared with three other ADR variants: The maximum ADR, which calculates the spreading factor and transmission power upon the maximum SNR value from a sequence of received packets; secondly, the minimum ADR, which utilizes the lowest SNR measured; lastly, the average ADR, which takes the average of all collected SNRs. Results for all these simulations are described and analyzed in the following subsections.

Low Attenuation Scenario Results

The training runs in 120 days, while the simulation tests comprise 20 rounds of 7 days each. This single device scenario was tested over the 4 ADR implementations at hand, and the results for all of them are shown in Figure 3.1. For this simple entry-level scenario, only energy per packet is assessed. It is pointless to compare the rate of successfully transferred packets because all methods are likely to achieve 100% given there are no interferences.

Moderate Attenuation Scenario Results

The second experiment utilizes the same amount of time as previous executions, 120 days for training and 20 rounds of 7 days for tests. The difference, in this case, is the presence of walls that impose signal attenuation, therefore yielding different SNR values. Results of these tests are shown in Figure 3.2 and 3.3.

High Attenuation Scenario Results

In this last simulation, times for training and testing are maintained. Three end nodes are positioned at the same radial distance of 2000 meters far apart from the gateway. Two signals are attenuated by walls while a third end node has a clear line-of-sight. This

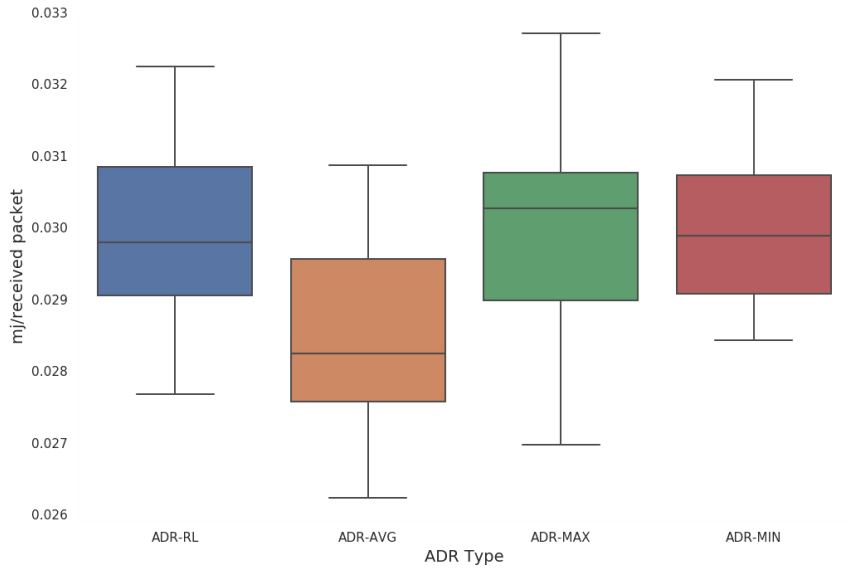

Figure 3.1: Energy in mJ per packet received for the 4 different ADR implementations in a low attenuation scenario.

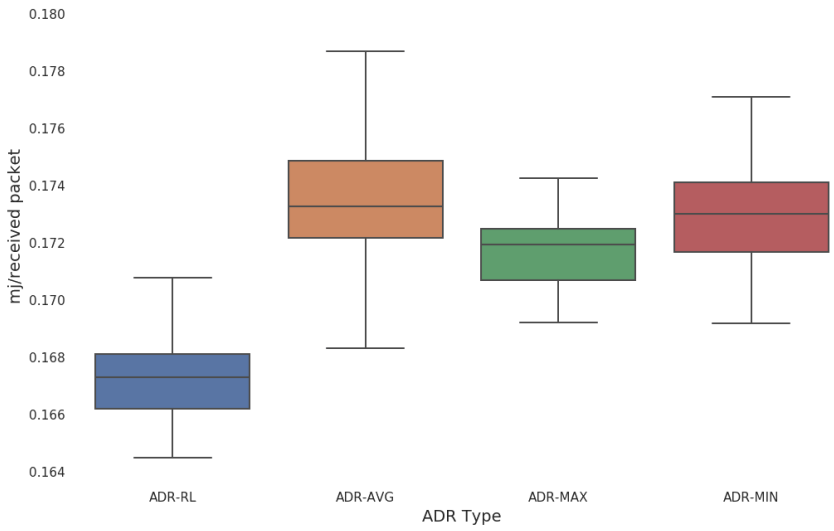

Figure 3.2: Energy in mJ per packet received for the 4 different ADR implementations in a moderate attenuation scenario.

arrangement will combine the two previous experiments into the same scenario, with the additional factor of concurrence. The competition for the medium access and obstacles imply different SNR measurements despite the positions and parameters remain the same as in previous experiments. The tests results are depicted Figure 3.4 and 3.5.

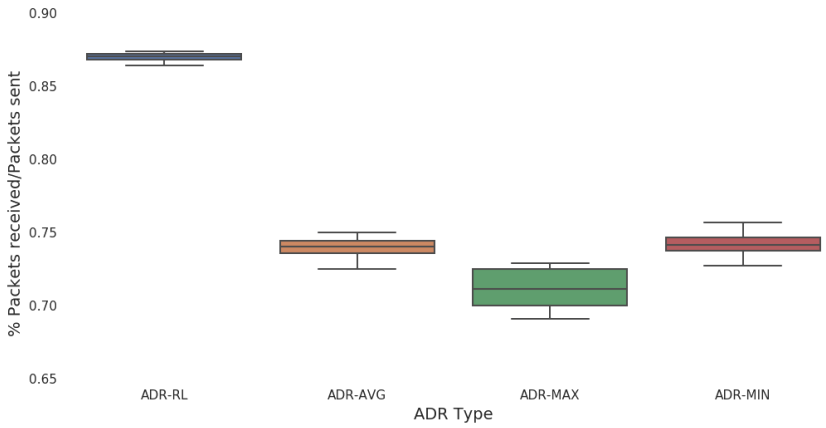

Figure 3.3: Percentage of received packets for the 4 different ADR implementations in a moderate attenuation scenario.

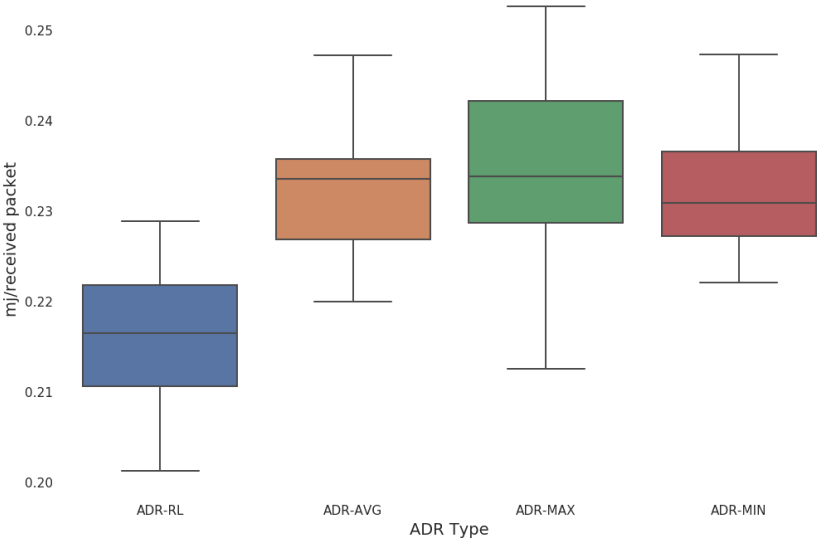

Figure 3.4: Energy in mJ per packet received for the 4 different ADR implementations in a high attenuation scenario.

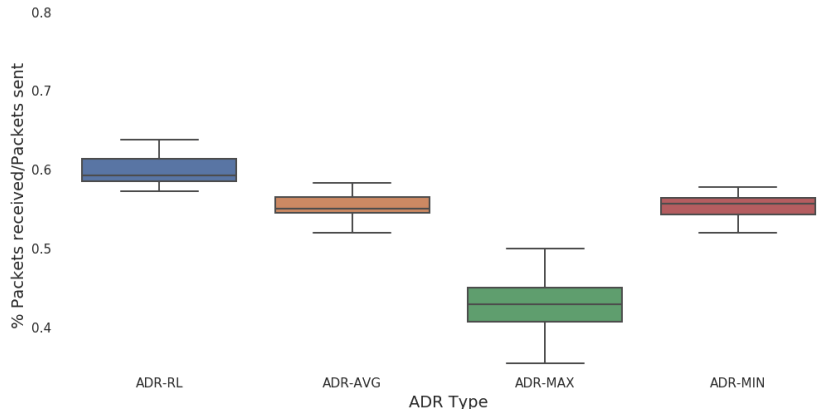

Figure 3.5: Percentage of received packets for the 4 different ADR implementations in a high attenuation scenario.

3.3.3 Summary of Results

In this section, the results are presented and analyzed in a converged way. To facilitate that, Table 6.2.3 compiles the average results of the 20 simulation rounds for each one of the ADR methods.

Scenario	Avg Goodput Packets	Avg Energy mJ/Packet
Low Attenuation RL-ADR	100 %	0.0299
Low Attenuation ADR AVG	100 %	0.0285
Low Attenuation ADR MAX	100 %	0.0299
Low Attenuation ADR MIN	100 %	0.0300
Moderate Attenuation RL-ADR	87 %	0.1673
Moderate Attenuation ADR AVG	74 %	0.1738
Moderate Attenuation ADR MAX	71 %	0.1716
Moderate Attenuation ADR MIN	74 %	0.1731
High Attenuation RL-ADR	60 %	0.2161
High Attenuation ADR AVG	55 %	0.2325
High Attenuation ADR MAX	43 %	0.2350
High Attenuation ADR MIN	55 %	0.2323

Table 3.3: ADR agent: goodput and energy results.

With respect to the low attenuation scenario, the ADR average approach has proven to perform better than other contenders in terms of battery saving. This was driven by default SF and TP values that were tuned enough to deliver optimal performance. The RL-ADR in its turn responded just as well as the maximum and minimum methods. Those seemed to be operating in the lower or upper limits of the SF and TP ranges, and because of that, they tend to be outperformed by a less aggressive approach, such as the average in the long run. It is important to highlight that no method had low performance in regards to the goodput.

Looking at the moderate attenuation tests, the RL-ADR starts to stands out on both fronts, goodput, and battery saving. The immediate response to any existing SNR margin gives this method an enormous advantage initially. This advantage is amplified and maintained as other methods still go under their interactive fine-tuning process.

Lastly, there is the high attenuation scenario where the RL-ADR once again proved to leverage the immediate response to deliver better performance. As the end node conditions are learned from previous experiences, the RL agent is able to configure the network with its optimal parameters for all nodes at the beginning of the communications. The overall difference is stressed out as each one of the nodes needs to go through their stabilization processes. The long-run results show that RL-ADR can perform well whenever trained for a sufficient period of time.

3.4 RL-ADR Conclusions

The proposed RL agent demonstrates itself as a viable alternative for replacing the *de-facto* standard of the ADR. After conducting comparative evaluations with other integrated ADR implementations, it can be asserted that reinforcement learning emerges as a strong candidate for optimizing the ADR process. The approach presented in this section offers a simplified solution to the inquiries explored in prior investigations [58]. Furthermore, when combined with other features, it has the potential to effectively tackle more intricate questions, such as the distribution of optimized configurations, as demonstrated in a previous work [57]. Although utilizing a converged RL-ADR system incurs a substantial computational burden due to the training of network scenarios, it offers advantages, given its quick adaptation.

After conducting these experiments, it has been verified that the RL-ADR enables overall stability and dynamic adaptation. The ability to adjust to various physical medium characteristics and communication patterns across different visited areas is a crucial aspect of the central question of this work. In conclusion, the agent's achievement is of utmost importance in ensuring optimal LoRa communication throughout the visited clusters. Ultimately, the experimental results confirm the justification for utilizing the RL-ADR in conjunction with the NS3 simulators in the design of the GMwES and A2PC agents. On top of that, this mechanism exhibits the potential for expansion into other domains and challenges. For example, one potential approach could involve implementing a DQN solution to address challenges related to compression and fragmentation. This could include the utilization of techniques such as Static Context Header Compression and Fragmentation (SCHC) [66] as well as the application of cross-layer orchestration techniques [61] to optimize performance.

CHAPTER 4

GMwES Agent as a Proof of Concept

4.1 Introduction

Following the proposed framework, this chapter concerns the implementation of a mathematical optimization model and a preliminary DRL agent to solve the trajectory planning with edge systems. During this phase of the work, the output generated from simulations is utilized to parametrize all variables and establish the common context. The first mathematical optimization model, referred to as MILP-I, is the component that sets up the KPI used to assess the DRL agent's performance. Such a KPI is the total amount of data processed in the shortest time possible. The idea of this PoC is to evaluate the ability of DRL methods to solve the problem in a quick and straightforward manner. In the end, the experiments and results are analyzed to validate the overall feasibility of the proposed techniques.

Contributions

- Rodrigo Carvalho, Noélia Correia, Faroq Al-Tam. "Mobility planning of LoRa gateways for edge storage of IoT data", Computer Networks, vol 221, 2023.

4.2 A Mixed Integer Linear Programming Approach

This subsection introduces the MILP-I model for gateway mobility planning, which aims to ensure data collection and delivery at the edge in the shortest time possible. The model takes into account the features of LoRa. Therefore, some relevant conditions inherented from this protocol are taken into consideration first.

4.2.1 LoRaWAN Premisses

Given that LoRa is based on the CSS technology, SF controls the chirp/symbol rate and, therefore, the data transmission speed. A large SF increases the ToA, which also increases energy consumption, reduces the data rate, and improves communication range.

The SF of an end device depends on its distance from the gateway. As for the data rate, it depends on both SF and bandwidth. More precisely, the bit time duration is given by:

$$T(f) = \frac{2^f}{BW} \times 1000 \tag{4.1}$$

where f is the SF in use and BW denotes the channel bandwidth. This determines the useful bit rate, which is given by:

$$R(f) = f \times \frac{BW}{2^f} \times CR \tag{4.2}$$

where CR is the code rate. Please note that the distance ends up determining not only the SF but also the maximum packet size (e.g., for the European 863-870MHz band, the maximum application packet size is 51 bytes when using SF10, SF11, and SF12 on 125kHz, which entails low data rates). The LoRaWAN protocol adds at least 13 bytes to the application payload. Also, the probability of the gateway receiving packets correctly, known as Packet Reception Ratio (PRR), is determined by the SNR when communicating under specific SF and CR. That is, PRR is modeled as a function of SF, CR, and SNR [57]. PRR vs SNR characterizations are described in [67] for different SF and CR values.

The LoRaWAN is characterized by a restrictive use of the channel [68]. A maximum Transmission Duty Cycle (TDC) is imposed on devices, where TDC is the ratio of the accumulated sum of transmission time per observation period. Hence, the maximum TDC will be the maximum percentage of time an end device can occupy in a channel per hour.

4.2.2 Problem Definition

Edge data center infrastructures bring flexibility regarding where to deliver data for storage and where to process data. Analytics closer to the end-user becomes possible, which can address many application performance challenges.

To define the problem of data collection and storage at the edge using LoRa mobile gateways, a set of node clusters is assumed, denoted by $\mathcal{C}$, and gateways are assumed to travel from one cluster center to another. In these conventions, a graph $\mathcal{K}(\mathcal{C}, \mathcal{L})$ where $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$ denotes the set of clusters, and $\mathcal{L} = \{l = (c_j, c_k) : c_j, c_k \in \mathcal{C}\}$ is the set of links. This will be a complete graph as the mobile gateway is allowed to travel between any two clusters. A link $l \in \mathcal{L}$ has a cost that may represent the time required to travel between the endpoints, battery consumption, or any other relevant information.

In the transmission range of the cluster centers, there are sensing IoT devices with data to be collected or edge IoT devices connected to the edge data storage system. This means that some clusters may exclusively have sensing IoT devices or edge IoT devices. The set of clusters that have a connection with the edge data storage system is denoted by

$\mathcal{E}$, where $\mathcal{E} \subseteq \mathcal{C}$. The traveling gateways serve as the bridge between sensing IoT devices and edge IoT devices. An IoT device $i \in \mathcal{I}$ is expected to have $pkt(i)$ packets to send per time unit (i.e., an hour) with average size $sz(i)$, and $pkt(i) = 0$ for edge IoT devices.

The formal definitions of the main elements that compose this problem are listed in the next items.

Definition 1 (IoT Cluster) *Data collection from IoT devices and delivery to an edge system is covered by a pre-determined location (center of the cluster) to which LoRa mobile gateways can travel.*

The set of LoRa mobile gateways is denoted by $\mathcal{G}$, and a gateway $g \in \mathcal{G}$ is expected to visit a subset of IoT clusters. Together all gateways must visit all clusters with data to be collected, besides visiting clusters with edge IoT devices for data delivery. The subset of clusters to be visited by each gateway is to be determined, but a gap-free path must be designed whatever choice is made.

Communication with sensing IoT devices (for data collection) and edge IoT devices (for storage) occurs within an arrival-departure time window, $[\tau_c^{\mathrm{A},g}, \tau_c^{\mathrm{D},g}]$, $\forall c \in \mathcal{C}$, where $\tau_c^{\mathrm{A},g}$ and $\tau_c^{\mathrm{D},g}$ represent the arriving and departure points in time, respectively. This means that the start of reception/transmission and overall service time are bounded by the arrival-departure time window.

Definition 2 (Arrival-Departure Time Window - ADTW) *The time interval that goes from the arrival of the gateway (at the IoT cluster center) to its departure, and during which data reception/transmission can occur.*

If arrival-departure time windows for gateways are found, then LoRa communication can be scheduled at devices, i.e. transmission/reception time windows can be set accordingly. Clearly, the SF-CR configuration in use by devices, which depends on how close a device is from the cluster center (visited by gateways), will influence the ToA of packet transmissions and, therefore, ADTWs.

Definition 3 (Gateway Mobility with Edge Storage - GMwES) *Given a deployment graph $\mathcal{K}(\mathcal{C}, \mathcal{L})$ and a set of mobile gateways $\mathcal{G}$, determine the traveling path for every gateway $g \in \mathcal{G}$, $\zeta(g) = \{c_i, c_j, ..., c_{|\zeta(g)|}\}$, and arrival/departure times at each cluster, $[\tau_c^{A,g}, \tau_c^{D,g}]$, so that the most effective data collection/delivery solution is found, while ensuring that data is still useful when stored (data freshness).*

Effectiveness refers to the power consumption for proper transmission/reception of data at IoT gateways and devices. This can be achieved when traveling times are minimized. Note that a whole panoply of deployment cases can emerge depending on how $\mathcal{C}$, $\mathcal{E}$, $\mathcal{I}$ and $\mathcal{G}$ are defined.

4.2.3 Mathematical Formulation

The mathematical formulation can be done assuming the following convetions:

$\mathcal{C}$	Set of clusters, where $c \in \mathcal{C}$ denotes one of the clusters.
$\mathcal{E}$	Set of clusters with edge devices in their transmission range, $\mathcal{E} \subseteq \mathcal{C}$.
$\mathcal{I}$	Set of IoT devices, where $i \in \mathcal{I}$ denotes a specific IoT device.
$\mathcal{I}(c)$	Set of IoT devices at the transmission range of the center of cluster $c \in \mathcal{C}$.
$v(c, i)$	Binary information stating if device $i \in \mathcal{I}$ is covered by $c \in \mathcal{C}$.
$t(c_j, c_k)$	Traveling time between the center of clusters c_j and c_k, where $c_j, c_k \in \mathcal{C}$.
$pkt(i)$	Amount of packets to be transmitted by IoT device $i \in \mathcal{I}$.
$sz(i)$	Average packet size in bits from device $i \in \mathcal{I}$.
$TTL(i)$	Time to live of packets from device $i \in \mathcal{I}$, which directly impacts the freshness of data.
$ToA(i, c)$	One-bit airtime when transmitting a packet from/to IoT device $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$PRR(i, c)$	Packet reception ratio when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$NoCl(i, c)$	No collision probability when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.

Note that $PRR(i, c)$ and $NoCl(i, c)$ assume an average packet size and are SF-CR related. These are defined by the NS3 simulations, according to the distance of IoT device $i \in \mathcal{I}$ from the center of cluster $c \in \mathcal{C}$. The traditional ALOHA is usually used to derive $NoCl(d, s)$, as in [57]. In this case, the decision variables are:

$\lambda^{g}_{c_j,c_k}$	One if gateway $g \in \mathcal{G}$ travels from the center of cluster $c_j \in \mathcal{C}$ to the center of cluster $c_k \in \mathcal{C}$; zero otherwise.
$\tau^{\mathrm{A},g}_{c}$	Arrival time of gateway $g \in \mathcal{G}$ at the center of cluster $c \in \mathcal{C}$.
$\tau^{\mathrm{D},g}_{c}$	Departure time of gateway $g \in \mathcal{G}$ from the center of cluster $c \in \mathcal{C}$.
$\tau^{\mathrm{T},g}_{c}$	Start of gateway g's reception/transmission time window when visiting cluster $c \in \mathcal{C}$.
μ^{g}_{c}	Service time of gateway $g \in \mathcal{G}$ at cluster $c \in \mathcal{C}$.
$\rho^{g}_{c,i}$	Percentage of packets of IoT device $i \in \mathcal{I}(c)$ that have been collected by gateway $g \in \mathcal{G}$ when visiting cluster $c \in \mathcal{C}$.
$\phi^{g,c_k}_{c_j,i}$	Percentage of packets of IoT device $i \in \mathcal{I}$ that have been collected by gateway $g \in \mathcal{G}$, when visiting cluster $c_j \in \mathcal{C}$, and are delivered in $c_k \in \mathcal{E}$.
$\theta^{g,c_k}_{c_j,i}$	One if $c_k \in \mathcal{E}$ is serving as a delivery cluster for the data of IoT device $i \in \mathcal{I}(c_j)$, $c_j \in \mathcal{C}$.
τ^{MAX}	Upper bound on gateways final arrival time.

For gateways to be able to start/end at any cluster, two virtual clusters are included in $\mathcal{C}$: i) c_1 with outgoing edges only, and $t(.) = 0$ for every outgoing edge; ii) $c_{|\mathcal{C}|}$ with incoming edges only, and $t(.) = 0$ for every incoming edge. These virtual clusters will be the starting and ending points for each gateway, and do not cover any device.

Objective Function

Given an amount of data that has to be collected, and delivered at the edge, gateways should make their journey as soon as possible for energy saving. Therefore, the final arrival time should be minimized for all gateways, which can be achieved through the minimization of τ^{MAX}:

$$\text{Minimize } \tau^{\mathrm{MAX}} \tag{4.3}$$

Minimizing an upper bound on final arrival times will increase energy saving and data freshness because travel and visit times are shortened. The imposition of TTL non-violation constraints (presented next) also ensures that any strict data freshness requirement is met.

Constraints

The following set of requirements must be fulfilled.

– Gateway journey:

Every gateway must start its journey at virtual cluster c_1 and must finish it at virtual cluster $c_{|\mathcal{C}|}$. As mentioned earlier, moving from c_1 to any non-virtual cluster, and moving from any non-virtual cluster to $c_{|\mathcal{C}|}$, has zero cost, so what is actually ensured is that a gateway is allowed to start/finish its path at any real cluster. This is achieved as follows.

$$\sum_{\{c_k \in \mathcal{C}\}} \lambda^g_{c_j,c_k} - \sum_{\{c_k \in \mathcal{C}\}} \lambda^g_{c_k,c_j} = \begin{cases} 1 & ,if\ c_j = c_1 \\ -1 & ,if\ c_j = c_{|\mathcal{C}|} \\ 0 & ,otherwise \end{cases} , \forall g \in \mathcal{G}, \forall c_j \in \mathcal{C} \tag{4.4}$$

Constraints (4.4) apply the flow conservation law to every gateway path, and any subset of $\mathcal{C}$ can be chosen. Such law ensures that paths from c_1 to $c_{|\mathcal{C}|}$ will have no gaps.

– Arrival, transmission and departure times:

It is important to determine when each gateway arrives at the center of the clusters, and when it may leave, to ensure that each gateway stays as long as necessary to collect/deliver the defined amount of data. This can be done as follows.

$$\tau^{\mathrm{A},g}_c \leq \tau^{\mathrm{T},g}_c \leq \tau^{\mathrm{D},g}_c, \forall g \in \mathcal{G}, \forall c \in \mathcal{C} \tag{4.5}$$

$$\tau^{\mathrm{T},g}_c + \mu^g_c \leq \tau^{\mathrm{D},g}_c, \forall g \in \mathcal{G}, \forall c \in \mathcal{C} \tag{4.6}$$

$$\tau^{\mathrm{D},g}_{c_j} + t(c_j, c_k) - \Delta \times (1 - \lambda^g_{c_j,c_k}) \leq \tau^{\mathrm{A},g}_{c_k}, , \forall g \in \mathcal{G}, \forall c_j, c_k \in \mathcal{C} \tag{4.7}$$

where Δ is a big value. Constraints (4.5) force transmission time windows to be included in arrival-departure period of time, and Constraints (4.6) include service time in it. Constraints (4.7) specify that a gateway can not arrive at the center of cluster c_k before $\tau^{\mathrm{D},g}_{c_j} + t(c_j, c_k)$, if it travels from c_j to c_k. The use of Δ allows constraints to be accomplished according to the value decided for $\lambda^g_{c_j,c_k}$.

– Data gathering:

Data collection will have to be done only by gateways that are in a position to do

so, and there should be no accounting for data collection in the other situations. This is accomplished as follows.

$$\rho^{g}_{c_j,i} \leq v(c_j,i) \times \sum_{\{c_k \in \mathcal{C}\}} \lambda^{g}_{c_j,c_k}, \forall g \in \mathcal{G}, \\ , \forall c_j \in \mathcal{C} : \mathcal{I}(c_j) \neq \emptyset, \forall i \in \mathcal{I} \tag{4.8}$$

$$\sum_{\{g \in \mathcal{G}\}} \sum_{\{c \in \mathcal{C} : i \in \mathcal{I}(c)\}} \rho^{g}_{c,i} = 1, \forall i \in \mathcal{I} \tag{4.9}$$

Constraints (4.8) ensure that data is collected at clusters that have the IoT device in their transmission range, and are part of gateway's path. Constraints (4.9) state that all device's data must be collected, although this can be done by more than one gateway.

– Data delivery:

Similarly to the previous set of constraints, data delivery (for storage at the edge) can only be done by gateways that are in a position to do so, which is accomplished as follows.

$$\sum_{\{c_k \in \mathcal{E}\}} \phi^{g,c_k}_{c_j,i} = \rho^{g}_{c_j,i}, \forall g \in \mathcal{G}, \forall c_j \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{4.10}$$

$$\sum_{\{c_j \in \mathcal{C}\}} \sum_{\{i \in \mathcal{I}(c_j)\}} \phi^{g,c_k}_{c_j,i} \leq \sum_{\{c_j \in \mathcal{C}\}} \lambda^{g}_{c_j,c_k} \times |\mathcal{I}| \times |\mathcal{C}|, \\ , \forall g \in \mathcal{G}, \forall c_k \in \mathcal{E} \tag{4.11}$$

Constraints (4.10) ensure that all collected data is delivered at the edge, while Constraints (4.11) ensure that data is delivered at edge places that are part of the gateway's path.

– Data freshness:

Whenever a gateway collects data from an IoT device, it must be ensured that the lifetime of the data does not expire until it is delivered at the edge storage system. This is ensured by the following constraints. Given the nature of the objective function, the freshness of the data is also maximized because, once specific lifetime constraints are met, the time for gateways to collect and deliver data is minimized.

$$\theta_{c_j,i}^{g,c_k} \geq \phi_{c_j,i}^{g,c_k}, \forall g \in \mathcal{G}, \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{4.12}$$

$$\tau_{c_k}^{\mathrm{T},g} - \tau_{c_j}^{\mathrm{T},g} \leq TTL(i) + \Delta \times (1 - \theta_{c_j,i}^{g,c_k}), \forall g \in \mathcal{G}, \\ , \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{4.13}$$

Constraints (4.12) determine the binary information stating if cluster c_k is supporting the delivery of data from $i \in \mathcal{I}(c_j)$, required for the following set of constraints, and Constraints (4.13) ensures that data is delivered to the edge before it becomes outdated.

– Total service time:

The time that a gateway stays in a cluster must be at least as long as the time it takes to collect and/or deliver the amount of data it is supposed to, which increases if $PRR(.)$ or $NoCl(.)$ decrease. It is worth noting that $PRR(.)$ and $NoCl(.)$ are applied here in the same way as in previous validated cases, like in [57].

$$\mu_{c_j}^{g} \geq \sum_{\{i \in \mathcal{I}(c_j)\}} \frac{pkt(i) \times sz(i) \times ToA(i, c_j)}{PRR(i, c_j) \times NoCl(i, c_j)} \times \\ \times [\rho_{c_j,i}^{g} + \sum_{\{c_k \in \mathcal{C}\}} \phi_{c_k,i}^{g,c_j}], \forall g \in \mathcal{G}, \forall c_j \in \mathcal{C} \tag{4.14}$$

Constraints (4.14) include spectrum utilization time of both data collection and delivery when determining the service time of a gateway in a certain cluster.

– Largest final arrival time:

The following constraints are required to determine the largest final arrival time (upper bound).

$$\tau^{\mathrm{MAX}} \geq \tau_{c_{|\mathcal{C}|}}^{\mathrm{A},g}, \forall g \in \mathcal{G} \tag{4.15}$$

– Non-negativity assignment to variables:

$$\lambda_{c_j,c_k}^{g}, \theta_{c_j,i}^{g,c_k} \in \{0, 1\}; \\ \tau_{c}^{\mathrm{A},g}, \tau_{c}^{\mathrm{D},g}, \tau_{c}^{\mathrm{T},g}, \mu_{c}^{g} \geq 0; 0 \leq \rho_{c,i}^{g}, \phi_{c_k,i}^{g,c_j} \leq 1. \tag{4.16}$$

Complexity

The above optimization problem can be solved using packages like CPLEX or Gurobi, [69, 70], which eventually finds the optimal solution given an instance of the problem. However, the GMwES problem is more complex than the vehicle routing problem with time windows, addressed in [55], as both time windows and collect/drop places are also being decided. Vehicle routing and scheduling problems are known to be NP-Hard, meaning that the GMwES problem will also be hard to solve. This implies in feasible solutions for small instances of the problem only. Even so, this optimization problem is useful to play a role as an upper bound and metric establishments, as its results can be regarded as optimal ones for not very large instances.

Given the drawback of mathematical optimization, a DRL solution is proposed next: The GMwES agent. In this approach, the DRL agent can incorporate any underlying dynamics into its decisions and, after training, make appropriate decisions in real-time [71]. Therefore, it can be used in any deployment without prior knowledge. Although training takes time too, the problem being addressed is smaller because the DRL agent does not have to be aware of which devices are being covered by the gateways, nor their transmission time windows and traffic pattern (dynamics captured by the agent when interacting with the environment).

4.3 The Reinforcement Learning Approach

RL agents simplify administration by allowing for the development of solutions that can be continually refined and adjusted to various contexts [62, 72, 73]. This trait will be leveraged to create the PoC described in this section. Following the model developed with a given a set of clusters $\mathcal{C}$ and set of IoT devices $\mathcal{I}(c)$ at the transmission range of the center of cluster $c \in \mathcal{C}$, the GMwES problem can be solved in two stages:

1. Select the best transmission configuration for a device to communicate with the center of its cluster. This is achieved in the RL-ADR simulations detailed in Chapter 3.

2. Solve the GMwES problem, assuming the previously selected transmission configurations.

As explained before, the best configuration for every gateway visit, can be provided by the RL-ADR as in Chapter 3, which jointly optimizes data rate and power. The solution of the second part is presented next.

4.3.1 Background and Assumptions

As already highlighted in Chapter 3.2, the Q-learning algorithm is used to maximize the accumulation of reward by an agent. In this algorithm, Q is represented as a lookup table, which can be initialized by random guesses and updated in each iteration using the Bellman Equation; same as presented in (3.5).

$$Q(s,a) = Q(s,a) + \alpha[r(s,a) + \gamma \max_{a \in \mathcal{A}} Q(s',a) - Q(s,a))], \tag{4.17}$$

where α is a learning rate.

Since it is empirically difficult to maintain a lookup table for Q, this is usually approximated using a Deep Neural Network (DNN). In this work two hidden layers of 512 neurons each are used for this purpose.

To balance exploration and exploitation, when in state s, the agent selects an action a as $a = \arg\max_{a' \in \mathcal{A}} Q(s,a')$ with probability $1-\epsilon$, otherwise a random action is selected with probability ϵ. During the training, ϵ is progressively updated via a decaying threshold δ_ϵ similar to what was employed in the RL-ADR agent, from Chapter 3.

4.3.2 GMwES Agent Design

State

At any given time t, the state of gateway g is represented by a vector specifying its current position, $s_t^{\mathrm{P},g}$, together with elapsed timesteps $s_t^{\mathrm{E},g}$, collected bytes $s_t^{\mathrm{B},g}$, data age $s_t^{\mathrm{A},g}$ and gateway selector $s_t^{\mathrm{F},g}$:

$$s_t(g) = [s_t^{\mathrm{P},g}, s_t^{\mathrm{E},g}, s_t^{\mathrm{B},g}, s_t^{\mathrm{A},g}, s_t^{\mathrm{F},g}], \tag{4.18}$$

where $s_t^{\mathrm{F},g} = 1$ if g is the gateway in focus, and $s_t^{\mathrm{F},g} = 0$ otherwise.

Then, the agent state s_t is the concatenation of the states of all gateways:

$$s_t = \bigcup_{\{g \in \mathcal{G}\}} s_t(g) \tag{4.19}$$

Action Space

The action space used by the model is a direct representation of existing clusters. This is because a gateway can move to the center of any cluster at any given time. Thus, the action space will be $\mathcal{A} = \{c_1, c_2, ..., c_{|\mathcal{C}|}\}$, where c_1 and $c_{|\mathcal{C}|}$ are two virtual clusters (also included in the mathematical formulation) for gateways to be able to start/end their true journey at any real cluster. Gateways are initially positioned at cluster c_1, and $|\mathcal{C}|$ is a terminal state. The traveling time when moving to cluster c_i, $i \notin \{1, |\mathcal{C}|\}$, is $t(.) = 0$. Such action space includes both collecting and delivering (edge) points.

Reward

The proposed reward function is composed of two components, one related with data collection and the other related with data delivery. The first component is required because the agent must be able to decide whether it is good to stay at the same place, or if it is better to move to another location. The second component is required to encourage the delivery of data at the edge. There shall be a reward only when the data has not expired. Therefore,

$$r(s_t, a_t) = \begin{cases} 1 & , if\ a_t \notin \mathcal{E} \wedge \text{message collected} \\ s_t^{\text{B},g} & , if\ a_t \in \mathcal{E} \wedge s_t^{\text{A},g} + t(a_{t-1}, a_t) \leq \\ & TTL \\ 0 & , \text{otherwise} \end{cases} \tag{4.20}$$

where TTL is the time to live of packets. $s_{t+1}^{\text{B},g}$ resets whenever $a_t \in \mathcal{E}$.

4.3.3 Implementation Considerations

Adopted Timestep

In the current implementation, a discrete timestep is adopted. This assumption is motivated by two facts: i) given this is a PoC, the adoption of a discrete unit of time enables a more straightforward agent design, thus having a faster convergence; ii) the agent has better performance when dealing with discrete space states. These issues have already been discussed in [74, 75].

The adopted timestep is based upon a minimum time that enables capturing the LoRa message in the worst-case scenario. This is based upon the formula used to calculate the ToA in LoRaWAN networks, given by [76]:

$$ToA = \frac{2^{SF}}{BW} \times [8 + \max(\left\lceil \frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)} \right\rceil \times (CR + 4), 0)], \tag{4.21}$$

where H is the header implicit flag, SF is the spreading factor, PL is the packet payload, CR is the coding rate, BW is the bandwidth, and DE is the low data rate optimization flag. With all that, the smallest possible timestep would be the ToA of $\sim 2793,5ms$. That number is obtained for a maximum payload of 51 bytes, considering the slowest data rate, i.e., $SF = 12$ at 125 kHz. Nevertheless, this could still be too strict in scenarios where gateways have mobility. The empirical evidence shows that a timestep of $5s$ (approximately twice the value of the ToA calculated above) can accommodate the worst cases,

i.e., moving gateways communicating with the farthest covered device (lowest data rate).

Cluster positions and travel times are assumed to be a known information, while the covered IoT devices and traffic patterns are learned by the agent.

Training

The training phase is a pivotal stage in implementing any RL agent. This section presents the parameters and logic employed in it.

input: ℓ_{episode}, M, ϵ, δ_ϵ, $\min_\epsilon$, $\mathcal{R}$.
output: trained neural network
Initialize state $\boldsymbol{s}$ randomly according to each state component range;
for $i = 1 : \ell_{episode}$ **do**
  Forward state $\boldsymbol{s}$ to the on-line Q neural network and get,via ϵ-greedy, the selected cluster: $c = \arg\max_{a \in \mathcal{A}} Q(\boldsymbol{s}, a)$;
  Anneal ϵ as: $\max\{\epsilon - \delta_\epsilon, \min_\epsilon\}$;
  Calculate the reward $r(\boldsymbol{s}, c)$ using (4.20);
  Calculate new state $\boldsymbol{s'}$;
  Add the tuple $(\boldsymbol{s}, c, \boldsymbol{s'})$ to the experience replay $\mathcal{R}$;
  $\boldsymbol{s} \leftarrow \boldsymbol{s'}$;
  Sample M mini-batches from $\mathcal{R}$ and train the on-line Q neural network;
  Update the target critic Q′ neural network parameters;
end
Return trained target critic Q′ neural network.

Algorithm 1: Training phase of the GMwES agent.

The algorithm used to train the GMwES agent is shown in Algorithm 1. It follows the Q-learning with double DQN networks (two neural networks, the online and the target). In this algorithm, a single episode with length ℓ_{episode} is shown but the algorithm can run for multiple episodes. At the end of each episode, it returns a trained target network. For a better sample efficiency, an experience replay memory $\mathcal{R}$ is used to memorize the visited states and a set of samples (mini-batch) of size M is used to train the online network. The parameters used in the GMwES agent are listed in Table 4.1.

Parameter	GMwES agent
Discount Factor γ	0.9
Learning Rate α	0.2
Exploring Decay ϵ	0.8

Table 4.1: Hyperparameters of the GMWES agent.

The GMwES agent is fully developed in Python, Pytorch, and OpenAI Gym [77]. The

LoRa environment is the LoRaWAN NS3 simulation model just as the ones used in the RL-ADR simulations from Chapter 3.

4.4 Performance Evaluation

The implementation steps required to evaluate the performance of the proposed DRL solution, as well as the KPIs of interest are explained next.

4.4.1 Framework Pipeline Setup

The output of the RL-ADR simulations represent the best LoRa parameters for IoT devices to communicate with the center of their cluster, where the gateways are eventually positioned. This means that the ToAs associated with possible packet transmissions become known.

A further step is to solve the MILP-I using CPLEX[1]. The optimal value obtained for each instance, τ^{MAX}, is the shortest time possible to collect and deliver all data. Moving next, this is used to evaluate the performance of the GMwES agent. More precisely, the GMwES agent learns its trajectory for processing all data under the time window τ^{MAX}. This means that the KPI will be the throughput achieved by the agent, which has as an upper bound of 100% stipulated by the optimization model.

To organize and define fair comparison criteria, the types of scenarios used in the experiments are classified according to a convention based on their size specifications. Large scenarios have twice as many clusters as small ones, while dense scenarios have twice as many devices (on average and per cluster) as sparse ones. Devices are randomly distributed using a uniform distribution. In a nutshell, the whole set of possible scenario combinations are: Small and sparse (SS), small and dense (SD), large and sparse (LS), and large and dense (LD), as outlined in Table 4.2.

Scenario	**No. Clusters**	**Avg No. Devices per Cluster**
Small & Sparse (SS)	8	1 to 7
Small & Dense (SD)	8	8 to 16
Large & Sparse (LS)	16	1 to 7
Large & Dense (LD)	16	8 to 16

Table 4.2: GMwES deployment scenarios.

Messages are 18-byte long on average, a value observed in real-world applications (see [65]). The number of messages sent per device ranges from 1 to 5 (the largest TDC-

[1]IBM ILOG CPLEX Optimizer version 12.8.

Information	Value
No. Gateways	1-3
No. Clusters	8 or 16
No. Edges*	2 or 4
Nodes Distribution	Sparse or Dense
TTL	1200s
Message Avg Size	18 bytes
No. Messages per Node	1-5

*Max. no. edges to cluster size ratio: 25%.

Table 4.3: GMwES simulation setup information.

compliant number of LoRaWAN messages, considering the worst SF). These values are generated using a uniform distribution at simulation time. In turn, the data freshness threshold, or TTL, is chosen in line with the travel times between clusters. A $TTL(i) = 1200s$, $\forall i \in \mathcal{I}$ is a reasonable value for the scenarios generated in the simulations. The simulation setup information is summarized in Table 4.3.

Finally, it is worth mentioning that the GMwES agent training elapses in the order of minutes while CPLEX may span through several hours when executed on the same PC, with 8 vCPU Intel i7, 16 GB RAM, and Nvidia GeForce RTX 2080 GPU GV102 with 11264 MB.

4.4.2 Analysis of Results

The mathematical model is a baseline to analyze the performance achieved by the agent. Since τ^{MAX} is the smallest time window to accommodate 100% delivery, it can be used as a hyperparameter (episode length) of the agent. There are three main reasons why τ^{MAX} can serve as an episode length: i) τ^{MAX} depends on the scenario and deployment (discussed further below); ii) it does not change frequently; iii) when training the agent to deliver as much data as possible, for the period τ^{MAX}, an agent's throughput analysis can be performed by comparing the achieved percentage of delivery to the 100% limit obtained by the mathematical model.

In the following subsections, the convergence of the agent is discussed, and then τ^{MAX} is analyzed for different scenario types and number of gateways. Finally, the throughput achieved by the agent, under τ^{MAX}, is evaluated. These analysis take the average values of 30 executions per scenario.

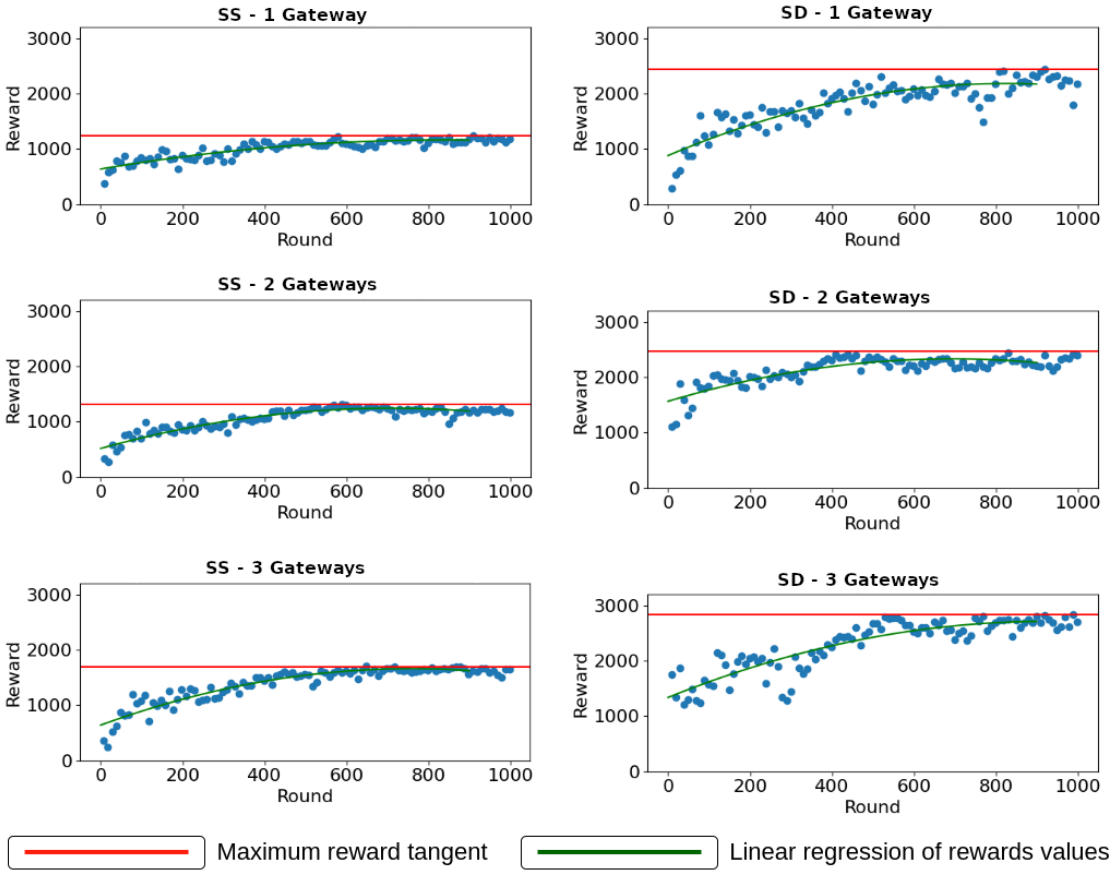

Figure 4.1: Reward curve for different trainings.

4.4.3 Summary of Results

Convergence

Convergence analysis can be performed based on the reward curve. The preliminary assessments of this DRL agent show that the size of the scenarios does not influence the convergence arrival. Hence, small scenarios were used during random training rounds to produce enough details about the convergence pattern. It is interesting to see that the convergence happened around 500 episodes regardless of the type of scenario or number of gateways being used. This stability in convergence, depicted in Figure 5.6, indicates the minimum number of episodes required to be used during the training of the GMwES experiments.

Final Arrival Time

Table 4.4 shows the average optimal values for τ^{MAX}, found by the mathematical optimization model, for each type of scenario and number of gateways. It is possible to observe that the first extra gateway (2nd column in this table) brings the greatest time reduction, particularly for LD scenarios. In these scenarios, adding a single gateway

reduces τ^{MAX} by more than 50%. When adding another extra gateway (3rd column) for the same scenarios, only $\approx$ 26% reduction was achieved. Therefore, the mathematical model not only provides an accurate estimate of the required τ^{MAX} for each scenario type but also allows finding the most cost-effective (and feasible) scenario given an acceptable travel time for the gateways. In addition, data freshness is ensured because visit and travel times are shortened as much as possible. In contrast, the TTL non-violation constraints in the mathematical optimization model meet any specific and more stringent data freshness requirement.

	No. of Gateways		
Scenario	**1**	**2**	**3**
SS	4028.53s	2122.80s	1472.57 s
SD	4256.30s	2154.00s	1550.33 s
LS	8142.45s	4125.88s	2745.00s
LD	9258.30s	4313.80s	3151.55s

Table 4.4: GMwES agent average gateway final arrival time (τ^{MAX}).

The findings from Table 4.4 are used next to analyze the agent's performance given a scenario type and number of gateways.

Throughput

Figures 4.2 and 4.3 show the average number of bytes successfully delivered at the edge by both the mathematical optimization and GMwES agent for small and large deployment scenarios, respectively. It is possible to see that the developed agent can produce near-optimal results, especially in larger and more dense scenarios. This means that the agent is able to use the available resources to deliver more bytes even when the input space increases.

Table 4.5 shows the average throughput percentages achieved by the agent compared to the 100% limit obtained by the mathematical model under τ^{MAX}. From these results, it is possible to observe that the developed agent can achieve better throughput performance under large and dense scenarios when compared with small or sparse deployments despite the intricacy of these scenarios (notice that larger scenarios have twice the number of clusters than small scenarios, and dense scenarios have twice the number of devices per cluster than sparse ones).

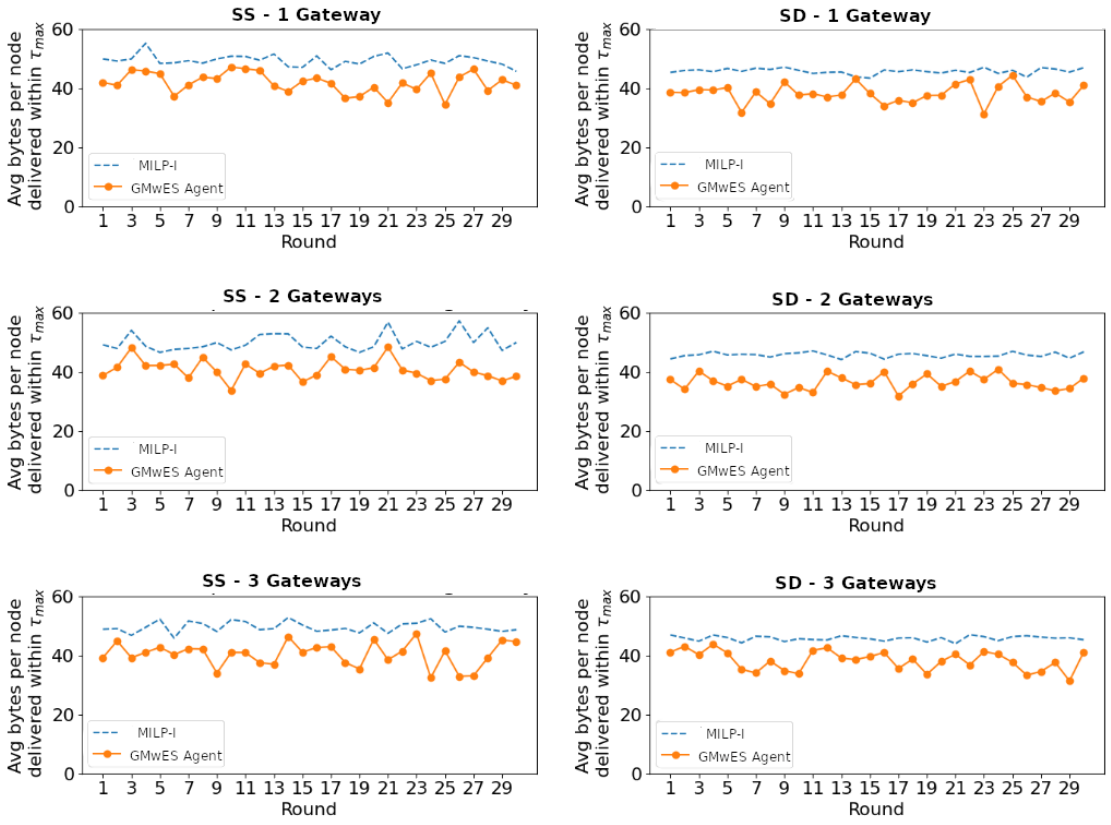

Figure 4.2: Small deployment scenarios; Results for 1,2 and 3 gateways.

	No. of Gateways		
Scenario	**1**	**2**	**3**
SS	84.43%	81.23%	83.20%
SD	82.83%	79.06%	80.93%
LS	76.58%	86.76%	85.63%
LD	79.05%	88.58%	88.77%

Table 4.5: GMwES agent average throughput (episode duration = τ^{MAX}).

It is worth noting that a 3rd gateway seems to bring negligible benefit or no benefit at all, and it should be further verified whether this is related to the strictness imposed by τ^{MAX} (Table 4.4). That is, more gateways will allow the optimizer to accommodate collections/deliveries in a very tight time window, which may hinder the agent's learning task. For the purpose of evaluating this point, tests were carried out in the GMwES agent for more relaxed episode lengths. To be more specific, 5%, 10%, and 15% above the initial value of τ^{MAX} each. It is possible to see that there is room for throughput improvement by the GMwES agent when time windows are not as strict as they are shown in Table 4.6. This may be the case in many realistic monitoring scenarios, and it may be an option that should be investigated when dealing with situations based on the real world.

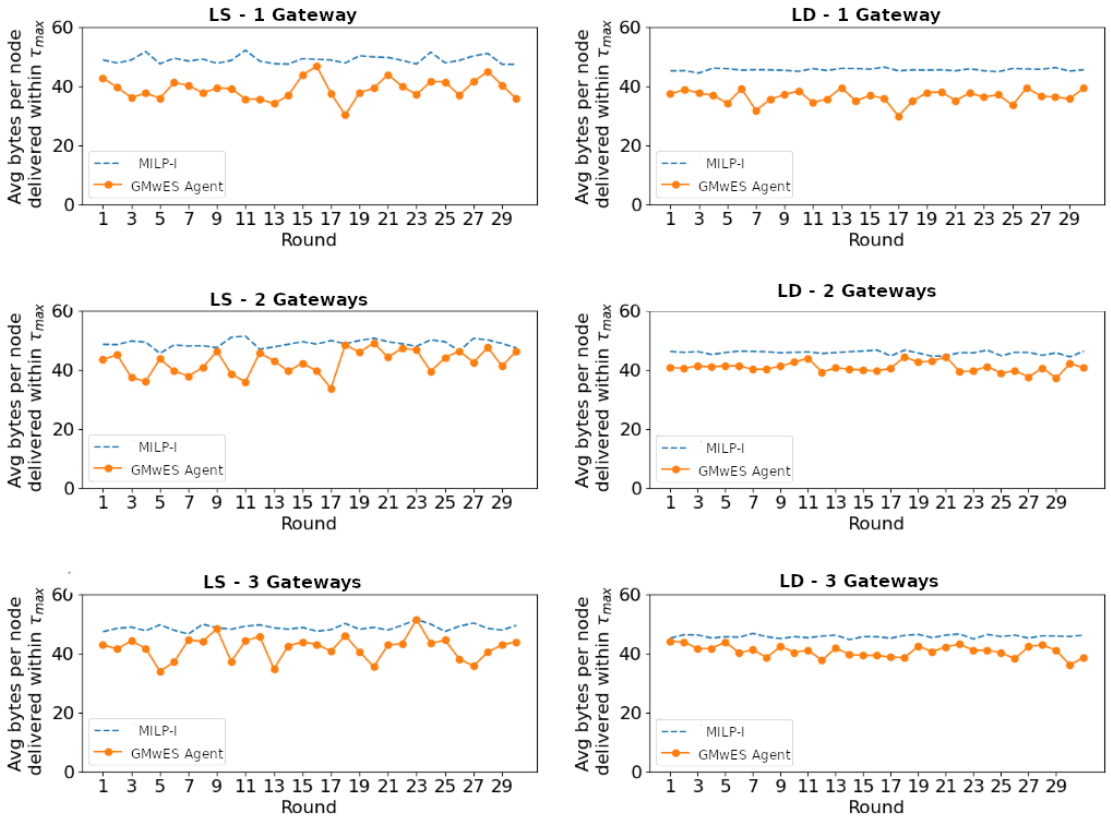

Figure 4.3: Large deployment scenarios; Results for 1,2 and 3 gateways.

	Additional Margin		
Scenario	**5%**	**10%**	**15%**
LS (3 GWs)	87.09 %	88.51 %	89.35 %
LD (3 GWs)	90.11%	91.38%	92.53%

Table 4.6: GMwES agent average throughput (episode duration = τ^{MAX} plus margin).

In summary, the developed GMwES agent can achieve a throughput near the optimal, mainly when final arrival times are not so strict and for large deployment scenarios. Given the advantages of having a DRL approach, where the agent does not have to be aware of which devices are being covered by the gateways, nor their transmission time windows and traffic patterns (contrarily to the mathematical optimization model)

4.5 GMwES Conclusions

In this section, a DRL GMwES agent is proposed for the planning of LoRa gateway mobility, considering data storage at the edge. The approach is compared with the results of the optimal solution given by the formulated optimization problem, which aims to minimize the time window required for complete data delivery. However, mathemat-

ical optimization requires too much time to solve large instances of the problem. It also requires full knowledge of the scenario (e.g., devices covered by each possible gateway location, transmission time windows of devices, and traffic pattern) being optimized, which is not always feasible in practice.

Results show that the proposed approach can produce near-optimal solutions, particularly for large deployment scenarios and when episode duration is not so strict. Performance is not affected by the number of gateways and kind of scenario, meaning it is a scalable solution. Such consistency is important for decision-making in real-world scenarios. These results and the fact that the agent does not have to be aware of which devices are being covered, nor their transmission time windows and traffic pattern, allow to confirm that the approach is an added value for developing adaptive and intelligent monitoring systems.

The success achieved on this assessment promotes extended paths for this investigation. Such extended possibilities can improve the agent, and they are at the core of the next chapter's contents. They represent the state-of-the-art and the ultimate techniques to wrap up the proposed framework.

CHAPTER 5

A2PC Agent

5.1 Introduction

This chapter presents the last part of the proposed framework for tackling the LoRaWAN gateway mobility with edge system problem. Based on the ideas and results from the PoC, this ultimate design includes making a better ML agent to determine the best course of action for the mobile gateway trajectory and the best times to collect and send data without breaching freshness. The goal is to build on the idea used in the previous agent and make it work better overall. This enhancement can be achieved by tackling the challenges in dealing with large dimensional action space.

Here, the DRL agent incorporates two advanced machine-learning techniques that are properly integrated for the purpose mentioned. One is a Ptr-Net AC model with masks, which enables scalability. The other is the action-branching technique, which is applied to the main AC neural network. This forking yields better learning capabilities when the action space is large. By splitting the action dimensions, the agent can better learn multiple patterns influenced by a single critic component. In the LoRaWAN gateway mobility with edge system problem, an ordinary case that illustrates this dimensional issue can be seen when 16 different hops need to be chosen in a journey of 12 different movements. This sequence alone generates a permutation within the range of billions of possibilities. Combining the visit time with the hop decisions creates a dimension that makes it challenging to achieve convergence when using conventional neural networks. In summary, by incorporating the forking into the core AC component, the agent can enhance its ability to learn and recognize multiple associated patterns.

Another mathematical model is used to evaluate this DRL agent's overall efficiency. Such model is referred to as MILP-II, and it is derived from Chapter 4. Its objective is to calculate the highest amount of data processed within an arbitrary time interval. The MILP-II differs from the MILP-I regarding the objective function and specific constraints. Consequently, they cannot be directly compared.

Contributions

- Rodrigo Carvalho, Faroq Al-Tam. and Noélia Correia "A2PC: Augmented Advantage Pointer-Critic Model for Low Latency on Mobile IoT with Edge Computing", IEEE Transactions on Machine Learning in Communications and Networking, Vol. 3, 2024.

5.1.1 A Mixed Integer Linear Programming Upper Bound

This MILP-II model determines an upper bound for the amount of data that can be processed within a specified time frame and is used as a reference when analyzing the DRL agent's performance. Once again, this model presupposes the existence of coverage zones in cluster partitioning, and the gateways traverse these zones to gather or transmit data to a central location. The main difference between the MILP-I, presented in Chapter 4, and this MILP-II is that the latter employs a single gateway instead of multiple ones. This setup is made because the action-branching DRL agent likely needs an architecture with more than one agent to accommodate various gateways. At first glance, the mathematical formalization of both models may seem to be repeated. The MILP-II definitions are the same as the ones used in the MLIP-I, however the conditions and constraints formalization are slightly different, given the divergences around multiple and single gateways. With that in mind, the entire MILP-II model formalization is detailed below so an easier distinction between MILP-I and MILP-II can be made:

$\mathcal{C}$	Overall set of clusters, where $c \in \mathcal{C}$ denotes one cluster.
$\mathcal{E}$	The set of clusters with edge devices inside their transmission range, $\mathcal{E} \subseteq \mathcal{C}$.
$\mathcal{I}$	Overall set of IoT devices, where $i \in \mathcal{I}$ denotes a specific IoT device.
$\mathcal{I}(c)$	The set of IoT devices at the transmission range of the center of cluster $c \in \mathcal{C}$.
$v(c, i)$	This flag identifies whether device $i \in \mathcal{I}$ is covered by $c \in \mathcal{C}$.
$t(c_j, c_k)$	The travel time between the centers of the clusters c_j and c_k, where $c_j, c_k \in \mathcal{C}$.
$pk(i)$	The amount of packets to be transmitted by IoT device $i \in \mathcal{I}$.
$sz(i)$	The average packet size (bits) transmitted by device $i \in \mathcal{I}$.
$TTL(i)$	The time to live for packets from device $i \in \mathcal{I}$. This parameter directly impacts data freshness as it sets the period when data are still valid for processing. After that, data are considered expired.

$ToA(i,c)$	This represents the airtime for one bit when transmitting a packet from/to IoT device $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$PRR(i,c)$	The packet reception ratio when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
$NoC(i,c)$	This is the non-collision probability when transmitting a packet from/to $i \in \mathcal{I}$ to/from the center of cluster $c \in \mathcal{C}$.
τ^{MAX}	The time window for data collection.

Once more, following what was explained in Chapter 4 and based upon [57], $PRR(i,c)$ and $NoC(i,c)$ holds an average packet size related to a specific SF-CR, thus the variables turn to be:

λ_{c_j,c_k}	This takes the value 1 if the gateway travels from the center of cluster $c_j \in \mathcal{C}$ to the center of cluster $c_k \in \mathcal{C}$; it is 0 otherwise.
τ_c^{A}	This denotes the arrival time of the gateway at the center of cluster $c \in \mathcal{C}$.
τ_c^{D}	Following the above, this is the departure time of the gateway from the center of cluster $c \in \mathcal{C}$.
τ_c^{T}	When visiting cluster $c \in \mathcal{C}$, this is the start of the gateway's reception/transmission time window .
μ_c	This is the period spent transmitting data. For simplicity, inside the MILP-II, this is referred as to as the *service time* of the gateway at cluster $c \in \mathcal{C}$.
ρ_c^i	This represents the percentage of packets of IoT device $i \in \mathcal{I}(c)$ collected by the gateway when visiting cluster $c \in \mathcal{C}$.
φ_{c_j,c_k}^i	This represents the percentage of packets from IoT device $i \in \mathcal{I}$ collected by the gateway when visiting cluster $c_j \in \mathcal{C}$, but also delivered in $c_k \in \mathcal{E}$.
β_{c_j,c_k}^i	This flag is set to 1 if $c_k \in \mathcal{E}$ is serving as a delivery cluster for the data of IoT device $i \in \mathcal{I}(c_j)$, $c_j \in \mathcal{C}$.

Two virtual clusters are included in $\mathcal{C}$: i) c_1 is a cluster that only has edges leading away from it, and its time cost $t(.)$ is equal to 0 for each of these outgoing edges; ii) $c_{|\mathcal{C}|}$ is a cluster that only has edges leading towards it, and time cost $t(.)$ is also equal to 0 for each of these incoming edges. These virtual clusters do not include devices and are connected to each non-virtual cluster to ensure the journey has the starting and ending

points for the gateway trajectory.

The constraints and data freshness limits are used to ensure some conditions are met. These TTL and other constraints are detailed next.

Constraints

– Gateway's journey:

The gateway must start its journey at virtual cluster c_1 and must finish it at virtual cluster $c_{|\mathcal{C}|}$. Moving out of c_1 or into $c_{|\mathcal{C}|}$, has zero cost. This condition is ensured by the following.

$$\sum_{\{c_k \in \mathcal{C}\}} \lambda_{c_j,c_k} - \sum_{\{c_k \in \mathcal{C}\}} \lambda_{c_k,c_j} = \begin{cases} 1 & ,if\ c_j = c_1, \\ -1 & ,if\ c_j = c_{|\mathcal{C}|}, \\ 0 & otherwise, \end{cases} \quad \forall c_j \in \mathcal{C} \tag{5.1}$$

The conservation law is again observed in (5.1) and no gaps are allowed in the circuit.

– Data collection constraints:

$$\rho^i_{c_j} \leq v(c_j, i) \times \sum_{\{c_k \in \mathcal{C}\}} \lambda c_j, c_k, \forall c_j \in \mathcal{C} : \mathcal{I}(c_j) \neq \emptyset, \forall i \in \mathcal{I} \tag{5.2}$$

$$\sum_{\{c \in \mathcal{C}: i \in \mathcal{I}(c)\}} \rho^i_c \leq 1, \forall i \in \mathcal{I} \tag{5.3}$$

Constraints (5.2) ensure that data is collected at clusters that have the IoT device in their transmission range, and are part of gateway's path. Constraints (5.3) state that data can be collected when the gateway is visiting any of the clusters covering it. The main difference in this model is that only one gateway is considered, whilst in the previous MILP-I, multiple gateways were allowed.

– Data delivery constraints:

$$\sum_{\{c_k \in \mathcal{E}\}} \varphi^i_{c_j,c_k} = \rho^i_{c_j}, \forall c_j \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{5.4}$$

$$\sum_{\{c_j \in \mathcal{C}\}} \sum_{\{i \in \mathcal{I}(c_j)\}} \varphi^i_{c_j,c_k} \leq \sum_{\{c_j \in \mathcal{C}\}} \lambda_{c_j,c_k} \times |\mathcal{I}| \times |\mathcal{C}|, \quad \forall c_k \in \mathcal{E} \tag{5.5}$$

Similar to what was used in Chapter 4, Constraints (5.4) ensure that all collected data are delivered at the edge, whereas Constraints (5.5) ensure that data are delivered at edge locations along the gateway's path.

– Data expiration constraints:

When the gateway receives data from an IoT device, it is necessary to ensure that they do not expire until their delivery at the edge storage system. The following restrictions ensure this condition. The objective function guarantees that the data are always as fresh as possible by minimizing the time required for gateways to collect and send data, assuming that lifetime constraints are met.

$$\beta^i_{c_j,c_k} \geq \varphi^i_{c_j,c_k}, \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{5.6}$$

$$\tau^{\mathrm{T}}_{c_k} - \tau^{\mathrm{T}}_{c_j} \leq TTL(i) + \Delta \times (1 - \beta^i_{c_j,c_k}), \quad \forall c_j, c_k \in \mathcal{C}, \forall i \in \mathcal{I}(c_j) \tag{5.7}$$

Constraint (5.6) determines the binary information, noting if cluster c_k supports the delivery of data from $i \in \mathcal{I}(c_j)$, Constraint (5.7) ensures that data are delivered to the edge before becoming outdated.

– Arrival, departure and transmission times constraints:

To ensure that the gateway remains in the cluster for the duration needed to collect or deliver the specified data, it is crucial to determine both the arrival and departure times of the gateway at the center of the cluster. The following conditions must be considered:

$$\tau^{\mathrm{A}}_c \leq \tau^{\mathrm{T}}_c \leq \tau^{\mathrm{D}}_c, \forall c \in \mathcal{C} \tag{5.8}$$

$$\tau^{\mathrm{T}}_c + \mu_c \leq \tau^{\mathrm{D}}_c, \forall c \in \mathcal{C} \tag{5.9}$$

$$\tau_{c_j}^{\mathrm{D}} + t(c_j, c_k) - \Delta \times (1 - \lambda_{c_j,c_k}) \leq \tau_{c_k}^{\mathrm{A}}, \forall c_j, c_k \in \mathcal{C} \tag{5.10}$$

where Δ denotes large values. Constraint (5.8) enforces the inclusion of transmission time windows within the arrival–departure period, whereas Constraint (5.9) incorporates the service time within it. Constraint (5.10) requires that the gateway is not allowed to reach the center of cluster c_k before $\tau_{c_j}^{\mathrm{D}}+t(c_j, c_k)$ when traveling from c_j to c_k. The utilization of Δ enables the fulfillment of the constraints based on the determined value of λ_{c_j,c_k}.

– Visit time constraints:

The duration of a gateway remaining in a cluster must be equal to or greater than the time required to collect and/or deliver the required data. This duration increases when the value of $PRR(.)$ or $NoC(.)$ decreases [57].

$$\mu_{c_j} \geq \sum_{\{i \in \mathcal{I}(c_j)\}} \frac{pk(i) \times sz(i) \times ToA(i, c_j)}{PRR(i, c_j) \times NoC(i, c_j)} \times \\ \times [\rho_{c_j}^i + \sum_{\{c_k \in \mathcal{C}\}} \varphi_{c_k,c_j}^i], \forall c_j \in \mathcal{C} \tag{5.11}$$

Constraint (5.11) includes the spectrum utilization time for both data collection and delivery when determining the service time of the gateway in a certain cluster.

– Final arrival time constraints:

A subsequent constraint is necessary to restrict the final arrival time (i.e., temporal window) for data gathering.

$$\tau_{c_{|\mathcal{C}|}}^{\mathrm{A}} \leq \tau^{\mathrm{MAX}} \tag{5.12}$$

– Non-negative assignment to variables:

$$\lambda_{c_j,c_k}, \beta_{c_j,c_k}^d \in \{0, 1\}; \\ \tau_c^{\mathrm{A}}, \tau_c^{\mathrm{D}}, \tau_c^{\mathrm{T}}, \mu_c \geq 0; 0 \leq \rho_c^i, \varphi_{c_j,c_k}^i \leq 1. \tag{5.13}$$

Objective Function

Lastly, it comes the definition of the objective function of this MILP-II. This is expressed in the maximization of the processed data throughout all nodes in all clusters.

$$\text{Maximize} \sum_{\{i \in \mathcal{I}\}} \sum_{\{c \in \mathcal{C}: d \in \mathcal{I}(c)\}} \rho_c^i \tag{5.14}$$

Several critical points need to be emphasized here. Beginning with the fact that reducing travel and visit times will lead to shortened processing of data, resulting in energy conservation and improved data freshness. A crucial assumption underlying this statement is that the total energy of the traveling gateway is inexhaustible. The SF setting under the most unfavorable conditions, combined with the greatest distance paths traversed by the gateway, are utilized in the maximum time for the simulations (including traveling between clusters). This ensures a well-defined τ^{MAX} that will not exhaust the gateway's battery until its journey is complete. By relying on that, the objective function will encourage the anticipated energy saving. Furthermore, the implementation of TTL non-violation constraints, as in 5.7, guarantees the fulfillment of any stringent data freshness requirement.

5.1.2 The Augmented Advantage Pointer Critic Model (A2PC)

In reinforcement learning, an agent learns to interact with an environment by observing it and responding with reactions. Accordingly, the environment state changes and the agent receives a reward (an outcome). Given a predefined rewarding system, the objective of the agent is to accumulate up to the highest possible outcome G over the interactions:

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_0 = s_t\right], \tag{5.15}$$

where $r(s_t, a_t)$ is the reward for taking action a_t in state s_t at time t. The leading coefficient γ is a discount factor to balance short and long-term rewards.

For the agent to interact efficiently with the environment, it needs to follow a policy π to determine which action to be taken at a given state. A policy is optimal π^* when following it produces the maximum G.

There are two main ways to learn a policy, value-based and policy-gradient learning. The current agent belongs to the latter.

In value-based learning, the policy is learned from the value function $V(s)$, which measures how good is it to be at state s:

$$V(s) = \mathbb{E}\left[G_t | s_t = s\right], \tag{5.16}$$

and to measure how good is it to take action a at state s, the quality function (Q-function) can be used:

$$Q(s, a) = \mathbb{E}\left[G_t | s_t = s, a_t = a\right] \tag{5.17}$$

Furthermore, to obtain a relative importance of an action a and determine if choosing a is better than the average performance of the policy we can use the advantage function:

$$A(s,a) = Q(s,a) - V(s), \tag{5.18}$$

which can be approximated as:

$$A(s_t,a_t) = r(s_t,a_t) + \gamma V(s_{t+1}) - V(s_t) \tag{5.19}$$

In policy gradient learning, the policy is represented as a parametric function where the objective is to learn the set of parameters that will maximize the outcome. Let Λ be defined as a trajectory (sequence of transitions) such that $\Lambda = s_0, a_0, \ldots, s_{T-1}, a_{T-1}, s_T$, for an episode with length T. Using the probability general product rule, π can be defined as:

$$\pi(\Lambda;\theta) = p(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t;\theta)p(s_{t+1}|s_t,a_t) \tag{5.20}$$

where p is a predefined transition probability function, and θ is the set of unknown parameters which can be estimated by maximizing an objective function J:

$$\theta^* = \arg\max J(\theta), \tag{5.21}$$

where J can be formulated as:

$$J(\theta) = E_{\Lambda\sim\pi(\Lambda;\theta)}r(\Lambda) = \int \pi(\Lambda;\theta)r(\Lambda), \tag{5.22}$$

following the logarithmic differentiation, we have:

$$\nabla_\theta J(\theta) = E_{\Lambda\sim\pi(\Lambda;\theta)}\left[\sum_{t=0}^{T-1}\nabla\log(\pi(a_t|s_t;\theta))G_t\right] \tag{5.23}$$

By applying the gradient ascent rule with a predefined learning rate α we have:

$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J(\theta) \tag{5.24}$$

The above three steps of: sampling a trajectory $\Lambda \sim \pi(\Lambda;\theta)$, (5.23), and (5.24) is known as the REINFORCE algorithm which will be used in training the model in this work. With these basis in mind, the proposed model is described next.

State

The state s_t at time t is a vector of five features:

$$s_t = [s_t^{\mathrm{P}}, t, s_t^{\mathrm{B}}, TTE, s_t^{\mathrm{RB}}], \tag{5.25}$$

where:

- s_t^{P}: is the current position of the gateway.
- t: is the elapsed time so far.
- s_t^{B}: is the amount of processed data up to that point in time.
- TTE: is the time to expire of the data in transit.
- s_t^{RB}: is the amount of the remaining data to be processed.

Action Space

This agent deals with two heterogeneous action spaces. The first one specifies where the gateway is moving to (location). While the second space represents the visit time associated with the selected move.

It is important to note that during the problem formulation, the LoRaWAN devices are grouped into clusters according to their spatial distributions. Hence, the first action corresponds to a cluster number:

$$\mathcal{A}^C = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}, \tag{5.26}$$

where c_1 and $c_{|\mathcal{C}|}$ are two virtual clusters (also included in the mathematical formulation).

This is done by considering real-world scenarios, like in Chapter 4, where the circuit trajectory's upper limit reaches 10800 seconds. Following that, the timestep conversion is done by stipulating units of 5s. In other words, each timestep is equivalent to 5s, which is valid since this arbitrary value can accommodate the transmission and reception of up to 51 bytes. These represent the largest messages at the worst case of LoRa spreading factors [76]. Such a time unit is also adopted in Chapter 4, and using the same parametrization makes sense to compare the solutions. Yet, respecting real-word scales and the above considerations, this action space is defined by:

$$\mathcal{A}^T = \{1, 20, ..., 160, 180\}, \tag{5.27}$$

where 1 time step = 5s. Visit times above 180 timesteps fall out of the range of real and suitable time periods.

Reward

Developing a reward function is one of the main challenges in RL. In this agent, it is even more challenging since there are two action spaces (location and visit time).

The reward obtained in every interaction is proportional to the collected or delivered bytes. This way, it is possible to filter out fruitless moves:

$$r(s_t, s_t^{\mathrm{P}}, v_t) = \begin{cases} s_t^{\mathrm{B}} - s_{t-1}^{\mathrm{B}}, \text{ if } \quad s_t^{\mathrm{P}} \notin \mathcal{E} \text{ and byte collected.} \\ s_t^{\mathrm{B}}/v_t, \text{ if } \quad s_t^{\mathrm{P}} \in \mathcal{E}, \text{ and } TTE > 0 \end{cases} \tag{5.28}$$

where TTE is the time to expire of the packets (LoRa messages), which decreases as time elapses. This value resets on every edge visit and it is assumed to start with a well-known initial and uniform value (i.e., the TTL) for all nodes.

5.1.3 A2PC Architecture

The size of the action space can be a bottleneck while developing and/or deploying DRL solutions. DRL models are usually static in terms of state and action spaces, meaning that the agent state-action dimensions are usually hard-coded and any changes in the action space are hard to incorporate without retraining. In realistic scenarios, retraining is expensively prohibitive. A typical example of a change in the action space is when one of the actions is not feasible or a new cluster is added/reactivated in the network.

Another issue is the complexity when additional decision is required. This is the case here, where not only the location of the gateway should be determined, but also the visit time. In such situations, there are two fundamental challenges. First, it is not easy to learn a high dimensional action space, since the distribution over actions (i.e, the policy) will be complex. Second, finding a suitable architecture and loss function is not trivial. To tackle both challenges, an augmented pointer-critic architecture and extended Advantage Actor-Critic (A2C) method is developed, like in [53], to incorporate the newly proposed architecture.

The Ptr-Net architecture can be used to represent variable-length dictionaries, which has been applied successfully for solving combinatorial problems, such as the Salesman Traveler Problem (STP) [54, 49]. Typically, in a Ptr-Net, the input sequence of elements is encoded into a representation space, where the decoder can generate pointers (indexes) to the input (one pointer at a time) [71, 78]. Such pointers will select available elements

from the input sequence until a predefined condition is met (e.g., an end of sequence token is generated).

Furthermore, the masked attention mechanism can be used to avoid non-repetitive actions, which makes sense in this agent since the gateway is intended to move to a different position on every new step. This whole process, as depicted in Figure 5.1, is comprised of pointers that select feasible tokens filtered by the mechanisms present in each element of the output sequence.

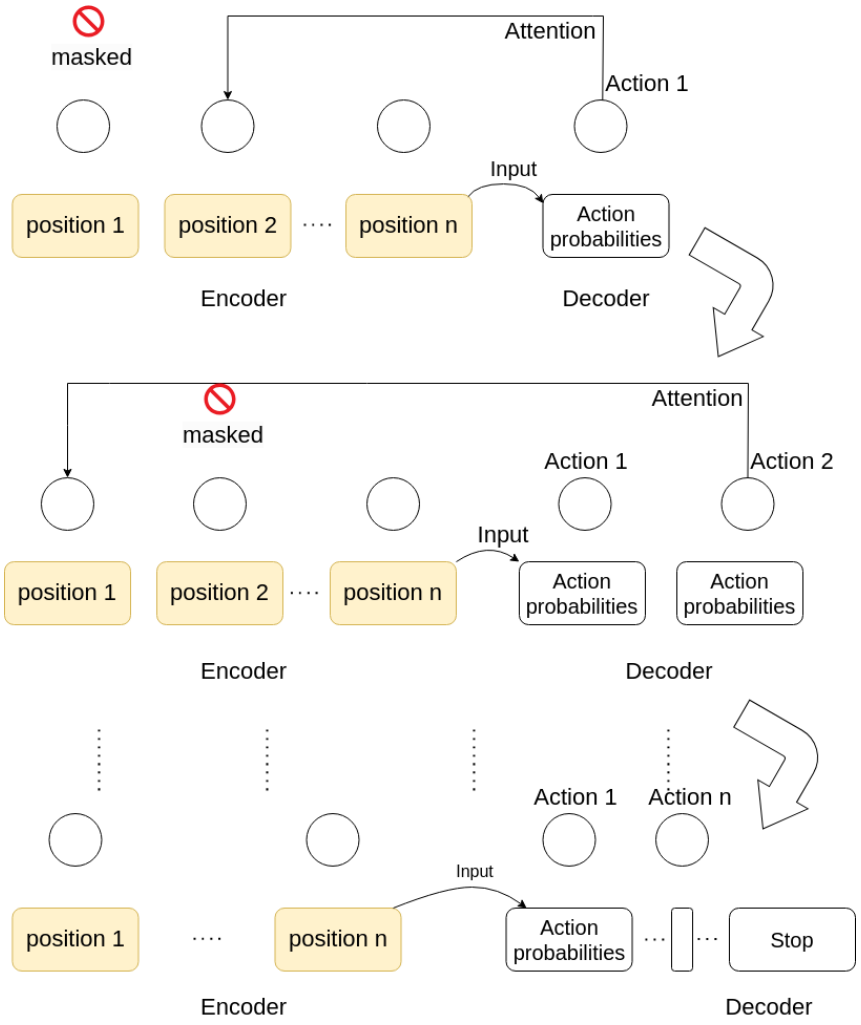

Figure 5.1: Masked attention concept in Ptr-Net.

5.1.4 The Action-Branching (Augmented) Model

The gateway travel planning problem revolves around selecting the positions and visit times. The dimension of the action space can grow fast depending on the scenarios. The combination of moves and visit time steps can't easily be achieved with a single output since they are fundamentally two different spaces. Therefore, two different heads (location and visit time heads) are used as shown in Figure 5.2. The location is selected

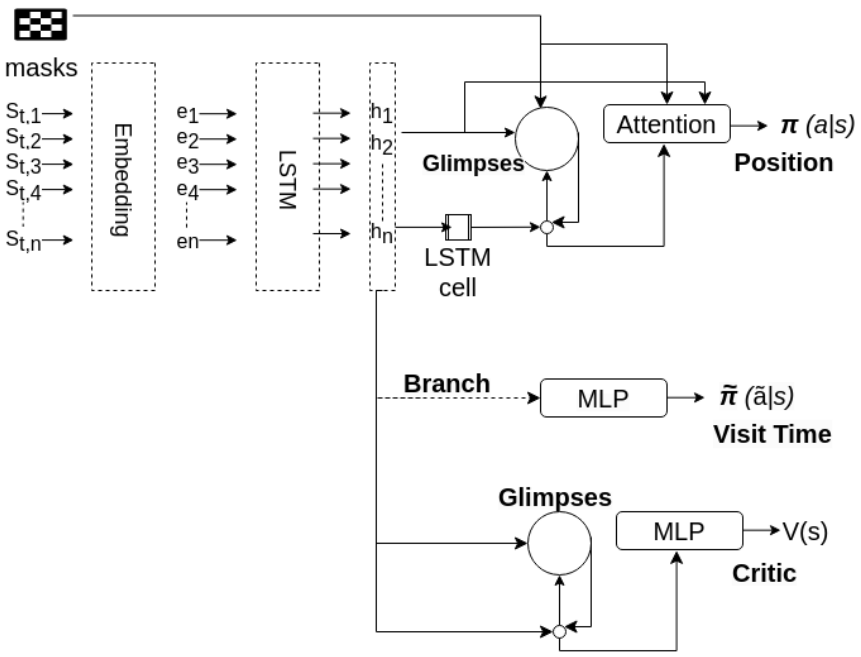

Figure 5.2: Action-branching high level diagram.

by a pointer head, which points back to the index of the position in the input state. On the other hand, the visit time decision is made by another head.

As shown in this scheme, a single baseline critic evaluates the policies that guide both actors (heads). The idea of using a single critic has a couple of advantages. It keeps the model simple, efficient, and easy to extend at the same time. Allowing the critic to capture the behavior of both actors makes it simple for the critic to focus on a common objective. Furthermore, the critic branch in the model estimates the value function which can be used in the losses of both actors.

The Figure 5.3 depicts the visit time head as a Multi-layer Perceptron (MLP) that is directly branched from the shared LSTM of the model. Rather than using a full copy of the location head actor, this more straightforward design has three advantages: i) visit times sequences do not benefit from the action masking as they may have repeated sequences of values; ii) the visit times sequence does not benefit from the pointer head as their prediction sequences do not point back to the input; and iii) as a design choice, the visit time head is lightweight and does not overcomplicate the existing Ptr-Net model.

5.1.5 Loss Functions

In order to incorporate the loss from the location, visit time, and critic's heads of the model, the A2C loss is augmented. This goes through the addition of a second term, corresponding to the second actor, in all A2C loss formulas, like the following:

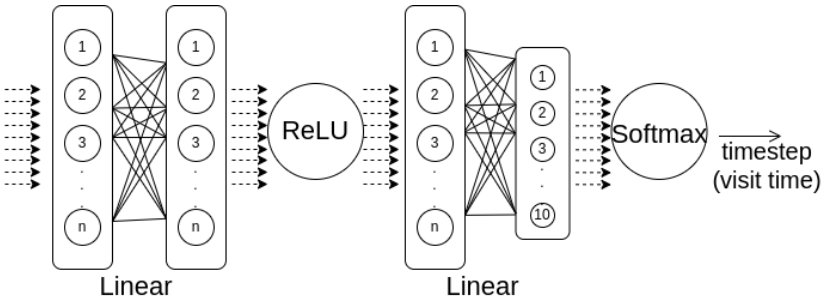

Figure 5.3: MLP (visit time) branch diagram.

- actor position loss:

$$-\sum_{t=0}^{T-1} log(\pi(a_t|s_t:\theta))A(s_t,a_t) \tag{5.29}$$

- actor visit time loss:

$$-\sum_{t=0}^{T-1} log(\tilde{\pi}(\tilde{a}_t|s_t:\theta))A(s_t,\tilde{a}_t), \tag{5.30}$$

  where $\tilde{\pi}$ is estimated from the visit-time actor.

- critic loss:

$$\frac{1}{2}\sum_{t=0}^{T-1} A(s_t,a_t)^2 + \frac{1}{2}\sum_{t=0}^{T-1} A(s_t,\tilde{a}_t)^2 \tag{5.31}$$

- entropy loss:

$$-\left\{\sum_{t=0}^{T-1} \pi(a_t|s_t)log(\pi(a_t|s_t)) + \sum_{t=0}^{T-1} \tilde{\pi}(\tilde{a}_t|s_t)log(\tilde{\pi}(\tilde{a}_t|s_t))\right\} \tag{5.32}$$

where $a \in \mathcal{A}^C$ and $\tilde{a} \in \mathcal{A}^T$,

5.1.6 A2PC Training

The proposed model is trained using the A2PC algorithm, which is represented in the Algorithm 2.

5.2 Performance Evaluation

In order to assess the proposed model this section outlines the evaluation process, explaining KPIs, scenarios setup, the convergence, and implementation details. With

input: T = episode length, α_{entropy} = entropy factor
output: Trained A2PC model

Initialize actors $\pi_\theta, \tilde{\pi_\psi}$ network parameters;
Initialize critic V_ϕ network parameters;
Initialize state s_0;

//sample the entire episode returning the
//list of state, actions, and reward
$s, a, \tilde{a}, r \leftarrow SampleEpisode(s_0, V_\phi, \pi_\theta, \tilde{\pi}_\psi, T)$;

$E \leftarrow ComputeEntropy(\pi_\theta, \tilde{\pi}_\psi)$;

$g_{\theta,\psi} \leftarrow \sum_{t=0}^{\mathrm{T}-1} A(s_t, a_t)\nabla_\theta log(\pi(a_t|s_t)) + A(s_t, \tilde{a}_t)\nabla_\psi log(\tilde{\pi}(\tilde{a}_t|s_t)) + \alpha_{\text{entropy}} E$;
$L_\phi \leftarrow \frac{1}{2}\sum_{t=0}^{\mathrm{T}-1} A(s_t, a_t)^2 + \frac{1}{2}\sum_{t=0}^{\mathrm{T}-1} A(s_t, \tilde{a}_t)^2$;

$\theta, \psi, \phi \leftarrow ADAM(\{\theta, \psi, \phi\}, (g_{\theta,\psi}, \nabla_\phi L))$;

Return trained actors (location and time) neural networks.

Algorithm 2: Training phase of the A2PC agent.

all those established, the model is evaluated and the results are presented for each KPI benchmark.

Similar to what is done in the PoC, the A2PC relies on a framework based on the NS3-LoRa simulator, CPLEX, and OpenAI Gym. Its source code is publicly available at [77].

KPIs

In this assessment there are two main KPIs. One is the amount of data being processed within an arbitrary travel duration, τ^{MAX}, and the other is the scalability of the DRL solution.

For the first KPI, the so-called arbitrary travel duration is obtained from the framework Step 1. That is assumed to be half the most extended period to process all data in all clusters. This predefined period ensures that the optimization upper bound will never reach 100% of all packets generated by the data collection simulations. Moreover, if long-lasting periods were used, then the A2PC agent and the MILP-II could process nearly all data available, thus defeating the idea of comparing one to another. The constrained time forces the A2PC agent and the MILP-II to determine the maximum data processed within that arbitrary time frame. This establishes a tangible limit that enables comparison between the two and follows the empirical deployment requirements.

The solution's scalability is the second KPI. This KPI is rarely addressed in the literature, but in here it is used to evaluate the capability of A2PC to scale under similar network dynamics. This refers to situations when the number of sensor nodes changes.

Information	Value
No. Clusters	8 or 16
No. Edges*	2 or 4
Nodes Distribution	Sparse or Dense
TTL	1200s
Message Avg Size	18 bytes
No. Messages per Node	1-5
τ_{max}	Half of Simulation Time

*Max. no. edges to cluster size ratio: 25%.

Table 5.1: A2PC simulation setup information.

Outside the KPIs, a third metric stands out with relevant influence is the visit time. Its pattern helps to understand the demands for the different scenarios. Given such an influence, this metric is also discussed during the analysis of the results.

5.2.1 Scenarios Setup

The configurations used in the data collection step follow the standard provided by the simulator. This is done because LoRaWAN-NS3 can represent the LoRa real-world systems accurately [64]. As a matter of validating the performance of A2PC in scenarios that scale in size (number of nodes/clusters) and volume (amount of data), same empirical conventions as the ones used in Chapter 4 are adopted. Therefore, the types of scenario continue to use the very same convention observed when experimenting the PoC GMwES as in 4.2. Still, the number of edge clusters is defined as being 25% of the total clusters. This ratio provides a more realistic proportion between edge and node clusters.

The moving gateway is assumed to maintain a constant speed of 60 km/h, and the variance in the visit times will be responsible for compensating any need for delay or acceleration during the journey. The average distance between any two clusters in the graph is 5.8 kilometers. In large scenarios, the maximum duration for the gateway's journey can reach 10800 *s* (5400 *s* being the half of the duration employed in the simulations), while in small scenarios it can reach up to 5300 *s*. These assumptions are all based upon real-world case specifications and they are corroborated by data published in [79, 80]. The Figure 5.4 displays a sample of a SD scenario, providing a more comprehensive illustration of how variations in size and density are characterized in the experiments. The dots represent the LoRaWAN nodes, while the circle areas represent the respective clusters.

In addition, the number of messages and sizes follow common patterns found in real scenarios [81, 82, 83]. The summary of these scenarios and the adopted classifications being are listed in Table 5.1

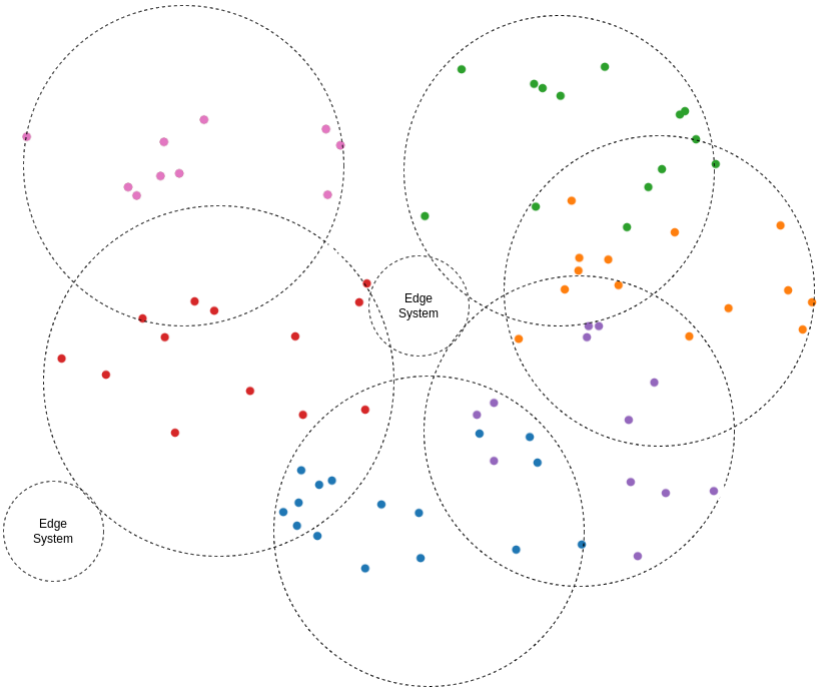

Figure 5.4: Small & Dense (SD) scenario sample.

A2PC Convergence

The learning process may be affected by the type of the scenario or topology. The convergence analysis can be performed based on the mean accumulated reward. Figure 5.5 shows the distribution of the number of episodes until the reward stabilizes. A preview set of 20 exploratory experiments is executed for each type of scenario. From those results, it is possible to notice that the convergence happens around 7k episodes for the small scenarios. In contrast, the same happens with about 14k episodes for the large ones. The results are better synthesized by the reward curves in Figure 5.6, which depicts the average of episodes required to convergence for each type of scenario.

Implementation Details

The A2PC is implemented in Python 3.8, Pytorch 1.10 [77], and Open-AI Gym v0.26.0. The model runs on Ubuntu 20.02 host systems. The hardware comprises 8 vCPU Intel i7, 16 GB RAM, and Nvidia GeForce RTX 2080 GPU GV102 with 11264 MB.

Lastly, the parametrization of A2PC follows the assumptions devised in Section 5.2.1 and Section 5.1.4. Hyper-parameters are also tuned to configurations which proved to be functional in the trials and initial experiments. The summary of these parameters is compiled in Table 5.2.

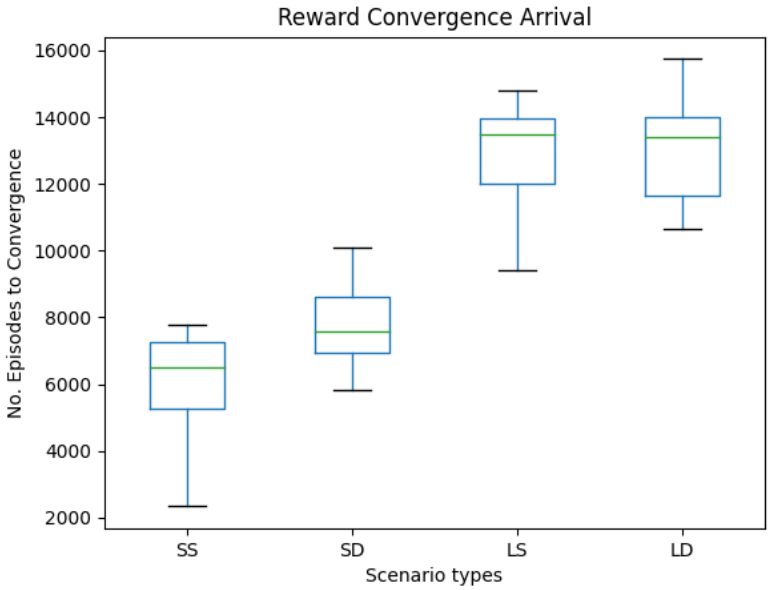

Figure 5.5: Reward convergence episodes for the different scenarios.

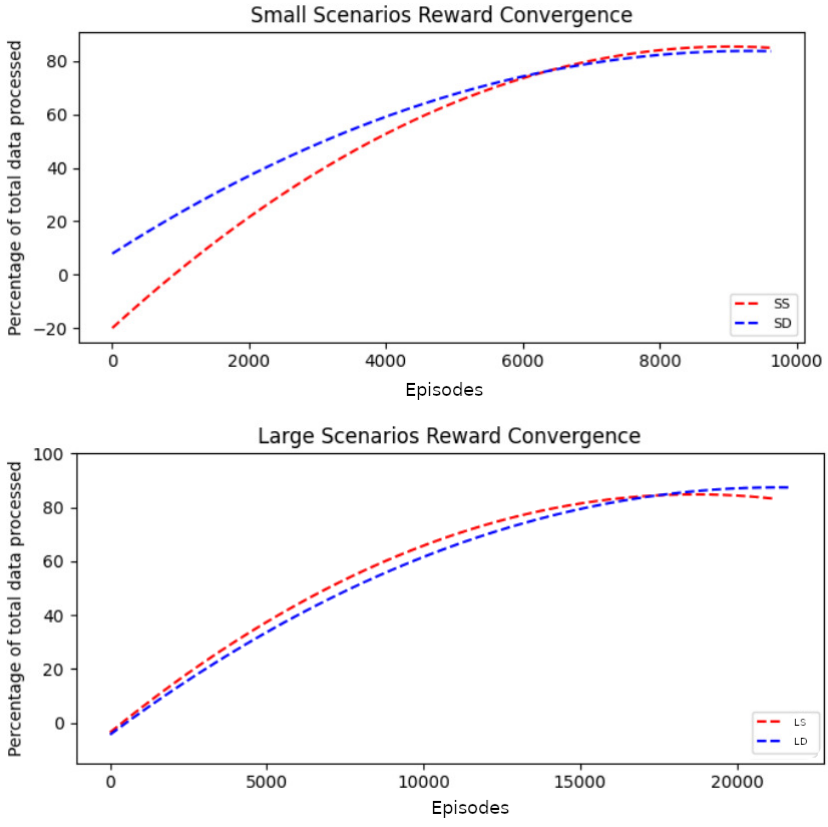

Figure 5.6: Avg. reward convergence curve for different scenarios.

5.2.2 Analysis of Results

In this section, the results for the KPIs and metrics are presented and discussed as follows.

Parameter	A2PC
Discount Factor γ	0.9
Learning Rate α	0.001
Entropy Factor ϵ	0.001
Training Episodes	7000 (small) & 14000 (large)

Table 5.2: A2PC agent hyperparameters.

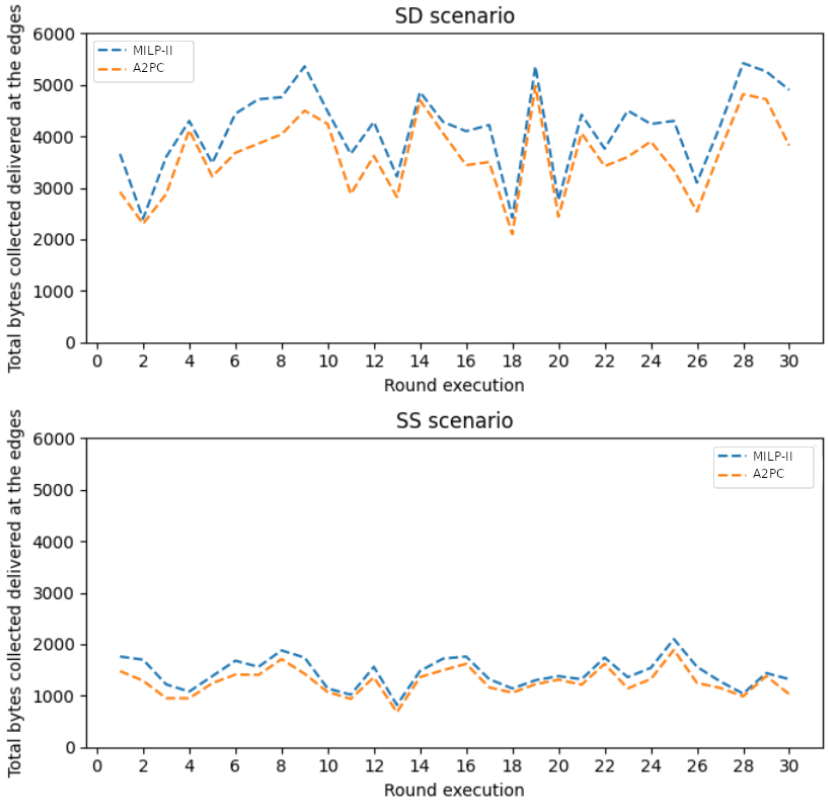

Figure 5.7: Bytes processed by A2PC vs optimal values - Small clusters.

5.2.3 Summary of Results

Processed data

After running 30 folds for each scenario, the results obtained by A2PC are illustrated in Figure 5.7 and Figure 5.8, respectively for small and large clusters.

In these figures, the plots show the number of bytes processed by the model versus the optimal amount of the data achieved by the MILP-II. It is clear that A2PC performance keeps a near-optimal results in all folds. Furthermore, in small clusters A2PC deals with less traffic overlapping and the range of actions to be taken is smaller than the large clusters scenarios. Therefore, a slight better performance is noticed when compared to large clusters.

When considering the total number of bytes processed in all rounds of execution, the analysis reveals that the A2PC system was unable to process 16,220 out of a possible

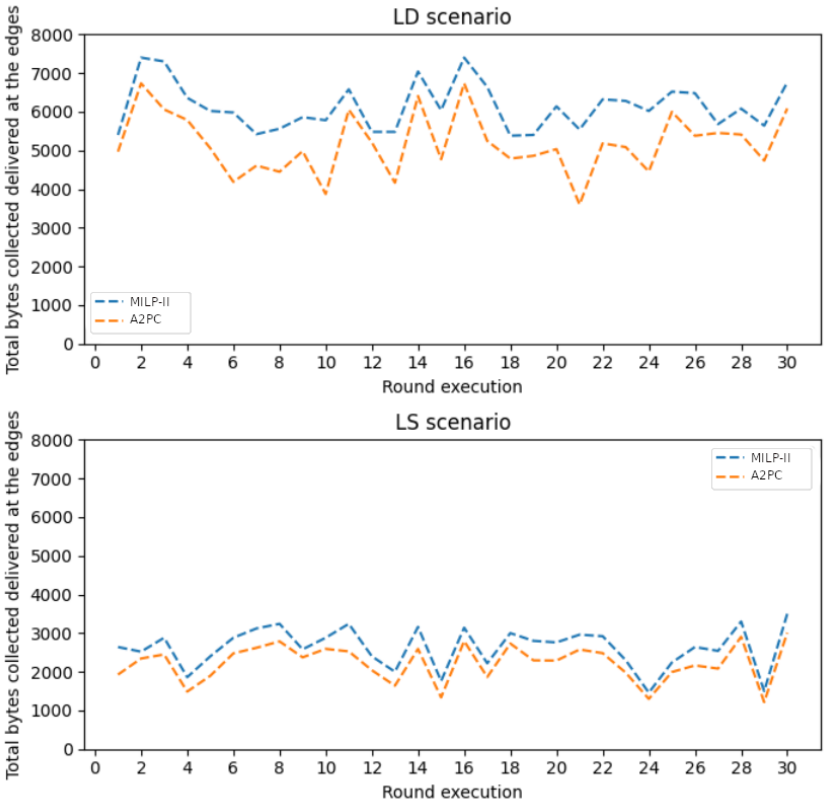

Figure 5.8: Bytes processed by A2PC vs optimal values - Large clusters.

Scenario	Total Bytes Available	Total Bytes Processed	Total Bytes Not Processed
SS	43340	38127	5213
SD	124460	108240	16220
LS	78840	66760	12080
LD	183980	155342	28638

Table 5.3: A2PC total data processed.

124,460 bytes for the SD scenarios. Similarly, for the LD scenarios, it was unable to process 28,638 out of a total of 183,980 bytes. These numbers support the idea that only a very small portion of the available data remains unprocessed, even in situations with a high density of data. The detailed absolute comparisons can be found in Table 5.3.

Furthermore, the averaged results of A2PC compared to the upper bound are listed in Table 5.4, highlighting that A2PC is consistent in terms of sparsity and a negligible drop can be noticed in dense scenario which confirms the A2PC efficiency regardless of the scenario type.

Scalability

Additional experiments are set using successful DRL trained models to inspect the agent's behavior under scaled-out or downscaled scenarios. For the expansion, the con-

Scenario	Percentage of Total Data Processed
SS	85.29%
SD	84.35%
LS	81.77%
LD	81.55%

Table 5.4: A2PC agent average data processed.

cept is based on cloning existing nodes, increasing the number of nodes inside the clusters. By doing so, the agent is expected to anticipate the pattern of communications, thus performing as well as the original environments. The scalability tests are based on SS clusters because this scenario produced the best overall results, as shown in Table 5.4. Thus, a random sample of 15 SS clusters is selected from the executed rounds, from 6.2.3, to serve as a baseline. Those scenarios were manipulated with the introduction of randomly cloned nodes. This cloning change was made by adding 1, 5, and 10 nodes to all these selected scenarios. A comparison between the baseline scenarios versus the scaled ones is done by the average performance difference between them. This means that a performance of 84% of bytes processed in a baseline execution, versus a performance of 84%, in the scaled out scenario, would remark 0%. Similarly, the reduced size of the clusters was tested following the same logic. A sample of 15 SS selected clusters is submitted to node removals at random. This withdrawal is done in steps of 1, 5, and 10. The execution of the same trained A2PC agents on the baseline and modified scenarios yields performance results that can be compared in a delta fashion.

In Figure 5.9, the set of the normalized Cumulative Distribution Function (CDF) graph depicts the performance divergence obtained by the execution of these agents. The values in the individual lines denote the percentage difference distribution in the agent's performance in the scaled scenario in relation to the baseline one. The positive values indicate that the agents outperformed the baseline scenarios, which is more prevalent in the scaled-out scenarios. On the other hand, the distribution starts increasing in negative incidences for the scenarios where nodes were removed. This is more tangible in the scenarios where 5 and 10 nodes were subtracted. That CDF appears reasonable, as the A2PC agents may have visited - according to previous learning - clusters with absent nodes, thus decreasing the overall performance.

Furthermore, it is possible to imply that scaled-out environments are not an issue for the trained models. Since they preserve similar LoRa communications characteristics, the agents can meet the behavior patterns by collecting and delivering the same, and often slightly more than the original scenarios.

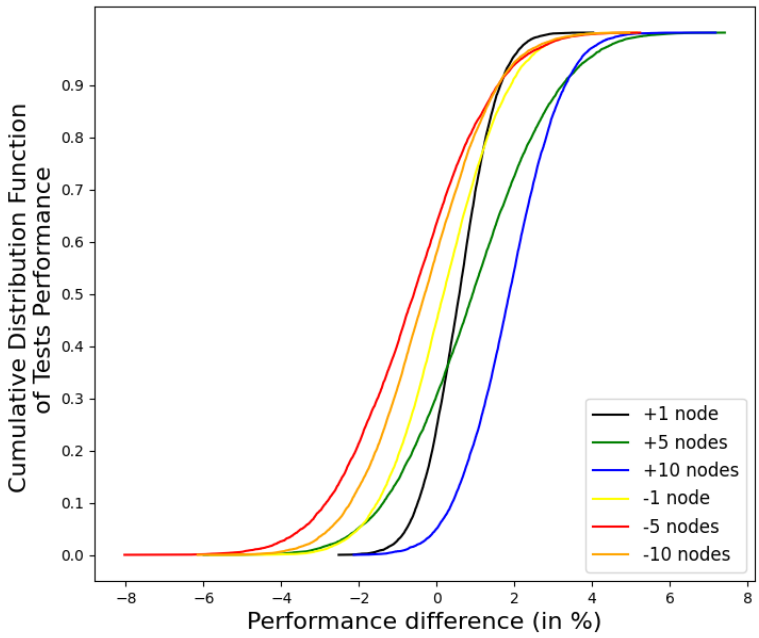

Figure 5.9: Variation of performance obtained by scaled scenarios in comparison with original scenarios.

Visit time

A technique that adds a singular characteristic to the A2PC is the action-branching method. Therefore it is plausible that its usage and impact are analyzed with more attention. Here, the samples of the visit-time actions learned by the A2PC (during the 30 pipeline executions) are compiled to provide an overview of its behavior and patterns. The Figure 5.10 summarizes the number of occurrences for each one of the timestep options available, as in Definition (5.27), in all trained scenarios.

There is a distinction between large and small scenarios. The A2PC agents tend to concentrate their visit times in the low part of the timesteps range for the large scenarios while distributing across for the small ones. Yet, the small ones also have a higher incidence of the high end of the timestep range. This outcome indicates that the A2PC can learn more rewarding behaviors according to the different topologies. Furthermore, the influential factor to this pattern is the size of the clusters rather than the node density.

5.3 A2PC Conclusions

After closing the tests with the A2PC, it is possible to conclude the proposal's success. The learning process for the multidimensional action space domains has demonstrated a significant improvement in the agent's performance in terms of throughput. This process

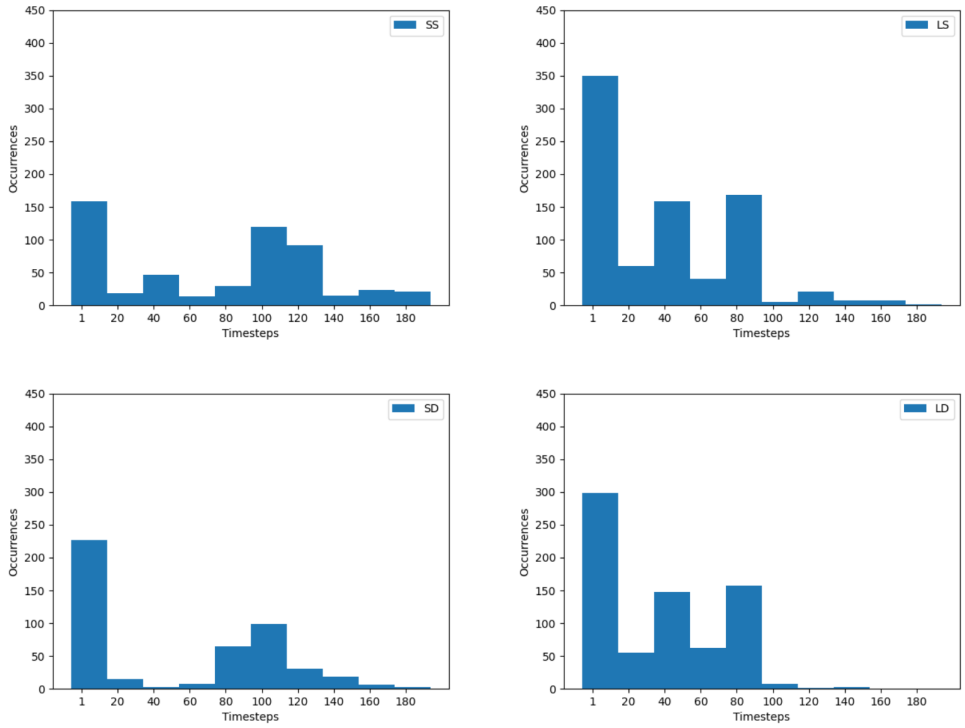

Figure 5.10: Timesteps occurrences for all scenarios.

has also been legitimated by the patterns captured when analyzing the distributions of visit times. Overall, the performance improvements are proved by the numbers accrued in the experiments. Still, and beyond that, it is possible to highlight the agent's actual scalability capability, as the same trained model has regularly performed for different sizes of the same problem. Next, a direct comparison between GMwES and A2PC is carried out to consolidate this analysis.

CHAPTER 6

Comparison of A2PC and GMwES Agents

6.1 Introduction

As a final and conclusive analysis, a direct comparison between the 2 developed DRL agents is performed in this chapter. The scenarios and parametrization of the experiments follow the same specifications and conventions already utilized in the Chapters 4 and 5.

Similar to what was done previously, the experiments start with executing NS3 simulations with the support of Python scripts [77]. This step generates the single gateway scenario configurations, including the amount of data for being processed. After completing the aforementioned task, the A2PC and GMwES agents can undergo training using the identical conditions configured in the simulations. The results produced by these agents are compared in terms of bytes processed.

Contributions

- Rodrigo Carvalho, Faroq Al-Tam and Noélia Correia, "Mobility Planning of Edge-Assisted IoT Systems Using DRL Action-Branching Approach", 2023 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS), 2023.

6.2 Assessment of the Agents

In terms of hyper-parameters, the GMwES agent continues to use all those established in Chapter 4, whilst A2PC adheres to the assumptions outlined in Chapter 5. The upper limit for the duration of travel for each gateway journey, specifically the training and test time, is determined to be half of the time intervals achieved by the NS3 simulations. The establishment of an arbitrary predetermined time frame guarantees that neither agent will encounter a scenario in which they can acquire 100% of the packets generated by the simulations. If a significant amount of time is allocated, there is a potential risk that the objective of comparing the performances of both agents may be undermined by the fact that they manage processing all the data available.

6.2.1 Scenarios Setup

The experiments adhere to the same conventions for cluster sizing and distributions that were previously employed. Therefore, the four types of scenarios are again: Small Sparse (SS), Small Dense (SD), Large Sparse (LS), and Large Dense (LD). For the specific instance of the GMwES, only executions that utilize a single gateway are taken into account. By implementing this approach, both agents would be subjected to identical conditions and standards, thereby facilitating an equitable and impartial comparison.

6.2.2 Analysis of Results

After running 30 folds for each scenario, the results obtained by A2PC versus the GMwES are illustrated in Figure 6.1 and Figure 6.2, respectively for small and large clusters. The summarized numbers are laid out in the following subsections.

6.2.3 Summary of Results

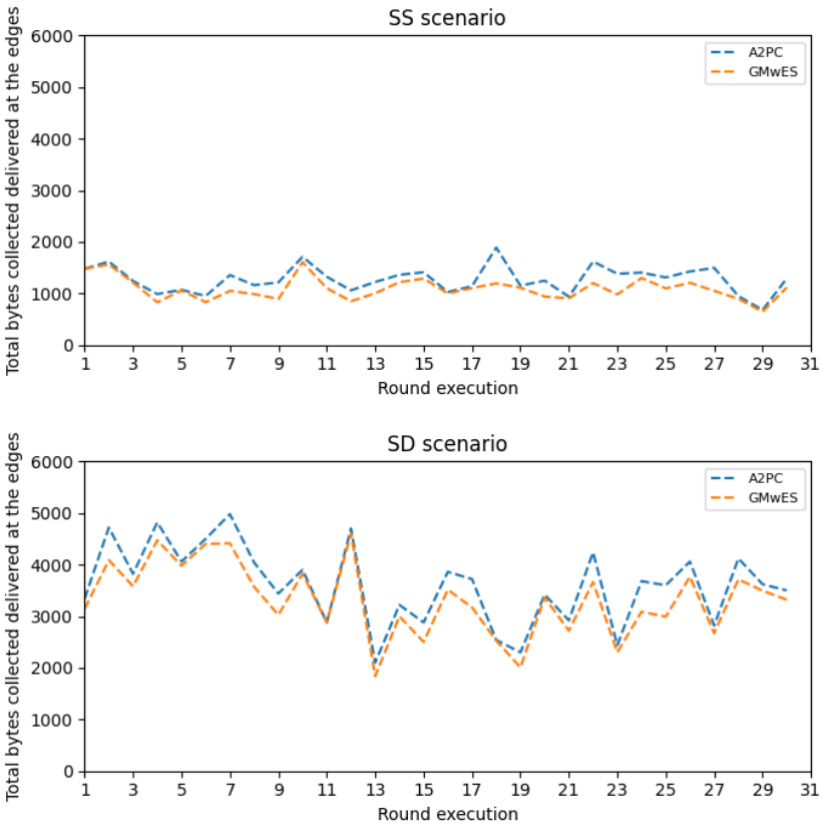

Figure 6.1: Bytes processed per agent - Small clusters.

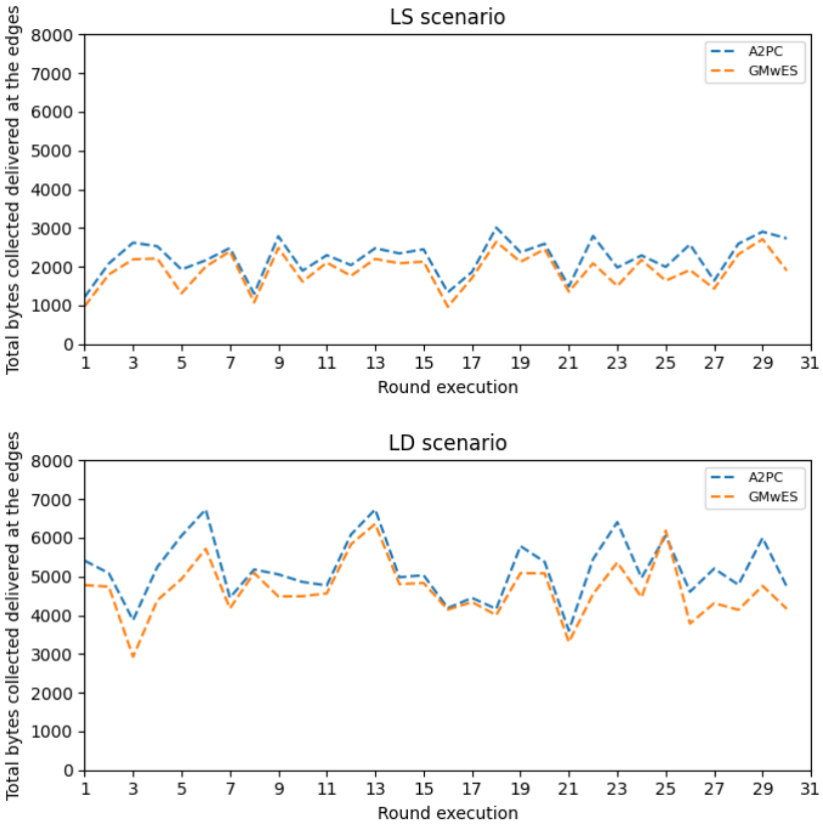

Figure 6.2: Bytes processed per agent - Large clusters.

The presented Figure 6.1, and Figure 6.2 illustrate that, based on the bytes processed metric, the performance of A2PC remains consistently near-optimal across all folds. Furthermore, it consistently outperforms GMwES. This average advantage is better summarized in the Table 6.1.

Scenario	A2PC outperformance*
SS	3.93%
SD	3.79%
LS	4.81%
LD	4.32%

*Measured in percentage.

Table 6.1: Average performance improvement from A2PC in relation to GMwES.

In a nutshell, the advantage of the A2PC approach becomes more apparent in low-density clusters, where accuracy in multiple dimensions is easier to achieve. Nonetheless, the improved efficiency over the conventional DRL GMwES is visible across all scenarios.

6.3 A2PC and GMwES Assessment Conclusions

The basis of the A2PC solution is centered around an AC agent that incorporates action-branching and Ptr-Nets. To that extent, this agent demonstrates improved performance upon less training demands if compared to prior DRL agents, such as the GMwES.

In addition to establishing the concept and confirming the effectiveness of the framework, the findings presented in this chapter highlight the significant capabilities of neural network branching, particularly in scenarios involving a high number of dimensions and combinations. The primary benefit of this approach includes the capacity to employ the model across various scales and the ability to approximate solutions that are near-optimal while utilizing reasonable computational resources.

CHAPTER 7

Conclusions

This work aims to describe a novel architecture that deals with the problem of mobility planning for a LoRaWAN mobile gateway with edge systems support. This study has established a framework comprising DRL agents and mathematical models to improve data processing while also considering data expiration restrictions. The framework is conceived in a pipeline format that, ultimately, generates the outcomes used to assess the effectiveness and achievement of the suggested solution as a whole. The following section summarizes the conclusions and inferences that may be drawn from this proposal. The last section briefly highlights additional considerations, including possible expansions of this work and its potential continuations.

7.1 Framework Results

The RL-ADR agent in Chapter 3 outperforms the usual ADR algorithm, making it an alternative option for LoRa implementations in real-world use cases or simulations. In this thesis, the usage of RL-ADR with NS3 was crucial because it aided in the establishment of the experimental scenarios as well as the DRL agent settings.

In turn, the mathematical models MILP-I and MILP-II were created from scratch in this work and have significantly contributed to establishing KPIs and metrics. More precisely, the MILPs' findings enabled developing reliable criteria for assessing the agents' efficiency. The biggest disadvantages of the MILPs are that they might become unfeasible when scenarios get too big, and they require delineating constraints expressions, which can make it hard to describe reality accurately.

Finally, DRL agents, GMwES, and A2PC have been presented as the key problem solvers. The former is considered a proof of concept given that it only analyzes one dimension of the issue: The visit position. Nonetheless, the GMwES can capture the key environmental patterns and translate them into a set of nearly ideal actions in terms of performance. A2PC is the ultimate update to the DRL method, and it utilizes cutting-edge approaches to bring improved performance and capabilities to the problem-solving process. Although it is only built for a single gateway, the findings indicate that using

A2C agents, Ptr-Nets, and branching may achieve near-optimal results with the desired scalability characteristics. Overall, the outcomes reported in this thesis prove the idea, validate the framework's usefulness, and highlight the significant potential of neural network branching.

7.2 Future Work

Ultimately, future research should focus on expanding the scope of study to encompass more complex situations. This might include building A2PC multi-agent systems that operate across several gateways or address cluster overlaps. Another front to explore in the A2PC improvement would be utilizing different action dimensions (e.g., time, battery level, transmission settings, etc.) that may have a more influential impact, depending on the desired objective.

At last, the framework's success encourages its use in different technologies and ecosystems. For example, the same framework may be slightly adapted to run similar cases on any star topology network. Furthermore, other IoT platform technologies, like Sigfox [84], could be a promising path for leveraging the positive outcomes of this work.

References

[1] Mohamed K. S., The Era of Internet of Things. ISBN : 978-3-030-18132-1, 2019.

[2] Jia X., Feng Q., Fan T., and Lei Q., RFID technology and its applications in Internet of Things (IoT), Second International Conference on Consumer Electronics, Communications and Networks (CECNet), 2012

[3] Sanchez G. J., Parnell G. S., Specking E., Pohl E. A., and Buchanan R., Smart cities — A structured literature review, Smart Cities, vol 6, 2023.

[4] Rajesh G., IoT: Ongoing challenges and opportunities in mobile technology, International Journal of Electrical, Electronics and Computers, vol 6, 2021.

[5] Wazir Z. K., Ahmed E., Hakak S., Yaqoob I., and Ahmed A., Edge computing: A survey, Future Generation Computer Systems, vol 97, 2019.

[6] Kong L., Tan J., Huang J., Chen G., Wang S., Jin X., Zeng P., Khan M., and Das S. K., Edge-computing-driven Internet of Things: A survey, ACM Computing Surveys, vol 55, 2023.

[7] Sun Z., Yang H., Liu K., Yin Z., Li Z., and Xu W., Recent advances in LoRa: A comprehensive survey, ACM Transactions on Sensor Networks, vol 18, 2022.

[8] Kufakunesu R., Hancke G. P., and Abu-Mahfouz A. M., A Survey on adaptive data rate optimization in LoRaWAN: Recent solutions and major challenges, Sensors, 2020.

[9] Schulz J., Applying mathematical optimization in practice, Operations Research Forum, vol 2, 2021.

[10] Liu F., Tang G., Li Y., Cai Z., Zhang X., and Zhou T., A Survey on edge computing systems and tools, In Proceedings of the IEEE, vol 107, 2019.

[11] Lu J., Gong P., Ye J., Zhang J., and Zhang C., A survey on machine learning from few samples, Pattern Recognition, vol 139, 2023.

[12] Dhal P., and Azad C., A comprehensive survey on feature selection in the various fields of machine learning, Applied Intelligence, vol 52, 2022.

[13] Perazzone J., Dwyer M., Chan K., Anderson C., and Brown S., Enabling machine learning on resource-constrained tactical networks, IEEE Military Communications Conference (MILCOM), 2022.

[14] Mendes B., Correia N., Passos D., On the optimization of LoRaWAN gateway placement in wide area monitoring systems, Internet of Things Advances in Information and Communication Technology (IFIP), 2022.

[15] Tapaswi V. A., and Prasenjit C. S., A Survey on path planning techniques for mobile sink in IoT-enabled wireless sensor networks, Wireless Personal Communications, 2021.

[16] Ikhsan M. G., Saputro M. Y. A., Arji D. A., Harwahyu R., and Sari R. F., Mobile LoRa gateway for smart livestock monitoring system, IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), 2018.

[17] Antonino P., Daniele C., Ilenia T., and Vitale G., A Survey on LoRa for smart agriculture: Current trends and future perspectives, IEEE Internet of Things Journal, 2022.

[18] Mojamed A. M., On the use of LoRaWAN for mobile Internet of Things: The impact of mobility, Applied System Innovations Using Recent Communications and Networking Advances, 2022.

[19] Ordonez A., Caicedo O. M., Villota W., Rodriguez-Vivas A., and Fonseca N. L. S., Model-based reinforcement learning with automated planning for network management, Sensors, vol 22, 2022.

[20] LoRaWAN specification v1.1. `https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/`, last accessed on Mar 2, 2024.

[21] Sarker V. K., Queralta J. P., Gia T. N., Tenhunen H., and Westerlund T., A survey on LoRa for IoT: Integrating edge computing, International Conference on Fog and Mobile Edge Computing (FMEC), 2019.

[22] Understanding LoRa adaptive data rate `https://lora-developers.semtech.com/uploads/documents/files/Understanding_LoRa_Adaptive_Data_Rate_Downloadable.pdf`, last accessed on Mar 2, 2024.

[23] Ferre G., Collision and packet loss analysis in a LoRaWAN network, In 25th European Signal Processing Conference (EUSIPCO), 2017.

[24] Bor M., and Roedig U., LoRa transmission parameter selection, 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2017.

[25] Slabicki M., Premsankar G., and Di Francesco M., Adaptive configuration of LoRa networks for dense IoT deployments, In IEEE/IFIP Network Operations and Management Symposium (NOMS), 2018.

[26] Li S., Raza U., and Khan A., How agile is the adaptive data rate mechanism of LoRaWAN?, IEEE Global Communications Conference (GLOBECOM), 2018.

[27] Santa J., Sanchez-Iborra R., Rodriguez-Rey P., Bernal-Escobedo L., and Skarmeta A. F., LPWAN-based vehicular monitoring platform with a generic IP network interface, Sensors, vol 19, 2019.

[28] Hussain F., Hassan S. A., Hussain R., and Hossain E., Machine learning for resource management in cellular and IoT networks: Potentials, current solutions, and open challenges, In IEEE Communications Surveys & Tutorials, vol 22, 2020.

[29] Gutiérrez S., Martínez I., Varona J., Cardona M., and Espinosa R., Smart mobile LoRa agriculture system based on Internet of Things, IEEE 39th Central America and Panama Convention (CONCAPAN XXXIX), 2019.

[30] Zhang Z., Zhou C., Sheng L., and Cao S., Optimization schemes for UAV data collection with LoRa 2.4 GHz technology in remote areas without Infrastructure, Drones, vol 6, 2022.

[31] Behjati M., Binti A., Noh M., Alobaidy H. A. H., Zulkifley M. A., Nordin R., and Abdullah N. F., LoRa communications as an enabler for Internet of drones towards large-scale livestock monitoring in rural farms, Sensors, vol 21, 2021.

[32] Zeng Y., Zhang R., and Lim T. J., Throughput maximization for UAV-enabled mobile relaying systems, IEEE Transactions on Communications, vol 64, 2016.

[33] Zhang S., Zhang H., He Q., Bian K., and Song L., Joint Trajectory and Power Optimization for UAV Relay Networks, IEEE Communications Letters, vol 22, 2018.

[34] Chen Y., Feng W., and Zheng G., Optimum placement of UAV as relays, IEEE Communications Letters, vol 22, 2018.

[35] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, 3-D Placement of an Unmanned Aerial Vehicle Base Station (UAV-BS) for Energy-Efficient Maximal Coverage, IEEE Wireless Communication Letters, vol 6, 2017.

[36] Montero E., Rocha C., Oliveira H., Cerqueira E., Mendes P., Santos A., and Rosário D., Proactive radio- and QoS-Aware UAV as BS deployment to improve cellular operations, Computer Networks, vol 200, 2021.

[37] Huang H., and Savkin A. V., Deployment of heterogeneous UAV base stations for optimal quality of coverage, IEEE Internet of Things Journal, vol 9, 2022.

[38] Saxena V., Jaldén J., and Klessig H., Optimal UAV base station trajectories using flow-level models for reinforcement learning, IEEE Transactions on Cognitive Communications and Networking, vol 5, 2019.

[39] Al-Ahmed S. A., Shakir M. Z., and Zaidi S. A. R., Optimal 3D UAV base station placement by considering autonomous coverage hole detection, wireless backhaul and user demand, Journal of Communications and Networks, vol 22, 2020.

[40] Lu H., Wei X., Qian H., and Chen M., A cost-efficient elastic UAV relay network construction method with guaranteed QoS, Ad Hoc Networks, vol 107, 2020.

[41] Baştürk I., Energy-efficient communication for UAV-enabled mobile relay networks, Computer Networks, vol 213, 2022.

[42] Zorbas D., Caillouet C., Hassan K. A., and Pesch D., Optimal data collection time in LoRa networks - A time-slotted approach, Sensors, vol 21, 2021.

[43] Stellin M., Sabino S., and Grilo A., LoRaWAN networking in mobile scenarios using a WiFi Mesh of UAV gateways, Electronics, vol 9, 2020.

[44] Sciullo L., Trotta A., and Felice M. D., Design and performance evaluation of a LoRa-based Mobile emergency management system (LOCATE), Ad Hoc Networks, 2020.

[45] Hamnache M, Kacimi R., and Beylot A. L., Enabling an inter-operator roaming capability in LoRaWAN networks, Ad Hoc Networks, 2022.

[46] Dias J., and Grilo A., LoRa WAN Multi-hop uplink extension, International Conference on Ambient Systems, Networks and Technologies (ANT), 2018.

[47] Nakamura K., Manzoni P., Redondi A., Longo E., Zennaro M., Cano J. C., and Calafate C. T., A LoRa-Based protocol for connecting IoT edge computing nodes to provide small-data-based services, Digital Communications and Networks, vol 8, 2022.

[48] Khoufi I., Laouiti A., Adjih C., and Hadded M., UAVs trajectory optimization for data pick up and delivery with time window, Drones, vol 5, 2021.

[49] Wang Z., Schaul T., Hessel M., Hasselt H. V., Lanctot M., and Freitas N. D., Dueling network architectures for deep reinforcement learning, In Proceedings of the 33rd International Conference on International Conference on Machine Learning (JMLR), 2016.

[50] Tavakoli A., Pardo F., and Kormushev P., Action branching architectures for deep reinforcement learning, In Proceedings of the Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, 2018.

[51] Shuai H., F. Li F., Pulgar-Painemal H., and Xue Y., Branching dueling Q-Network-Based online scheduling of a microgrid With Distributed energy storage systems, In IEEE Transactions on Smart Grid, vol 12, 2021.

[52] Wei F., Feng G., Sun Y., Wang Y., Qin S., and Liang Y. -C., Network slice reconfiguration by exploiting deep reinforcement learning with large action space, In IEEE Transactions on Network and Service Management, 2020.

[53] Al-Tam F., Mazayev A., Correia N., and Rodriguez J., Deep PC-MAC: A deep reinforcement learning pointer-critic media access protocol, IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2020.

[54] Vinyals O., Fortunato M., and Jaitly N., Pointer Networks, Advances in Neural Information Processing Systems, 2015.

[55] Almi'ani K., Viglas A., and Libman L., Mobile element path planning for time-constrained data gathering in wireless sensor networks, IEEE International Conference on Advanced Information Networking and Applications (AINA), 2010.

[56] Hauser V., and Hégr T., Proposal of adaptive data rate algorithm for LoRaWAN-based infrastructure, IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), 2017.

[57] Sandoval R. M., Garcia-Sanchez A. -J, and Garcia-Haro J., Optimizing and updating LoRa communication parameters: A machine learning approach, In IEEE Transactions on Network and Service Management, vol 16, 2019.

[58] Park G., Lee W., and Joe I., Network resource optimization with reinforcement learning for low power wide area networks, Journal on Wireless Communications and Networking (EURASIP), 2020.

[59] Shen Y., Shi Y., Zhang J., and Letaief K. B., LORM: Learning to optimize for resource management in wireless networks with few training samples, In IEEE Transactions on Wireless Communications, vo19, 2020.

[60] Ferre G., Collision and packet loss analysis in a LoRaWAN network, 25th European Signal Processing Conference (EUSIPCO), 2017.

[61] Callebaut G., Ottoy G., and Perre L. V. D., Cross-layer framework and optimization for efficient use of the energy budget of IoT nodes, IEEE Wireless Communications and Networking Conference (WCNC), 2019.

[62] Kalmbach P., Zerwas J., Babarczi P., Blenk A., Kellerer W., and Schmid S., Empowering self-driving networks, In Proceedings of the Afternoon Workshop on Self-Driving Networks, 2018.

[63] Zhang Z., Ma L., Poularakis K., Leung K. K., and Wu L., DQ Scheduler: Deep reinforcement learning based controller synchronization in distributed SDN, IEEE International Conference on Communications (ICC), 2019.

[64] Marais J. M., Abu-Mahfouz A. M. ,and Hancke G. P., A review of LoRaWAN simulators: Design requirements and limitations, International Multidisciplinary Information Technology and Engineering Conference (IMITEC), 2019.

[65] Blenn N., and Kuipers F., LoRaWAN in the wild: Measurements from the things network, ArXiv /abs/1706.03086, 2017.

[66] Moons B., Karaagac A., Haxhibeqiri J., Poorter E. D., and Hoebeke J., Using SCHC for an optimized protocol stack in multimodal LPWAN solutions, IEEE 5th World Forum on Internet of Things (WF-IoT), 2019.

[67] Sandoval R. M., Garcia-Sanchez A. -J., Garcia-Haro J., and Chen T. M., Optimal Policy Derivation for Transmission Duty-Cycle Constrained LPWAN, IEEE Internet of Things Journal, vol 5, 2018.

[68] Benkahla N., Tounsi H., Song Y., and Frikha M., Enhanced Dynamic Duty Cycle in LoRaWAN Network, International Conference on Ad-Hoc Networks and Wireless, 2018.

[69] International Business Machines (IBM) Corporation CPLEX Optimizer Documentation, `https://www.ibm.com/support/pages/ilog-cplex-optimization-studio-product-documentation`, last accessed on Jun 10, 2024.

[70] Gurobi Optimization LLC Optimizer Reference Manual, `https://www.gurobi.com/documentation/current/refman/index.html`, last accessed on Jun 10, 2024.

[71] Mazayev A., Al-Tam F., and Correia N., Attention-Based Model and Deep Reinforcement Learning for Distribution of Event Processing Tasks, Internet of Things, vol 19, 2022.

[72] Al-Tam F., Correia N., and Rodriguez J., Learn to Schedule (LEASCH): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5G MAC Layer, IEEE Access, vol 8, 2020.

[73] Al-Tam F., Mazayev A., Correia N., and Rodriguez J., Radio Resource Scheduling with Deep Pointer Networks and Reinforcement Learning, IEEE International Workshop on Computer Aided Modeling and Design (CAMAD), 2020.

[74] Wu H., Judd P., Zhang X., Isaev M., and Micikevicius P., Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation, ArXiv /abs/2004.09602, 2020.

[75] Tessler C., Shpigelman Y., Dalal G., Mandelbaum A., Kazakov D. H., Fuhrer B., Chechik G, and Mannor S., Reinforcement Learning for Datacenter Congestion Control, ACM SIGMETRICS Performance Evaluation Review, vol 49, 2021.

[76] Semtech LR1110 User Manual, 2024 https://www.mouser.com/pdfDocs/UM_LR1110_W_APP_V10-2.pdf, last accessed on April 21, 2024.

[77] Rzuolo/NS3-LoraRL, 2022. https://github.com/rzuolo/NS3-LoraRL, last accessed on November 20, 2023.

[78] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., and Polosukhin I., Attention is All you Need, Advances in Neural Information Processing Systems. Curran Associates, 2017.

[79] Top 10 Drones with the Longest Flight Time in 2023, https://www.jouav.com/blog/drone-with-longest-flight-time.html, last accessed on November 20, 2023.

[80] Guillaume J., Weidmann Y., Dongen E. V., Lüthi M. P., Vieli A., and Ryan J. C., High-Endurance UAV for Monitoring Calving Glaciers: Application to the Inglefield Bredning and Eqip Sermia, Greenland, Frontiers in earth science, vol 7, 2019.

[81] Liando J. C., Gamage A., Tengourtius A. W., and Li M., Known and Unknown Facts of LoRa: Experiences from a Large-scale Measurement Study, Transactions Sensor Networks (ACM), 2019.

[82] Takwa A., Martin H., Bernard T., and Andrzej D., Experimental Characterization of LoRaWAN Link Quality, IEEE Global Communications Conference (GLOBECOM), 2019.

[83] Rahmadhani A., and Kuipers F., When lorawan frames collide, Proceedings of the 12th International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (ACM), 2018.

[84] Chochul M., and Sevcik P., A Survey of Low Power Wide Area Network Technologies, 18th International Conference on Emerging eLearning Technologies and Applications (ICETA), 2020.