Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

PhD Thesis

Taking Advantage of Correlated Information for Energy-aware Scheduling in the IoT: A Deep Reinforcement Learning Approach

Author:
Jernej Hribar

Supervisor:
Prof. Luiz A. DaSilva

7th May 2020

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Signed:

_____

Jernej Hribar, 7th May 2020

Summary

The goal of the research conducted in this thesis is to provide insight into how it is possible to improve the energy efficiency of sensing devices in the Internet of Things (IoT) by taking advantage of the correlation exhibited in the information they collect. With massive deployments of low-power sensing devices in cities, factories, fields, etc., and with many of them being energy-constrained due to their reliance on a non-rechargeable battery or on energy harvesting, energy-efficient scheduling of updates transmitted by such devices is of extreme importance to future networks.

We focus on a system of low-power sensing devices, observing a phenomenon distributed in space and evolving in time. In such a scenario, the collected information may exhibit correlation in time that can be leveraged to improve devices' energy efficiency without impacting the performance of the services that rely on the information collected. To gain insight on how beneficial it can be for the system to rely on the correlated information, we employ the recently proposed Age of Information (AoI) metric, which quantifies the freshness of information, to analyse the system. Building on the obtained insight, we propose a design of an energy-aware scheduling mechanism using a Deep Reinforcement Learning (DRL) algorithm to improve sensing devices' energy efficiency.

First, we comprehensively analyse a system of two correlated sensors using the AoI metric. We prove that there is an optimal waiting time for the first update from the correlated source such that the benefits of employing correlated information are the highest. However, there is a limit of how much one sensor can benefit by leveraging correlation of information collected by others. Furthermore, we show that there is a fundamental trade-off between that gain and the correlated sensor's update rate. Additionally, by using different covariance models, which we employ to model the correlation between sensors, we can show their impact on the gain of using correlated information. To provide intuition regarding the use of covariance models, we analyse data collected in a real sensor network.

With the gained insight, we are able to design an energy-aware scheduling mechanism using two DRL algorithms, namely Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). The proposed mechanism is capable of determining the frequency with which sensors should transmit their updates, to ensure the accurate collection of observations, while simultaneously considering the energy available. To evaluate our scheduling mechanism, we use multiple datasets containing multiple environmental observations obtained in two real sensor deployments. We show that our solution is capable of significantly

extending sensors lifetime. We compare our mechanism to an idealized, all-knowing, scheduler to demonstrate that its performance is near-optimal. Additionally, we analyse a key feature of our design, energy-awareness, by displaying the impact of sensors' energy levels on the set frequency of updates.

In this thesis, we, by proposing a scheduling mechanism, demonstrate how it is possible to take advantage of correlation exhibited in the information collected by low-power sensors to extend the lifetime of the network. Our work considers information collected by sensors as a network resource that can be used to improve their energy efficiency.

Acknowledgements

A famous Slovenian Alpine climber Nejc Zaplotnik once wrote: “Anyone looking for goal will remain empty when it will be reached, but whoever finds a way, will always carry the goal inside.” In the last four years, this thesis was my goal, and luckily I had the privilege to work alongside great people who helped me find my way.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Luiz A. DaSilva, who took me as his student and believed in me from the start. His expert advice helped me immensely in carrying out the research presented in this thesis.

To Nick and Maice, I will never forget your help and thank you for setting me on the right path. I would also like to thank Alex, Andrei, and George, for their invaluable contributions to my work.

I want to thank everyone in our lab who made my long journey enjoyable and cheered me up when necessary. To Jacek for providing me with his immense wisdom and being there for me when I needed a piece of advice. To Boris for all the inspiring discussions and for lending me a tool (any tool) when I needed it. To Conor for always offering me a helping hand. To Erika and Joao for being ready to listen whenever I needed to complain. I would also like to thank everyone else who helped me on my journey in no particular order: Andrea, Andre, JB, Eamon, Harleen, Parna, Merim, Fadhil, Alan, Sandip, Nima, Diarmuid, Yi, Carlo, Pedro, Francisco, and Danny. Also special thanks to Eoghan, Patrick, Craig, Tiernan, and Marina, for all the beautiful Thursday D&D nights we spent together.

To my parents Stanka and Dusan, my sister Sabina and her family (Klemen, Patrik, and Ela), and my friends: Andrej, Blaz L., Blaz P., Klemen, and Tomaz, for their never-ending support and encouragement. And a special thanks to my girlfriend Susanah for supporting me in the last (most stressful) few months of my PhD.

Contents

List of Figures

List of Tables

1 Introduction

Introduction

> *Everything is related to everything else, but near things are more related than distant things.*
>
> Waldo Tobler's First Law of Geography, *1969*

Over the course of the next few years, billions of Internet of Things (IoT) sensing devices will be deployed [1]. These devices will collect information essential for various services to make decisions and take actions. However, many of these sensors will be battery-powered or rely on energy harvesting to obtain the necessary energy to transmit an observation. Providing accurate and up-to-date information to services while keeping the battery-powered sensors functional for as long as possible is one of the primary challenges in the IoT.

In this thesis, we investigate how it is possible to prolong the lifespan of battery-powered sensors by leveraging the massive scale of the Internet of Things (IoT). In a network with billions of sensors, it is expected that information collected by them exhibits high temporal and spatial correlation. By using correlated information, it is possible for individual devices to reduce their transmission requirements and save energy. Timeliness of information, i.e. a measure of how much time has elapsed since the information was generated, plays a crucial role in determining how beneficial an update received from a sensor is. We propose a mechanism that enables sensors to decrease the frequency with which they transmit updates. Our mechanism takes advantage of newer information from correlated sensors, thereby reducing the energy consumption of devices without impacting the system performance. In other words, we use the IoT's immense scale to our advantage, as more devices mean a higher probability of sensors transmitting correlated information. The closer to each other the sensors are, the more likely it is that the information they are collecting is correlated in time.

We have designed the proposed scheduling mechanism by considering the IoT architecture and the role of sensing devices within it. More specifically, we consider a publish-subscribe scenario, in which services "subscribe" to observations "published" by sensing devices. Such an approach usually requires a central entity, e.g., a server, on which the

information collected by sensing devices is stored. At the same time, services connect to it to obtain the required information rather than connecting to sensing devices themselves. In theory, the approach could be disturbed, which means that each device would store its observations and each service would connect directly to the device from which it requires information. However, in practice, due to the many limitations of sensing devices, the centralized approach is more common. We take advantage of such a centralized approach by deploying the proposed scheduling mechanism at the location of the central entity. This means that the proposed mechanism is located at the gateway(to which wireless sensors directly connect), or further up the IoT architecture in the cloud.

1.1 Utilising Correlated Information in the IoT

Since 2017, IoT devices have outnumbered[1] humans, and it is estimated that by the end of 2025 there will be more than 20 billion[1] IoT devices deployed worldwide [2]. With IoT devices becoming more ubiquitous, network providers have to find a way to accommodate the diverse requirements these devices have in comparison to a traditional cellular users (humans); this is a trend that is reflected in the recently proposed Long Term Evolution for Machines (LTE-M) for the 4^{th} generation (4G) and Narrowband IoT (NB-IoT) as part of the 5^{th} generation (5G) of cellular network technology. Additionally, other Low-Power Wide-Area Network (LPWAN) protocols designed specifically for IoT devices have been proposed, such as SigFox, Weightless, and Long Range Wide Area Network (LoRaWAN). The main reasons behind the emergence of such protocols is the requirement of IoT devices to transmit data in an energy-efficient manner [3].

On top of facilitating energy efficient communication protocols, the energy efficiency of IoT devices may be improved by tapping into the resources provided by devices themselves, namely data. The generated data is essential for a variety of services in smart cities [4], Industry 4.0 [5], smart agriculture [6], and many other IoT applications. The role of a service is to make decisions based on the provided information. For example, in smart agriculture, a service controlling an irrigation system requires information from various environmental sensors to decide which fields to water. When information from a sensor at one location is outdated or unavailable, information correlated in time and space obtained by other sensors may be used to aid the service's decisions. We believe that leveraging collected data has untapped potential in the IoT. A sensor can decrease the frequency with which it transmits its updates by relying on the use of correlated information, consequently reducing its energy consumption without impacting the services' performance.

The decision on how often a sensor should transmit an update primarily depends on the service requirements. In general, the higher the required accuracy is, the more often a sensor should transmit its observations. However, whenever sensors transmit information which

[1] This number excludes computers, phones, and tablets.

is correlated in time, the timeliness of the information plays a crucial role in determining how informative, i.e. useful, an update is for the service. The timeliness of a status update depends on the time elapsed since the last update from the information source, e.g., a sensor[2], and can be characterised through the metric called the *Age of Information (AoI)* [7]. The AoI is relevant in a network in which the receiver is interested in fresh information, but information sources are limited in how often they can send updates. For example, a source using energy harvesting can transmit only when it has collected enough energy to transmit a new update. Or a source might be unable to transmit due to excessive contention for the transmission channel. The underlying assumption is that the more recently the information was generated, the greater its importance in the decision-making process [8]. For example, real-time applications need a constant rate of fresh information to make the right decision, e.g., a sensor in an industrial environment measuring production-critical parameters. In this thesis, we show the AoI can be leveraged to determine how often a sensor should transmit its information [9], which is an important parameter in a variety of scenarios such as Cyber-Physical Systems (CPSs), Unmanned Aerial Vehicle (UAV) control, stock trading, robotics, etc. As pointed out in [10], finding the optimal update rate for transmitting timely updates as a function of system parameters remains an open question.

The decision of when a sensor should transmit a new update depends on various factors such as the sensor's available energy, timeliness of updates from other correlated sensors, contention for the transmission channel, etc. Additionally, the decision has to take into account the ever-changing environment as sensors may be added or removed, energy levels may change due to energy harvesting, etc. In such an environment, controlling the rate at which the sources transmit updated information about the observed phenomenon is a non trivial task. To tackle this challenge, we split the problem into two parts. In the first part, we analyse a system of correlated sources. Then, in the second part, we use our obtained intuition to propose a Deep Reinforcement Learning (DRL) based updating mechanism capable of decreasing the frequency with which IoT sensors transmit updates without compromising the accuracy of the information collected. The proposed mechanism learns from the content of the information collected to improve the energy efficiency of low-power sensors, thus making IoT deployments more sustainable, both economically and environmentally.

1.2 The Scenario of Interest

In many IoT scenarios, sensors will be deployed over a large area with limited energy for their operation. In such a scenario, an efficient way of scheduling their updates can significantly prolong their lifespan. We focus on inexpensive low-power sensing devices deployed to monitor a physical phenomenon over large areas. For example, in a smart city, sensors

[2]Throughout this thesis we refer to source and sensor interchangeably.

monitor phenomena distributed in space and evolving in time such as temperature, precipitation, traffic, quality of air, etc. In such a case, the change in an observation at one location may reveal a similar change in observations at other monitoring points. Therefore, an individual device can lower its transmission rate, by taking advantage of available correlated information from other sources, thus save energy without impacting the performance of services.

Inexpensive low-power sensors are typically equipped with processors that are usually small in size, consume very little energy, and have limited processing power. For example, ARM micro-controller processors are frequently used for such devices [11], capable of performing a basic task, such as to measure temperature, etc. Additionally, such devices are being equipped with energy-efficient communication technology. In our scenario, we assume that devices rely on one of many LPWANs protocols [3], reflecting the trend of LPWANs as the fastest-growing protocols in IoT deployments. It is estimated that over one billion new IoT devices, outfitted with one of the various LPWAN based radios, will be deployed in the next few years [12]. Most devices in such deployments will use a star topology, meaning that the device will communicate directly with a gateway.

Additionally, these devices rely on the sleep mode to preserve energy, as when the device is "sleeping" its energy consumption is negligible. The downside of sleep mode is that while the device is in this mode, the gateway can not communicate with the device. Consequently, the gateway is not able to ask for new observations on demand. The decision regarding when the device should wake-up and send its next observation has to be made while the device is still in the active mode. Typically, the device is in the active mode after it has transmitted an observation. In addition to the two mentioned constraints (limited processing power and use of sleep mode), we also consider that devices in our scenario have limited energy available. Deployed devices are powered only by a non-rechargeable battery, rather than connected to the power grid. In real deployments, that is often the case as connecting devices to the power grid may be too costly. For such devices, every transmission counts, as it brings the device closer to expiration.

1.3 Thesis Outline

In this thesis, we present the key quantitative results of our work in chapters 3 and 4. In both chapters, we focus on the question of how a network can utilise available correlated information collected by sensors. Specifically, in chapter 3, we focus on mathematical analysis to obtain intuition on how the timeliness of information impacts the accuracy of the observations collected by sensors. We then build on this acquired intuition and propose an energy-aware scheduling mechanism in chapter 4 using a Deep Reinforcement Learning algorithm. The organisation of the thesis is as follows:

Chapter 2, **Background**, provides the relevant background for the thesis, discussing the analytical tools and frameworks we rely on to provide insights on how to utilise correlated

information in IoT. Additionally, we describe how IoT emerged and define it as an ecosystem connecting billions of devices. As our work relies on the use of correlated information, we provide an in-depth overview of works on energy-efficient transmitting carried out in the context of Wireless Sensor Networks (WSNs). Because our work considers the AoI, which was proposed in the recent literature on information theory, we provide the necessary background and list relevant literature. We conclude the chapter by proving a short discussion on the applicability of DRL in the IoT, as we employ DRL in our proposed scheduling mechanism, and list some relevant applications of DRL to communication networks.

In *Chapter 3*, **Using Correlated Information to Extend the Lifetime of Low-power Sensors**, we outline how observations gathered by one sensor can be used to improve the timeliness of updates of other correlated sensors in the network. We consider a system of two correlated information sources, i.e., sensors, which periodically send updates to a gateway. Each update carries information regarding the observed physical phenomenon distributed in space and evolving in time. The optimal use of updates in such a system greatly depends on the correlation between the two sources. To explore this effect, we investigate three different models of the covariance between the sensors. Using the covariance, we then model the information correlation between sources. To show the trade-offs of using updates from the correlated sources, we analyse the system for two distinct cases, in which the source of interest updates more and less frequently than the correlated source. In the last part of the chapter, we make use of observations collected by nine sensors which we selected from the Intel Berkeley Research laboratory[3]. The presented data analysis provides the reader with practical insights into the benefits of using correlated information by the system.

In *Chapter 4*, **Energy-Aware Deep Reinforcement Learning Based Scheduling Mechanism**, we explore how to design a scheduling mechanism capable of taking advantage of the available correlated information. We implement the scheduling mechanism using two different DRL algorithms, Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). To be able to employ DRL algorithms, we describe the problem as a decision-making process and highlight the practical challenges that the mechanism has to overcome. We then describe in detail how we implemented the energy-aware scheduling mechanism using DQN and DDPG, and evaluate the mechanism in a simulated environment. In total, we rely on five different observation sets obtained from two real sensor deployments. We benchmark the scheduling mechanism performance over five observation sets and compare it to an ideal, all-knowing, scheduler. We show that our mechanism can on average prolong the lifetime of sensors to 90 percent of the lifetime that an ideal, all-knowing, scheduler would be able to achieve. Additionally, we demonstrate how a key feature of our scheduler, energy balancing, performs in a simulated environment. To simulate the environment with which the proposed scheduling mechanism interacts as realistically as possible, we leverage observations obtained from the Intel Berkeley Research laboratory and the SmartSantander testbed[3]. The Intel dataset provided us with temperature and humidity observations from

[3] We provide a more in-depth description of the dataset in Appendix B.

fifty sensors collected in an office environment over nine days. The SmartSantander observations were collected in an outdoor environment from twenty temperature, ten humidity, and eight ambient noise sensors over nineteen days.

In *Chapter 5*, **Conclusions and Future Directions**, we conclude our work by summarizing our key contributions and discuss potential extensions of our work. More specifically, we examine possible continuation studies and future directions.

1.4 Contributions

In this section, we list the main research contributions presented in this thesis. Our contributions range from providing conceptual intuition obtained analytically to simulation-based results demonstrating the performance of our proposed design. To summarize it in one sentence, we provide insights on how it is possible to effectively take advantage of correlated information collected by low-power sensors to improve their energy efficiency. We list our contributions based on the chapter in which they appear.

Chapter 2 - Background

In this chapter, we present background information for the thesis and list related work. As part of our discussion we make the following contributions:

- Through historical examples we show that the idea behind IoT, i.e., every object connected into a single network, has always been present in the field of wireless communication. The ramification of this is that the insights provided by our work will be applicable even in the network that will replace the IoT as we know it today.
- We make a case for the design of IoT networks that account for the energy and computational limitations of typical IoT devices, and more broadly the sustainability of the network. As we highlight in this chapter, the IoT offers new possibilities for improving the performance of sensors, especially due to rise of various Machine Learning (ML) algorithms capable of extracting patterns from data.

Chapter 3 - Using Correlated Information to Extend the Lifetime of Low-power Sensors

In the third chapter, we analyse the timeliness of information for a system comprising two correlated sources using the recently proposed AoI metric. The main aim of our work is to establish an updating strategy that a source should adopt to maximise the benefits of using correlated information.

- We derive both the minimal and optimal time shifts between two sources' updates, indicating when it is beneficial for the system to use an update from a correlated source.

- We analytically calculate the maximum gain from using correlated information and show the trade-off between reduced error gain and update rate, which must be considered when setting the optimal updating strategy.
- By employing multiple covariance models to represent the correlation between information sources, we can demonstrate their impact on the gain of leveraging correlated information.
- We analyse data, collected in a real sensor network, to provide intuition regarding the applicability of different covariance models to capture temporal and spatial correlation.
- We compare the energy savings achieved by adopting our proposed updating strategy to the state-of-the-art approach proposed in the recent literature.

Chapter 4 - Energy-Aware Deep Reinforcement Learning Based Scheduling Mechanism

In this chapter, we design an energy-aware updating mechanism using a DRL algorithm capable of taking advantage of the correlation exhibited in information collected by low-power sensors. The mechanism is capable of determining a sensor's next update time based on its available energy, the freshness of the information collected, and the expected lifetime of other sensors in the network. Its main goal is to reduce the energy consumption of sensors while at the same time ensuring that services using the collected information are not adversely affected. The main contributions we make in this chapter are as follows:

- We propose a mathematical framework that the scheduling mechanism can rely on to characterise the accuracy of collected information in the absence of fresh observations from sensors.
- We define the decision-making problem that our proposed mechanism is capable of solving by identifying states, actions, and a reward function.
- We implement the proposed scheduling mechanism using two DRL algorithms, DQN and DDPG, and provide a detailed description of our implementations.
- We use multiple datasets containing multiple environmental observations obtained in a real deployment to show that the proposed mechanism is capable of significantly extending the lifetime of sensors.
- We compare the mechanism to an idealized, all-knowing, scheduler to demonstrate that its performance is near-optimal.
- We evaluate a key feature of our design, energy-awareness, by quantifying the impact of sensors' energy levels on the set frequency of updates.

1.5 Dissemination

In this section, we list accepted and under-review scientific papers written during the PhD project. Note that the work we marked with an asterisk is directly relevant to this thesis.

Journals:

- ∗J. Hribar, A. Marinescu, A. Chiumento, and L. A. DaSilva, "Energy Aware Deep Reinforcement Learning Based Scheduling for Sensors Correlated in Time and Space", (Under review)
- Z. Jiang, Y. Liu, J. Hribar, L. A. DaSilva, S. Zhou,and Z. Niu, "SMART: Situationally-Aware Multi-Agent Reinforcement Learning-Based Transmission Control", (Under review)
- ∗J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Using Correlated Information to Extend Device Lifetime", IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2439-2448, Apr. 2019.

Conferences:

- ∗J. Hribar, A. Marinescu, G. Ropokis, and L. A. DaSilva, "Using Deep Q-Learning to Prolong the Lifetime of Correlated Internet of Things Devices", IEEE ICC Workshops, Shanghai, China, 20-24 May 2019.
- ∗J. Hribar and L. A. DaSilva,"Utilising Correlated Information to Improve the Sustainability of Internet of Things Devices", IEEE World Forum on Internet of Things, Limerick, Ireland, 15-18 April 2019.
- ∗J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Updating strategies in the Internet of Things by taking advantage of correlated sources", IEEE Globecom, Singapore, 2017.

Patents:

- ∗ J. Hribar, and L. A. DaSilva, "Method and System For Energy Aware Scheduling For Sensors", U.S. Provisional Patent Application, December 2019.

Other non-peer reviewed work, presentations, and posters,:

- ∗J. Hribar, and L. A. DaSilva, "Using Deep Reinforcement Learning to Improve the Energy-Efficiency of Low-Power IoT Devices", Poster Session at 6th Training School on Machine and Deep Learning Techniques for (Beyond) 5G Wireless Communication System, Barcelona, April 2019.
- ∗E. Fonseca, J. Hribar, and L. A. DaSilva, "SATORI-Smart Networking in the Era of AI", Poster Session at "The Future of Machine Learning" Conference with MIDAS Ireland, Qualcomm and UCC, Cork, February 2019.
- J. Kibilda, N. Afraz, J. Hribar, and Stephen O'Farrell, "Own Your Own Skype - WebRTC Setup and Test", Transition year immersion week for students at CONNECT, Dublin, April 2017.
- J. Hribar, "Achieving Greener IoT with Sharing", The Next-GWIN, Student challenge, Dublin, Presentation only, September 2016.

2 Background

Background

> *When wireless is perfectly applied the whole Earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole... and the instruments through which we shall be able to do this will be amazingly simple compared with our present telephone. A man will be able to carry one in his vest pocket.*
>
> Nikola Tesla, *1926*

The main purpose of this chapter is to present the essential tools we rely on in our work. We provide basic intuition behind discussed topics and provide reference to numerous manuscripts in which a reader may find a more in-depth description of them.

This chapter consists of five sections. In the first section, we describe how the Internet of Things (IoT) emerged. In its essence, the IoT represent the idea of a connected world, and while the exact definitions change over time, the core idea remains the same. Therefore, even though our work is tailored to its current iteration, our findings should be applicable in its next manifestation. In the second section, we define the IoT as a pervasive environment which connects billions of devices into one colossal ecosystem. Such an ecosystem offers new possibilities and options on how to improve low-power sensors' energy efficiency, the main focus of our work. In the third subsection, we describe various energy saving schemes that, similar to us, rely on leveraging correlated information collected by sensors. Our work differentiates from others by relying on the recently proposed Age of Information (AoI) metric. In the fourth section, we provide a comprehensive description of the AoI metric and discuss how it can be utilized in a system of sensors collecting correlated information. Finally, because we employ Deep Reinforcement Learning (DRL) to prolong the lifetime of the network, we provide a short description or DRL, discuss when it is suitable to adopt DRL in an IoT ecosystem, and review existing energy-aware applications of DRL.

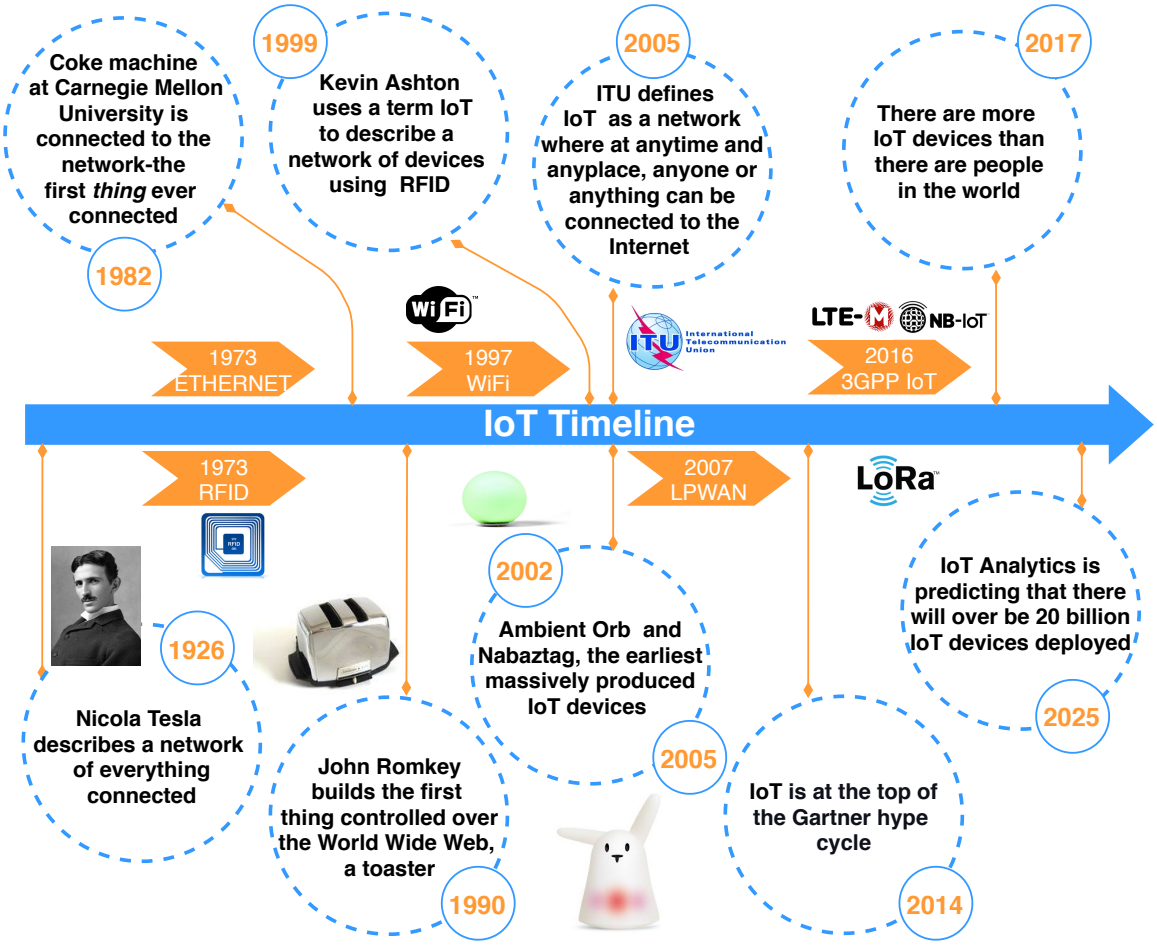


Figure 2.1: Timeline of enabling technologies and important events for the development of IoT.

2.1 The emergence of Internet of Things

The IoT represents a network of devices such as home appliances, Unmanned Aerial Vehicles (UAVs), mechanical machines, etc., with added sensors, actuators, and connectivity that enables them to interact with each other, e.g., exchange data. The generally accepted definitions of the term IoT describe it as a Machine-to-Machine (M2M) or a Human-to-Machine (H2M) interaction with the Internet as a communication medium [13–17]. Some IoT definitions are more user-centric, i.e., H2M interactions. For example, the authors in [14] describe it as a communication standard agnostic interconnection of devices with an ability to share data across different platforms to enable various new applications. In contrast, the International Telecommunications Union (ITU) provided a more M2M centric definition by defining the term IoT as the network where at anytime and anyplace, anyone or anything can be connected to the Internet [17]. Nevertheless, all IoT definitions share the same core idea that exists since the inception of wireless technology, namely, every object (thing) connected in one network.

Nearly a century ago, Nicola Tesla had an idea of connecting the entire surface of the

planet into one colossal brain [18]. He understood that with wireless technology, it would be possible to connect every object into one unified network. Nowadays, we refer to this unified network as the Internet. Interestingly, the term *"Internet of Things"* was coined much later in 1999 when Kevin Ashton used the term IoT in his presentation on applications of Radio-frequency Identification (RFID) in industrial environments [19]. Recently, new names have been proposed to describe a network in which devices communicate with each other in massive scales. For example, Industrial IoT (IIoT), Internet of Everything (IoT), or Internet of Nano Things (IoNT) [20]. However, regardless of the name that will be used to denote the IoT as we know today, the core idea will remain the same, i.e., every object connected into one single network.

In Fig. 2.1 we present important events and technologies that helped IoT to emerge. In 1982 a group of students at Carnegie Mellon University connected a Coke machine to the network to save themselves a walk to it when it was empty [21] - this is now recognized as the first object connected to the network. Eight years later, in 1990, John Romkey built a toaster he could control over the World Wide Web (WWW) [22], supposedly the first device with such a capability. Later, in the early 2000s, various product manufacturers picked up interest in the area and designed the first massively produced devices with Internet connectivity in mind. For example, Ambient Orb and Nabaztag [23] relied on the Internet connectivity to change colour to inform users regarding the weather or stock market conditions. None of those devices had significant market success, but they paved a path towards the IoT as we know today. These examples of early IoT devices reveal the true motivation of connecting an object to the Internet: By adding sensors and connectivity to *things*[1] we can simplify our daily tasks.

2.2 IoT as an Ecosystem

Based on a variety of possible applications, technologies, and domains of use [24–26], we consider the IoT as the combination of devices, data, connectivity, software, and users, that form an ecosystem, as illustrated in Fig. 2.2. The ecosystem enables a pervasive environment where devices interact among themselves and users, collecting information, processing, and transferring it to be used by different applications by and for people, influencing and modifying everyday life.

The focus of our work is on the low-power sensing devices generating information, and we envision their role within the IoT ecosystem as follows:

> **Low-power Devices** represent the core of the IoT, as they enable sensing and actuating in the environment. These devices tend to be inexpensive, and as a result, they are constrained in terms of processing power, energy, and communication capabilities. With recent advances in technology, the devices are becoming

[1] objects we interact with on a daily basis such as washing machines, keys, flowerpots, etc.

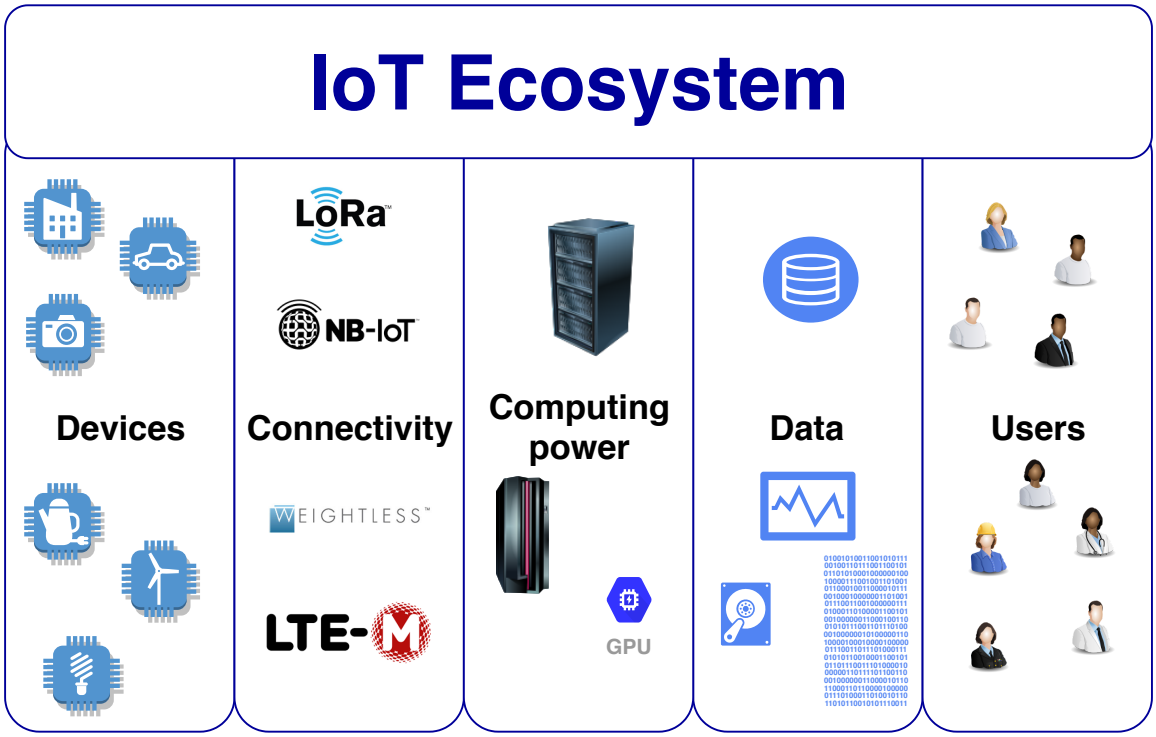


Figure 2.2: The IoT ecosystem.

smaller and more energy-efficient. However, their constraints remain.

Connectivity enables devices to deliver data and to receive instructions. Wireless communication allows design flexibility and eases the deployment process. However, many energy-constrained devices require energy-efficient data communications technology. For example, technologies such as ZigBee or 6lowPAN, based on the 802.15.4 standard, provide energy-efficient communication at the expense of low data rate (as low as few Kb/s). Such a low data rate is acceptable for numerous IoT applications and, recently, many Low-Power Wide-Area Network (LPWAN) standards have emerged. Such as Long Range Wide Area Network (LoRaWAN), SigFox, Weightless, Long Term Evolution for Machines (LTE-M), Narrowband IoT (NB-IoT), etc., all tailored to address different requirements of various IoT applications.

Computing power plays a vital role in an IoT ecosystem, as services require it to process available data and to make decisions. Services can access the highest computing power in the cloud at the expense of longer response times. If services require a faster response, the processing can be carried out closer to end devices, i.e., at the access points or gateways. The third option is to use processing power available on the IoT devices themselves. However, many low-power devices lack the necessary processing power for complex computational tasks.

Data is generated in high volume and with great speed by a variety of different IoT devices. It is also considered a precious resource that enables services to

operate as desired. Additionally, the desire to generate data seems to be is the driving force behind IoT deployments. Recently, there has been a notable trend of the enormous amount of data is being leveraged by Machine Learning (ML) techniques in a multitude of scenarios.

Users are the essential part of the IoT ecosystem. The primary objective of IoT is to improve every aspect of users' lives. Users and devices are meant to interact, but moreover, devices are supposed to automate and communicate among themselves to realize the user's goal.

As we established in the introduction chapter, we focus on a scenario of multiple sensors deployed over a large area with limited available energy. These sensors are part of the ecosystem and play an essential role within it. However, services rarely interact with low-power sensors directly. Instead, they rely on information that is already processed and stored inside the network. For example, in a smart city, deployed sensors observe many environmental parameters, e.g., weather, traffic congestion. A traffic control service will not directly connect to a sensor installed on the street but rather access information for a specific street saved on a server in the cloud. Therefore, the data used by the service can originate from any relevant sensor. Following this logic, the infrastructure providers deploying sensors can take advantage of massive deployments by leveraging correlation in the collected observations. By reducing the number of transmissions, the sensors can preserve their energy, thus decreasing deployment and replacement cost.

In our work, we answer the question of how often sensors collecting correlated information should transmit their observations while maintaining the accurate collection of information as required by services. By increasing the time between transmissions of their observations, the sensors can save energy. As we discuss in the next section, the topic of sensors' energy efficiency has been of great interest to the community in the context of Wireless Sensor Networks (WSNs).

2.3 Making Low-power Devices Energy Efficient Trough the Use of Correlated Information

The use of spatial and temporal correlation has inspired energy saving schemes in the context of WSNs [27–31]. The IoT differs from WSNs as it encompasses a variety of different elements in a much more extensive network, and to a certain extent the WSN can be understood as a part of the IoT ecosystem. Most works proposed to improve the low-power sensors' energy efficiency in WSNs rely on detection or reconstruction of the observed phenomena, through data prediction or model-based active sampling methods.

The initial intention of using correlation to improve the sensing process in WSNs was to use only a fraction of all available sensor nodes to reconstruct the observed physical phenomenon [32]. This objective bears similarities to approaches proposed in estimation

and detection theory, where the primary aim was to determine the necessary density to detect the observed physical phenomenon correctly [33]. Several data-driven techniques that utilise the correlation between sensor nodes for energy conservation have been proposed in the past [27]. Of these, the one most related to our work is the data prediction technique, in which information sensors use mathematical models to estimate the value of the observed physical phenomenon at a given time and place. In some techniques, the sink estimates the value of the observed phenomenon by aggregating information from multiple sensors. Sensors transmit updates only when the information available at the sink results in an error greater than a certain threshold. The authors in [34] show that, by using this approach, the sensors can save energy by sending information only when necessary. However, the approach requires that each of the sensors calculates the estimated value of the observed phenomenon, which may present a challenge for sensors with limited processing capability. Another option is a model-based active sampling technique, which relies on models used only at the sink. In this case, the sink has to control when sensors send an update. As demonstrated in [35], the energy savings are two-fold: fewer transmissions and less energy needed to capture the data. However, the downside of this approach is that the sensor has to constantly listen to discover when to send an update.

Recently a few approaches relying on correlated information to preserve energy were proposed [36, 37]. In [36], the authors propose an algorithm that clusters sensors based on their spatial and temporal correlation. Each time an update is required a different sensor in a cluster will transmit it, thus preserving energy and simultaneously maintaining real-time reporting. Similarly, in [37], the authors propose a scheduling algorithm that groups sensors based on their physical locations to preserve energy. In both cases, the main energy savings arrive due to the mesh network type, as an efficient clustering and routing can result in significant energy savings. However, recent trends for low-power devices indicate that devices will rely on LPWAN standards [38]. In such a case, the network has a star topology and solutions presented in [36] and [37] have limited applicability.

Other methods include employing kriging, i.e., Gaussian process regression, to take advantage of correlated information [39, 40]. With kriging, it is possible to reduce the number of transmission by interpolating observations from non-reporting sensors. For example, in [39] the authors create an interpolation map. Using it, they can reduce the amount of communication from sensors while keeping the accuracy of the observed phenomenon within service requirements. The authors in [40] design a similar solution, proposing an algorithm that selects which subset of sensors should transmit. Such approaches require a significant amount of computing power for kriging. Additionally, a large number of overhead messages is needed to orchestrate the sensors' selection process, i.e., inform the sensor that it is its turn to send an observation.

An interesting approach was proposed in [41], where the authors designed a scheduler that exploits logical correlation and sleep-wake patterns of low-power devices in an ambient-assisted home. The logical correlation is a correlation that describes the relationship between

two or more measured quantities caused by one or more common independent quantities. For example, an open window causes correlated changes in both temperature and humidity measurements. The authors save energy by deactivating sensors when they are not expected to sense anything relevant for the user living in the home. The downside of their approach is that it has limited applicability, as ascertaining the logical correlation for a given scenario is a non-trivial task.

In a more distantly related research area, environmetrics, researchers have looked into how it is possible to leverage correlation between different monitoring sites to optimise monitoring system design [42–45]. These works provide valuable insight on how to model the correlation between sensors. The underlying assumption is that with less required monitoring sites, the overall efficiency of the network will increase. For example, in [42], the authors propose criteria on how to select the optimal sampling sites to observe a process distributed in space. Similarly, in [43], the authors propose an optimal placement algorithm for sampling sites, while the works in [44] and [45] provide an in-depth analysis of the impact of sensors' location on the monitoring process. However, in practice, these approaches have limited applicability in a real network deployment. The determined optimal location might be inaccessible, or costs of preparing the location for a sensor's deployment are too high. Additionally, the proposed approaches rely on a priori knowledge of statistical models representing the observed phenomenon. Therefore, for these approaches to work, the infrastructure provider would have to collect a certain amount of measurements before it could determine the optimal sensor placement.

Our approach to improving energy efficiency in IoT relies on the concept of Age of Information (AoI). Next, we present the AoI metric.

2.4 Age of Information in the IoT

2.4.1 A Metric Representing Information Freshness

In any network, the information generated by sources, e.g., sensors, is always outdated due to delays, latency, traffic congestion, etc. In Fig. 2.3 we depict an information source, e.g. a temperature sensor, generating status updates with information regarding the observed physical phenomenon, e.g. temperature, and sending this information to the sink, e.g., a thermostat. In addition to the observation, the status update also contains a time-stamp revealing when the updates were generated. The time-stamping is crucial, as the sink relies on the time-stamp to calculate the outdatedness of the received status update.

The AoI is defined as the time that has elapsed since a status update was generated at the source:

$$\Delta_i(t) \coloneqq t - t_i, \tag{2.1}$$

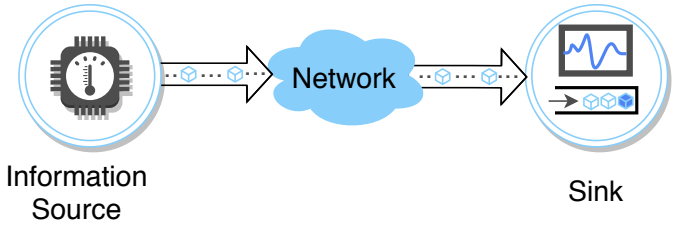


Figure 2.3: An information source sending status updates to the sink.

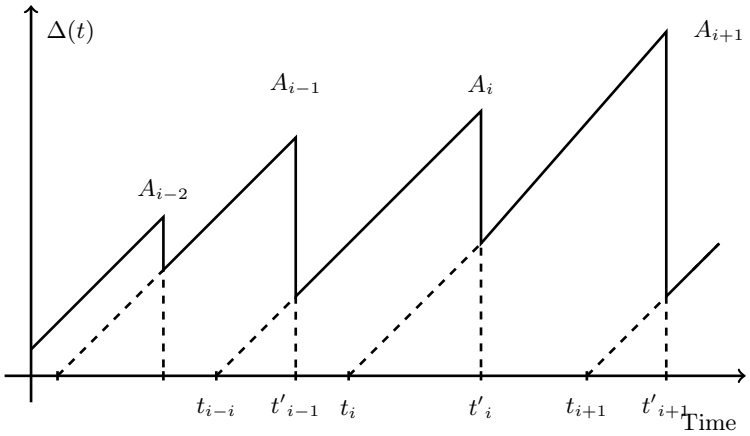


Figure 2.4: Sample path for the process $\Delta(t)$.

where t_i represents the time the source generated the i^{th} update. We show the behaviour of AoI at the sink in Fig. 2.4. The value t'_i represents the time when the sink receives that status update. Whenever the sink receives the update, the AoI becomes $\Delta_i(t'_i) = t'_i - t_i$. The AoI of a status update always increases linearly with time until the next status update lowers the AoI value, at which point the AoI starts increasing linearly again. The delay between the time the update was generated and received at the sink can be due, for example, to latency in the network.

As demonstrated in [7], the behaviour of AoI is different from the delay. The delay depends on transmission channel parameters, such as the distance between source and sink or contention for the transmission channel. In general, delay usually refers to the transmission process. AoI, on the other hand, is a broader concept, which encompasses delay, in addition to other processing times, including the time elapsed since the information was generated. For example, if sources transmit new updates rarely, the delay may be low, but the value of the AoI of those status updates will be high.

Next, we present the basic method to measure the AoI in the system.

2.4.2 Characterizing the Age of Information

Various age-based metrics have been proposed to characterize the impact of outdated information on the system. The most basic measures the average [46] and provides valuable insights on which transmission policy a source should undertake. We calculate the *average*

AoI [47] by integrating the area under the process $\Delta(t)$, and normalize it by the time of observation T as follows:

$$f_{avg}(T) = \frac{1}{T}\int_0^T \Delta(t)dt, \tag{2.2}$$

and extending the observation interval T to infinity:

$$\Delta = \lim_{T\to\infty} f_{avg}(T). \tag{2.3}$$

High average AoI indicates that status updates may be experiencing high delays on the transmission channel or that the source is not transmitting frequently enough.

Minimizing the average AoI in a system may lead to surprising conclusions on when the information source should send status updates. For example, in an energy harvesting system, the source should not send an update when it has gathered enough energy, but rather when a sufficient amount of time has elapsed since the last update [10]. Each new update should deliver new information to the sink. The authors in [47] demonstrated that the status updates should be processed on a last-come-first-served basis to minimize the average AoI because it is expected that the packet at the end of the queue is the most up to date. Even the most obvious updating policies may not be optimal. For example, *zero-wait* policy in which sources transmit another update as soon as the last one was delivered is not always optimal [9]. Such an approach may maximise throughput and minimize delay, but does not always minimize the AoI.

The main drawback of the AoI metric is that the value of AoI does not reveal how valuable to the system the information is. For example, information with AoI of five minutes might be acceptable for a system monitoring temperature in a park, while the same AoI value for information from a temperature sensor located in a combustion engine is not. Merely measuring the time does not directly reveal how informative a status update is. The solution is to employ a metric that also encompasses other relevant system parameters when characterizing the impact of outdated information on the system, namely the *effective* AoI [48].

2.4.3 On the Effective Age of Information

The idea of the effective AoI metric is to provide a measure indicating the impact of outdated information on system performance. Ideally, such a metric would always return a number that would intuitively express how informative an update is. Energy efficiency is an example of such an intuitive metric, where a number reveals easy to comprehend system performance. However, the effective AoI value depends on a variety of system parameters, and to this day, it remains the most notable unanswered question in the AoI literature.

Whenever the observed process changes rapidly, the value of fresh information is higher than when the observed process evolves slowly over time. In an extreme case, when the system would observe a constant, the effective AoI would always be zero. Because once the system obtains the value of the constant process, it holds that information for all-time instances. Ergo, a new update carries no further information for the system. In such a case, the AoI of the sample would increase over time, but the effective AoI would remain zero as an "old" sample is still as good as a "new" one. Effective AoI is fundamentally connected to the concept of how much information the system obtained regarding the observed process. The more information the system has about the observed process, the lower will be the effective AoI.

A few attempts have been made to define an effective AoI metric [49, 50]. However, none of the proposed approaches is practically applicable to a system of multiple correlated sources, the system we consider in our work. For example, in [49] the authors proposed to measure the impact of AoI using case-specific cost functions. However, they fail to propose a fair mechanism for selecting the right cost function for a specific use case. They proposed that in a system with fast changes, an exponential cost function would be acceptable, while a logarithmic cost function could be an appropriate choice for a system with slow changes. Unfortunately, in such a case, the selection of the cost function is arbitrary. Consequently, any results and obtained insights would be directly impacted by the cost function selection. A much better alternative to a cost function is a well-defined metric already widely used by the community, that can capture the impact of outdated information on the system performance.

The authors in [51] used Minimum Mean Square Error (MMSE) to quantify the usefulness of outdated information. They consider a system consisting of a source sampling a Wiener process and sending updates through a channel to an estimator. They study an optimal sampling policy, which can minimise the MMSE on the estimator. They find that a zero-wait policy (the AoI optimal strategy) is not optimal. The optimal sampling policy is a threshold policy, i.e., a source should wait a certain amount of time after the estimator processed the last update before sending a new sample. Such behaviour indicates that considering AoI alone is not enough for a system. Additionally, the results illustrate that the effectiveness of outdated information on the system performance can be characterised through the value of estimation error. In our work, we adopt estimation error to quantify the effectiveness of outdated information in a system of multiple correlated sources.

2.4.4 The AoI in a System of Multiple Correlated Sources

In a system of multiple sources transmitting correlated information, an update from one source lowers the need for a fresh update from all other sources. Sources should adopt status sending policies which are AoI optimal, e.g., minimise average AoI. However, following such an approach requires to translate the benefit of correlated information into the time domain,

i.e., devise an effective AoI metric that encompasses the impact of correlated information. Unfortunately, such a metric does not exist. In what follows, we first discuss the impact of multiple sources on the system AoI performance and then conclude by describing how others tackle the impact of correlation on the AoI value.

Determining the impact on the timeliness of information in a system of multiple sources is a non-trivial task. An update from one source impacts the timeliness of the information on all others. For example, if one source transmits an update, it may lead to additional congestion on the transmission channel and long queues at the sink. The authors in [46] considered a pair of independent sources sending a status update through a queue to a sink. They show that there exists an optimal update rate, i.e., the time between a source's two consecutive updates, with which the source should transmit new updates to minimise the average AoI for both sources. A system of multiple sources sending updates through a queue to a sink is considered in [52]. They demonstrated using a simple queue management technique that it is possible to minimise the AoI in the system. In their approach, upon arrival of a new update from a source, they remove every other status update from the same source that is already in the queue. A more in-depth analysis of a system of multiple independent sources is provided in [53]. The authors conclude: "..that there are nontrivial gains in trunking efficiency when N users share the system capacity. However, achieving these gains appears to require coordinated load balancing of the sources." This means that sources need to adopt updating strategies to optimize their AoI related performance.

The challenge intensifies in a system of dependent sources, i.e., sources transmitting correlated information. In such a case, an update from one source impacts the value of receiving a fresh status update from other correlated sources –a widespread scenario involving IoT sensing devices. And interestingly, such a scenario is relatively unexplored in the AoI literature.

A few works, including our own, have explored the topic [54–59]. In [54, 55] we have proposed possible updating strategies correlated sources can adopt to take advantage of correlated information and demonstrated potential benefits for the system. We were the first to identify the potential benefits of leveraging more timely information from a correlated source. We analysed a system of two sources, one primary source whose information the system uses to base its decisions on, and one secondary source collecting correlated information. We demonstrated when it is beneficial for the system to rely on the information collected by a correlated source. Additionally, we showed that by utilising more timely information from a correlated source, the primary source can reduce the frequency with which it transmits its status update, thus saving energy. In [56] the authors considered a similar system of multiple sensors, i.e., sources, periodically sampling from a one-dimensional static random field. For such a system, they derived an optimal sampling rate, i.e., the frequency with which a source should transmit status updates to keep the accuracy of observations within a preset boundary. However, they did not show any potential gains, e.g., improved energy efficiency, that sources can obtain by adopting the optimal policy. The other three

works [57–59] examined the role of the AoI in a system of multiple correlated sources from different perspectives.

The authors in [57] analysed a system of only one source observing a temporally correlated physical phenomenon. In their system, the source is transmitting status updates over an encoded channel to a monitor(sink). In such a system, two consecutive updates from the source exhibit high correlation which the authors exploited when setting the sources' update scheme. They demonstrate that it is possible to minimise decoding failures by adopting their proposed solution(update scheme). However, their approach has limited applicability in a system observing a spatially and temporally correlated physical phenomenon - the system we consider in this thesis.

The authors in [58] and [59] considered a similar problem, a system in which multiple devices can obtain the same information. Nevertheless, their scenarios slightly differ. The authors in [59] focused on a system of multiple sensors collecting information from various sources of information. Note that the authors in [59] make a distinction between a source of information and a sensor, meaning, that in their system, any of the sensors can obtain information from any information source. For such a system, they propose an updating scheme that minimises the AoI by limiting the number of transmission sensors have to do. They prevent sensors from transmitting information collected from a source that another sensor has already reported. A scenario of multiple wireless cameras observing the same scenery and transmitting their live feed to a fog server for processing was considered in [58]. In such a case, cameras observe the same scenery from multiple angles. Therefore, since they observe the same space, the captured pictures are correlated. The authors focus on minimising the peak AoI of status updates (images of the scenery) from sources (cameras). Note that the peak AoI denotes the maximal AoI value, i.e., the worst case [8]. By reducing the peak AoI of status updates, the fog server can process images faster. Consequently, the entire system can respond more quickly. The correlation considered by the authors in [58] and [59] is not the same as correlation as in our work. They considered correlation as different sensors/sources capable of obtaining the same information. We, on the other hand, consider the correlation that arises when sources(sensors) observe spatially and temporally correlated physical phenomena.

In all of the papers mentioned in this subsection, the observed phenomenon ware assumed to be static, e.g., multiple sensors observing a stationary physical phenomenon, or the same scenery captured by multiple cameras from different angles. In such settings, it is possible to find an optimal updating strategy which the sources should adopt to optimise their performance. However, such an idealised environment is not realistic. For example, correlation of sources is not known apriori. To that end, to take advantage of correlated information, the system has to adopt techniques that are capable of adapting source update times to dynamics found in the real world, e.g., sensors observing a non-stationary process.

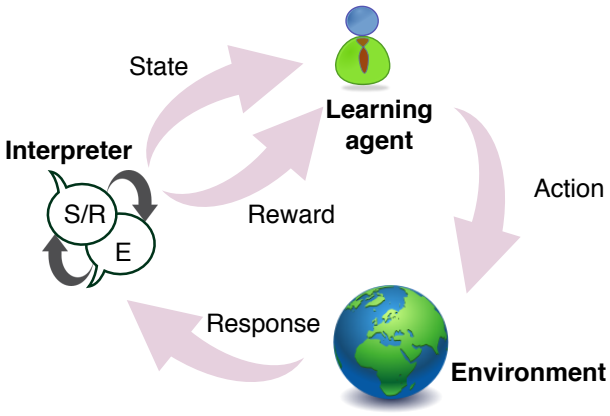


Figure 2.5: Reinforcement Learning cycle.

2.4.5 Difference Between AoI and Temporal Correlation

At first glance, due to topics discussed in this chapter, the temporal correlation might seem similar to the concept of AoI. However, these two concepts are very different. Temporal correlation describes how values of the observed phenomenon are connected in time. Meaning, temporal correlation reveals how similar is a measurement acquired in one moment to a measurement obtained at another moment. In short, the AoI metric measures the timeliness of updates and effect of outdated information on the decision-making process, while temporal correlation describes how the value of observed phenomena is related to its past values.

In the next section, we present the Deep Reinforcement Learning (DRL), a technique we employ in our work, describe when it can be applied in the IoT ecosystem, and list existing energy-saving applications.

2.5 Applying Deep Reinforcement Learning in the IoT Ecosystem

In Reinforcement Learning (RL), the system deploys a learning agent capable of interacting with the environment. The agent will take action. As a result, its state in the environment will change. The interpreter will determine the agent's state, and what the reward for the taken action is. Then the agent will take another action and repeat the cycle. We illustrate the DRL cycle in Fig. 2.5. Over time, the agent will learn which actions are best to take for every state, the best action being the ones that will yield the highest discounted cumulative reward. For the agent to find the best actions possible, the system can employ one of many RL or DRL algorithms [60].

To be able to utilise the RL algorithms, the problem has to be translated into a decision-making process. Using Markov Decision Processs (MDPs) it is possible to mathematically

formalize the sequential decision-making process [61, 62]. MDP offers a way to frame the problem abstractly and flexibly by defining states, actions, and rewards. However, framing the problem ”correctly“ is usually one of the hardest steps in applying DRL to solve a problem and requires extensive insight on the problem itself. By ”correct “ we mean that the selected states, actions, and rewards, will enable the agent to learn the desired behaviour. The selected states have to offer enough knowledge for the agent to have a full representation of the relevant environment, i.e., the state space should encompass all relevant factors. The selected action space should include all actions that an agent could use to achieve optimal behaviour. For example, if a robot wants to learn how to move, its action space should be connected to its moving part. The state space should be based on its sensory inputs. Assigning the reward to a state-action pair is also a complicated task. The main challenge of the reward is to keep its design simple yet still capable of stimulating the agent's desired behaviour. An ill-designed reward may have unintended consequences that will guide the agent into learning an undesired behaviour. Therefore, the agent will never be able to find the optimal solution. A multitude of problems can be described efficiently using MDP and can then be solved with a DRL algorithm.

The environment in which agents act may be complex and often requires an enormous amount of possible states to describe fully. Unfortunately, the higher the number of states, the longer will the agent need to learn the optimal behaviour. Additionally, in real-world deployments, the agent often never experiences the same state again. To resolve the issue, we can use function approximation. With function approximation, it is possible to generalise the received reward for a particular action-state to other similar states and actions. One of the widely used methods to perform nonlinear function approximation is an Artificial Neural Network (ANN) [63].

Whenever the ANN consists of multiple layers of neurons, it gains the ability to extract higher-level features from the input data [64], an approach also referred to as deep learning. In the case of DRL, the so-called deep[2] ANN can recognize patterns in the sequence of agent's actions, an ability that was very well demonstrated in 2013 by DeepMind's learning agent capable of playing Atari games achieving human-level performance [65]. Following their initial success, they designed, using deep learning, an agent capable of defeating the world champion of Go [66]. The popularity of DRL resulted in a numerous newly proposed algorithms [65, 67, 68]. Additionally, many new tools have become available, e.g., Tensorflow, Pytorch[3], etc., adding to the rising popularity of applying DRL to a variety of areas and problems, including telecommunications.

[2]In this thesis, we consider any ANN that consists of more than three layers, including output layer, as deep ANN.

[3]For more information, the reader may visit https://www.tensorflow.org; https://pytorch.org, respectively.

2.5.1 When to Use a DRL-based Solution in the IoT Ecosystem?

In the last few years, we have seen an exponential increase in the use of DRL to solve complicated problems in telecommunications [69–74]. However, applying DRL in favor of a traditional method, e.g., optimization, is not always desirable. Often, well-known tools, already employed by the telecommunications community, provide a much more efficient way of solving a problem. Therefore, the use of DRL is limited to specific scenarios and particular use-cases. Some of the criteria for the use of DRL are summarised as follows:

1. **Decision-making process:** As briefly discussed in the previous section, the problem needs to be translatable into a sequential decision-making process. Describing the problem using MDP in most cases requires an in-depth understanding of the problem.
2. **Model or Algorithm Deficit:** When no precise mathematical model exists to describe the problem, leveraging deep learning to extract patterns from available data is a widely accepted approach in the telecommunications research community. An algorithm deficit, on the other hand, means that mathematical models exist. However, the existing algorithms are too complex for a real deployment, e.g., they require too much processing power [69]. In such a case, DRL can be applied.
3. **Dataset is available:** The use of large amounts of data is crucial for DRL algorithms, allowing the agent to extract patterns from the data. In the case of DRL, where the agents are interacting with the environment, the data should be leveraged to aid the design of the agent's simulated environment.
4. **System Dynamics:** A highly dynamical system, in which conditions change dramatically over a short period of time, is not suitable for any ML based solutions. For example, if system conditions change entirely in a matter of few time steps, the data collected in the past and used in the training process will not be valid when an agent acts in the changed environment. The system does not have to be stationary, but changes should happen gradually enough to provide the agent with the time to learn and adjust its behaviour.

2.5.2 Applying RL to Improve Energy Efficiency in IoT

The IoT ecosystem provides the enormous amount of data and processing power, the two key ingredients to enable deployments of DRL-based solutions. In such settings, as expected, RL and DRL has been applied to many different energy-aware networking solutions [75–87]. Most proposed solutions focus on devices' physical layer performance to improve their energy efficiency, while others try to improve devices' energy collection when they rely on energy harvesting.

The authors in [75] used Q-learning to enhance the spectrum utilisation of industrial IoT devices. They demonstrated that devices are capable of learning which channel is most

suitable to avoid collisions. By reducing the number of re-transmissions, devices improved their energy efficiency. Similarly, in [76], the authors leveraged Q-learning to enhance the ALOHA protocol to minimise collisions on the transmission channel. In their approach, devices not only improved throughput but also improved their energy efficiency. The authors in [77] proposed the deployment of an agent that is capable of scheduling device transmission times and selecting the transmission channel to maximise the system throughput. They considered one relay trough which multiple devices receive packets. The goal of the agent was the timely delivery of updates to devices. The updates were stored in a buffer located on the relay. The authors' solution allowed for timely delivery of updates and simultaneity saved energy (due to fewer transmissions). The authors in [78] also employed Q-learning to improve the energy efficiency of an embedded device, e.g., a phone or tablet. The energy improvement arrived from a better way of scheduling the periodic processing of tasks by devices. However, their approach is limited to devices with substantial computational power available. The use of DRL was also investigated in [79], where authors relied on Bluetooth signal energy strength to improve indoor users' location estimation.

The works in [82] and [83] analysed how an energy-harvesting device collects energy and schedule its transmissions accordingly. In both cases, the objective was more effective power management. They applied RL to prevent power outages, i.e., to avoid the situation where an energy-harvesting device completely depletes its energy. A multi-agent RL approach in a system of devices using energy harvesting was considered in [84]. They investigated a two-hop scenario, in which a source transmitted information to a sink through a relay. They designed a transmission policy that used available energy efficiency and also maximised the throughput. Similarly, in [85], the authors employed a multi-agent approach (with a Deep Q-Network (DQN) algorithm) to power control a system of devices relying on energy harvesting. The main aim of their approach was to maximise devices' data throughput. The DQN algorithm was also used by authors in [80] to solve a task offloading problem for vehicles. By selecting which computational task will be performed by the vehicle and which by a processing unit located at the network edge, the authors demonstrated that it is possible to save energy. In [81], the authors employed deep learning to design a data recovery mechanism for sensors that collect spatio-temporally correlated information. Their solution was capable of determining which observations from other sensors could be used to replace missing or corrupted observations.

The trends in papers listed above, which to the best of our knowledge represent the current state of the art of RL and DRL applications applied in telecommunication to preserve devices' energy, indicate that low-power sensors are mostly overlooked. Furthermore, in most listed works, energy savings arrived from either fewer transmissions due to improved collisions avoidance [75–77], more efficient edge device management [80, 85], or a more efficient collection of energy [82–84]. Additionally, most proposed solutions would be hard to implement in a system consisting of low-power sensors due to their constraints. For example, low processing power available on such devices limits computationally demanding

tasks, e.g., ANN training. This thesis proposes a DRL-based solution that takes advantage of correlated information collected by IoT devices to increase their lifetime.

3 Using Correlated Information to Extend the Lifetime of Low-power Sensors

Using Correlated Information to Extend the Lifetime of Low-power Sensors

> *Nothing puzzles me more than time and space; and yet nothing troubles me less, as I never think about them.*
>
> Charles Lamb, *1810*

In this chapter we investigate the use of correlated sources of information to improve the lifetime of low power sensors in the Internet of Things (IoT). We consider sensors as information sources that transmit periodic updates to a gateway regarding the status of an observed process. Status updates contain the value of the process observed, and a timestamp to indicate when the observation was made. We focus on the benefits of using a second information source, possibly one of reduced cost, in order to deliver timely information.

The technical work presented in this chapter is based on our works "Updating strategies in the Internet of Things by taking advantage of correlated sources" presented at at IEEE Globecom 2017 and "Using Correlated Information to Extend Device Lifetime" published in IEEE IoT Journal.

3.1 Introduction

We described IoT as an ecosystem connecting the physical world with a digital one using ubiquitous computing, various communication technologies, and embedded devices with sensors and actuators. In such an ecosystem, many of the connected devices are battery-powered sensors, with the primary function to observe phenomena (e.g., weather, vehicular traffic, etc.) to provide a system with information regarding the physical world. With more sensors being deployed, the resulting higher density inevitably leads to high correlation in time and space between different observations of the same phenomenon. By leveraging the correlation between sensors, i.e., information sources, we show that it is possible to improve their energy-efficiency.

The system model we analyse in this chapter consists of one information source of interest and one correlated secondary source. The secondary source's role in the network may be to collect information of its own, and the correlation between information collected by multiple sources allows usage of those updates to benefit the source of interest. Alternatively, the correlated source may be present because it was installed in the network with the sole intention of assisting the source of interest. An example of the first scenario is vehicular traffic monitoring sensors installed at selected sites in a city. While the main purpose of each sensor is to collect information from its site, the correlation between the information collected by all sensors can be leveraged to reduce their update rates. An example of the second scenario would be the installation of a second sensor that is capable of providing data that is strongly correlated to the data collected by the sensor of interest but can operate at a lower cost. Regardless of the reason for the presence of the secondary source, it is always possible to use its information to improve the timeliness of updates from the source of interest by leveraging information collected by the secondary source.

In areas where sensor nodes, i.e., information sources, are densely deployed, information sent from neighbouring sensor nodes observing the same phenomenon (e.g., temperature, humidity, noise, etc.) often exhibits high correlation in time and space. Hence, the information gathered at a given time or location can be used to predict the phenomenon at a different position and at a future time instant. A new observation from any sensors can help improve the timeliness of the information on all others. In other words, considered correlation is a two-way relationship. However, in our work, to provide insights, we focus on a one-way perspective, i.e., how much can the source of interest (primary source) gain by leveraging newer information from the correlated source. By taking advantage of correlation in such a way, it is possible to decrease the rate at which an information source transmits observations to a gateway. Such an approach is desirable when an information source has a limited energy source (e.g., a battery), as fewer updates lead to a longer lifetime.

In our work, we rely on analysing the timeliness of updates, i.e., measuring how outdated the information is, to improve the low-power sensors energy efficiency. By determining the value of newer information from correlated sensors, we can reduce the rate at which sensors transmit their updates. Therefore, our approach differs from the Wireless Sensor Networks (WSNs) approaches we listed in the previous chapter, which relied on detection or reconstruction of the observed phenomena to improve sensors' energy efficiency. The trade-off in the WSNs approaches and ours is the same: the higher the accuracy required in the data gathering, the more energy sensors will use. However, our approach is entirely data-driven, making it ready to utilise Machine Learning (ML) methods. The most significant originality of our work, in comparison with WSNs related research, is in the utilisation of the Age of Information (AoI) metric to provide new insights for a network of correlated sensors.

Using the AoI metric, we can characterise the value of a newer update from the secondary source to the source of interest. However, quantifying the exact impact of outdated

information is hard as it depends on multiple system parameters. To that end, we rely on the measure of estimation error to determine how relevant a particular transmitted status update is, similarly to the authors in [51]. However, we focus on a significantly more complex scenario. Specifically, we consider a system of multiple correlated sources. Such a scenario is of great interest to the AoI community [54–59], and we were first to provide insights to the problem. Following our initial work in [54, 55], the authors in [56, 58, 59] analysed from different perspectives a system of correlated sources. Additionally, in contrast to other authors, we focus on how to take advantage of newer information from correlated sources to improve the energy efficiency of battery-powered sources by prolonging the times between sources' consecutive updates.

Our main contribution presented in this chapter is the analysis of the timeliness of information for a system of correlated sources. We determine the optimal update strategy for a system of two correlated sources, i.e., determining when are updates from a secondary source beneficial to the source of interest. We show that the time difference between consecutive updates from the sources is crucial. We derive the optimal time difference between the two sources' updates, for which the presence of correlated sources will bring the highest benefits to the timeliness of the information collected. We model the correlation using three different covariance models. Using multiple models, we can illustrate how various covariance models impact the timeliness of updates. We use one separable and two non-separable models [88, 89]. We also extract values for the parameters in the covariance models from real IoT data. By doing so, we demonstrate that a system can extend the lifetime of a source by relying on newer information from a correlated source.

The remainder of the chapter is structured as follows. In the next section, we present the system model. In that section, we also define the covariance models which we use in our analysis and show how we employ the estimation error metric to quantify the value of status updates. In section 3.3, we analyse the case in which both sources, the one of interest and secondary, transmit with equal update rates. Meaning that time between constitutive updates is the same for both sources. In the fourth section of this chapter, we analyse the case in which either the source of interest or the secondary source updates more often than the other. In the following section, we focus on data analysis to provide the reader with a realistic feel for how beneficial using correlated sources can be. Finally, we provide a conclusion for the chapter.

3.2 System Model

Our system consists of two sensors periodically sending updates to the gateway, regarding the observed physical phenomenon distributed in space and evolving in time. We represent the observed phenomenon at a position $\boldsymbol{x}$ in 3-dimensional space, at a given time instant t, with $Z(\boldsymbol{x}, t)$. The observations are made available to the gateway by means of status update messages, which carry the observation and a time stamp indicating when the observation

was made. We quantify the timeliness of the received updates using the AoI, denoted as $\Delta(t)$.

The AoI is defined as the time that has elapsed at the gateway since the last update from a given source [8], meaning that in a system where sources update deterministically, the AoI will also be deterministic. For the i-th source the AoI is as follows:

$$\Delta_i(t) \coloneqq t - t_i, \tag{3.1}$$

where $i \in \{1, 2\}$, and t_i represents the time of the last update generated by source S_i. We assume that each sensor sends updates periodically at a rate λ_i, i.e., the interval between consecutive updates is $1/\lambda_i$. In such a case, the AoI is also deterministic and varies between 0 and $1/\lambda_i$.

The system can estimate the value of the observed physical phenomenon at the desired location using aged information originating from one of the two available information sources. The accuracy of estimation also depends on the distance between the desired location of estimation and the location of the information source. With d_i we denote the Euclidean distance between the S_i source and the exact location at which the system estimates the value of observed physical phenomena. Our system as represented in Fig. 3.1 is interested in estimating the value of $Z(\boldsymbol{x_1}, t)$ at the location of information source S_1, meaning that updates sent by S_1 are perfectly spatially correlated to the desired value of $Z(\boldsymbol{x_1}, t)$, i.e., $d_1 = 0$. The only effect to be taken into account, in this case, is the AoI of the available information, i.e., $\Delta_1(t)$. The information from the correlated source S_2 may be outdated, i.e., $\Delta_2(t) \geq 0$, and separated in space, i.e., $d_2 > 0$. As a result, the information sent by S_2 is subject to the effects of spatial correlation and aging. The system may use outdated information $Z(\boldsymbol{x_1}, t_1)$ from source S_1 or outdated and spatially correlated information $Z(\boldsymbol{x_2}, t_2)$ from source S_2, when estimating the value of $Z(\boldsymbol{x_1}, t)$. Given the most recent observation available, the estimator which minimises the mean square error is the conditional expectation:

$$\widehat{Z}_i = \boldsymbol{E}[Z(\boldsymbol{x_1}, t) | Z(\boldsymbol{x_i}, t_i)], \;\; i \in \{1, 2\}. \tag{3.2}$$

The conditional variance is proportional to $1 - \mathrm{C}_i{}^2$, where C_i is the correlation coefficient, defined assuming unit variance as

$$\mathrm{C}_i \coloneqq cov\big(Z(\boldsymbol{x_1}, t), Z(\boldsymbol{x_i}, t_i)\big), \tag{3.3}$$

which is a function of the age and the spatial separation between the selected sources.

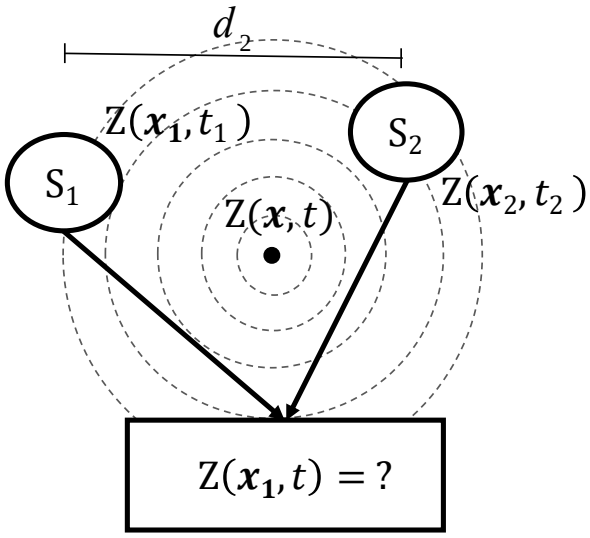


Figure 3.1: Two correlated sources observing the same physical phenomenon and sending the information back to the sink.

3.2.1 Covariance Models

A covariance model can be used to describe how information from two correlated sources observing the same physical phenomena jointly varies over time and/or space. We consider that the observed phenomenon can be represented by a stationary Gaussian process. Therefore, the observations from the two sources $Z(\boldsymbol{x_1}, t_1)$ and $Z(\boldsymbol{x_2}, t_2)$ are assumed to be jointly Gaussian. The covariance indicates how likely it is that if the value of the observed phenomenon changed at the distant correlated source, i.e., S_2, the value at the point of interest, i.e., S_1, has also changed, and vice versa. In our analysis, we consider a separable covariance model [89]:

$$\mathrm{C}_i^I(d_i, t_i) = exp(-\theta_2 d_i - \theta_1 \Delta_i(t)), \tag{3.4}$$

and two non-separable models [88]:

$$\mathrm{C}_i^{II}(d_i, t_i) = \frac{\theta_1 \Delta_i(t) + 1}{\left((\theta_1 \Delta_i(t) + 1)^2 + \theta_2^2 d_i^2\right)^{(D+1)/2}}, \tag{3.5}$$

$$\mathrm{C}_i^{III}(d_i, t_i) = \exp(-\theta_2^2 d_i^2 - \theta_1 \Delta_i(t) - \theta_3 \Delta_i(t) d_i^2). \tag{3.6}$$

Note that D in (3.5) refers to dimensions: we consider 3-dimensional space, i.e., $D = 3$. In the models above, θ_1, θ_2, and θ_3 are the scaling parameters of time and space.

Covariance functions provide tractable mathematical modelling for the spatial-temporal variation of the observed physical phenomenon. However, there is a significant difference between the separable and the non-separable class regarding mathematical tractability.

With a separable covariance class, it is possible to formulate the covariance as a separate product of only the spatial covariance and only the temporal covariance, while for non-separable classes the spatial and temporal components are inseparable. In general, the non-separable class models space-time interactions and is therefore applicable to a broader range of data gathering scenarios in comparison to the separable class. However, as we illustrate in Section 3.5, all three covariance models defined above can be applicable in realistic scenarios

With the correlation between the two sources established, it is possible to determine the estimation error resulting from the use of spatially and temporally correlated information.

3.2.2 System Estimation Error

Whenever the system uses outdated information to estimate the current value of the physical phenomenon, it incurs an estimation error, which depends on the age and the location of the used information. We consider that the observed phenomenon can be represented by a stationary Gaussian process. Therefore, status updates sent from two information sources are assumed to be jointly Gaussian. We can then define estimation error as:

$$\varepsilon_i(d_i, t) = 1 - \mathrm{C}_i^2(d_i, t_i). \tag{3.7}$$

Note that C_i represents one of the covariance models defined in previous subsection, i.e., C_i^I, C_i^{II}, or C_i^{III}. With ε_i we calculate the estimation error for the last update received from the i-th source. The higher the error, the lower the value of the information in the status update, and the less likely is that the system will use that update.

The system always uses the information from the source which results in the lower estimation error, meaning that the system estimation error is:

$$\varepsilon_{sys}(d, t) = min\big(\varepsilon_1(0, t), \varepsilon_2(d_2, t)\big). \tag{3.8}$$

Fig. 3.2 illustrates how the estimation error changes over time when both sources have the same update rate, with (3.4) describing the correlation between sources. Let T_1 denote the period of updates from source S_1, i.e., $T_1 = 1/\lambda_1$. Every time that S_1 transmits an update, ε_1 drops to zero. As information becomes more outdated, the error slowly rises according to the expression in (3.9). We denote T_2 as the status update period for S_2. Updates for S_2 are transmitted at time instants $\tau_\Delta + jT_2, j \in \{1, 2, \ldots\}$, at which times $\Delta_2(t) = 0$, and the error based on information received from S_2 is $\varepsilon_2(d_2, 0) = 1 - exp(-2\theta_2 d_2)$ and then slowly rises according to the expression in (3.10). In Fig. 3.2, the system estimation error is zero every time S_1 sends an update and then increases along with expression (3.9) until

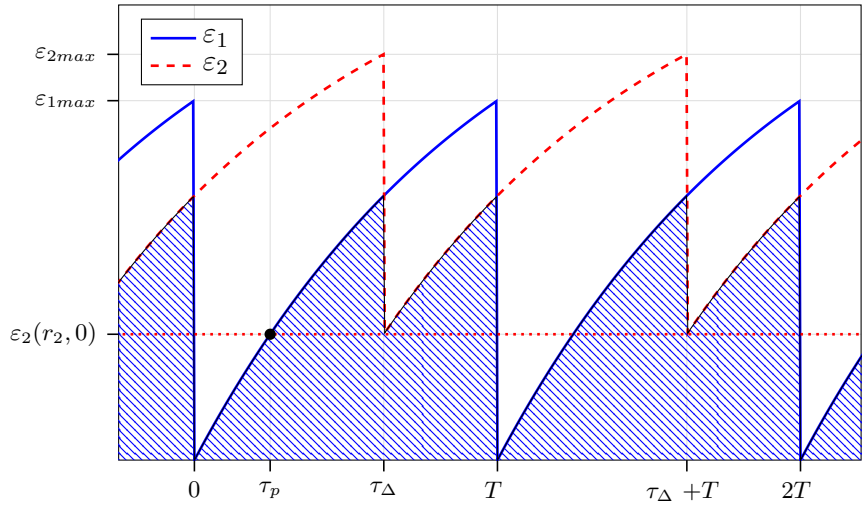


Figure 3.2: Error change over two status updates from S_1, for two correlated sources with equal update rates. At $t = 0$, S_1 sends an update, followed by an update from S_2 with time shift τ_Δ. The dashed area is the system estimation error. The covariance model adopted here is the one in Equation (3.4).

an update from S_2 is received, after which the system estimation error equals the error corresponding to the information from S_2, until S_1 updates again and repeats the cycle.

$$\varepsilon_1(0, t) = 1 - exp(-2\theta_1\Delta_1(t)) \tag{3.9}$$

$$\varepsilon_2(d_2, t) = 1 - exp(-2\theta_2 d_2 - 2\theta_1\Delta_2(t)). \tag{3.10}$$

In the next sections, we will analyse the proposed system and determine when sources should send updates such that the system estimation error will be minimal.

3.3 Equal Update Rates

In this section, we consider a system in which both sources update with equal update rates, i.e., $\lambda_1 = \lambda_2$, meaning that $T_1 = T_2 = T$. In such a case, the system estimation error depends only on the time shift between the sources' updates. We denote this time shift as τ_Δ, which may take values in $[0, T)$. In Fig. 3.2, when S_2 sends its first update, the age of information from S_1 equals τ_Δ, i.e., $\Delta_1(t) = \tau_\Delta$. Whenever S_2 produces an update, this update may incur a higher or lower estimation error than that from the latest update from S_1, depending on τ_Δ and the correlation between the two sources. As seen in Fig. 3.2, the error at the time of an update from S_2 equals $\varepsilon_2(d_2, 0)$.

3.3.1 When is Correlated Information Beneficial?

An update from the correlated source can replace only sufficiently aged updates from the source of interest. We refer to τ_p as the minimal age an update from S_1 needs to have so that an update from S_2 can reduce the system estimation error. When S_2 updates with time shift τ_p, i.e., $\tau_\Delta = \tau_p$, the estimation error from S_2 at $\Delta_2 = 0$ equals the error from source S_1 at time τ_p. Hence, we can calculate τ_p by setting:

$$\varepsilon_2(d_2, 0) = \varepsilon_1(0, \tau_p). \tag{3.11}$$

τ_p results[1] for C^{I}, C^{II}, and C^{III}, respectively, are:

$$\tau_{p_I} = \frac{\theta_2 d_2}{\theta_1}, \tag{3.12}$$

$$\tau_{p_{II}} = \frac{(1 + \theta_2^2 d_2^2)^{\frac{2}{3}} - 1}{\theta_1}, \tag{3.13}$$

$$\tau_{p_{III}} = \frac{\theta_2^2 d_2^2}{\theta_1}. \tag{3.14}$$

τ_p is independent of update rates and depends only on the scaling parameters for the covariance model and the spatial separation between sources. Furthermore, it imposes a condition on the system, that for the information from the correlated source to be beneficial in reducing the system estimation error, the update period for the two sources has to be higher than τ_p, i.e., $T > \tau_p$.

With an increase in distance between the information sources, τ_p always increases, regardless of which covariance model describes the correlation between the sources, as shown in Fig. 3.3. When the correlation can be described by the separable model C^{I}, τ_p increases linearly, while with non-separable covariance models C^{II} and C^{III} the increase is exponential. In the case depicted in Fig. 3.3, the scaling parameter θ_1 is larger than parameter θ_2, meaning that sensors are more temporally than spatially correlated. The scaling parameters have a direct impact on the τ_p value. The more spatially correlated sources are, i.e., $\theta_2 >> 0$, the higher the value of τ_p. However, regardless of the scaling parameter values, the curve shape remains the same. The increase of τ_p with distance illustrates how the shorter the update period, the closer together the sources must be for the updates from S_2 to be advantageous in reducing the system estimation error.

To calculate the system estimation error when two correlated sources update with equal update rate, it is necessary to integrate $\varepsilon_{sys}(d, t)$ over one period T, which can be expressed as:

[1] We provide a detailed description of the procedure in Appendix A.

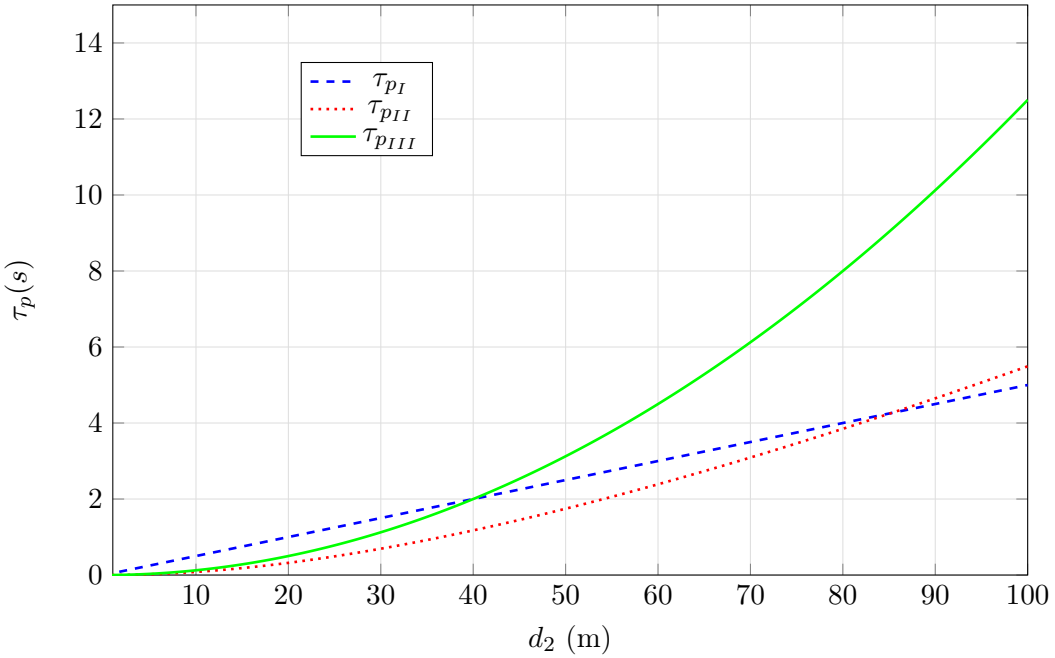


Figure 3.3: The graph shows the increase of τ_p with the distance between information sources, for all three covariance models. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.025$, and $\theta_3 = 0.0125$.

$$\overline{\varepsilon}_{sys} = \frac{1}{T}\left(\int_0^{\tau_\Delta} \varepsilon_1(0,t)dt + \int_{\tau_\Delta}^{T} \varepsilon_2(d_2,t)dt \right), \tag{3.15}$$

where, for $0 \leq t \leq T$:

$$\Delta_1(t) = t \text{ and } \Delta_2(t) = t - \tau_\Delta,$$

and with the condition:

$$\tau_\Delta \geq \tau_p.$$

The system estimation error varies with the time shift τ_Δ between the updates from the two information sources. To illustrate this effect, we define the *error gain*, which captures the impact that the updates from S_2 have on reducing the system estimation error. We define the gain as the ratio between the average error if the system relied solely on updates from S_1 to the system estimation error when updates from S_2 are also considered:

$$G(d,t) = \frac{\overline{\varepsilon}_1(0,t)}{\overline{\varepsilon}_{sys}(d,t)}. \tag{3.16}$$

Fig. 3.4 depicts the change in the gain for different τ_Δ values. For $\tau_\Delta < \tau_p$, the gain is one, and the graph exhibits a plateau, meaning that an update from S_2 does not reduce the system estimation error. When $\tau_\Delta > \tau_p$, the update from S_2 reduces the error and the

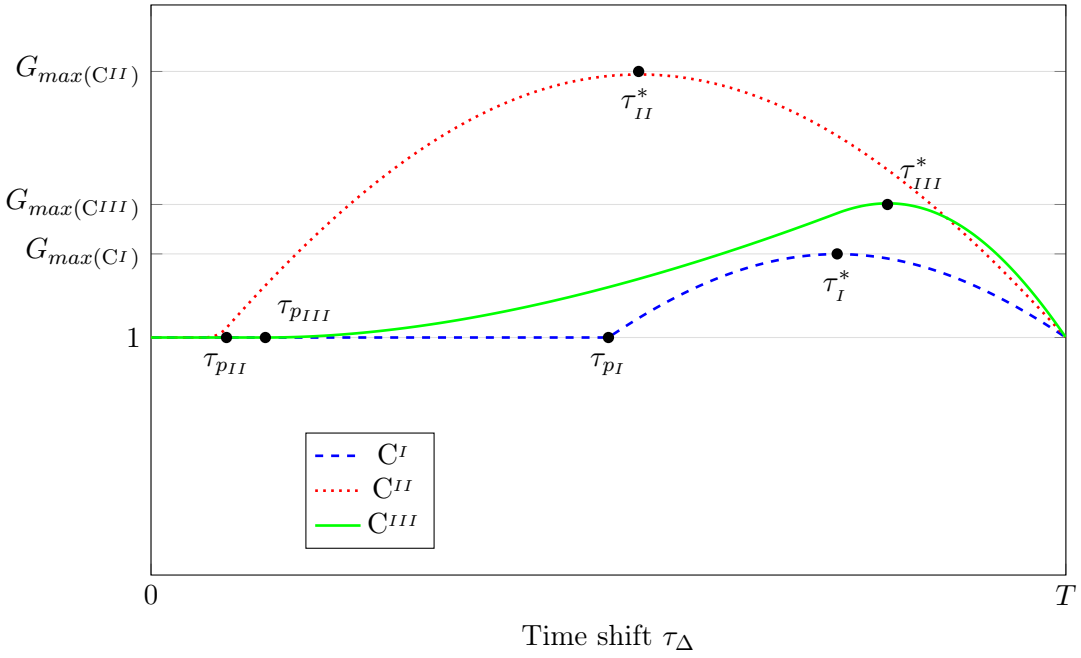


Figure 3.4: Gain change for two correlated sources with equal update rates, as the time shift between consecutive updates from the two sources changes from zero to T, for three different covariance classes. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.025$, $\theta_3 = 0.0125$, and $T = 1$, with distance $d_2 = 10m$.

gain increases. At $\tau_\Delta = \tau^*$, the system generates the lowest system estimation error and the highest gain. When the time shift for the update from S_2 is greater than τ^*, the gain decreases again. The reason for such behaviour lies in the fact that the error caused by outdated information from S_1 is no longer effectively compensated by an update from S_2. Furthermore, the scaling parameters influence the values of τ_p and τ^*; however, they do not affect the shape of the curve. Higher dependence on spatial separation in the second non-separable covariance model C^{III} causes non-symmetrical behaviour as seen in Fig. 3.4. For all three covariance models, we can establish that there always is an optimal time shift, i.e., $\tau_\Delta = \tau^*$, for which receiving an update from correlated source is most beneficial.

3.3.2 Optimal Time Shift

To determine τ^* we calculate the derivative of (3.15), i.e., $\partial\overline{\varepsilon}_{sys}/\partial\tau_\Delta$. For separable model C^I and non-separable model C^{III}, we are able to derive closed-form expressions for the optimal time shift τ^* (equations (3.17) and (3.18)). For the second non-separable model C^{II} we cannot produce a closed form expression[2], and instead used a numerical solution to obtain Fig. 3.5.

$$\tau^*_I = \frac{d_2\theta_2 + T\theta_1}{2\theta_1}, \tag{3.17}$$

[2]We provide a detailed description of the procedure in Appendix A.

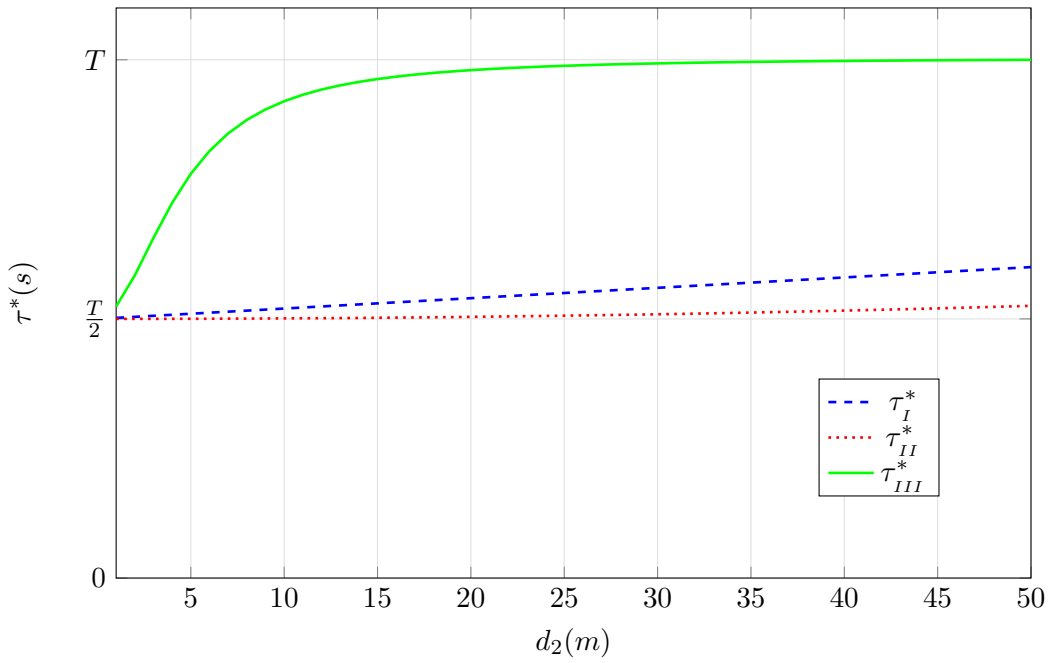


Figure 3.5: The graph shows the increase of τ^* with the distance between the two information sources, for all three covariance models. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.1$, $\theta_3 = 0.0125$, $d_2 = 10m$, and $T = 50$.

$$\tau^*_{III} = \frac{(\theta_3 d_2^2 + \theta_1)T + 2\theta_2^2 d_2^2}{\theta_3 d_2^2 + 3\theta_1}. \tag{3.18}$$

The optimal time shift depends on the update period, the distance between sources d_2, and the value of the scaling parameters. With an increase in distance the value of τ^* increases, but the value is always relative to T, as illustrated in Fig. 3.5. For the separable model (3.4), it is clear from Eq. (3.17) that τ^* is always larger than $T/2$. The same applies for non-separable model (3.5), which we determined numerically. The increase of τ^* is linear with distance for both models. The second non-separable model (3.6) has no such limitation in the value of τ^* and as the distance between the sources increases τ^* approaches T.

In this section we showed that the observed system has two notable points of interest, τ_p and τ^*, and a desirable condition: $\tau_\Delta > \tau_p$. τ_p represents the minimal time shift the update from the correlated source should have to reduce the system estimation error. Furthermore, τ_p is independent of update rates. τ^* denotes the time shift at which a received update from the correlated source would result in the minimal system estimation error. The system dynamics change when sources update with different update rates, as we explore in the next section.

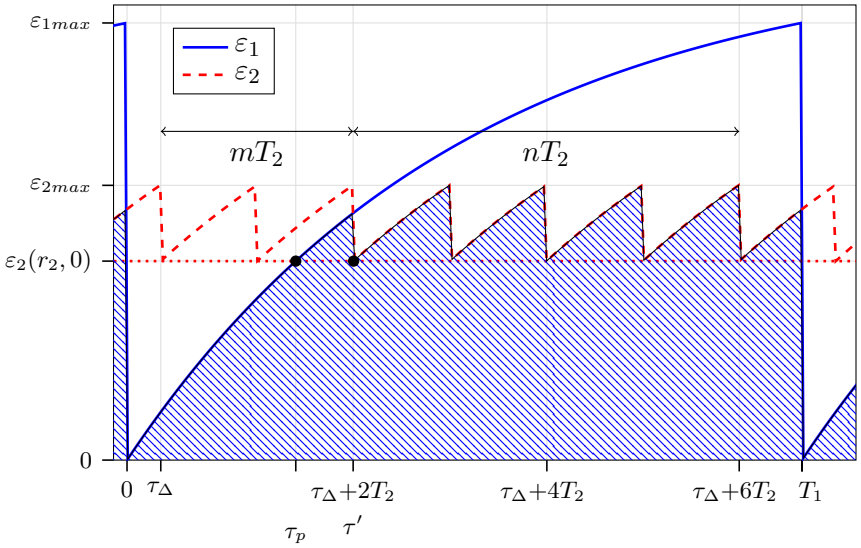


Figure 3.6: Error change over one period of T_1 for two sources when S_1 updates less often than S_2, i.e. $\lambda_1 < \lambda_2$. The dashed area represents the system estimation error. The covariance model adopted here is the one in Equation (3.4).

3.4 Different Update Rates

Information sources observing the same physical phenomenon in an IoT network may, for many reasons, update with different update rates. For example, one of the sources may transmit more frequently because the cost of its transmission is lower than the cost of transmission of the other (e.g., it may rely on a more stable energy source). In this section, we analyse two possible cases, namely when S_1 updates more or less frequently than S_2.

3.4.1 Primary Source Updates Less Frequently Than Secondary

When S_1 updates less frequently than S_2, i.e., $\lambda_1 < \lambda_2$, the system estimation error changes depending on the update rate of S_2 and the time shift τ_Δ. After S_1 sends an update, there is a time period during which any update sent from S_2 is irrelevant, because it does not reduce the error. As illustrated in Fig. 3.6, the first m updates from S_2 do not influence the average error value. The time period during which updates from S_2 are irrelevant is described as τ_p. Note that τ_p is independent of update rates and is calculated as shown in (3.11).

To calculate the system estimation error we integrate $\varepsilon_{sys}(d,t)$ over one period of T_1 as expressed in Eq. (3.19).

$$\overline{\varepsilon}_{sys}(d,t) = \frac{1}{T_1}\left(\int_0^{\tau'} \varepsilon_1(0,t)dt + n\int_0^{T_2} \varepsilon_2(d_2,t)dt + \int_0^{T_2-\tau_\Delta} \varepsilon_2(d_2,t)dt \right) \tag{3.19}$$

where, for $0 \leq t \leq T_1$:

$$\Delta_1(t) = t \text{ and } \Delta_2(t) = t - \tau_\Delta,$$

with:

$$n = \left\lfloor \frac{T_1 - \tau'}{T_2} \right\rfloor, \tag{3.20}$$

$$\tau' = \tau_\Delta + \lceil m \rceil T_2, \tag{3.21}$$

$$m = (\tau_p - \tau_\Delta)\lambda_2, \tag{3.22}$$

and with the condition:

$$\tau' \geq \tau_p.$$

τ' represents the time shift between an update from S_1 and the first update from S_2 which reduces the system estimation error (see Fig. 3.6). Before that update, the first m updates from S_2 are irrelevant as they do not reduce the error. The number n represents the number of full transmissions from S_2 between τ_p and the next update from S_1. For example, in Fig. 3.6, $n = 4$. It is necessary to integrate $\varepsilon_2(d_2, t)$ n times over one period of T_2, as every time S_2 sends an update $\Delta_2(t) = 0$. What remains is only part of an S_2 update which reduces the system estimation error only partially because an update from S_1 intervenes.

We illustrate how the gain of using correlated information increases as the update rate of the secondary source increases in Fig. 3.7. To obtain results we assume that the update rates are related by an integer factor k as follows:

$$k\frac{1}{\lambda_2} = \frac{1}{\lambda_1} \tag{3.23}$$

where $k \in \{1, 2, 3, ...\}$. We determine the gain using Eq. (3.16). As expected, the more often the secondary source transmits, i.e., higher factor k, the more beneficial the updates from the secondary sources are. However, the benefits of using correlated information are limited, indicating that the primary source can not entirely rely just on the information from a correlated source.

For the source of interest the benefit of using updates from a correlated source is limited, as an update from correlated source can reduce estimation only to a value of $\varepsilon_2(d_2, 0)$. This limit can be calculated by increasing k to infinity. The result is the maximum possible reduction in the system estimation error, as presented in Equations (3.25), (3.26), and (3.27), for each covariance model. In all cases, the maximum gain falls rapidly with distance

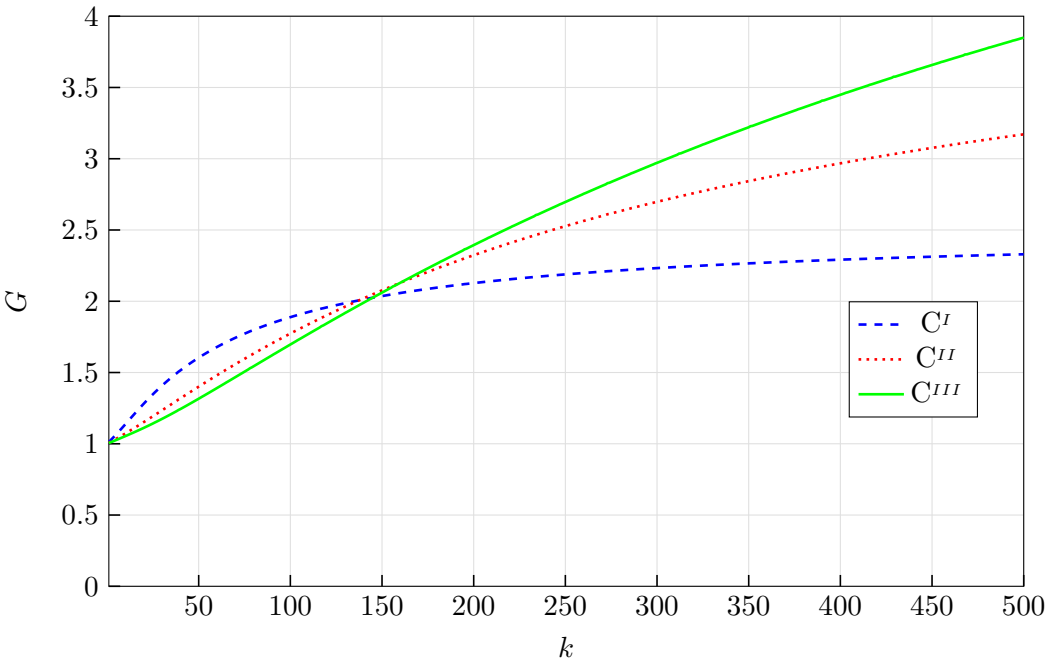


Figure 3.7: The error gain change as S_2 increases the frequency of its updates, while the period between updates from S_1 stays the same, for three covariance models. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.025$, $\theta_3 = 0.0125$, $d_2 = 10$, and $T_1 = 50$.

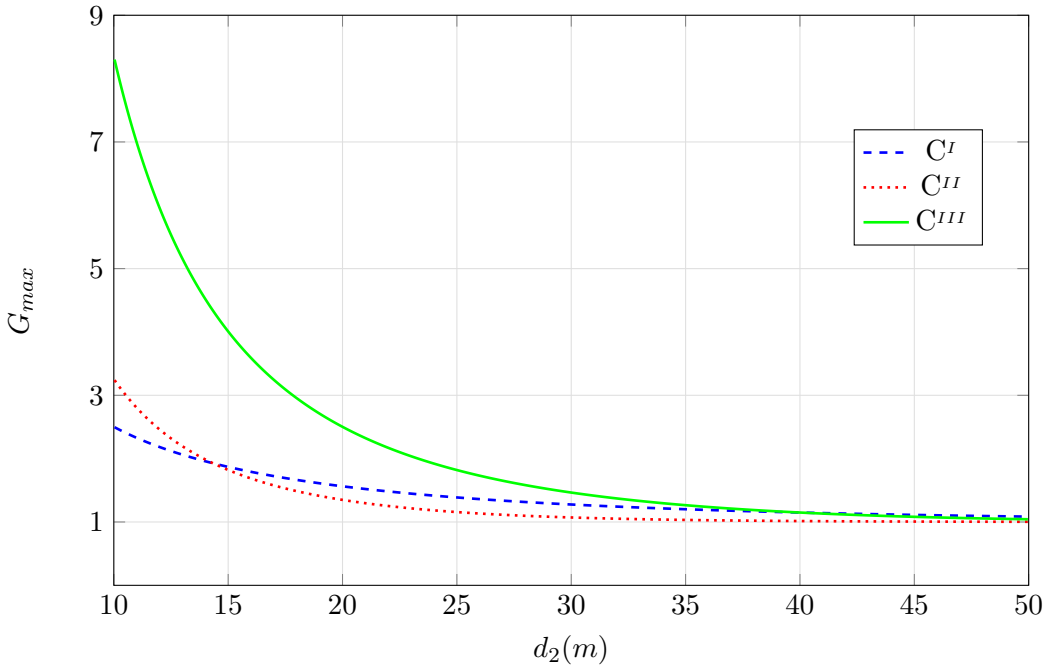


Figure 3.8: The graph shows the decrease of G_{max} with the distance between information sources, for all three covariance models. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.025$, $\theta_3 = 0.0125$, and $T_1 = 50$.

as presented in Fig. 3.8. G_{max} depends only on the scaling parameters and the distance between the sources.

$$G_{max}(d,t) = \frac{\overline{\varepsilon}_1(0,t)}{\lim_{\lambda_2\to\infty}\overline{\varepsilon}(d,t)}, \tag{3.24}$$

and the limits when sources are correlated according to models C^{I}, C^{II}, and C^{III}, respectively, are:

$$\lim_{\substack{\lambda_2\to\infty\\ T_2\to 0}}\overline{\varepsilon}_{I}(d,t) = \frac{1}{T_1}\left(\tau_{p_I} + \frac{exp(-2\theta_1\tau_p)-1}{2\theta_1} + \big(T_1-\tau_{p_I}\big)\big(1-exp(-2d_2\theta_2)\big)\right), \tag{3.25}$$

$$\lim_{\substack{\lambda_2\to\infty\\ T_2\to 0}}\overline{\varepsilon}_{II}(d,t) = \frac{1}{T_1}\left(\tau_{p_{II}} + \frac{1}{5\theta_1(\theta_1\tau_{p_{II}}+1)^5} - \frac{1}{5\theta_1} + (T_1-\tau_{p_{II}})(1-\frac{1}{(1+-2\theta_2^2d_2^2)^4})\right), \tag{3.26}$$

$$\lim_{\substack{\lambda_2\to\infty\\ T_2\to 0}}\overline{\varepsilon}_{III}(d,t) = \frac{1}{T_1}\left(\tau_{p_{III}} + \frac{\exp(-2\theta_1\tau_p)-1}{2\theta_1} + (T_1-\tau_{p_{III}})(1-\exp(-2\theta_2^2d_2^2))\right). \tag{3.27}$$

Note that we derive above expression in Appendix A along with a complete $G_{max}(d,t)$ expression for each covariance model. Whenever the correlated source updates more often, only updates which effectively reduce the system estimation error are beneficial to the system. In general, the more updates S_2 sends, the greater is the gain of using updates from the correlated source. However, due to information being collected at a different location, there is a limit to the gain, but as demonstrated in this subsection, using multiple updates from the correlated source can be useful to the overall performance of the system.

3.4.2 Primary Source Updates More Frequently Than Secondary

When S_1 updates more frequently than S_2, i.e., $\lambda_1 > \lambda_2$, using correlated information becomes less beneficial. In such a case, an update from S_2 reduces the system estimation error only every few updates from S_1, as illustrated in Fig. 3.9, where updates from S_2 reduce the system estimation error every third update from S_1. Furthermore, the update from the S_2 has to arrive when the error, due to outdated information from S_1, is higher than $\varepsilon_2(d_2,0)$, i.e., $\varepsilon_1(\Delta_1(t),0) > \varepsilon_2(d_2,0)$ with $\Delta_1(t) = \tau_\Delta$. In this case, the system estimation error depends mostly on updates from S_1.

We assume that the update rates are related through an integer factor j as following:

$$\frac{1}{\lambda_2} = j\frac{1}{\lambda_1}, \tag{3.28}$$

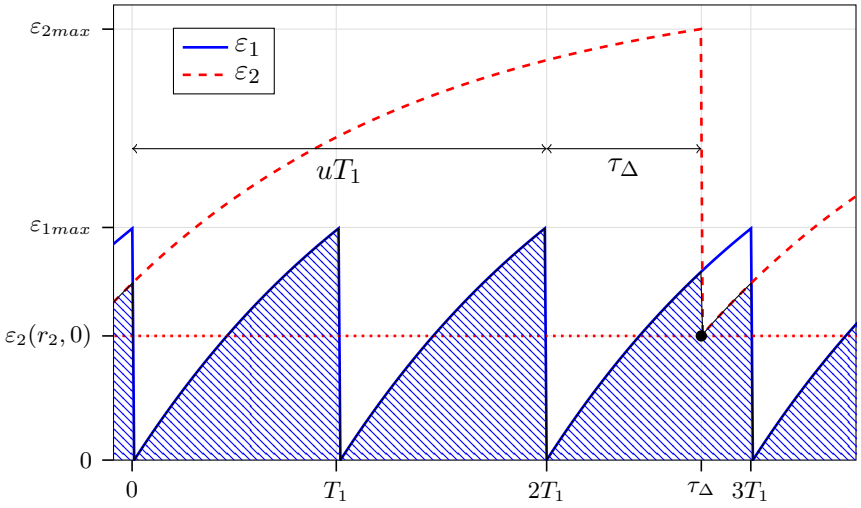


Figure 3.9: Error change over one period of T_2 for two sources when S_1 updates more often than S_2, i.e. $\lambda_1 > \lambda_2$. The dashed area represents the system estimation error. The covariance model adopted here is the one in Equation (3.4).

where:

$$j \in \{1, 2, 3, ...\}.$$

To calculate the system estimation error, we integrate $\overline{\varepsilon}_{sys}(d, t)$ over one period of T_2, as expressed in Eq. (3.29).

$$\overline{\varepsilon}_{sys}(d,t) = \frac{1}{T_2}\left(u\int_0^{T_1}\varepsilon_1(0,t)dt + \int_0^{\tau_\Delta}\varepsilon_1(0,t)dt + \int_0^{T_2-\tau_\Delta-nT_1}\varepsilon_2(d_2,t)dt\right), \tag{3.29}$$

where, for $0 \leq t \leq T_2$:

$$\Delta_1(t) = t \text{ and } \Delta_2(t) = t - nT_1 - \tau_\Delta,$$

with:

$$u = j - 1,$$

and with the condition:

$$\tau_\Delta \geq \tau_p.$$

An update from S_2 can reduce only a fraction of the overall system estimation error, as depicted in Fig. 3.9. The higher the update rate for S_1, the lower the benefit of using updates from S_2 is. Furthermore, the timing of updates from the correlated source plays a

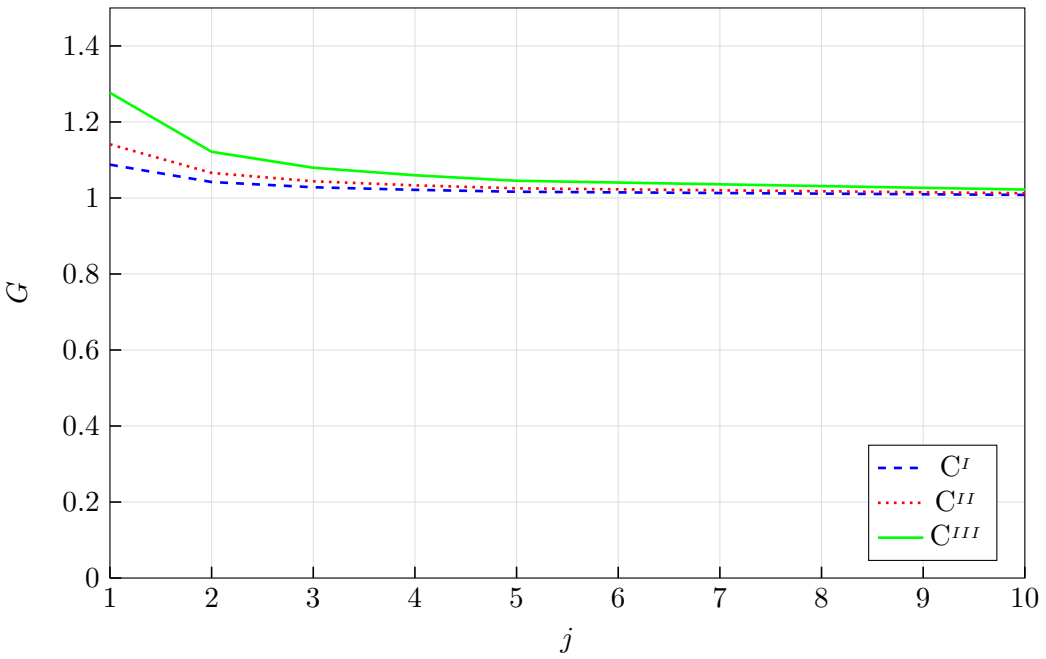


Figure 3.10: The error gain changes as S_2 decreases the frequency of its updates, while the period between updates from S_1 stays the same, for three covariance models. The graph was obtained using scaling parameters $\theta_1 = 0.5$, $\theta_2 = 0.025$, $\theta_3 = 0.0125$, $d_2 = 10$, and $T_1 = 1$.

crucial part whenever the source of interest updates more often than the correlated source. It may happen that updates from S_2 will not reduce the system estimation error; in such a case, an update from S_2 arrives with delay smaller than τ_p after S_1 sends update, i.e., $\tau_\Delta < \tau_p$.

The higher the frequency of updates from the source of interest is, i.e., the higher the factor j is, the lower the chance that an update from the correlated source will arrive at the right time. The results in Fig. 3.10 demonstrate such behaviour and indicate that when the primary source updates more often using correlated information is much less beneficial. However, updates from the correlated source may still be advantageous to the system, provided they arrive at the right moment.

3.5 Data Analysis

In this section, we bridge the gap between the theoretical analysis presented in the previous two sections and the real world. In the first part of this section, we analyse data provided by the Intel Berkeley Research laboratory [90] to extract scaling parameters for the covariance models described earlier. In the second part, we use the covariance model with the extracted scaling parameters to demonstrate how the lifespan of the source of interest, S_1, can be extended by using updates from a correlated source, S_2. We conclude the section by comparing our approach to clustering algorithm as presented in the literature.

3.5.1 Extracting Correlation

In our analysis, we use temperature and humidity measurements collected from nine sensors deployed at Intel Berkeley Research laboratory [90], University of California. The data was gathered in March 2004. We focus on a subset of available sensors residing in one large open space office. Sensors deployed in the same room are subject to roughly the same environmental factors, e.g., air circulation, thus ensuring that causation for humidity and temperature changes is the same for every sensor we have selected in our analysis[3].

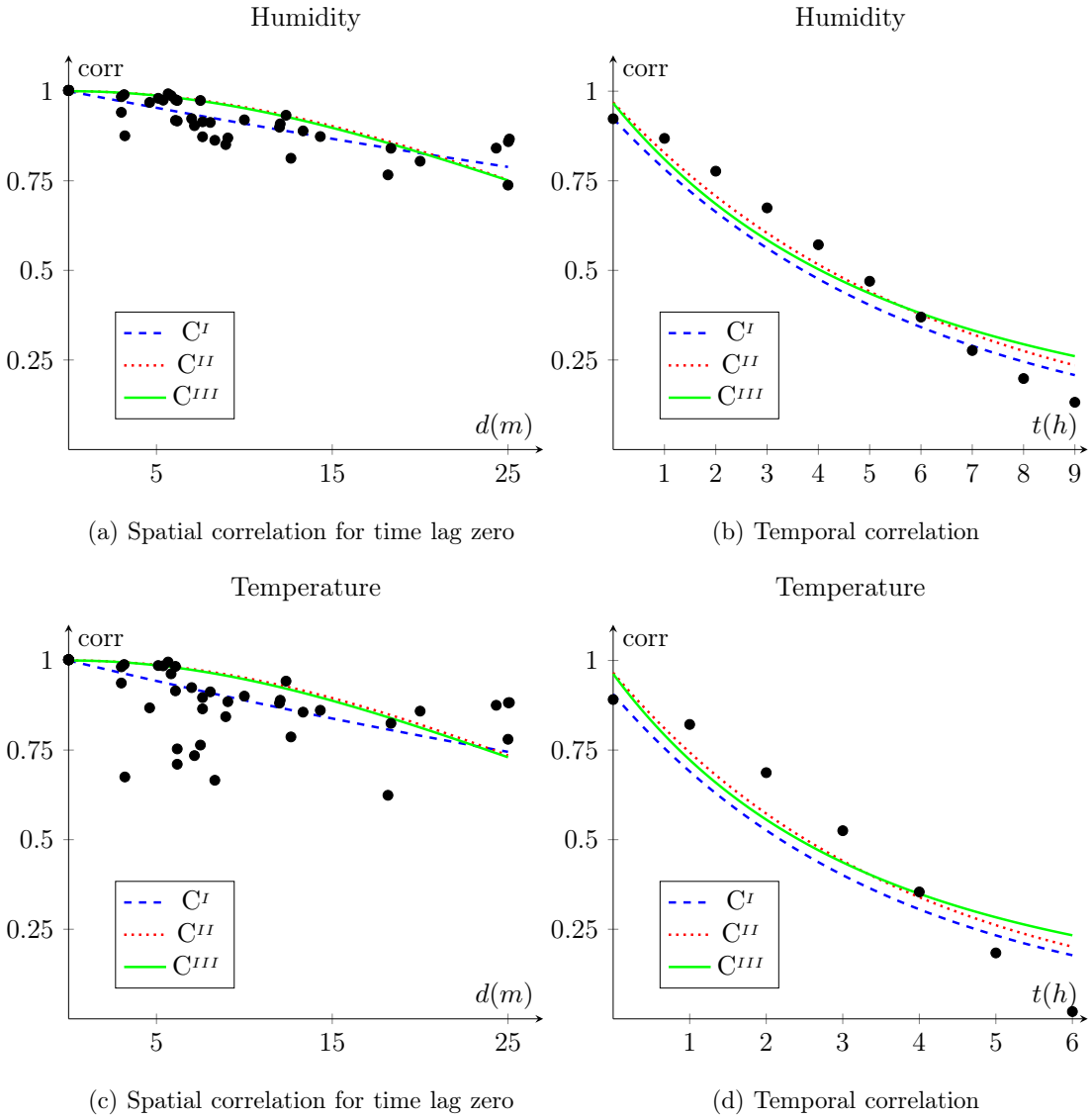
We apply the technique presented in [91] to determine the scaling parameters value for all three covariance models. To be able to use the technique, we have to ensure time-domain consistency. The time consistency being that for every sensor, we have the same amount of readings(measurements) all obtained at the same time instance. Such consistency is vital when comparing readings obtained at different time lags. For example, in [91] the authors use daily mean wind measurements to determine the scaling parameters and their time lag unit is a day. In our work, we average multiple temperatures and humidity readings reported by each sensor over an hour to ensure time-domain consistency[4]. We selected an hour as changes in temperature are gradual, and the hourly average is as good as 30 minutes or 15 minutes one would be. Additionally, averaging enables us to reduce the random measurement error.

We calculate empirical space correlation for hourly averages of measurements from the data set using Pearson's correlation coefficient formula for samples. We start by calculating the spatial correlation for time lag zero. At time lag zero, we compare measurements taken by each sensor at the same hour. Each point in Fig. 3.11(a) and 3.11(c) represents the empirical spatial correlation coefficient for one pair of sensors separated by distance d.Nine sensors yield 81 different pairs; of those, nine pairs are a sensor paired with itself. For such a pair, correlation at time lag zero is one. Therefore, in Fig. 3.11(a) and Fig. 3.11(c) there is only 37 points visible, as there are in total nine points with the same value(when sensor is paired with itself), and all others are points doubled. We repeat this process for multiple time lags, i.e., compare measurements taken from different sensors hours apart, to calculate temporal correlation.

To obtain temporal correlation, we calculate the average of empirical spatial correlation coefficients for a specific time-lag. For example, we obtain the temporal correlation at time zero in Fig. 3.11(b) by calculating the average of spatial correlation coefficient points in Fig. 3.11(a). As can be seen in Fig. 3.11(b) and 3.11(d), the higher the time lag between the data points, the lower the empirical correlation. The empirical graphs show better temporal correlation for humidity in comparison to temperature. One noticeable difference is that humidity data is correlated even when time lag is nine hours, in contrast to temperature, which has negligible correlation after around six hours.

[3]We provide a more in-depth description of the dataset and selection of sensors in Appendix B.

[4]A reader may check the available online IPython notebook to understand the scaling parameters extraction process better: Github.com/hribarjernej89/IntelLabDataAnalysis



(a) Spatial correlation for time lag zero

(b) Temporal correlation

(c) Spatial correlation for time lag zero

(d) Temporal correlation

Figure 3.11: Empirical space and time correlation points plotted alongside covariance models with scaling parameters extracted from the data.

With correlation coefficient points determined, the last step in our data analysis is to fit models to calculated points. All three covariance models (3.4), (3.5), and (3.6) provide a reasonable fit to the data. In Fig. 3.11 we plot all three covariance models, with scaling parameters extracted from the data, alongside the empirical space-time correlation points. For example, the separable covariance model (3.4) should have scaling parameter values $\theta_1 = 0.272$ and $\theta_2 = 0.012$ to describe temperature spatial-temporal correlation. The pattern repeats in non-separable covariance models where temporal scaling parameter, θ_1, also has bigger value in comparison to spatial scaling parameter,θ_2. The same is true for humidity, where the non-separable model (3.5) scaling parameters are $\theta_2 = 0.016$ and $\theta_1 = 0.062$ to best describe spatial-temporal correlation. In general, scaling parameters describing humidity correlation have lower values in comparison to scaling parameters describing temperature

correlation. In the next subsection, we adopt the separable covariance model, i.e., (3.4), for temperature and the first non-separable model, i.e., (3.5), for humidity, with scaling parameters values mentioned above.

3.5.2 Extending Lifespan

Sensors consume energy to create information, i.e., measure the observed physical phenomena. In particular, battery-powered sensors are of great concern as every update brings the device closer to expiration. By using updates from the correlated source, the source of interest, S_1, can reduce its update rate and consequently extend its lifespan at the expense of the correlated source. The lifespan of a source is measured from the moment the sensor is deployed in the network, to the moment the source is considered non-functional. We consider a source non-functional when the source depletes all its available energy and ceases to transmit updates. We model source lifetime $\mathcal{L}_i$ as in [92]:

$$\mathcal{L}_i = \frac{E_0}{P_c + \lambda_i \mathbb{E}[E_r]}. \tag{3.30}$$

E_0 represents the initial total non-rechargeable energy. P_c is the continuous power consumption that the information source uses for sensing and basic operations. $\mathbb{E}[E_r]$ denotes the expected energy needed to transmit a status update, and λ_i represents the source update rate. In our case, we selected the lifetime model parameters to reflect the power consumption in a battery-powered sensor. We base our power consumption parameters on the Mica2Dot board, a sensor board used in Intel Berkeley Research laboratory deployment [90]. We estimate that the sensor requires five milliseconds to produce the measurements and that the sensor's radio is active for one millisecond to transmit the update. Using information from the datasheet we determine that $P_c = 45\mu W$ and $0.2mJ$ of energy is required to obtain and transmit the measurement, i.e., $\mathbb{E}[E_r] = 0.2mJ$. We consider that the sensor would be powered by a $620mAh$ Lithium coin battery suitable for the desired application and sensor type. Note that λ_i in our case refers to the number of updates by a sensor transmitted in an hour.

In our experiment, we focus on a system with a pair of correlated sources as described in the previous sections. The system has a primary interest in information gathered from one source, S_1, and can use updates from a correlated source, S_2, to prolong the lifetime of the source of interest. An example of such a system would be a battery-powered sensor S_1 deployed directly at the center of the observed physical phenomenon, with S_2 deployed as a backup at a more distant location but with a connection to the power grid. In such a case, the system can use updates from S_2 to reduce the number of updates sent by S_1 and prolong its lifetime while simultaneously maintaining the required accuracy of information, i.e., keep the system estimation error below a certain threshold. In other words, the source of interest S_1 can reduce its update rate because information received from correlated source

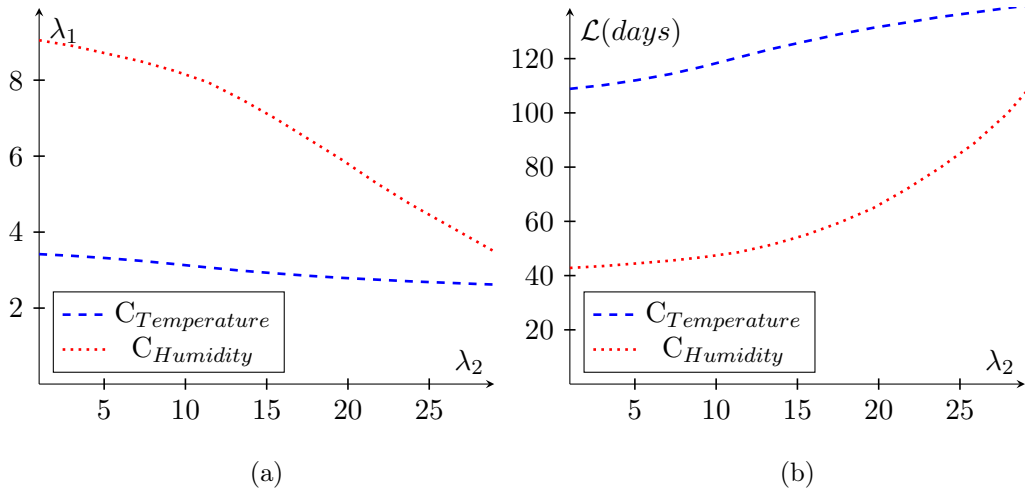


Figure 3.12: (Left) The decrease in update rate for S_1 when S_2 update rate increases while the system estimation error is fixed. (Right) The increase in expected device lifetime by taking advantage of correlation as a function of λ_2, as the system estimation error remains fixed. Both graphs were obtained by using distance $d_2 = 4m$.

S_2 is reducing the system estimation error. We set the system estimation error value to 7.5% for observing temperature and 2% for humidity. To obtain results presented in Fig. 3.12 we calculated required update rate from source S_1, λ_1, to satisfy the system estimation error requirement depending on the S_2 update rate, λ_2.

The higher the rate of updates from the correlated source S_2 is, the lower the required rate of updates from the source of interest and the longer its expected lifetime. Fig. 3.12 shows the change of update rates and the lifetime increase of the sensors by taking advantage of correlated updates, for cases of temperature and humidity. In both cases, significant improvement in the lifetime of sensors is achievable. For humidity, benefits are more significant because humidity exhibits better spatial-temporal correlation in comparison to temperature. As noted in the previous subsection, covariance scaling parameters for humidity were significantly smaller in comparison to temperature, indicating that humidity variation over time and space is much lower. Consequently, the gain of using correlated information is much higher for humidity.

Translating our experiment into a real world deployment where it is expected that information for temperature and humidity are transmitted simultaneously, the correlated source S_2 should transmit 30 updates, while battery-powered source S_1 would transmit only three updates in a hour. Such a system is capable of gathering information accurately for three months. Our results show that even a small decrease in the sensor update rate has a notable impact on the sensor lifetime. The gain depends on the variation of the observed physical phenomenon across time and space. In our example, the temperature data exhibit high variation, yet sensor lifetime increase is 20%. Whenever data is more highly correlated, e.g., humidity, relying on more frequent updates from a correlated sensor can more than double the source lifetime.

3.5.3 Energy Saving Comparison

In this subsection, we compare our energy savings approach with the approach presented in [37]. Both approaches have to meet the same accuracy constraint, meaning that the estimation error threshold is equal in both cases. We set the accuracy constraint to 95%, in other words, we set the maximum estimation error to 5%. Note, that we calculate estimation error using Eq. (3.7). For our approach, we calculate the update rate such that each source satisfies the set accuracy constraint.

In our implementation of the approach presented in [37] we try to reproduce their scheduling and clustering algorithm as it would work in our simulated sensor network. We achieve that by clustering sources based on their proximity to cluster heads; each source joins the cluster group of the cluster head with the shortest distance to it. Once the clusters are formed, we calculate the super cycle. A supercycle is a period within which every source in a cluster sends one update to the sink, while still meeting the information accuracy requirements. We schedule the transmission of source updates evenly throughout the super cycle. This means that sources' update rate is the reciprocal of the time of the supercycle, i.e., $\lambda_c = 1/T_{sc}$. The first source to transmit in the cycle is the cluster head, followed by the nearest source, followed by second nearest and so forth. With such an implementation, we effectively mimic the proposed approach.

The source's energy consumption is directly related to its update rate in both approaches. We calculate the energy savings (P_{es}) as the ratio between P_0 and P_n as follows:

$$P_{es} = \frac{P_0 - P_n}{P_0} \times 100\%. \tag{3.31}$$

P_0 represents the energy a source uses when transmitting with an update rate that enables the source to meet the required accuracy constraint on its own. Furthermore, P_0 is the same for every source in both cases. P_n is the energy sources use when taking advantage of correlated sources or when using Yu et al. approach. We calculate P_n using an update rate we obtain using simulation. We calculate both P_0 and P_n as $P_c + \lambda\mathbb{E}[E_r]$, with the values for P_c and $\mathbb{E}[E_r]$ as we used in the previous subsection.

Sources in our simulation are uniformly randomly distributed in a room of size 15×15 m. Such a room size is comparable to the room size in the Intel lab deployment where the sensor data was collected. Furthermore, we use non-separable covariance model (3.5) with scaling parameters extracted from humidity data. In [37] the authors showed that three clusters yield a good result. Therefore we decided to cluster sources into three groups. In our comparison, we use the average energy consumption of sources in the deployment. In both cases, we assumed no contention for the transmission channel and instantaneous transmission of updates.

In Fig. 3.13 we show the increase in energy savings as the number of sources in the room increases. When the number of sources is low, the clustering approach is very efficient due

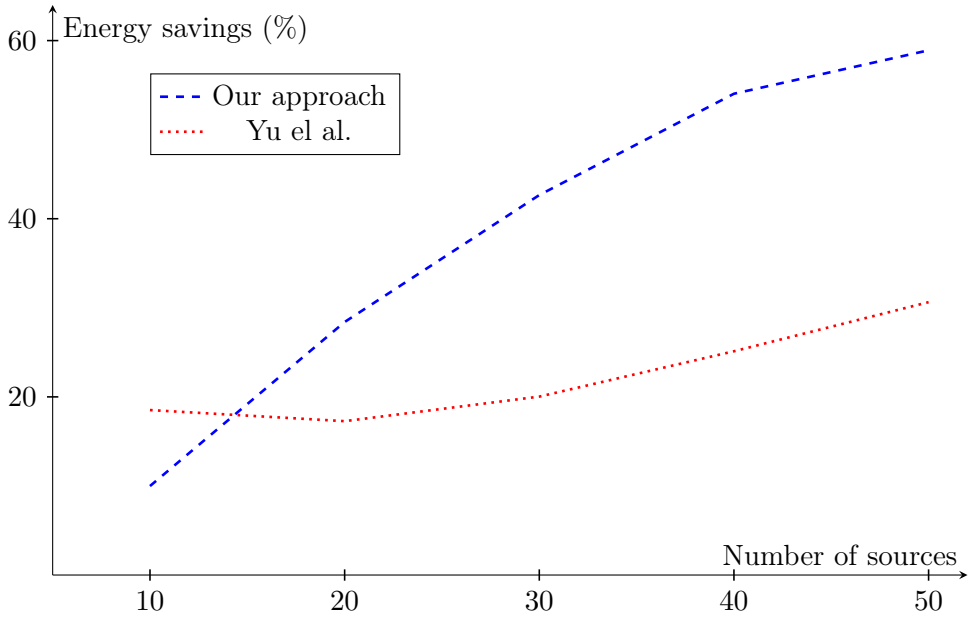


Figure 3.13: Average energy saved per source by using information from correlated information sources in comparison to energy saved used by clustering and scheduling algorithm proposed in the literature.

to the evenly scheduled updates within the supercycle. However, as the number of sources increases the same scheduling becomes a handicap and reduces the efficiency of clustering approach. As the number of sources increases our approach of taking advantage of correlated information proves to be better. Our result indicates that energy savings can be improved much more than what the current state of the art is offering.

3.6 Conclusion

In this chapter, we analysed two correlated information sources and showed how the different spatial-temporal variation of the observed physical phenomenon influences the optimal use of updates from the two sources. The system employs information from the correlated information source to minimise the system estimation error when estimating the value of an observed physical phenomenon. We established that there exists a minimal time shift between the two sources' updates, for all three tested covariance models, so that the arrival of an update from a correlated source will reduce the system estimation error. We also derived the optimal time shift between the two sources, for all three tested covariance models, for which the system estimation error is minimal. We examined some real data and extracted the covariance model parameters, to provide the reader with a realistic feel for scaling parameters values and the applicability of our analysis in a real scenario. We demonstrated that using correlated information results in a significant increase of sensor lifetime.

When billions of sensors will be connected to one network, as the IoT vision promises,

using available correlated information will become essential to improve the performance of energy constrained sensors. In this chapter, we focused on extending sensor lifespan; another gain of our approach is reduced contention for the transmission channel. By relying on the concept of AoI, we offered an alternative take on the problem of accuracy, decision-making, and lifespan of sensors in a network of sensors. A more accurate decision requires more frequent updates, and more updates lead to a shorter lifespan for battery-powered sensors. However, by sharing information and making use of correlated information, benefits such as more accurate decisions, or an increase in sensors' lifespan, or reduction of contention for the transmission channel can be obtained.

In the next chapter, we focus on a scheduling mechanism design for a system of multiple correlated sources. The main challenge we are addressing is on how can we effectively take advantage of correlated information. To that end, we employ Deep Reinforcement Learning (DRL) to design a scheduling mechanism to increase sensor lifespan for the energy-constrained sensors, by prolonging the time between consecutive updates transmitted by these sensors.

4 Energy-Aware Deep Reinforcement Learning Based Scheduling Mechanism

Energy-Aware Deep Reinforcement Learning Based Scheduling Mechanism

> *Applications of reinforcement learning are still far from routine and typically require as much art as science.*
>
> Richard S. Sutton and Andrew G. Barto , *2018*

In this chapter, we propose an updating mechanism capable of learning the frequency of updates, i.e., how often a low-power sensor should transmit updated readings. Our approach prolongs battery-powered sensors' lifetime by leveraging correlation exhibited in observations collected, without hindering the accuracy of the information provided to services relying on these observations.

The technical work presented in this chapter is based on U.S. provisional patent application "Method and System For Energy Aware Scheduling For Sensors" and our works "Using Deep Q-Learning to Prolong the Lifetime of Correlated Internet of Things Devices" presented at IEEE ICC 2019 Workshop, "Utilising Correlated Information to Improve the Sustainability of Internet of Things Devices" presented at IEEE World Forum on Internet of Things 2019, and the journal paper "Energy Aware Deep Reinforcement Learning Based Scheduling for Sensors Correlated in Time and Space" currently under submission.

4.1 Introduction

The Internet of Things (IoT) ecosystem which we presented in chapter 2 connects billions of low-power sensors into one gargantuan network. These sensors are being deployed to provide services in smart cities [4], Industry 4.0 [5], smart agriculture [6], and other IoT applications. Many of these devices are low-cost sensors powered by non-rechargeable batteries. Their role is to provide sensed information to services, which use this information to make decisions. For example, in smart agriculture, a service controlling an irrigation system requires information from various sensors to decide which fields to water. The main

challenge is to provide accurate and up-to-date information to services while keeping the battery-powered devices functional for as long as possible.

In a system of multiple sensing devices observing the same physical phenomenon, it is expected that the information collected will be correlated in time and space. In the previous chapter we have demonstrated that by relying on this correlation it is possible to increase the time between consecutive updates by each sensor, thereby increasing its lifetime, without compromising the accuracy of the information provided to the IoT service. In the absence of up-to-date information from one sensor, the system can rely on more recent information obtained from a correlated sensor. In this chapter, we propose a Deep Reinforcement Learning (DRL)-based scheduling mechanism capable of determining how frequently each low-power sensor should transmit its observations so as to furnish the service with accurate information while maximising the lifetime of the network.

We consider an IoT system where low-power sensors transmit periodic updates to a gateway. The gateway is able to schedule when the next update by each sensor should occur. The gateway relies on a data-driven approach to make this determination, by considering the energy available to each low-power sensor and the need for fresh updates, according to concepts related to the Age of Information (AoI) we discussed in previous chapters. Using the DRL algorithm we can design a solution capable of improving its behaviour by learning from past experiences.

Multiple, often non-trivially connected, factors impact the decision of when a particular low-power sensor should transmit new information. These factors can be external to the sensor, such as whether a nearby sensor has recently transmitted updated information, changes in the observed physical phenomenon, etc., or internal to the sensor, e.g., the remaining energy, transmission power, location, etc. The use of DRL enables us to design a scheduling mechanism capable of determining the sensors' update interval by weighing all relevant factors to make an efficient decision.

In our design we leverage AoI logic and insights we obtained from the analysis we carried out in the previous chapter. As we pointed out in the background chapter, the optimal update rates with which sources should send information are non-trivial [10]. Additionally, considering the correlation between status updates adds to the complexity of the problem, as an update from one source lowers the requirement for fresh information on all other correlated sources. In the previous chapter we proposed to take advantage of newer information from correlated sources to improve the energy efficiency of battery-powered sources by prolonging the times between sources' consecutive updates. The authors in [56, 58, 59] analysed from different perspectives a system with correlated sources and all considered only the impact of correlation on the timeliness of information to establish the desired frequency of updates. We present a solution that considers both the timeliness of the information and the energy available to the sources to arrive at a scheduling of information updates that improves network-wide energy efficiency.

Our work is also related, but differs in crucial aspects, to the various approaches proposed for energy efficiency in the context of Wireless Sensor Networks (WSNs) [27–31]. Most proposed works for WSN, which we discussed in chapter 2, rely on detection or reconstruction of the observed phenomena, through data prediction or model-based active sampling methods, to improve the low-power sensors' energy efficiency. In contrast, we focus on the timeliness of updates, i.e., the value of AoI, and then employ DRL to determine how to utilize correlated measurements to reduce the rate at which sources transmit their updates.

Reinforcement Learning (RL) has been applied to other energy-aware networking solutions [75–79, 81–84]. However, in chapter 2 we revealed that those works focus on exploiting devices' behaviour in the physical layer or improving energy management to improve devices' energy efficiency. In our approach we learn from the content of information collected to prolong the sensors' lifetime. We employ a DRL algorithm, to arrive at an efficient transmission schedule for sensors based on their available energy, the freshness of the information collected, and the expected lifetime of all other sensors in the network, without compromising the accuracy of the information delivered to the service.

The main contribution we present in this chapter is the design of a DRL energy-aware scheduler that is capable of determining when an IoT sensor should transmit its next observation. To design the mechanism we first take a few important steps by identifying the constraints which the low-power sensors have, quantifying the observation accuracy, and defining a multiple objective decision our mechanism will take. We employ two DRL algorithms (Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG)) to arrive at an efficient transmission schedule for sensors based on their available energy, the freshness of the information collected, and the expected lifetime of all other sensors in the network, without compromising the accuracy of the information delivered to the application. A unique feature of our solution is energy balancing: our mechanism is capable of determining to what extent the energy available to one sensor can be used to prolong the lifetime of others. We compare the performance of the two proposed mechanisms to an optimal scheduler and a baseline. Additionally, we validate it over five different datasets to demonstrate its near-optimal performance in a variety of scenarios.

The remainder of the chapter is structured as follows. In the next section, we first describe how a system of sensors collecting correlated information can estimate the accuracy of their observations. Then we define the objective of the decision-making problem that our proposed scheduling mechanism is capable of assisting with. To solve the proposed problem using a DQN [65], we describe the system dynamics using states, actions, and rewards from an RL perspective and tailor them for a DQN algorithm (Section 4.3). In Section 4.4, we perform the same to be able to employ a DDPG algorithm for the mechanism design. In both sections, we adopt a similar structure. First, we describe the DRL algorithm, proceed to define the problem in the form of a decision-making process, and conclude the section by discussing our implementation. We utilize data obtained from real deployments to show that the learned behaviour significantly prolongs the sensors' lifetime and achieves near-optimal

performance (Section 4.5). Additionally, we demonstrate the scheduling mechanism's energy awareness when deciding on sensors' transmission time. Finally, we discuss open issues and our future work in Section 4.6.

4.2 Problem Formulation

The scenario of interest to our work, as we described in details in the introduction chapter, is the use of inexpensive battery-powered sensors transmitting observations to a gateway for collection. The gateway aims to schedule the transmission of observations in such a way that the accuracy of the information collected will satisfy the service requirements while, simultaneously, trying to prolong the lifetime of the battery-powered sensors. The updating mechanism residing in the gateway decides on the low-power sensors' next update time by evaluating the accuracy of collected observations and the sensors' available energy. In what follows, we present our methodology for modelling the accuracy of the observations, and multi-objective decision our updating mechanism undertakes while considering the constraints of low-power sensors.

4.2.1 Quantifying the accuracy of observations

We consider a sensor network with N geographically distributed sensors transmitting observations to a gateway for collection. The main purpose of these sensors, denoted as $\{S_1, \ldots, S_N\}$, is to observe a physical phenomenon $Z(\boldsymbol{x}, t)$ distributed in space $\boldsymbol{x}$ and evolving in time t[1]. Sensors are deployed at positions $\boldsymbol{x}_n$ and transmit periodic observations with an update interval $T_n, n = 1, \ldots, N$. In our system, we assume that the latest received observation from a sensor replaces the previously received information as, according to the AoI paradigm, the freshest information is the most relevant in the decision making process [7]. Whenever the system receives an observation from location $\boldsymbol{x}_n$ at time t_n, the system will anticipate the arrival of the next observation from location $\boldsymbol{x}_n$ at time instance $t = t_n + T_n$. We write the collected observations into a vector $\mathbf{y} = [y_1, \ldots, y_N]^T$ with $y_n = Z(\boldsymbol{x}_n, t_n)$ where t_n is the latest time at which sensor n has reported an observation.

The system can estimate the value of the observed physical phenomenon at the desired location $\boldsymbol{x}_i$ at any time instant t using the collected information, as presented in Fig 4.1. We denote the Euclidean distance between sensor S_n and the location of interest $\boldsymbol{x}_i$ as $d_{n,i}$. With $\Delta_{n,i}(t)$ we denote the time elapsed since the system received the latest observation from sensor S_n, i.e., the AoI, $\Delta_n(t) \coloneqq t - t_n$. Using every available observation, we apply a Linear Minimum Mean Square Error (LMMSE) estimator as in [94], to estimate the observed physical phenomena using correlated observations. This approach offers a mathematically

[1]We perform the physical modeling of the observed phenomenon using observations obtained in a real IoT deployment [90, 93].

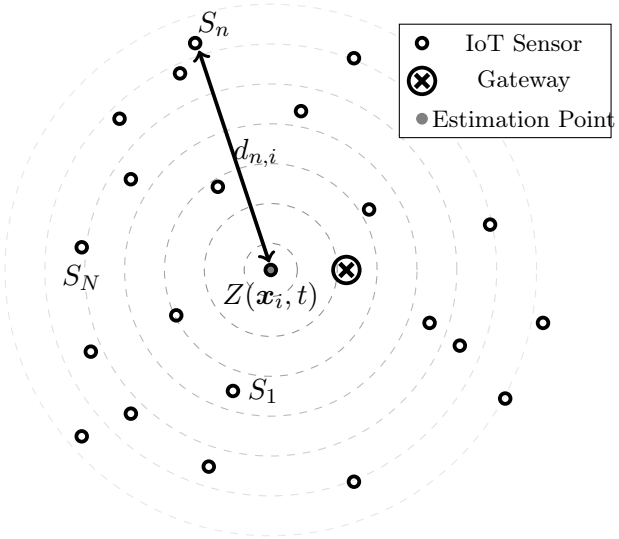


Figure 4.1: A system of N randomly distributed sensor nodes, whose observations are used to estimate the value of observed physical phenomenon, $Z(\boldsymbol{x}, t)$, at location $\boldsymbol{x}_i$ at time t.

tractable way of estimating the value of the observed phenomena and the resulting estimation error. We can approximate the value of the observed physical phenomenon at position $\boldsymbol{x}_i$ at time instant t, as:

$$\widehat{y}_i(t) = \sum_{n=1}^{N} w_{n,i}(t) y_n, \tag{4.1}$$

where $w_n(t), n = 0, \ldots, N$ are LMMSE estimator weights.

Following the analysis in [95], we obtain the LMMSE estimator weight vector $\boldsymbol{W}_i(t) = [w_{0,i}(t), \ldots, w_{N,i}(t)]^T$ as follows:

$$\boldsymbol{W}_i(t) = \Big(\boldsymbol{C}_{\boldsymbol{YY}}(t)\Big)^{-1} \boldsymbol{c}_{i,\boldsymbol{Y}Z}(t). \tag{4.2}$$

The matrices $\boldsymbol{C}_{\boldsymbol{YY}}(t), \boldsymbol{c}_{i,\boldsymbol{Y}Z}(t)$ are covariance matrices, required to determine $\boldsymbol{W}_i(t)$:

$$\boldsymbol{C}_{\boldsymbol{YY}}(t) = \begin{bmatrix} \mathrm{C}_{1,1}(t) \ldots \mathrm{C}_{1,N}(t) \\ \vdots \qquad\qquad \vdots \\ \mathrm{C}_{N,1}(t) \ldots \mathrm{C}_{N,N}(t) \end{bmatrix}; \boldsymbol{c}_{i,\boldsymbol{Y}Z}(t) = \begin{bmatrix} \mathrm{C}_{1,i}(t) \\ \vdots \\ \mathrm{C}_{N,i}(t) \end{bmatrix}; \tag{4.3}$$

in which $\mathrm{C}_{j,k}(t); j, k = 1, \ldots, N$, is the covariance of observations y_j and y_k, and $\mathrm{C}_{j,i}(t)$ is the covariance of y_j and the observed process Z at the desired location of the estimation. To obtain the required matrices we can rely on a covariance model and utilize past observations to determine its values. We adopt a separable covariance model defined in [89](the model we also adopted in the previous chapter: Eq.(3.4)). With it we model how observations collected at different instants in time and different locations relate to each other. We express

the covariance between two observations or one observation and the estimation point, with time difference $\Delta_{j,k}(t)$ and distance $d_{j,k}$ apart as:

$$\mathrm{C}_{j,k}\big(d_{j,k}, t|\theta_1(t), \theta_2(t)\big) = exp(-\theta_2(t)d_{j,k} - \theta_1(t)\Delta_{j,k}(t)). \tag{4.4}$$

Note that $\theta_1(t)$ and $\theta_2(t)$ are scaling parameters of time and space, respectively. With $d_{j,k}$ we denote the Euclidean distance between sensor S_j and the location at which the system estimates the value of the observed physical phenomenon, or between sensor S_j and sensor S_k. Both scaling parameters change over time and are extracted from the obtained observations. In our work, we follow a scaling extraction method with Pearson's correlation coefficient formula for samples, as described in [91].

The selected covariance function provides a good fit to model spatial and temporal variation for many physical phenomena. For example, in [89], the authors showed that such a covariance model could be applied to wind-speed data. In general, such a covariance model should be applicable to any data exhibiting spatio-temporal correlation. Such correlation can be observed in many IoT sensor deployments: examples include IoT systems in a smart city measuring air pollution, precipitation, or noise [4], and smart farm applications in which an IoT system monitors soil parameters [6]. Additionally, we demonstrated in [55], that the selected model is applicable to the temperature and humidity sensor data used in the evaluation section.

Every time the system employs Eq. (4.1) to estimate the value of the observed physical phenomenon it makes an error. By using matrices $\boldsymbol{C_{YY}}(t)$ and $\boldsymbol{c_{YZ}}(t)$ it is possible to determine the Mean Square Error (MSE) in the estimation as:

$$\varepsilon_i\big(\boldsymbol{x}_i, t|\theta_1(t), \theta_2(t)\big) = \sigma_Z^2 - \boldsymbol{c_{ZY}}(t)\boldsymbol{W}_i(t), \tag{4.5}$$

where $\boldsymbol{c_{ZY}}$ is the transpose of $\boldsymbol{c_{YZ}}$ defined above, and σ_Z^2 represents the variance of the observed phenomenon. The estimation error provides a measure with which the gateway can quantify the quality of the information currently provided by the sensing process: the lower the value of the estimation error, the more accurate the estimated values and the lower the need for an additional update. Hence, by measuring the average estimation error between two consecutive updates the gateway can asses how accurate and up-to-date the observations collected by the system are. In our work, we control the accuracy of our sensing process by setting as a constraint the maximum MSE of the estimator, ε^*. In short, the purpose of our proposed updating mechanism is to set sensors' update times in such a way that the average estimation error will not exceed the set target. In the next section, we describe the optimisation problem that the gateway must solve: maximising the network lifetime, constrained by the target accuracy in the measurements.

4.2.2 Gateway's Multi-objective Decision for Constrained Low-Power Devices

The goal of our updating mechanism is to prolong sensors' lifetime while maintaining the collection of information within the pre-specified accuracy range (ε^*). Additionally, the mechanism should aim for a balanced use of sensors' available energy, i.e., prevent one sensor from consuming its entire energy much earlier than others. Each sensor's lifetime depends on the frequency of transmitted observations, i.e., the time between two consecutive updates, and on the continuous power consumption that is independent of transmissions.

We assume that a non-rechargeable primary battery powers the low-power sensors. Therefore, we can model a sensor's lifetime $\mathcal{L}_n(T_n)$ using the same formulation as in the previous chapter:

$$\mathbb{E}[\mathcal{L}_n(T_n)] = \frac{E_0}{P_c + \frac{\mathbb{E}[E_{tr}]}{T_n}}, \tag{4.6}$$

where E_0 represents the sensors' starting energy and P_c is the continuous power consumption, and $\mathbb{E}[E_{tr}]$ represents the expected energy required to acquire and transmit the observation. The continuous power consumption is the power that the sensor requires to function regardless of mode of operation and depends solely on the sensor hardware components. For low-power IoT sensors, P_c is in range of a few μW or less. The energy required to transmit the observation, i.e., $\mathbb{E}[E_{tr}]$, depends on many factors such as the size of the transmitted packet, the energy required to take the measurement, and channel conditions.

Energy is not the only factor the updating mechanism has to take into account. As described in [96], low-power sensors are also constrained in terms of available computing power, memory, communication capabilities, etc. Limited processing power and memory prevent the use of a sophisticated algorithm on the sensor itself. Therefore, computationally demanding tasks when making a decision should be carried out at the gateway. Additionally, these sensors rely on low data rate transmission, meaning that communication messages between sensors and gateway should be kept to a minimum. Furthermore, to extend their lifetime, low-power sensors rely on the use of sleep mode. When a sensor is in sleep mode, the rest of the network cannot communicate with it. Consequently, the gateway has to inform each sensor, while the sensor is still in active mode, when it should wake up again and transmit the next observation. Sleep mode is supported by most Low-Power Wide-Area Network (LPWAN) standards, such as SigFox, Weightless, Long Range Wide Area Network (LoRaWAN)[2], and Narrowband IoT (NB-IoT)[97]. The low-power sensor is usually in active mode only after it has transmitted. For example, a sensor using a LoRaWAN class A radio will listen for two short time-windows after it has transmitted, as illustrated in the LoRaWAN message sequence in Fig. 4.2 [98], meaning that the updating mechanism only has a short time-window to provide a response.

[2]For more information, the reader may visit www.sigfox.com; weightless.org; lora-alliance.org, respectively.

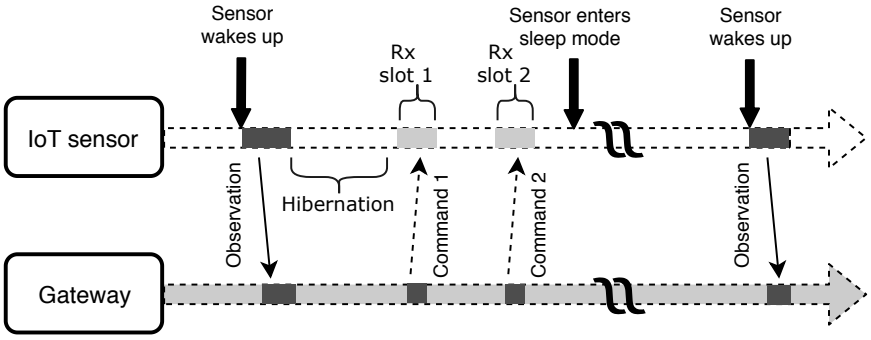


Figure 4.2: Message sequence of a low-power sensor using LoRaWAN.

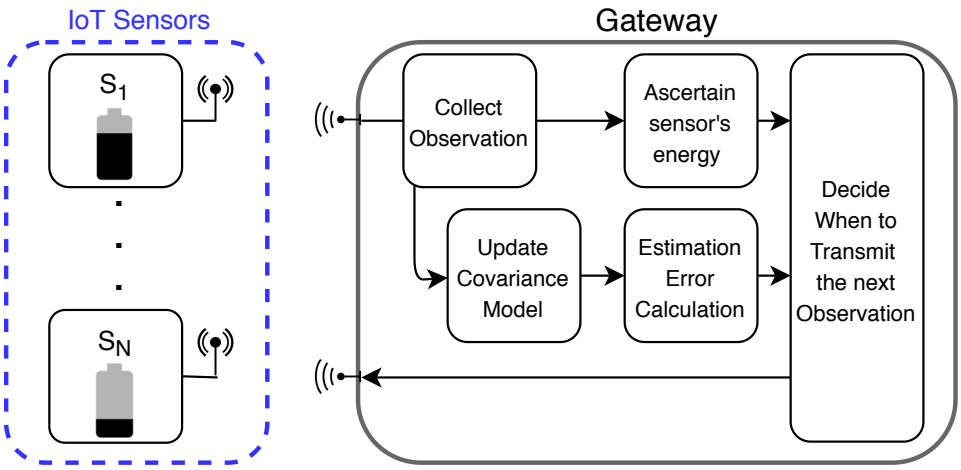


Figure 4.3: High level overview of the decision making process in the gateway.

The gateway's goal is to prolong the network lifetime. We define the network lifetime as the lifetime of the sensor with the shortest lifespan in the deployment. In other words, the network lifetime expires the moment one sensor depletes all of its energy. To that end, the gateway should aim to minimise transmissions by all sensors, i.e., increase T_n, and, when updates are required, favour sensors with higher remaining energy, all while keeping the average estimation error of the observed physical phenomenon at every location of interest below the pre-specified value, i.e., ε^*. In a real deployment, services dictate which locations are of interest for the system. In this chapter, we consider every sensor location, i.e., $\boldsymbol{x}_n$, to be a location of interest, meaning that system has to make accurate estimations at the location of every sensor while keeping sensors' power consumption to a minimum. We summarise the decision-making process in Fig. 4.3. The gateway decides on each sensor's next update time by evaluating the accuracy of the collected observation and the sensors' available energy, which it can determine from the sensor's reported power supply measurement. The gateway can then decide when the sensor should transmit its next observation.

At the time t_n when the gateway receives an observation from the n-th sensor, the gateway has to, while the sensor is in hibernation mode, decide when the sensor's next

update will be by altering the update interval T_n. If we discretise time, we can formulate the problem the gateway has to solve as follows:

$$\begin{aligned} \underset{T_n}{\text{maximize}} \quad & \min\Big(\mathcal{L}_1(T_1),\ldots,\mathcal{L}_n(T_n),\ldots,\mathcal{L}_N(T_N)\Big) \\ \text{subject to} \quad & \bar{\varepsilon}_n(t) \leq \varepsilon^*, \forall\, n=1,\ldots,N, \\ & \bar{\varepsilon}_n(t+1) \leq \varepsilon^*, \forall\, n=1,\ldots,N, \\ & \vdots \qquad , \\ & \bar{\varepsilon}_n(t+T_n) \leq \varepsilon^*, \forall\, n=1,\ldots,N., \\ & T_n \in \{1,\ldots,T_{max}\}. \end{aligned} \tag{4.7}$$

The system has to select T_n such that in every-time step until the next update from the n-th sensor the accuracy constraint is met. A new update from one sensor impacts the accuracy on every other sensor in the deployment. Note that T_{max} represents the maximum update interval the system allows any sensor to have. In real deployments, and as we also assume later in our simulations, the maximum update interval can be up to a few hours.

Intuitively, when the n-th sensor has more energy available than others, it should transmit more often, to enable other sensors (which will transmit in the future) to increase their update intervals. Such a problem is ideal for a RL approach because, the agent (in our case the gateway) can learn how to take actions that might bring negative reward in the near future but will ultimately increase the long-term reward [60] in prolonging the network lifetime.

Due to the matrix inversion in Eq. (4.2), required to determine the MSE value using Eq. (4.5) for N sensors, the expected computational complexity to solve the problem in Eq. (4.7) is of the order of $\mathcal{O}(N^4 T_{max})$. Using a deep learning approach can significantly reduce the required computational time, as it depends only on the number of neurons and the number of layers. For example, for $N = 10$, we estimate that the computation time to resolve the optimisation problem will be over ten seconds, while on the same machine, the decision using a DRL approach would take a couple of milliseconds. Therefore, using a DRL approach, the gateway can easily obtain the T_n value in the available hibernation time (1-2 seconds) for the sensors.

Furthermore, our system is highly dynamical, as each received observation impacts the covariance model's scaling parameters. As a result, the value of the MSE (Eq. (4.5)) continuously varies over time. These changes are application-specific: for example, in a smart factory, environmental changes are persistent due to many factory processes simultaneously impacting the observed environment. In contrast, the changes in a smart farming scenario tend to be much more gradual. The updating mechanism has to anticipate such changes, and by employing DRL, we enable the updating mechanism to learn how to set sensors' transmission times to near-optimal values in the presence of an ever-changing environment.

In the two next sections, we model the problem in the form of relevant states, actions, and rewards to be able to apply DQN and DDPG algorithms, respectively. Then, by employing either one of the two DRL algorithms, the gateway can arrive at a long-term updating policy to collect accurate observations and prolong the network lifetime.

4.3 Deep Q-Learning Based Approach

As we presented in the background chapter, the use of RL allows an agent to learn its optimal behaviour, i.e., a set of actions to take in every state, solely from interactions with the environment. In our case, the learning agent resides in the gateway. The agent interacts with the environment by setting sensors' update intervals, and through the MSE it can assess the impacts of those decisions. Additionally, the remaining energy available on each low-power sensor and their latest update interval aid the agent's decision-making process. Such a system exhibits non-Markovian properties - a part of the environment is hidden from the agent's view and is only partially observable. However, DRL has been proven to be applicable even when the system has non-Markovian properties [99]. In such a case, using an Artificial Neural Network (ANN) enables the agent to reconstruct the hidden part of environment in the ANN. In this section, we implement the updating mechanism, i.e., the gateway decision process of when a sensor should transmit its next observation, using deep Q-learning.

4.3.1 Q-learning

The Q in Q-learning [100] stands for the quality of an action in a given state. The learning agent should take the action with the highest Q-value, unless the algorithm decides to explore. The agent learns the best action possible by updating the Q-value every time it takes an action and observes a reward. When the agent takes enough actions in every state of the environment, it can correctly approximate the real action values, i.e., Q-values, associated with every state. With Q-values determined, the agent can choose the optimal action in every state. A Q-value is calculated as follows:

$$Q^{new}(\mathbf{s}_n, a_n) \leftarrow Q(\mathbf{s}_n, a_n) + \alpha\left(R(\mathbf{s}'_n) + \gamma \max_{a'_n} Q(\mathbf{s}'_n, a'_n) - Q(\mathbf{s}_n, a_n)\right) \tag{4.8}$$

where $Q(\mathbf{s}_n, a_n)$ is the previous Q-value, α is the learning rate, γ is the discount factor, and R is the reward observed in the new state $\mathbf{s}'_n$ after taking action a_n in state $\mathbf{s}_n$. The $\max_{a'_n} Q(\mathbf{s}'_n, a'_n)$ stands for an estimate of the optimal future value an agent can acquire from the next state $\mathbf{s}'_n$.

The role of the agent, i.e., the gateway, is to determine the sensor state and to select the action with the highest Q-value. Every time a sensor transmits an observation, the gateway will respond by instructing the sensor for how long it should enter sleep mode, i.e., set its

next update time. Next, we define states, actions, and reward functions, i.e., a tuple in $\langle \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$ that enables the gateway to determine sensors' optimal update intervals.

4.3.2 States, Actions, and Rewards for DQN

The agent's *state*, $\mathbf{s}_n$, captures three critical aspects of the decision-making process in the gateway: a sensor's current update interval, its available energy, and the estimation error value. Whenever a sensor transmits an observation, the gateway stores the information regarding sensor's update interval, i.e., T, available energy, i.e., E, and value of average MSE since the last transmission, i.e., $\overline{\varepsilon}$. The learning agent then uses those values to represent each sensor state. The agent requires information regarding sensors' update intervals and MSE values to reconstruct the hidden part of the environment, while the energy levels enable the agent to ascertain which sensor in the network most needs to save energy. We express state $\mathbf{s}_n$ as a $3N$ dimensional vector:

$$\mathbf{s}_n = \left(T_1, E_1, \overline{\varepsilon}_1, \cdots, T_n, E_n, \overline{\varepsilon}_n, \cdots, T_N, E_N, \overline{\varepsilon}_N \right) \tag{4.9}$$

with N representing the number of sensors under the agent's control. Using ANN can efficiently solve problems associated with a sizable non-linear state space [65].

In contrast to the state space, we limit *actions*, i.e., the cardinality of set $\mathcal{A}$, to five. Actions enable an agent to learn the best update interval by decreasing or increasing the update interval in time-steps. Furthermore, the agent can select to make a large or small step to adapt more quickly or more gradually. We denote an action to increase the update interval for one time-step U_{incr1} and for ten U_{incr10}. With U_{dec1} and U_{dec10} we denote a decrease of update interval for one and ten time-steps, respectively. If the agent decides to maintain the current update interval unchanged, it selects action U_{cons}. As we show in the Section 4.5, the system can, by using this action space, adapt to any changes in the environment promptly. Additionally, a limited action space prevents rapid fluctuations in the system when the learning algorithm decides to explore.

We designed the *reward function* to aid the agent to quickly adapt to the changing environment. To achieve both gateway objectives, i.e., collecting accurate observations and prolonging sensors' lifetime, we split the reward function into two parts as follows:

$$r_n(\overline{\varepsilon}_n, E_n) = \phi r_{acc}(\overline{\varepsilon}_n) + (1 - \phi) r_{en}(E_n). \tag{4.10}$$

$r_{acc}(\overline{\varepsilon}_n)$ is the reward for accurate collection of observations and $r_{en}(E_n)$ is the reward related to the energy aspect of the problem. The weight $\phi \in [0.25, 0.75]$ controls the balance between the reward for accurate collection of observations and the sensors' energy preservation. We restrict the range of the weight ϕ to avoid the reward from being overly weighted towards one goal or the other.

The accuracy reward depends on whether the set accuracy boundary, i.e., ε^*, was satisfied in the episode. We compare the average MSE in an episode, i.e., $\overline{\varepsilon}_n$, to the target MSE. The accuracy reward is as follows:

$$r_{acc}(\overline{\varepsilon}_n) = \begin{cases} \text{when} \quad \overline{\varepsilon}_n \leq \varepsilon^* : \\ \quad \left(\frac{\overline{\varepsilon}_n}{\varepsilon^*}\right)^2 + \Upsilon \ \Delta\overline{\varepsilon}_n \\ \text{when} \quad \overline{\varepsilon}_n > \varepsilon^* : \\ \quad \left(\frac{\overline{\varepsilon}_n - \varepsilon^*}{\varepsilon^*}\right)^2 - \Upsilon \ \Delta\overline{\varepsilon}_n \end{cases} \tag{4.11}$$

where $\Delta\overline{\varepsilon}_n$ is the difference in the average estimation error since the last transmission. Our objective is for the average MSE observed in an episode to be as close as possible to the target ε^*, without exceeding it. The accuracy reward in Eq. (4.11) accomplishes that.

Our energy reward function depends on the change in the update intervals and how a sensor's available energy compares to the average sensor's energy in the network. We write the energy reward for the n-th sensor as follows:

$$r_{en}(E_n) = \begin{cases} 2 - \frac{2NE_n}{\sum_{i=1}^{N} E_i}, & \text{if } \ T_\Delta > 0 \\ 1 - \frac{1NE_n}{\sum_{i=1}^{N} E_i}, & \text{if } \ T_\Delta = 0 \\ \frac{2NE_n}{\sum_{i=1}^{N} E_i} - 2, & \text{if } \ T_\Delta < 0 \end{cases} , \tag{4.12}$$

where T_Δ represents the change in update interval, E_n is the sensor's available energy.

4.3.3 DQN based Scheduling Mechanism

Figure 4.4 shows a high-level model of our proposed mechanism. Sensors collecting information, along with the part of the gateway responsible for processing information, represent the external environment to our learning agent. An observation sent by an IoT sensor starts the learning cycle. The gateway then passes the necessary information (state and reward) to the learning agent. The learning agent then updates the state space and passes these updates to the ANN. The output of the ANN indicates which action the gateway should take (i.e., the action with the highest Q-value). The updating mechanism uses an ϵ-greedy approach: in our case, $\epsilon = 0.15$. Due to constant changes in the environment, the learning agent has to sometimes explore random actions to find the optimal action. In the last step, the gateway then transmits the action, i.e., the new update interval, to the IoT sensor.

We implemented the ANN, as presented in Fig. 4.5, using Keras [101], a deep learning Python library. To train the ANN, the learning agent requires the values of state spaces of N sensors and corresponding Q-values. The first inserted state space values $(T_1, \overline{\varepsilon}_1, E_1)$ are from the sensor that transmitted last, followed by the state space values of the second last sensor $(T_2, \overline{\varepsilon}_2, E_2)$, and so on. We train the ANN periodically, using batches of recently

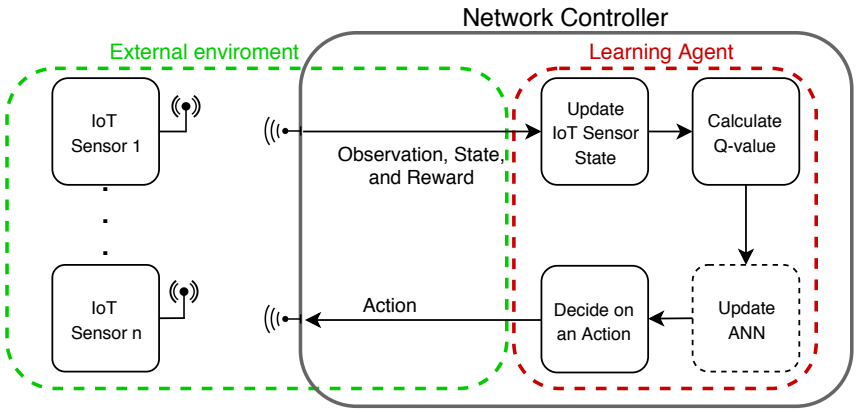


Figure 4.4: Diagram of the proposed updating mechanism.

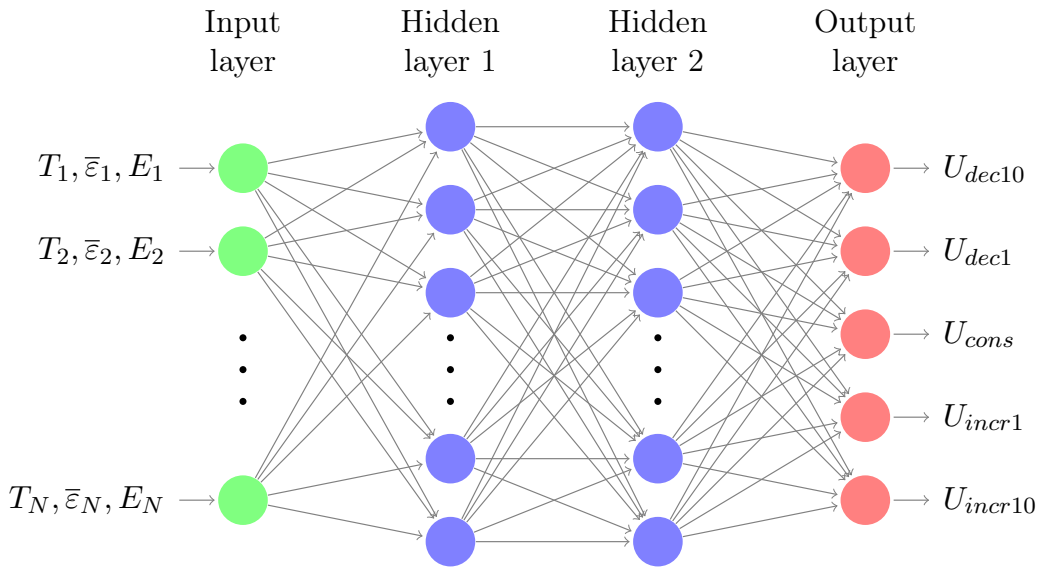


Figure 4.5: We employ ANN with two hidden layers, each with 24 neurons. We use MSE loss function and Adam optimization algorithm to train ANN weights. We apply ReLU activation function for the hidden layers and linear activation function for the output layer.

observed states, rewards, and actions, to shorten the response time. Our learning agent is capable of responding within a few milliseconds, thus satisfying the timing constraints set by sensors' communications technology. For example, a device using LoRaWAN radio typically enters hibernation mode for more than a second. Additionally, response messages impose no additional energy cost to a sensor because the communication standard requires it to always listen to the channel before entering sleep mode.

In the next section, we describe how we implement the scheduling mechanism using DDPG algorithm.

4.4 Deep Deterministic Policy Gradient Based Approach

Employing DDPG algorithm enables us to design a scheduling mechanism with a higher number of possible actions regarding setting the sensor's next update time than a DQN algorithm. Meaning that the granularity of possible update interval change is higher. However, to be able to implement the DDPG algorithm, we had to redefine the states, actions, and rewards.

4.4.1 Deep Deterministic Policy Gradient Algorithm

DDPG is an actor-critic algorithm and, as the name suggests, consists of two entities/neural networks: the actor taking actions, and the critic which evaluates them [67]. The output action has a continuous value, and the actor's policy is represented by a parametric probability distribution function. By sampling the stochastic policy, i.e., exploring the action space, the algorithm can learn the optimal behaviour, i.e., adjust its parametric probability distribution towards a greater cumulative reward. Note that exploration in a DDPG algorithm is carried out by adding a random value, i.e., noise, to the actor's selected value. Additionally, the algorithm we selected is deterministic, meaning that the policy gradient is integrated only over the state space, thus requiring much fewer samples to find the optimal policy in comparison to stochastic algorithms [67].

We follow the DDPG implementation as described in [102]. In their approach, the critic is implemented as a DQN and we denote its ANN as $Q(\mathbf{s}, a|\theta^Q)$ where θ^Q are weights of the critic's ANN, with a denoting the action the agent takes in state $\mathbf{s}$. Next, we define the actor as a parametric function $\mu(\mathbf{s}|\theta^\mu)$ in which θ^μ represents the actor's ANN weights. In addition, during the training process we initialize the target ANN $Q'(\mathbf{s}, a|\theta^{Q'})$ and $\mu'(\mathbf{s}|\theta^{\mu'})$ with weights $\theta^{Q'}$ and $\theta^{\mu'}$ for critic and actor respectively. The agent selects actions according to its current policy with added noise $a = \mu(\mathbf{s}|\theta^\mu) + \mathcal{N}$, where $\mathcal{N}$ represent added random noise. Then the agent transitions into a new state $\mathbf{s}'$ and receives reward r. Then the transition $(\mathbf{s}, a, r, \mathbf{s}')$, also referred to as experience, is stored in memory. When the algorithm is training the ANN it first samples a mini-batch of M experiences from the batch and calculates the target values:

$$h_m = r_m + \gamma Q'(\mathbf{s}'_m, \mu'(\mathbf{s}_m|\theta^{\mu'})|\theta^{Q'}), \tag{4.13}$$

with m denoting the selected experience from the batch. After determining h_m we can update the critic by minimizing the loss L as:

$$L = \frac{1}{M}\sum_{m=1}^{M}(h_m - Q(s_m, a_m|\theta^Q))^2. \tag{4.14}$$

With the loss function calculated, the algorithm then updates the actor's policy using the

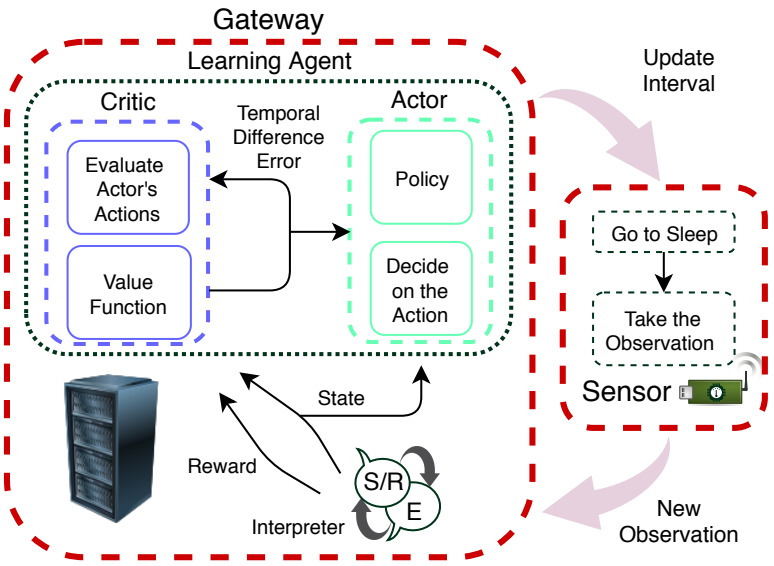


Figure 4.6: A high-level overview of the proposed scheduling mechanism implemented with a Deterministic Policy Gradient Algorithm.

sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_a Q(\mathbf{s}, a|\theta^Q)|_{\mathbf{s}=\mathbf{s}_m, a=\mu(\mathbf{s}_m)} \nabla_{\theta^\mu} \mu(\mathbf{s}|\theta^\mu)|_{\mathbf{s}_m}. \tag{4.15}$$

In the last step the DDPG algorithm updates the target ANNs:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau_C)\theta^{Q'}, \tag{4.16}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau_A)\theta^{\mu'}, \tag{4.17}$$

where τ_A and τ_C represent the target networks update factor. Note that the role of the target ANNs is to calculate h_m. Using separate target networks along with the replay buffer provide stability during the training process as was established in [65].

We illustrate how we adopt DDPG algorithm in a high-level overview in Figure 4.6. In our approach, the gateway performs every computationally demanding task. A low-power sensor only receives a command message that instructs the sensor for how long it should enter sleep mode. By setting the duration of each sensor's sleep mode, the gateway effectively schedules the sensors' transmission of updates. In our mechanism the actor's actions are limited to increasing and decreasing the sleep time, or equivalently the sensor's update interval T_n, and the critic's role is to evaluate whether the selected change is beneficial for the system. The critic derives a value representing the quality of the selected action using the reward for the actor. The actor then uses the provided quality of action value to adapt its policy accordingly.

Next, we define states, actions, and rewards, i.e., a tuple in $\langle \mathcal{S}, \mathcal{A}, R \rangle$ that enables the gateway to determine each sensor's optimal update intervals.

4.4.2 States, Actions, and Rewards for DDPG

For each sensor, the state $\mathbf{s}_n \in \mathcal{S}$ must capture three critical aspects of the decision-making process in the gateway: a sensor's current update interval, its available energy, and the value of estimation error. Whenever a sensor n transmits an observation, the gateway stores the information regarding the sensor's update interval, T_n, available energy, E_n, and the value of the average MSE since the last transmission, $\overline{\varepsilon}_n$. The inclusion of these three pieces of information from every sensor would result in an overly complex state space: we therefore turn to the geometric mean to compress the number of state inputs. The state $\mathbf{s}_n$ can be expressed as a six-dimensional vector:

$$\mathbf{s}_n = (T_n, E_n, \frac{\overline{\varepsilon}_n}{\varepsilon^*}, \left(\prod_{\substack{i=1 \\ i\neq n}}^{N} T_i\right)^{\frac{1}{N-1}}, \left(\prod_{\substack{i=1 \\ i\neq n}}^{N} E_i\right)^{\frac{1}{N-1}}, \left(\prod_{\substack{i=1 \\ i\neq n}}^{N} \frac{\overline{\varepsilon}_i}{\varepsilon^*}\right)^{\frac{1}{N-1}}) \tag{4.18}$$

with N representing the total number of sensors under the agent's control. The learning agent always interprets states from the perspective of the sensor that has transmitted an observation, and for which it is making the decision. The first three inputs correspond to the transmitting sensor's update interval, available energy, and the ratio between average MSE and target estimation error. Relying on the ratio enables the learning agent to perform well even if the target estimation error changes, as we demonstrate in the next section. Using the geometric mean enables us to reduce the number of state inputs, while simultaneously making sure the learning agent captures the most significant information about the environment. For example, the geometric mean provides information to the agent regarding whether the energy level in the majority of the sensors is low or high.

The learning agent's *action* is limited to either increasing or decreasing the sensors' current update interval. As we mentioned in the previous section, the DDPG algorithm returns a continuous value. In our implementation, the returned action value ($a_n \in \mathcal{A}$) is between -1 and 1, i.e., $\mathcal{A} = [-1, 1]$. To determine the sensors' new update interval, we multiply the received action value by a constant, representing the selected maximum update interval change U_{max}. We calculate the new update interval as follows:

$$T_n = \min\Big(\max\big(T'_n + \lfloor U_{max} a_n \rceil, 1\big), T_{max}\Big), \tag{4.19}$$

where T'_n is the sensors' previous update interval. Note that the value of U_{max} can be relatively large, e.g., hundreds of seconds, and our approach will still perform well. Additionally, the value of T_n is limited due to practical reasons, i.e., $T_n \in \{1, \ldots, T_{max}\}$.

We form the *reward* with the learning agent's goals in mind. The learning agent has to ensure that information collection is frequent enough to maintain the freshness of the information and simultaneously try to prolong the sensors' lifetime. We express the reward

for the DDPG approach using two parts, each representing one of the two main objectives of the scheduling mechanism, as we did for the DQN approach:

$$r_n(\overline{\varepsilon}_n, E_n) = \phi r_{acc}(\overline{\varepsilon}_n) + (1-\phi) r_{en}(E_n). \tag{4.20}$$

The accuracy reward depends on whether the set average accuracy, ε^*, was satisfied since the n-th sensor's last transmission and the change in the estimation error since the last transmission. We use the same accuracy reward as we did for the DQN approach:

$$r_{acc}(\overline{\varepsilon}_n) = \begin{cases} \text{when} \quad \overline{\varepsilon}_n \leq \varepsilon^* : \\ \quad \left(\frac{\overline{\varepsilon}_n}{\varepsilon^*}\right)^2 + \Upsilon \, \Delta\overline{\varepsilon}_n \\ \text{when} \quad \overline{\varepsilon}_n > \varepsilon^* : \\ \quad \left(\frac{\overline{\varepsilon}_n - \varepsilon^*}{\varepsilon^*}\right)^2 - \Upsilon \, \Delta\overline{\varepsilon}_n \end{cases} \tag{4.21}$$

where $\Delta\overline{\varepsilon}_n$ represents the change in the average estimation error since the previous transmission. The closer the estimation error is to the target, the greater will be the reward from the first term of the expression. The second term of the accuracy reward steers the learning agent towards keeping the average MSE as close as possible to the set ε^*, without exceeding it. The factor Υ is used to balance the contributions of the two parts of the accuracy reward.

Our energy reward exploits the relationship between the update interval and a battery-powered sensor's lifetime. The longer the time between the consecutive updates, the longer the sensors' lifetime will be. Therefore, the selected energy reward is based on how the update interval is increased or decreased, as follows:

$$r_{en}(E_n) = \begin{cases} 1 - \frac{NE_n}{\sum_{i=1}^{N} E_n}, & \text{if } \; T_n > T'_n \\ 0, & \text{if } \; T_n = T'_n \\ \frac{NE_n}{\sum_{i=1}^{N} E_n} - 1, & \text{if } \; T_n < T'_n \end{cases}, \tag{4.22}$$

where E_n is the sensor's available energy. If a sensor has above average available energy, the energy reward should encourage the learning agent to make sure that such a sensor updates more often, and vice-versa if a sensor has below average energy. We altered the energy reward for the DDPG approach in comparison to the energy reward we apply when we employ a DQN algorithm (Eq. (4.12)). The main reason behind such a decision is the fact that DDPG approximates the action space, which means that the output of its neural network is a continuous value that proportionally corresponds to the action the agent should take. In contrast, for DQN, each output neuron corresponds to a particular action value the agent should take. Consequently, the reward that yields the desired behaviour when we employ DQN does not result in the desired behaviour for when we use DDPG. More specifically, if the DQN approach would adopt the reward we defined in Eq. (4.22) it would learn to alternate around a stable update interval as a constant increase or decrease of the

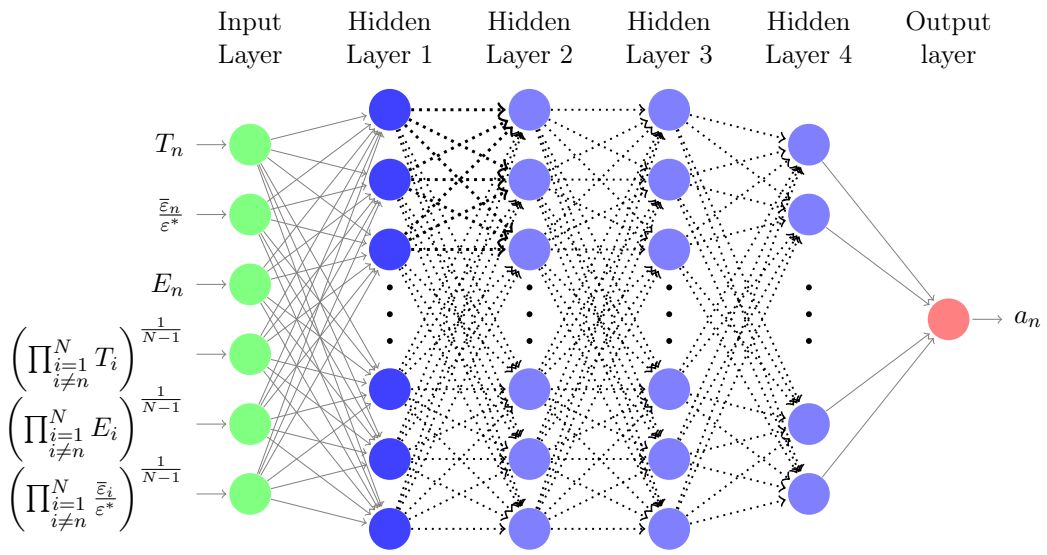


Figure 4.7: Actor's ANN structure

update interval would yield a greater cumulative reward for the DQN agent. To that end, we have designed different energy reward for the two approaches.

In the next subsection, we present our implementation of the DDPG algorithm.

4.4.3 Implementation

To implement the learning agent using DDPG as described in [102] we used Pytorch [103], a standard Python-based library for implementing DRL algorithms. We employ similar ANN structures for the actor and the critic. The actor's ANN consists of an input layer (state inputs), output layer (action value), and four feed-forward hidden layers, as shown in Figure 4.7. We use 75 neurons in the first three hidden layers and 25 neurons in the fourth layer. Between each hidden layer, we implemented a 50% dropout layer. We use batch normalization after activation in the first hidden layer. The dropout layers prevent over-fitting, and batch normalization improves learning speed. We employ the same structure for the critic's ANN, with a slight difference in the activation function used. We use ReLU activation function for every layer in both ANNs. The only exception is the output layer of the actor's ANN, where we use a Hyperbolic function. Such a difference is required as the actor's output value is limited to values between -1 and 1 while the critic's is not. To train the ANNs we periodically perform batch learning. In each batch training we use 128 experiences. Each experience consists of a state, corresponding action and reward, and the state to which our sensor transits after taking the selected action. Note that 128 experiences are randomly selected from a memory pool of up to 100,000 experiences.

In the gateway we deploy only one set of actor's and critic's ANN for the DDPG algorithm to control the update intervals of all N sensors in the deployment. Upon the n-th sensor's transmission the gateway constructs the state input from its perspective. The first

three inputs to the ANN represent the n-th sensor information (current update interval, average estimation error, and energy), and the last three inputs represent the information from every other sensor in the deployment as we defined in Eq. (4.18). Following such an approach, we can extrapolate the knowledge obtained from one sensor behavior to all others.

We set the start of an episode to when a sensor transmits an observation. When the gateway receives an observation, it uses the interpreter to determine the sensor's state and calculates the reward for the action taken, i.e., the change in the update interval. The sensor state information is then passed to the learning agent to determine the new action; the learning agent calculates the sensor's new update interval and informs the sensor. The sensor will enter sleep mode for the amount of time determined by the learning agent. As soon as the sensor wakes up, the episode ends, and the new episode starts as the sensor transmits a new update. Note that multiple episodes happen in parallel, one for each sensor under the mechanism's control. While one sensor is in sleep mode, another wakes up and transmits an observation. Exploration in DDPG algorithms is performed by adding noise, i.e., a random value, to the actor's output value. We use the Ornstein-Uhlenbeck process [102] to generate a random value to be added to the selected action value.

In a real deployment, our learning agent will be capable of responding within 2 to 3 ms, thus satisfying the timing constraints set by sensors' communications technology. For example, a sensor using LoRaWAN radio typically enters hibernation mode for more than a second. Additionally, response messages from the agent impose no additional energy cost to the sensor, as most LPWAN standards require the sensor to listen to the communication channel after its transmission.

In the next section, we evaluate the proposed mechanism performance using data obtained from a real sensor network deployment.

4.5 Evaluation

In this section, we evaluate our proposed scheduling mechanism using observations provided by the Intel Berkeley Research laboratory [90], as well as data collected from multiple sensors deployed in the city of Santander, Spain, as part of the SmartSantander testbed [93]. In our experiments, a simulated sensor transmits an observation with the exact value as was obtained by its real world counterpart. In other words, the data enables us to realistically represent how the observed physical phenomenon varies over time. We evaluate our mechanism using five different datasets: two from Intel (temperature and humidity), and three from SmartSantander (temperature, humidity, and ambient noise)[3]. We split the evaluation into two parts. In the first part, we demonstrate that the proposed scheduling mechanism is capable of learning the optimal behaviour and that using our approach can

[3]We provide a more in-depth description of the datasets in Appendix B.

significantly extend the sensors' lifespan. In the second part of our evaluation, we highlight the energy-aware aspects of our proposed scheduling mechanism.

We use part of the data for training the agent, and the remainder for validating the learned behaviour. Such a split ensures that during validation, the learned behaviour is not a consequence of over-fitting to the environment encountered during the exploration. In our evaluation, we use real sensor locations in the Intel laboratory deployment and transmitted observations collected over nine days of measurements from 50 sensors: we use six days of data for exploration, and three days for exploitation. The Intel data comprises observations collected very frequently (every 31 seconds), providing us with a ground truth of observed values for the evaluation process. The SmartSantander data represents a realistic deployment of sensors in a smart city environment. In our analysis, we use data obtained in the first nineteen days of April 2018. The evaluation data comprises the last six days. We rely on data from twenty temperature, ten humidity, and eight ambient noise sensors. In contrast to the Intel data, every sensor transmitted observations at different intervals of time, and sometimes there were a few hours during which a sensor did not send an observation. We use the Amelia II software program [104], a tool for missing data, to generate missing values. To generate these values, we used observations from the sensor that collected the highest number of observations; as a consequence, we had to remove that sensor from each SmartSantander dataset, to avoid adding bias in the evaluation process.

Table 4.1: Static Simulation Parameters

Parameter	Value	Parameter	Value
U_{max}	$250s$	P_c	$15uW$
E_0	$6696J$	$\mathbb{E}[E_{tr}]$	$78.7mJ$
T_{start}	$900s$	Υ	10
ε^*	0.01	ϕ	0.5

The data above serves as input for our simulated network, where data transmitted by an individual sensor is collected and processed by a gateway. Using real data from sensor deployments, we are able to replicate a real environment with which our learning agents interacts. We list system parameters that are kept constant throughout our evaluation process in Table 4.1. We select static simulation energy parameters by assuming that each of the sensors is powered by a single non-rechargeable Lithium coin battery with a capacity of 620 mAh, which provides us with the value for E_0. The selected energy consumption parameters, i.e., P_c and $\mathbb{E}[E_{tr}]$, mimic the power consumption of an IoT sensor using a LoRaWAN radio. We obtain the power parameters following the analysis presented in [105]. The selected U_{max} yields the best average performance at the end of the training phase for all five datasets. T_{start} represents a suitable starting update interval value, while T_{max} represents the maximal value that should be allowed between two consecutive updates from one sensor. At the start of every simulation, we set the same initial update interval for every sensor. ϕ and ε^* are set to the value stated in the Table unless they are parameters that

we change in the presented experiment. We perform simulations in ten-second time-steps. In Table 4.2 we list the DRL solution hyperparameters which we determined through a grid search to be most suitable. Note that we multiply the calculated rewards by a factor of 10 to improve the training process, as higher reward values tend to reduce the time required for the DDPG algorithm to arrive at the optimal policy [106].

Table 4.2: DQN and DDPG Hyperparameters

	Hyperparameter	Value
DQN	Learning rate	10^{-3}
	Discount factor γ	0.5
	Explore rate ϵ	0.15
	Batch size M	32
	Memory size R	2×10^4
	Maximal Number of Senors N_{max}	16
	Optimizer	Adam
	Loss Function	MSE
DDPG	Actor's ANN learning rate τ_A	10^{-4}
	Critic's ANN learning rate τ_C	10^{-4}
	Target ANN soft update	10^{-3}
	Discount factor γ	0.99
	Batch size M	128
	Memory size R	10^5
	Optimizer	Adam

4.5.1 Performance Evaluation

In this subsection, we evaluate our scheduling mechanism's ability to maintain accurate observations and compare it to other scheduling approaches.

To illustrate how the scheduling mechanism is capable of finding the optimal solution, i.e., capable of determining the maximal update interval possible that still maintains the observation error below the threshold ε^*, we test the mechanism's performance in a system with two sensors. We set one sensor in learning mode while the other keeps a constant update interval. Furthermore, we vary the covariance model scaling parameters only in selected episodes. Such a simplified system enables us to obtain the optimal update interval for comparison purposes. We changed the scaling parameters in episode 50, and in episode 100 we changed ε^*. The mechanism implemented with either DQN and DDPG algorithm is always capable of adapting and finding the optimal update interval, as illustrated in Fig. 4.8(a) and Fig. 4.8(b). To obtain the optimal solution, we used a simple search algorithm to resolve the problem we posed in Eq. (4.7) for a system of two sensor nodes with fixed scaling parameters. Such a result indicates that our proposed mechanism is capable of finding the optimal solution. In Fig. 4.8(c) and Fig. 4.8(d), we show the average observation error $\overline{\varepsilon}$ over a number of episodes: the mechanism always converges toward the selected ε^*. Note

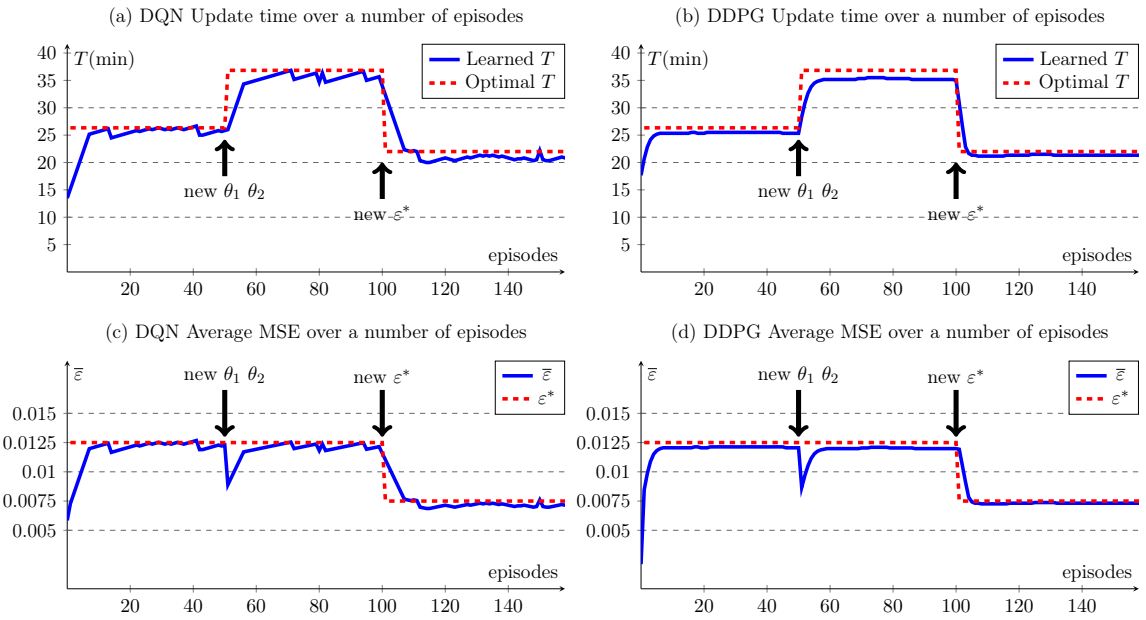


Figure 4.8: Updating mechanism searching for the optimal update interval, arrows indicate change in the covariance model scaling parameters or ε^*. Note that in episode 100 we change ε^* from 0.0125 to 0.0075.

when we abruptly lower ε^* in episode 100, the target is violated for a few episodes until the system adapts to the new condition. The main difference between the two approaches is that when the scheduling mechanism is implemented with the DDPG algorithm it is capable to adapt much faster to the change.

Next, we test the updating mechanism performance as the number of sensors, N, under its management increases. The Intel datasets offer us a maximum of 50 sensors, and for cases in which $N < 50$, we randomly selected a subset of these sensors and then repeated the experiment several times. As expected, increasing the number of sensors leads to more correlated information available, and therefore, the gain of using correlated information increases with the number of sensors. The benefits of using correlated information is higher when observing temperature, due to higher correlation exhibited in the observations collected. We calculate the expected lifetime using Eq. (4.6). In Fig. 4.9(a) and Fig. 4.9(b) we use both Intel datasets, humidity and temperature, and calculate the expected lifetime using the average update interval that sensors achieved in the test portion of the experiment. Results indicate that regardless of which algorithm the scheduling mechanism is using, the performance is very similar.

Additionally, to demonstrate the performance improvement brought by our solution, we calculate the lifetime gain η, plotted in Fig. 4.9(c) and Fig. 4.9(d). We define this gain as the ratio between the lifetime achieved using our mechanism and that achieved in the original datasets. The gains of using our approach are significant, as the achieved lifetime exceeds five years, while the original update intervals observed in the datasets would lead sensors to last only a month (a time duration that coincides with the original time the sensors in the

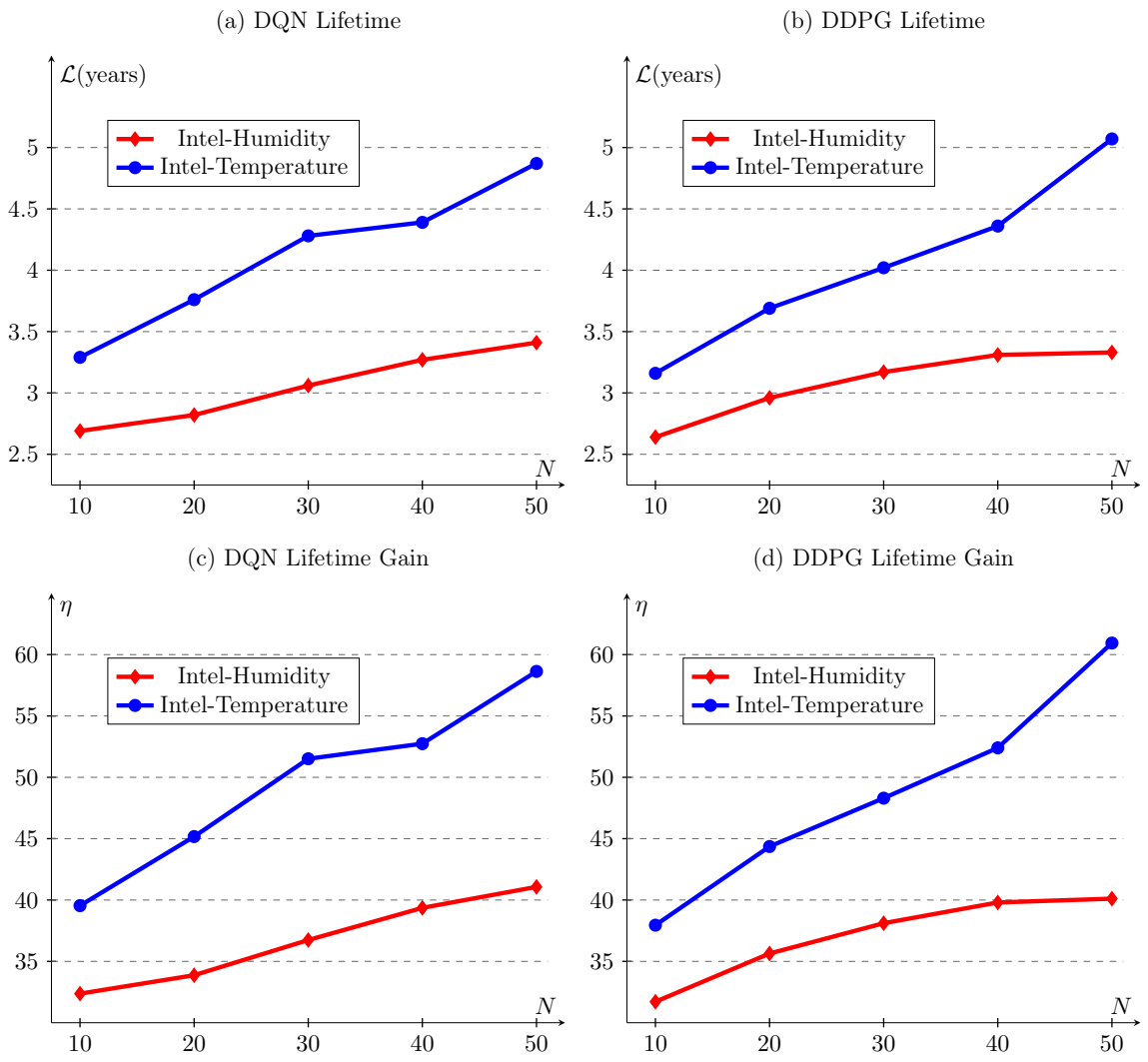


Figure 4.9: Sensors' lifetime and lifetime gain achieved by our updating mechanism as the number of sensors under its control increases.

Intel lab were deployed).

To further investigate the performance of the two approaches we look into how they behave when we alter the target estimation error. We show the increase of sensors update interval as set target increases in Fig. 4.10 (a). In Fig. 4.10 (b) we show how close to the set target, ε^* , the two approaches come. As we can see, both DQN and DDPG perform very well both can achieve the different set targets without overshooting. The achieved average update interval is also very similar, the DQN approach achieves a slightly higher update interval. Note, that to achieve a fair comparison we re-use a neural network we trained to obtain results we showed in Fig. 4.9, and then trained on the train part of the dataset for the same number of days for each different set target.

In Fig. 4.11 we compare the performance of our DQN and DDPG-based scheduling mechanism, in terms of achieved sensor lifetime, to two different methods of obtaining update intervals:

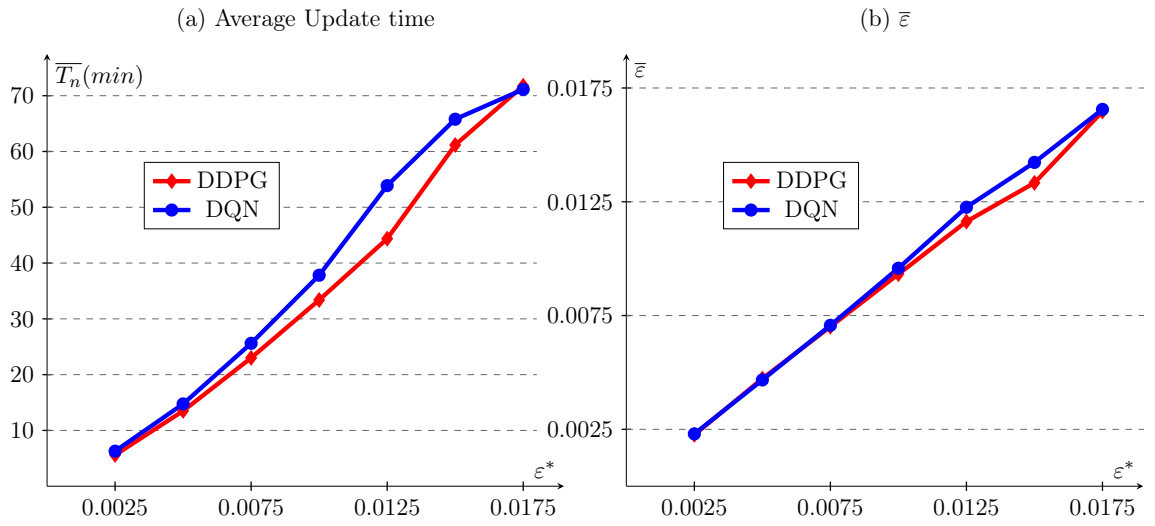


Figure 4.10: Sensors' update interval and average estimation error with the change of target, i.e., ε^*.

1. **Original**: as a baseline, we take the update intervals adopted in the Intel Lab and Smart Santander datasets. The baseline case reveals the original expected lifetime of deployments, if the sensors were battery-powered. Note that for the SmartSantander datasets, we use the update interval of the sensor that collected the most observations.
2. **Ideal**: in the ideal case, the network controller acts essentially as an oracle that knows the ground truth and can obtain observations from any sensor on demand. With the ideal scheduler, sensors are asked to transmit only when the average error of estimated values (of temperature, humidity, ambient noise) exceeds the set threshold ε^*.

In Fig. 4.11, we show the expected lifetime we achieved using all four approaches for all five datasets. We calculated the lifetime using Eq. (4.6) and using values of average update intervals that we obtained with each approach. The performance results observed with DDPG and DQN are similar. The results also show that our mechanism is capable of finding update intervals that are very close to the ideal one. The only exception is for the case of Intel lab temperature data. In that case, a higher U_{max} would enable the agent to get results closer to the ideal case. However, by keeping the U_{max} constant throughout our validation we can demonstrate the system's robustness to multiple different scenarios.

To improve the performance in comparison to the ideal case, the apparent solution is to provide the agent with more information, i.e., expand the ANN input state space. For example, adding information regarding the average distance to neighbouring sensors would help. However, adding the average distance as part of the state information would decrease the generality of our solution. Intuitively, providing the agent with direct information regarding all sensors in the system (current update interval, average estimation error, and energy level) should lead to better results. Unfortunately, a significantly larger state space would lead to longer computational time and even possibly a degradation in performance,

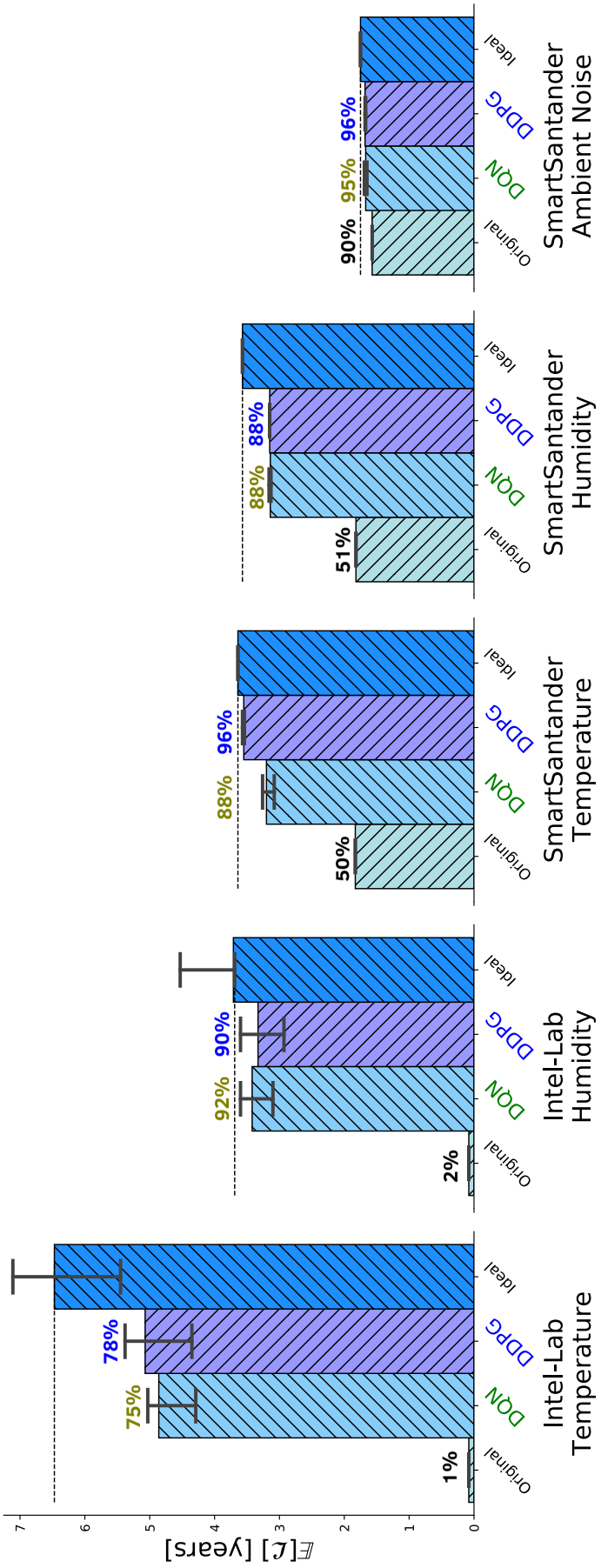


Figure 4.11: Achieved expected lifetime in years with every approach for all five datasets. The number on top of each bar plot reveals how close each approach is to the Ideal case.

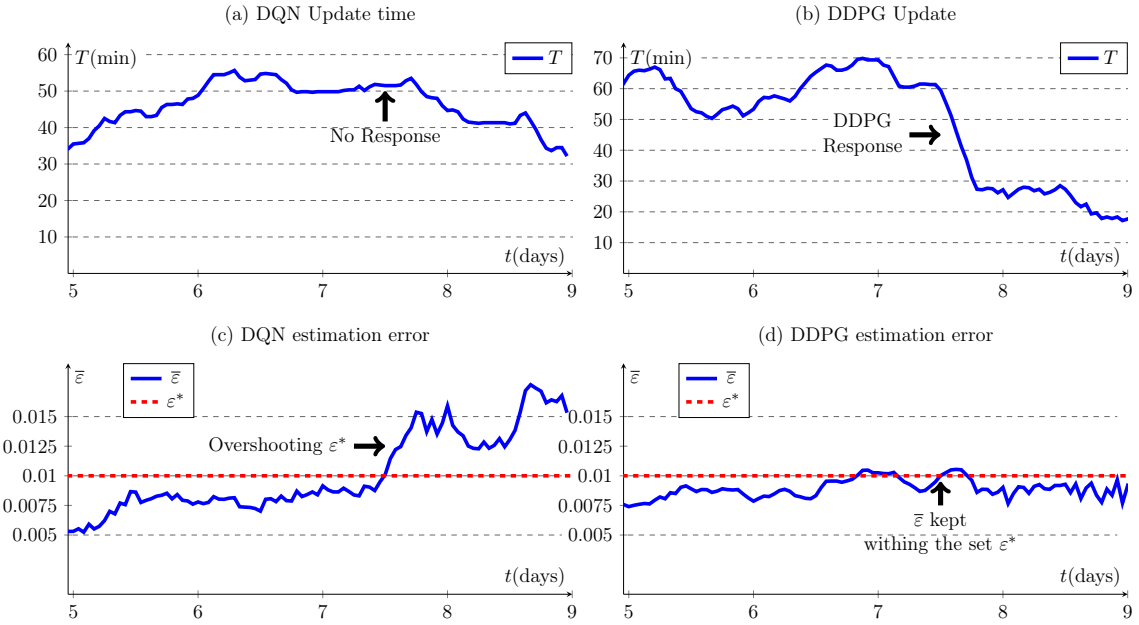


Figure 4.12: The change of update interval and $\overline{\varepsilon}$ for Intel lab sensor (number 9) in the last four days of humidity dataset (the test part of the dataset). On the left, we show the change when a sensor is controlled by a scheduling mechanism implemented with DQN algorithm and on the right when the scheduling mechanism is implemented with a DDPG algorithm.

as an agent could have a hard time differentiating between the more and the less relevant information, e.g., which sensor's updates are most relevant for the current decision.

Table 4.3: Obtained $\overline{\varepsilon}$ per dataset for each approach.

Dataset	ε^*	DQN	DDPG	Ideal
Intel-Temperature	0.01	0.0088	0.0092	0.01
Intel-Humidity	0.01	0.0093	0.0091	0.01
SmartSantander-Temperature	0.01	0.0091	0.0091	0.01
SmartSantander-Humidity	0.01	0.0094	0.0091	0.01
SmartSantander-Ambient Noise	0.01	0.0098	0.0092	0.01

Looking only at the achieved average sensor lifetime for DQN and DDPG obscures the real differences between the two solutions we proposed, as both achieve a similar result. However, the DQN approach is less stable in achieving the set average estimation error target ε^*. In Table 4.3 we show that the DDPG approach achieves $91 - 92\%$ of the set target, for every dataset. The DQN approach leads to larger variation in achieving the set estimation error target.

In Fig. 4.12 we show the difference in the response to changes in the environment depending on which algorithm our scheduling mechanism adopts. In the middle of the 7-th day, a sudden change in the scaling parameters happens, resulting in a drastic rise of ε^*. Note that we noticed such a large anomaly only in the case of Intel Humidity dataset. Regardless, it serves as nice example of how the mechanisms respond differently. It appears,

that in the case of DQN the scheduler has learned to wait, we assume that in the past once such sudden changes occurred it was a better strategy to wait than to adapt. The DDPG on the other hand reacts preemptively to avoid crossing the set target. As soon as the average error approaches the set target it reacts.

Such a behaviour begs the question, for how long is mechanism violating the set target. Note that such violations occur due to sudden change of scaling parameters and are independent of mechanism actions. In Table 4.4, we list for how long (in the percentage of test dataset time) the mechanism violates the set target. From the obtained results, we can assume that when the mechanism adopts DDPG algorithm will more easily meet the set requirement. The time during which the mechanism violates the set target is marginal except for Intel dataset humidity dataset. However, even in that cases DDPG still outperforms the DQN.

Table 4.4: The amount of time the mechanism is violating the set target per dataset.

Algorithm	Intel-Tem	Intel-Hum	SmartS-Tem	SmartS-Hum	SmartS-Amb
DQN	31.5%	25.1%	13.6%	30.8%	32.2%
DDPG	5.7%	14.7%	7.5%	1.7%	1.2%

The reasons why DQN under-performs in terms of achieving the set target in comparison to DDPG are three. The first is the limited available set of actions to the agent: in the DDPG approach the agent can select the change in update interval with much finer granularity. It would be possible to improve the performance of DQN by including additional actions (each action represents the number of time-steps by which the transmitting sensor may change its update interval). However, each added action will significantly prolong the learning time. The second reason is that the maximum change in update interval can be greater for the DDPG case, i.e., 250 s, while in DQN implementation the maximal change is limited to $100s$. Therefore, the DDPG approach can more easily adapt to rapid changes in the system, e.g., a sudden change in scaling parameters in the covariance model. The third reason is in the way the two techniques perform exploring. In a dynamic environment, the agent has to keep exploring. Otherwise, its performance will degrade over time [67]. In the DQN approach, the agent takes a random action. As such, it can easily happen that, for example, the agent instead of increasing it will decrease the sensor's update interval. We set the exploring rate to acceptable 15% the random action still impacts the results slightly but ensures approaches ability to adapt to enviroment changes. When using DDPG, exploring is performed by adding noise to the selected action. The actor's ANN is capable of returning a general direction, e.g., an agent increases the sensor's update interval. And then we add a random value that slightly modifies the change in update interval as the actor's ANN calculated. Consequently, the DDPG can better follow possible environmental changes.

The importance of faster adaptability to an ever-changing environment is paramount in real deployments. Given this criterion, the DDPG solution substantially outperforms

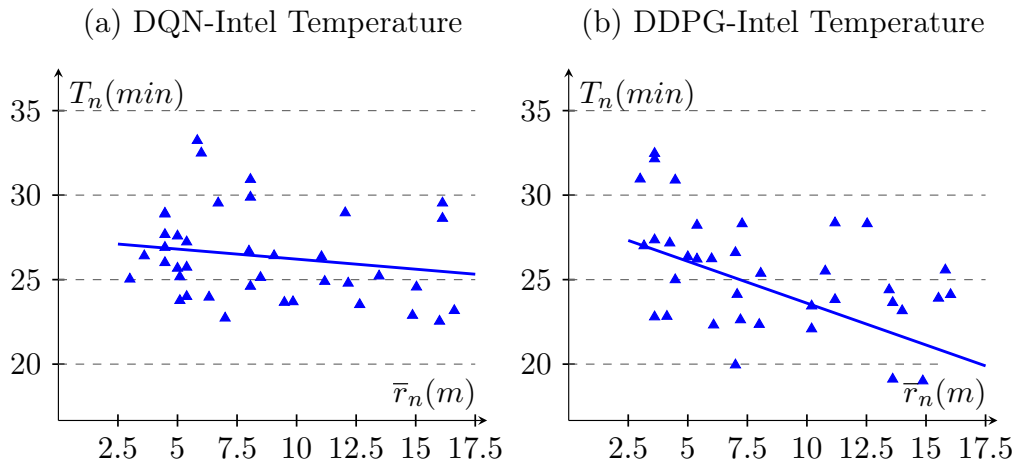


Figure 4.13: Sensor's update interval change depending on the average distance to other sensors in the network.

the DQN approach, even if results in Fig. 4.11 indicate very similar performance in terms of achieved lifetime. Additionally, we noticed that the DQN approach requires a longer exploration time, i.e., a larger amount of experiences needed for the agent to learn how to set sensors' update interval. On average, Intel Temperature sensors transmit every 30 to 40 minutes, meaning that each sensor on average generates anywhere between 50 and 30 experiences. In a deployment of 50 sensors such an update interval yields around 2000 experiences per day. In this regard, DDPG is also much better. As for the Intel Lab dataset we iterated two times (24 000 experiences) to train the system using DDPG, in comparison to the three iterations over the training dataset which we needed for the DQN approach (36 000 experiences).

In Fig. 4.11, we have also shown that sensors have a different lifetime expectation (grey line shows the maximal and minimal expected lifetime). The variation is especially noticeable in the case of Intel-Lab data - indicating, that some sensors benefit more from using correlated information than others. In the case of Intel laboratory data, sensors are deployed on the same floor, and those in the middle of the room have other sensors closer in comparison to the ones deployed at corners. However, distances are not part of the information of the learning agent. To see if our mechanism can capture the impact of distances between sensors we plot the average distance to nearby sensors and achieved average update interval on test data of Intel-lab Temperature in Fig. 4.13. We can see that indeed, the average distance to other sensors impacts the achieved update interval. In the case of DDPG even more so. Note that we obtain the lines in Fig. 4.13 using linear regression.

As we hinted with the last results, if sensors in the deployment start with the same battery level, over time, their energy levels will be different as one sensor will consume its energy faster than other. To that end, we designed an energy-aware solution to prevent one sensor from running out of energy much sooner than others. We evaluate this feature of our proposed scheduling mechanism in the next section.

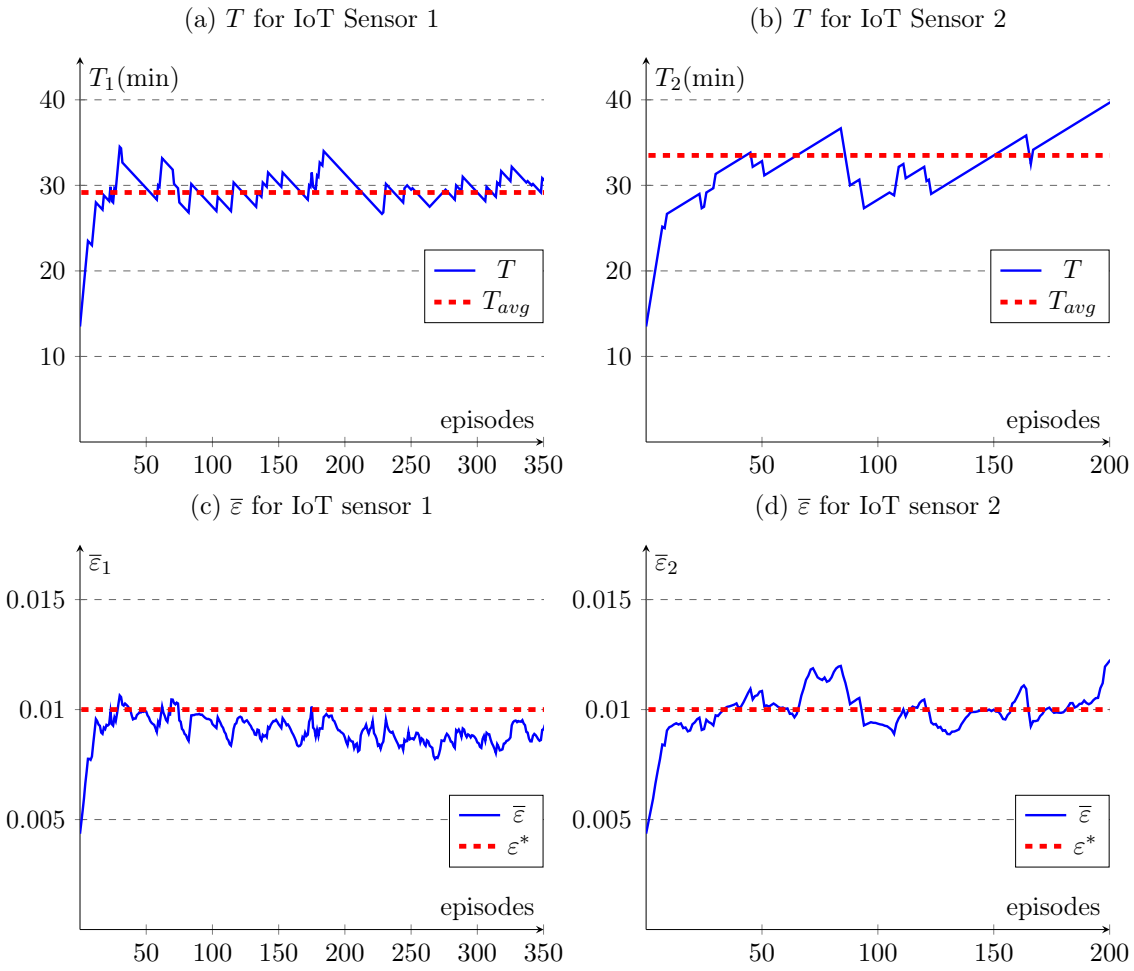


Figure 4.14: Scheduler implemented with DQN algorithm. Two IoT sensors with different battery levels learning over a number of episodes. Sensor 1 battery level is at 75% while the sensor 2 has 25% of the battery left.

4.5.2 Energy Awareness

In this subsection, we demonstrate the energy-aware capabilities of the proposed mechanism. We evaluate the energy-aware performance using Intel-laboratory temperature data and in all experiments the scheduling mechanism controls 50 sensors.

First, we show the change of update interval and $\overline{\varepsilon}_n$ over a number of episodes for two sensors with different energy levels than the others. We iterate over the Intel temperature test dataset twice. Note that the mechanism controls 50 sensors in total and energy levels on all others is 50%. Furthremore, the two sensors were randomly selected. In Fig 4.14 we plot the results we obtained with the DQN algorithm-based scheduling mechanism and in Fig. 4.15 with the DDPG algorithm. In Fig 4.14(a) and Fig. 4.15(a) we plot the update interval over a number of episodes for a sensor with above-average available energy (75%), while in Fig 4.14(b) and Fig. 4.15(b) we plot update intervals of sensors with below-average energy (25%). As we show, our updating mechanism sets the update interval of a sensor with less energy significantly higher in comparison to the update interval of a sensor with more

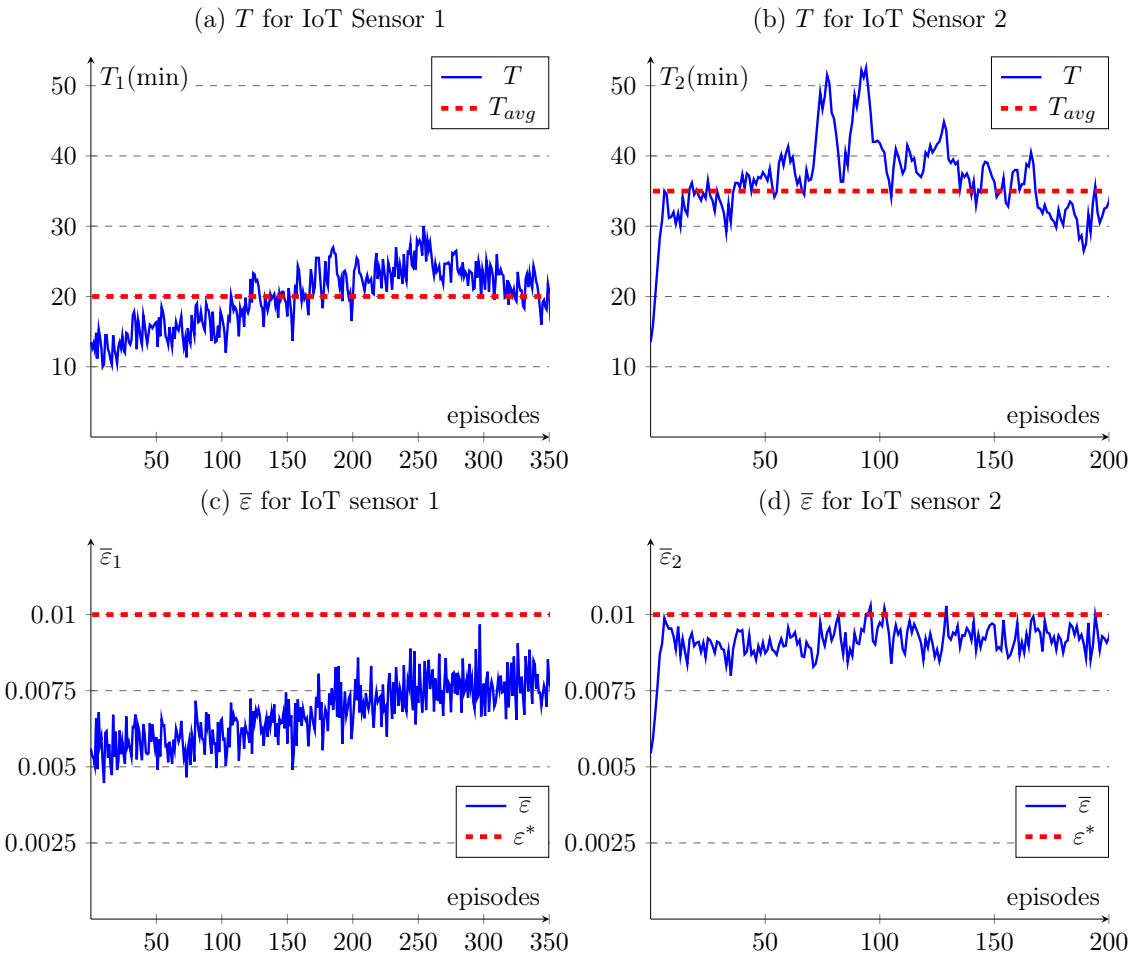


Figure 4.15: Scheduler implemented with DDPG algorithm. Two IoT sensors with different battery levels learning over a number of episodes. Sensor 1 battery level is at 75% while the sensor 2 has 25% of the battery left.

energy available. The trend is more noticeable in the case of a DDPG-based mechanism as in the DQN-based one. By setting different update intervals, the mechanism can balance the energy levels among sensors. Simultaneously, for the case of DDPG, as we show in Fig. 4.15(c) and Fig. 4.15(d) set target is not exceeded. The mechanism keeps the average estimation error for a sensor with more energy far below the threshold, while the sensor with more energy is always close. In other words, the mechanism forces the sensor with more energy to update more often than it should to help preserve the energy of the sensor with less energy. While for the DQN approach as we show in Fig. 4.14(c) and Fig. 4.14(d) the mechanism will try to keep both sensors close to the set target. However, for a sensor with less energy, it will exceed the set target more often.

In the next experiment, we show how the mechanism, implemented with a DDPG algorithm, sets the ratios between sensors with different energy levels. We set the energy level of 90 per cent of sensors (45 out of 50) in the dataset to a fixed value. Note that we denote the values associated with the 90 per cent majority with a subscript 90%, while with the

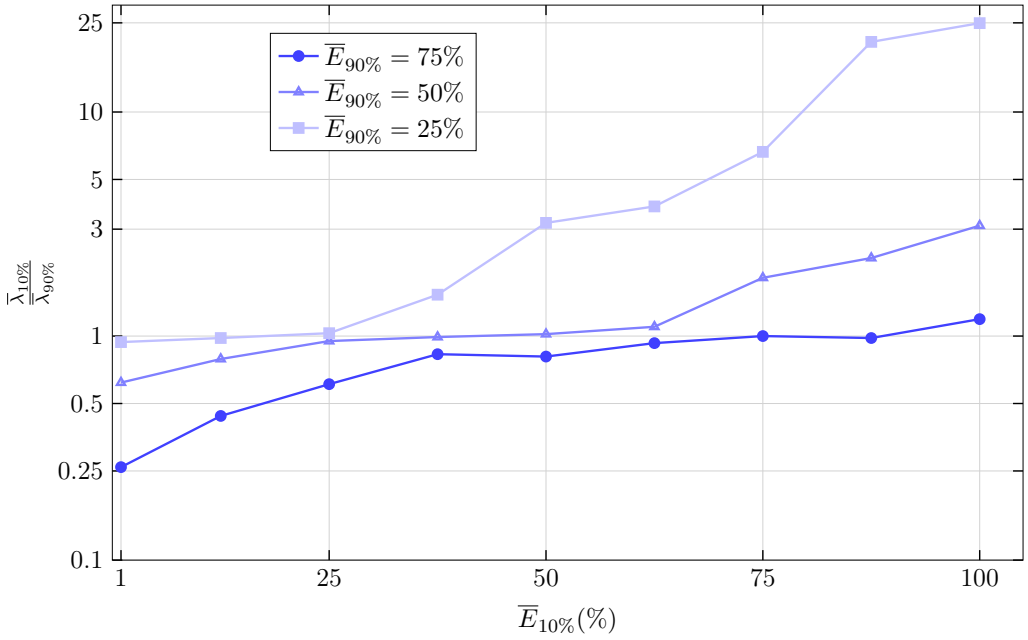


Figure 4.16: The change in update interval ratio as the percentage of the energy levels in sensors change.

subscript 10% we denote the values related to the minority of sensors. In the experiment, we change the energy level of the remaining ten per cent sensors (5) in steps from 1% to 100% of available energy. In each step we iterate over the test data and report the ratio between achieved average update interval of the two groups, i.e., $\frac{\overline{\lambda}_{10\%}}{\overline{\lambda}_{90\%}}$. We show the obtained results in Fig. 4.16. As expected, when the sensors have the same energy level, they will transmit with roughly the same update interval. We can observe an intriguing behaviour when sensors are close to depleting their energy. When a few sensors have much more energy than others, they will transmit new observations much more often. In an extreme case, even 25 times more often as in the case when the minority of sensors have full energy while the majority energy level of sensors is at 25%. By doing so, the mechanism decided to use the energy of a few sensors to prolong the lifetime of all others. Furthermore, due to the higher frequency of updates, the sensors with more energy that will transmit more often should eventually catch up with energy levels of the majority. In contrast, whenever a few sensors are close to depleting their energy, the majority will transmit more often. However, the ratio is not as high as in the first case. Such behaviour indicates that the mechanism favours the lifetimes of the majority over the minority.

The resulting energy balancing behaviour also depends on the value of the reward weight, i.e., ϕ. By changing it, it is possible to control the impact of sensors' energy levels on the scheduling process. The weight provides the system designer with the only way to control, to a certain extent, the scheduling process of our proposed scheduling mechanism. In an extreme case, i.e., by setting the value of ϕ to 0.75, the energy will play only a minor part in the decision-making process. While setting it to 0.25 would mean that the scheduling mechanism might be willing to overshoot the set target to preserve sensor's energy.

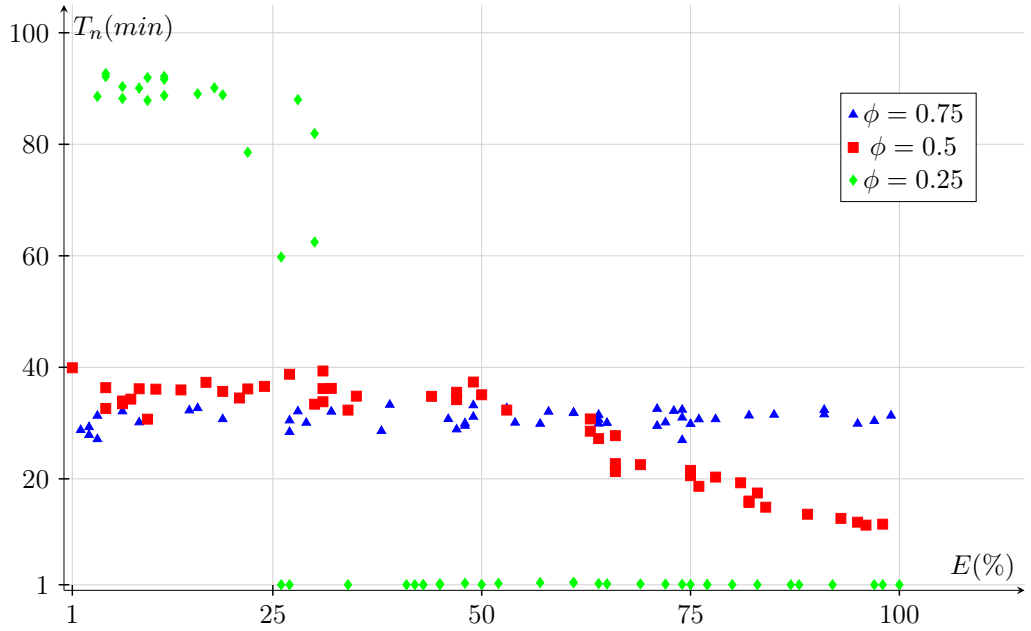


Figure 4.17: Impact of weight ϕ on sensors' lifetime.

In Fig. 4.17, we demonstrate the impact of the weight ϕ on the mechanism's performance (we performed test using DDPG implementation). As expected, when the value of ϕ to 0.75, the energy level makes little to no difference in the achieved average update interval. When we set the ϕ is 0.5, we can observe an intriguing decision the mechanism makes. When a sensor has an energy level above the average (for the case we show in Fig. 4.17 is 50%), the mechanism will decrease the sensor update interval. The reduced update interval is proportional to the sensors energy level, the higher the energy level, the lower is the sensor's update interval. When a sensor has below average energy, the sensors update interval is increased in comparison to a case in which the mechanism is not energy-aware, i.e., ϕ to 0.75. Interestingly, the mechanism tries to increase the update interval of every sensor with below-average energy level equally. When we set the ϕ to 0.25, we can observe a fascinating behaviour. The mechanism will schedule the sensors with high energy towards a minimum possible update interval, while for sensors with very low energy level, it will try to schedule towards the maximum possible update interval. Note that we set the minimum possible update interval to one minute and maximum to three hours. The mechanism never sets the sensor's update interval to the maximum possible. Due to the set target, it is not desirable, and the actual maximum we obtained in the test data is two hours. The lower the weight value, the lower will be the difference in the lifetime between the sensors with different energy levels. Meaning that by controlling ϕ, it is possible to set to what extent the lifetime of one sensor is prolonged at the expense of another.

The energy balancing capability of our mechanism ensures that sensors deployed at the same time, will also deplete their energy at the same time. Due to one sensor benefiting more from the use of correlated information, the sensors' energy levels will become unbalanced over time. The energy-awareness compensates the benefits and assures that sensors that benefit

the most will return the benefits to others by transmitting more often once the difference between energy levels becomes apparent. Such behaviour is desirable in deployments as it will enable infrastructure provider to replace entire sensors network at the same time.

4.6 Conclusion

In this chapter, we have proposed a DRL-based energy-aware scheduling mechanism capable of significantly prolonging the lifetime of battery-powered sensors without hindering the overall performance of the sensing process. We have demonstrated, using real-world observations, that the performance of our proposed mechanism is near-optimal. We have shown that the mechanism can be implemented either with a DQN or DDPG algorithm. However, our simulations indicate that DDPG-based mechanism would most likely be a better choice. Additionally, the proposed mechanism is capable of setting update intervals depending on the energy available on devices. Such behaviour assures that battery-powered sensor deployed at the same time will also expire at the same time. Thus this mechanism enables the infrastructure providers to replace the entire sensor deployments simultaneously. With the increase of energy-limited devices, the decision on when should a sensor transmit an update is complex yet crucial. As such, the energy-aware scheduling mechanism we proposed in this chapter can have a profound impact on the suitability of the future deployments of IoT sensing devices.

5 Conclusions and Future Directions

Conclusions and Future Directions

> *So, ask yourself: If what you're working on succeeds beyond your wildest dreams, would you have significantly helped other people? If not, then keep searching for something else to work on. Otherwise you're not living up to your full potential.*
>
> Andrew Ng, *Quora, 2016*

This thesis analysed how to take advantage of correlation exhibited in the information collected by low-power sensors. The main contribution of our work is the design of a Deep Reinforcement Learning (DRL) based scheduling mechanism capable of prolonging the lifetime of battery-powered sensors while simultaneously ensuring the accuracy of the observations reported to services from the sensors. In the remainder of this chapter, we first summarise our contributions and then propose directions for future research.

5.1 Contributions and Findings

In this section, we summarise the contributions of our work. In broad terms, we can divide them into two categories: findings of the analysis we presented in the chapter 3 and the lessons we learned from the implementation of the proposed scheduling mechanism presented in chapter 4.

5.1.1 In-depth Analysis of the System With Correlated Sources

In chapter 3, we focused on a system of two correlated sensors, i.e, information sources. We leveraged information collected by one source to improve the performance of the other. We established the optimal time shift between sources' updates such that the gain of using correlated information is maximised. We showed a trade-off between the gain and the source's update rate, which must be considered when setting the updating strategy for correlated sources. Additionally, by considering multiple covariance models to describe the correlation between the two sources, we could demonstrate the impact of correlation.

Furthermore, we also performed data analysis to provide intuition on the achievable benefits of correlated information for the system.

Using the Age of Information (AoI) concept, we were able to quantify which update from two sources is more informative to the system. In the case of correlated sources, an update from any source contains some information regarding the value of the observed physical phenomenon. Therefore, an update from any correlated source decreases the need for a fresh update from other sources. We relied on the estimation error to indirectly characterise the value of each update. The lower the estimation error resulting from an update, the higher the value of the information in that particular update. It is reasonable for sources to adopt the scheduling policy that results in the lowest estimation error. In other words, we have demonstrated that the value of information in an update depends on two factors: the AoI of the last received status update, and the correlation between the information sources.

The updating strategy that results from our work may be applied in an Internet of Things (IoT) network to address the challenge of scaling [107]. However, due to many factors, determining the updating strategy analytically for a system of multiple sensors is complicated. And, considering the dynamic environment, we turned to DRL to design a solution to obtain the updating strategy that takes advantage of correlated information.

5.1.2 Design of an Energy-aware Scheduling Mechanism

In chapter 4 we built on obtained insight and designed a DRL based energy-aware scheduler mechanism. The mechanism is capable of determining when a sensor should transmit its next observation. We employ a DRL algorithm to resolve a multi-objective decision as the mechanism's aim is to prolong sensors' lifetimes while maintaining the collection of information within the pre-specified accuracy range. We show that the performance of our mechanism is near optimal as we compare it to an ideal, all-knowing, scheduler and validate it over five different datasets obtained from real sensor deployments. Additionally, we demonstrated the unique feature of our solution, namely energy balancing. The mechanism is capable of determining to what extent the energy available to one sensor can be used to prolong the lifetime of others.

We envision the role of the mechanism in the IoT ecosystem as an entity that will enable the infrastructure providers to manage large-scale sensor deployments. The main benefit of deploying our scheduling mechanism will be to reduce deployments cost by prolonging battery-powered sensors' lifetimes. Additionally, we expect the performance of the scheduling mechanism to improve over time as it will be better able to extract the correlation from collected observations.

As mentioned, the unique feature of our mechanism is energy balancing. A good application of our work would be in smart agriculture. Sensors are usually deployed in groups, so, in the case of 20 soil moisture sensors in a field, some sensors might be in a part of the area prone to flooding. Our mechanism recognises the correlation between sensors and leverages

it to reduce their transmission rate since the reading from one sensor can be used in place of others. This could, in turn, lead to one sensor being requested more often, and its battery running down more rapidly than the others. To avoid such a situation, our proposed mechanism also takes into account the energy available and makes sure that all the sensors deplete their energy simultaneously. This means the battery depletion rate of the sensors can be controlled, thus preventing multiple trips to a field to replace the battery of just one or two sensors. Instead, a more efficient situation is created where sensor batteries can all be replaced at the same time.

5.1.3 Summary

The findings of this thesis have a few important implications. The most notable one is that we have shown that information collected by devices can be leveraged to extend the lifetime of battery-powered sensors. Perhaps, a more subtle implication of our work is that different infrastructure providers, i.e., sensor owners, can benefit by sharing the information their sensors are collecting. Such an approach is easily achievable as services rely on information stored in a database into which different infrastructure providers can input collected information. In short, we consider *information* collected by sensors as a *resource* that can be used to *improve network performance*, namely their energy efficiency.

5.2 Future works

The qualitative analysis and simulation-based performance of the scheduling mechanism indicate the possible high energy-savings an infrastructure provider will be able to obtain by integrating our solution into their networks. However, before the proposed mechanism can become an integral part of future network deployments, a more detailed investigation of certain aspects of our proposed solution is necessary. In what follows, we discuss possible continuation studies and future research directions.

5.2.1 Continuation Studies

The implementation of the scheduling mechanism we presented in chapter 4 opened intriguing practical issues that require further investigation. In the following, we briefly discuss the mechanism's feasibility, how to improve its performance using Gaussian Processes (GP), and how to better explain its decisions.

Feasibility

A comprehensible feasibility study should identify potential components that can be used in deployments and possible impacts that our mechanism can have on the overall system performance.

Our scheduling mechanism requires sufficient computation power and memory storage capacity. The Artificial Neural Network (ANN) training requires processing power, while storage is needed to save sensors' observations. We can place the mechanism anywhere in the network. However, the mechanism has to respond within the specified time. For example, if we set the mechanism in the cloud, the response time might be too large due to delays in the network. Therefore, the mechanism position relative to sensors is a crucial design factor. In the near future, deploying DRL-based solutions closer to IoT sensors will become easier. Several companies are currently designing components for deep learning, e.g., Xilinx's Adaptive Compute Acceleration Platform (ACAP) [108]. Using such components, deploying the mechanism closer to the sensors' location will be much more feasible.

On the other hand, the most noticeable impact of the mechanism on the network performance is in the additional messages used to alter sensors' update times. These messages are short, and their frequency depends on the number of sensors under mechanism control and the type of observed physical phenomena. For example, in our analysis, we noticed that the mechanism transmitted up to a few hundred control messages every hour. Such frequency should not pose any challenge to overall system performance. However, a much higher rate of control messages, e.g., a few hundred messages every minute can potentially negatively impact the system performance. Therefore, the feasibility study should estimate the number of control messages and, if necessary, design a strategy to reduce this amount, e.g., small changes in update time should not be transmitted, etc.

In general, the feasibility study should look into how it is possible to integrate our proposed scheduling mechanism into the network without any negative impact.

Gaussian Processes

Analysing data, we discovered that the observed physical phenomenon is more highly correlated within specific geographical areas. For example, in the Intel lab dataset [90], the sensors deployed closer to the window observe higher temperatures in comparison to sensors located in the middle of the room. In our work, we relied on extracting scaling parameters periodically. Instead, we could employ GP [109] to more efficiently model correlation. Using GP it is possible to use a kernel function, i.e., a covariance model, describing regions in which the information collected by sensors is more correlated. Additionally, we could leverage metadata which would help to describe the correlation between sensors better. For example, for the SmartSantander dataset [93], knowing if it is a cloudy or sunny day can help GP to provide better estimations.

In our proposed mechanism design, it is easy to accommodate the use of GP as only a small alteration is required. The measure of estimation error should be replaced by the variance resulting from the use of a GP estimation process, in both the state space and the reward function. In Fig. 5.1 we illustrate how GP would fit in the proposed scheduling mechanism. However, the main downside of using GP is that it requires a higher

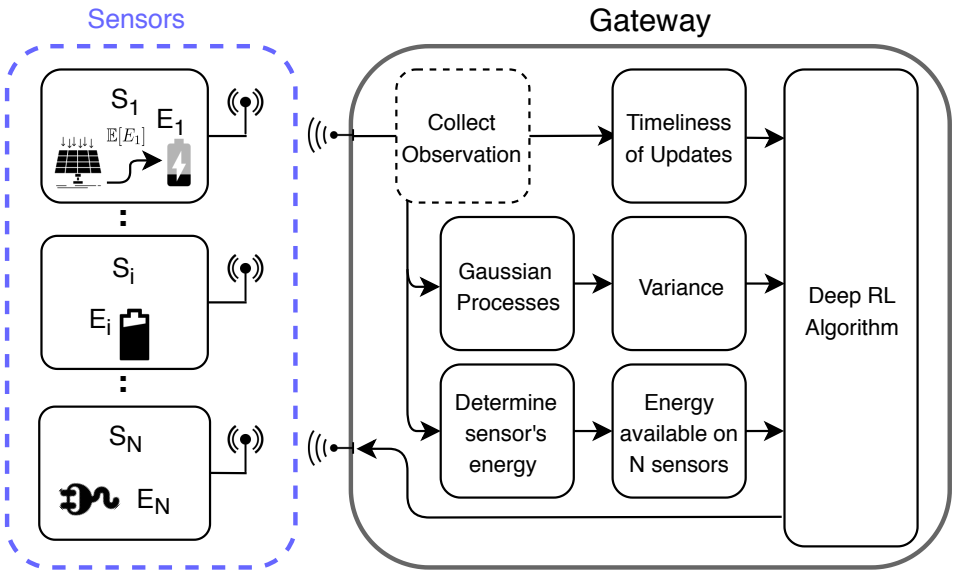


Figure 5.1: High-level scheduling mechanism design if implemented using GP.

computational power than the estimation error calculation.

Explaining the Mechanism's Decisions

In the background chapter, we mentioned that ANN is often regarded as a black-box in which decisions are often difficult to understand. When we made design choices for the ANN inputs, we relied on the intuition obtained from the extensive analysis of the system of correlated sources (chapter 3). However, our work would benefit from a more in-depth analysis of how our mechanism arrives at a particular decision. Such a study would serve to convince potential infrastructure providers to deploy our solution in their network. As demonstrated by the authors in [110], a better explanation of how the ANN derives a decision leads to a more sophisticated verification process and a possible better overall performance. A few tools are available to provide a better explainability such as Layer-wise Relevance Propagation (LRP) [111], or Sensitivity Analysis (SA) [112]. Offering a fully explainable solution would reassure infrastructure providers that our mechanism-learned behaviour is as expected.

5.2.2 Future Research Directions

In this subsection, we provide a short description of possible future research directions.

A system of sources relying on energy harvesting

The most interesting future direction of our work would be to consider a network of sensors using diverse primary power sources, e.g., mains-powered or event-based energy harvesting.

In such a case, the resulting scheduling policy will depend on the type of power source used by the sensor.

When the sensor has a stable power source, i.e., mains-powered, the energy cost is minimal. However, sensors face different restrictions such as contention for the transmission channel and low duty-cycles, e.g., Long Range Wide Area Network (LoRaWAN) duty-cycle is 1%[98]. Consequently, the sensor will not be able to transmit at every time instance. The challenge for the scheduling mechanism then becomes to determine how much more often should a mains-powered sensor transmit an observation in comparison to a battery-powered one. To obtain such behaviour, an alteration to the reward function is probably necessary, as well as perhaps a different ANN design.

Even more challenging would be a system with sensors relying on energy harvesters. In such a case, the mechanism will have to learn that a sensor with an energy harvester can replenish its energy. Additionally, the rate with which a sensor replenishes its energy is unknown and depends on many external factors. Therefore, the mechanism will have to learn that some sensors can replenish their energy, and others (battery-powered) do not. It is intuitively known to us that the sensor with an energy harvester should transmit more often than the battery-powered sensors (when their energy levels are the same). Our scheduling mechanism, on the other hand, has to experience a battery-powered sensor depleting its energy multiple times before it learns this. The mechanism has to be trained over the entire life-cycle of the network numerous times before it will be able to learn the difference between a sensor with an energy harvester and a battery-powered sensor. Following such an approach, the scheduling mechanism should learn the desired behaviour. However, the ANN training time would be significantly longer.

Distributed Approach

Multi-Agent Reinforcement Learning (MARL) could potentially be used to design a fully distributed version of the scheduling mechanism. In such a case, each sensor would be capable of making its own decision on when it should transmit an observation, i.e., decide on its sleep time. However, the sensor's limited processing power and increased energy consumption, in addition to the agents having only partial information regarding the actions of other sensors, limit the applicability of a distributed approach. Nonetheless, in the future an energy efficient processing unit capable of holding the trained ANN could be available and would make a distributed approach viable. In such a case, the most suitable option will probably be to train sensors offline and then transfer the trained ANN to them.

The main challenge would be to achieve cooperation between the agents as they have only limited knowledge of the actions and state of other agents. Consequently, in MARL, agents often learn to act greedily. In the scenario that we consider in our work, an agent could decide to not transmit for days, with the aim that other agents(sensors) will transmit more often to compensate for its decision of not transmitting. Additionally, obtaining energy

balancing in such a distributed manner would prove to be a significant challenge.

A distributed approach would also be to deploy agents in multiple gateways located in the same environment, e.g., a smart city. In such a case, each agent would control sensors that are connected to the gateway in which it resides. To improve its performance, an agent would need to cooperate with agents residing in other gateways. Designing such a system could prove to be an intriguing research challenge as a protocol on how agents should collaborate is required. Additionally, in such a setting, the agent could potentially greatly benefit from applying transfer and federated learning.

Transfer and Federated Learning

Transfer learning is a Machine Learning (ML) method in which a model trained by one agent is reused by another agent in different settings or for similar problem settings. By using an already trained model, it is possible to speed up the training process and to achieve better performance with less training data [113]. However, in some cases, the transferring ANN's weights result in degraded performance, meaning that the transferred ANN performs worse than a randomly initialized ANN would. The main challenge is to determine under which conditions it is beneficial for the system to transfer already trained weights. For example, it may be possible to reuse an already trained agent observing a particular phenomenon, e.g., temperature. Or agents that are deployed in a similar environment, e.g., field or a city, could reuse the ANN. Using transfer learning can enable the scheduling mechanism to significantly reduce the training time, thus making our solution easier to deploy.

Another method that we could potentially employ to improve our scheduling performance in real deployments is federated learning [114]. The federated learning approach could be used in a scenario in which we would deploy multiple scheduling mechanisms. In such a case, numerous agents could aggregate their locally trained ANNs models via a coordinating (federated) server without sharing data. Such an approach is desirable as it would be possible to train learning agents owned by different infrastructure providers. The main research objective of the future work leveraging federated learning would be to explore how to implement it. The benefit for the network would be to enable faster deployments, an ability to test multiple different ANN models simultaneously, and reduced power consumption on the gateways.

Appendix

A Derivations for Covariance Models

In the following we derive expressions for τ_p, τ^*, and G_{max} for a separable and two non-separable covariance classes, we employed in chapter 3 analysis.

We calculate the τ_p by calculating:

$$\varepsilon_2(d,0) = \varepsilon_1(0,\tau_p) \tag{5.1}$$

where

$$\varepsilon_i(d_i,t) = 1 - \mathrm{C}_i^2(d_i,t). \tag{5.2}$$

To calculate the τ^* we first have to calculate system error and then determine derivate:

$$\overline{\varepsilon}_{sys} = \frac{1}{T}\left(\int_0^{\tau_\Delta} \varepsilon_1(0,t)dt + \int_{\tau_\Delta}^{T} \varepsilon_2(d,t)dt\right), \tag{5.3}$$

where, for $0 \leq t \leq T$:

$$\Delta_1(t) = t \text{ and } \Delta_2(t) = t - \tau_\Delta,$$

and with the condition:

$$\tau_\Delta \geq \tau_p.$$

We defined G_{max} as following:

$$G_{max}(d,t) = \frac{\lim_{\lambda_2 \to \infty} \overline{\varepsilon}(d,t)}{\overline{\varepsilon}_1(0,t)}, \tag{5.4}$$

Note that when we calculate the G_{max} we denote the update time for the primary source as T_1 and T_2 denotes the update time of the secondary source.

A.1 Separable Model

$$\mathrm{C}_i^I(d_i, t_i) = exp(-\theta_2 d_i - \theta_1 \Delta_i(t)), \tag{5.5}$$

Following Eq. (5.1) we calculate τ_p as:

$$1 - \exp(-2\theta_2 d_2) = 1 - \exp(-2\theta_1 \tau_p).$$

Simplifying the above expression yields the following result for τ_p:

$$\tau_{p_I} = \frac{\theta_2 d_2}{\theta_1} \tag{5.6}$$

We continue by determining the optimal time-shift, τ^*:

$$\overline{\varepsilon}_{sys} = \frac{1}{T}\bigg(\int_0^{\tau_\Delta} \varepsilon_1(0,t)dt + \int_{\tau_\Delta}^{T} \varepsilon_2(d,t)dt \bigg),$$

The result of the integral is:

$$\overline{\varepsilon}_{sys} = \frac{\frac{\exp(-2d_2\theta_2)(\exp(2\theta_1(\tau_\Delta - T))-1)}{2\theta_1} + \frac{\exp(-2\tau_\Delta\theta_1)-1}{2\theta_1} + T}{T}$$

$$\frac{d\overline{\varepsilon}_{sys}}{d\tau_\Delta} = \frac{\exp(2\theta_1(\tau_\Delta - T) - 2d_2\theta_2) - \exp(-2\tau_\Delta\theta_1)}{T} = 0$$

which yields:

$$\tau_I^* = \frac{d_2\theta_2 + T\theta_1}{2\theta_1}. \tag{5.7}$$

Next, we determine the gain $G_{max}(d,t)$

$$G_{max}(d,t) = \frac{\lim_{\lambda_2 \to \infty} \overline{\varepsilon}(d,t)}{\overline{\varepsilon}_1(0,t)},$$

$$\lim_{\lambda_2 \to \infty} \overline{\varepsilon}(d,t) = \frac{1}{T_1}\bigg(\int_0^{\tau_p} \varepsilon_1(0,t)dt + (T_1 - \tau_p)\varepsilon_2(d,0) \bigg) =$$

$$\lim_{\lambda_2 \to \infty} \overline{\varepsilon}(d,t) = \frac{1}{T_1}\bigg(\tau_p + \frac{\exp(2\theta_1\tau_p) - 1}{2\theta_1} + \big(T_1 - \tau_p\big)\big(1 - \exp(-2d_2\theta_2)\big) \bigg) \tag{5.8}$$

$$\overline{\varepsilon}_1(0,t) = \frac{1}{T_1}\int_0^{T_1} 1 - exp(-2\theta_1 t) = \frac{1}{T_1}\big(t\Big|_0^{T_1} + \frac{\exp(-2\theta_1 t)}{2\theta_1}\Big|_0^{T_1}\big)$$

$$\bar{\varepsilon}_1(0,t) = \frac{1}{T_1}\big(T_1 + \frac{\exp(-2\theta_1 T_1)+1}{2\theta_1}\big) \tag{5.9}$$

By combing Eq. (5.8) and (5.9) we can obtain $G_{max}(d,t)$ as:

$$G_{max}(d,t) = \frac{\tau_p + \frac{\exp(2\theta_1\tau_p)-1}{2\theta_1} + \big(T_1 - \tau_p\big)\big(1-\exp(-2r\theta_2)\big)}{T_1 + \frac{\exp(-2\theta_1 T_1)+1}{2\theta_1}} \tag{5.10}$$

A.2 Non-separable Model I

$$\mathrm{C}_i^{II}(d_i,t_i) = \frac{\theta_1\Delta_i(t)+1}{\Big((\theta_1\Delta_i(t)+1)^2 + \theta_2^2 d_i^2\Big)^{(D+1)/2}}, \tag{5.11}$$

Following Eq. (5.1) and inserting $D=3$ we can calculate τ_p as follows:

$$\frac{(0+1)}{\Big((0+1)^2+\theta_2^2 d_2^2\Big)^2} = \frac{(\theta_1\tau_p+1)}{\Big((\theta_1\tau_p+1)^2+\theta_2^2 0^2\Big)^2}$$

$$\frac{1}{(1+\theta_2^2 d_2^2)^2} = \frac{1}{(\theta_1\tau_p+1)^3}$$

$$(\theta_1\tau_p+1)^3 = (1+\theta_2^2 d_2^2)^2$$

$$\tau_{p_{II}} = \frac{(1+\theta_2^2 d_2^2)^{\frac{2}{3}}-1}{\theta_1} \tag{5.12}$$

System estimation error for a case of when the two sources update with equal rates:

$$\bar{\varepsilon}_{sys} = \frac{1}{T}\Big(\int_0^{\tau_\Delta} 1 - \Big(\frac{\theta_1 t+1}{\big((\theta_1 t+1)^2\big)^{(D+1)/2}}\Big)^2 dt + \int_{\tau_\Delta}^{T} 1 - \Big(\frac{\theta_1 t+1}{\big((\theta_1 t+1)^2+\theta_2^2 d_2^2\big)^{(D+1)/2}}\Big)^2 dt\Big),$$

We calculate each integral separately and insert $D=3$:

$$\int_0^{\tau_\Delta} 1 - \Big(\frac{(\theta_1 t+1)}{\big((\theta_1 t+1)^2\big)^{(D+1)/2}}\Big)^2 dt = \int_0^{\tau_\Delta} 1 - \Big(\frac{(\theta_1 t+1)}{\big((\theta_1 t+1)^2\big)^2}\Big)^2 dt =$$

$$\int_0^{\tau_\Delta} 1 - \Big(\frac{1}{(\theta_1 t+1)^3}\Big)^2 = \int_0^{\tau_\Delta} 1 - \frac{1}{(\theta_1 t+1)^6}$$

To solve the integral we use 2.117 (3) in [115], which is:

$$\int \frac{dx}{Z_1^m} = -\frac{1}{(m-1)bZ_1^{m-1}}$$

where $Z_1 = (a + bx)$ and result is:

$$t\Big|_0^{\tau_\Delta} + \frac{1}{5\theta_1(\theta_1 t+1)^5}\Big|_0^{\tau_\Delta} = \tau_\Delta + \frac{1}{5\theta_1(\theta_1\tau_\Delta+1)^5} - \frac{1}{5\theta_1} \tag{5.13}$$

Unfortunately, the second integral does not have a rational solution. Therefore, we are unable to determine τ_{II}^* analytically. Instead we focus on calculating G_{max} by following Eq. (5.4):

$$G_{max}(d,t) = \frac{\lim_{\lambda_2\to\infty}\overline{\varepsilon}(d,t)}{\overline{\varepsilon}_1(0,t)},$$

$$\lim_{\lambda_2\to\infty}\overline{\varepsilon}(d,t) = \frac{1}{T_1}\bigg(\int_0^{\tau_p}\varepsilon_1(0,t)dt + (T_1-\tau_p)\varepsilon_2(d,0)\bigg) =$$

Following the method we applied to determine the integral in Eq. (5.13), we can resolve the above integral and result is as follows:

$$\frac{1}{T_1}\int_0^{\tau_p}\varepsilon_1(0,t)dt = \frac{1}{T_1}\big(t\Big|_0^{\tau_p} + \frac{1}{5\theta_1(\theta_1 t+1)^5}\Big|_0^{\tau_p}\big) = \frac{1}{T_1}\big(\tau_p + \frac{1}{5\theta_1(\theta_1\tau_p+1)^5} - \frac{1}{5\theta_1}\big)$$

while

$$(T_1-\tau_p)\varepsilon_2(d,0) = (T_1-\tau_p)(1-\frac{1}{(1+-2\theta_2^2 d_2^2)^4})$$

We can determine the $\overline{\varepsilon}_1(0,t)$ by following the solution we obtained in Eq. (5.13):

$$\overline{\varepsilon}_1(0,t) = t\Big|_0^{T_1} + \frac{1}{5\theta_1(\theta_1 t+1)^5}\Big|_0^{T_1} = T_1 + \frac{1}{5\theta_1(\theta_1 T_1+1)^5} - \frac{1}{5\theta_1}$$

by combining the above three solutions we can determine the $G_{max}(d,t)$:

$$G_{max}(d,t) = \frac{\tau_{p_{II}} + \frac{1}{5\theta_1(\theta_1\tau_{p_{II}}+1)^5} - \frac{1}{5\theta_1} + (T_1-\tau_{p_{II}})(1-\frac{1}{(1+-2\theta_2^2 d_2^2)^4})}{T_1 + \frac{1}{5\theta_1(\theta_1 T_1+1)^5} - \frac{1}{5\theta_1}} \tag{5.14}$$

A.3 Non-separable Model II

$$\mathrm{C}_i^{III}(d_i,t_i) = \exp(-\theta_2^2 d_i^2 - \theta_1\Delta_i(t) - \theta_3\Delta_i(t)d_i^2). \tag{5.15}$$

Following Eq. (5.1) we calculate τ_p as:

$$1 - \exp(-2\theta_1 0 - 2\theta_2^2 d_2^2 - 2\theta_3 0 d_2^2) = 1 - \exp(-2\theta_1\tau_p - 2\theta_2^2 0^2 - 2\theta_3\tau_p 0^2)$$

$$\exp(-2\theta_2^2 d_2^2) = \exp(-2\theta_1\tau_p)$$

$$-2\theta_2^2 d_2^2 = -2\theta_1\tau_p$$

$$\tau_{p_{III}} = \frac{\theta_2 d_2^2}{\theta_1} \tag{5.16}$$

System estimation error for a case of when the two sources update with equal rates:

$$\overline{\varepsilon}_{sys} = \frac{1}{T}\left(\int_0^{\tau_\Delta} 1 - \left(\exp(-\theta_1 t)\right)^2 dt + \int_{\tau_\Delta}^{T} 1 - \left(\exp(-\theta_1 t - \theta_2 d_2^2 - \theta_3 t d_2^2)\right)^2 dt \right.$$

We calculate each integral separately:

$$\int_0^{\tau_\Delta} 1 - \left(\exp(-\theta_1 t)\right)^2 dt = t\Big|_0^{\tau_\Delta} + \frac{\exp(-2\theta_1 t)}{2\theta_1}\Big|_0^{\tau_\Delta} = \tau_\Delta + \frac{\exp(-2\theta_1\tau_\Delta)}{2\theta_1} - \frac{1}{2\theta_1}$$

$$\int_{\tau_\Delta}^{T} 1 - \left(\exp(-\theta_1 t - \theta_2 d_2^2 - \theta_3 t d_2^2)\right)^2 dt = \int_{\tau_\Delta}^{T} 1 dt - \int_{\tau_\Delta}^{T} \exp(-2\theta_1 t - 2\theta_2^2 d_2^2 - 2\theta_3 t d_2^2) dt =$$

$$\int_{\tau_\Delta}^{T} 1 dt - \exp(-2\theta_2^2 d_2^2) \int_{\tau_\Delta}^{T} \exp(-2t(\theta_3 d_2^2 + \theta_1)) dt$$

To solve the above integral we substitute $u = -2t(\theta_3 d_2^2 + \theta_1)$ and $dt = du/(-2(\theta_3 d_2^2 + \theta_1))$. Resulting in:

$$\int_{\tau_\Delta}^{T} 1 dt + \frac{\exp(-2\theta_2^2 d_2^2)}{2(\theta_3 d_2^2 + \theta_1)} \int \exp(u) du = t\Big|_{\tau_\Delta}^{T} + \frac{\exp(-2\theta_2^2 d_2^2)\exp(-2t(\theta_3 d_2^2 + \theta_1))}{2(\theta_3 d_2^2 + \theta_1)}\Big|_{\tau_\Delta}^{T} =$$

$$T - \tau_\Delta + \frac{\exp(-2\theta_2^2 d_2^2)\exp(-2(T-\tau_\Delta)(\theta_3 d_2^2 + \theta_1))}{2(\theta_3 d_2^2 + \theta_1)} - \frac{\exp(-2\theta_2^2 d_2^2)}{2(\theta_3 d_2^2 + \theta_1)}$$

$$\overline{\varepsilon}_{sys} = \frac{1}{T}\left(\frac{\exp(-2\theta_1\tau_\Delta)}{2\theta_1} - \frac{1}{2\theta_1} + T + \frac{\exp(-2\theta_2^2 d_2^2)\exp(-2(T-\tau_\Delta)(\theta_3 d_2^2 + \theta_1))}{2(\theta_3 d_2^2 + \theta_1)} - \frac{\exp(-2\theta_2^2 d_2^2)}{2(\theta_3 d_2^2 + \theta_1)} \right)$$

The derivative is:

$$\bar{\varepsilon}'_{sys} = \frac{1}{T}\Bigg(- \exp(-2\theta_1\tau_\Delta) + \frac{2(\theta_3 d_2^2 + \theta_1)\exp(-2\theta_2^2 d_2^2)\exp(-2(\theta_3 d_2^2 + \theta_1)(T-\tau_\Delta))}{2(\theta_3 d_2^2 + \theta_1)}\Bigg) =$$

$$\frac{1}{T}\Bigg(- \exp(-2\theta_1\tau_\Delta) + \exp(-2\theta_2^2 d_2^2)\exp(-2(\theta_3 d_2^2 + \theta_1)(T-\tau_\Delta))\Bigg) = 0$$

$$\exp(-2\theta_1\tau_\Delta) = \exp(-2\theta_2^2 d_2^2)\exp(-2(\theta_3 d_2^2 + \theta_1)(T-\tau_\Delta))$$

Therefore, the optimum is:

$$\tau^*_{III} = \frac{(\theta_3 d_2^2 + \theta_1)T + 2\theta_2^2 d_2^2}{\theta_3 d_2^2 + 3\theta_1} \tag{5.17}$$

We follow Eq. (5.4) to calculate G_{max}:

$$G_{max}(d,t) = \frac{\lim_{\lambda_2\to\infty}\bar{\varepsilon}(d,t)}{\bar{\varepsilon}_1(0,t)},$$

$$\lim_{\lambda_2\to\infty}\bar{\varepsilon}(d,t) = \frac{1}{T_1}\Bigg(\int_0^{\tau_p}\varepsilon_1(0,t)dt + (T_1-\tau_p)\varepsilon_2(d,0)\Bigg) =$$

$$= \frac{1}{T_1}\Bigg(\int_0^{\tau_p} 1 - \exp(-2\theta_1 t)dt + (T_1-\tau_p)(1-\exp(-2\theta_2^2 d_2^2))\Bigg) =$$

$$= \frac{1}{T_1}\Bigg(\tau_p + \frac{\exp(-2\theta_1\tau_p)}{2\theta_1} - \frac{1}{2\theta_1} + (T_1-\tau_p)(1-\exp(-2\theta_2^2 d_2^2))\Bigg)$$

$$\bar{\varepsilon}_1(0,t) = \frac{1}{T_1}\int_0^{T_1}\varepsilon_1(0,t)dt = \frac{1}{T_1}\Bigg(t\Big|_0^{T_1} + \frac{\exp(-2\theta_1 t)}{2\theta_1}\Big|_0^{T_1}\Bigg) = \frac{1}{T_1}\Bigg(T_1 + \frac{\exp(-2\theta_1 T_1)}{2\theta_1} - \frac{1}{2\theta_1}\Bigg)$$

By combining above solutions we can determine the $G_{max}(d,t)$ as:

$$G_{max}(d,t) = \frac{2\theta_1\tau_p + \exp(-2\theta_1\tau_p) + 2\theta_1(T_1-\tau_p)(1-\exp(-2\theta_2^2 d_2^2)) - 1}{2\theta_1 T_1 + \exp(-2\theta_1 T_1) - 1} \tag{5.18}$$

B Dataset Descriptions

In the following we provide short descriptions of the datasets which we use in the data analysis in chapter 3, and as environmental information in the evaluation of the proposed scheduling mechanism in chapter 4.

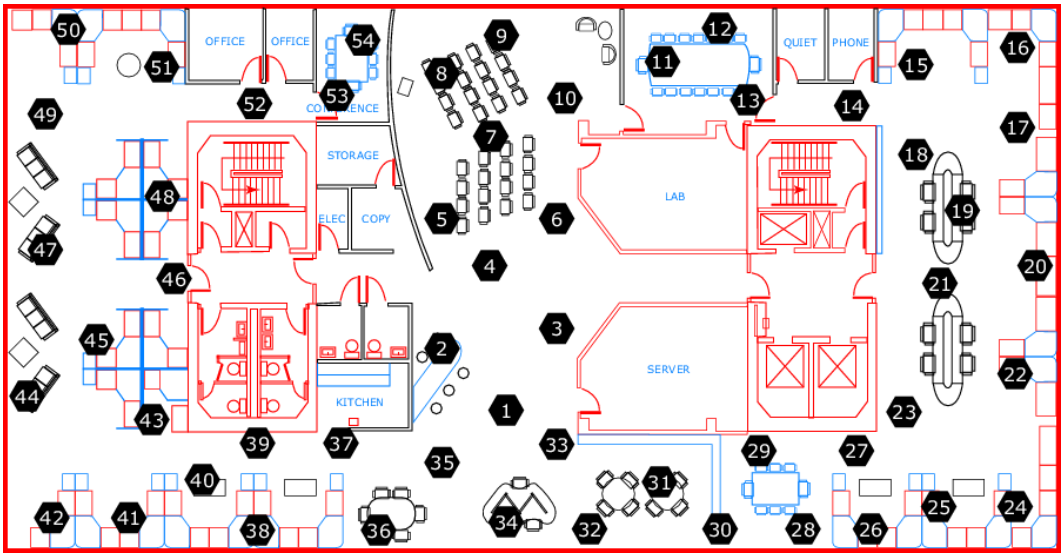


Figure B.1: Intel Berkeley Research lab dataset layout. Source: [90].

B.1 Intel Berkeley Research Lab Dataset

In this thesis, we leverage the Intel dataset in our analysis in chapter 3 and in the validation process in chapter 4. In total, the dataset contains observations collected from 54 sensors deployed in the Intel Berkeley Research laboratory, California U.S. The observations were collected between February 28th and April 5th, 2004. Collected measurements provide us with a great representation of how a physical phenomenon is distributed in space and evolves in time. In other words, the Intel dataset gives us a good ground truth as the data comprises observations collected very frequently (every 31 seconds). As such, it is an excellent choice in the analysis as well as for the mechanism design. However, we limit the number of days and the sensors we use.

In our analysis in chapter 3, we rely on nine sensors, all located in one room, that have collected observations between 1st and 21st of March 2004. We use sensor number 14, 22, 23, 24, 25, 26, 27, 28, and 29. In Fig. B.1 each number represents the location of the sensor. There are other sensors located in the same room, however, due to missing observations (for a day or more in the selected three weeks period), they were not part of our analysis.

In the evaluation process in chapter 4 the Intel dataset provides us with a great asset as it enables us to validate our mechanism performance on the data which exhibits high periodicity (daily and weekly). We use data collected between March 1st and 9th 2004. In our analysis, we exclude sensors number 5, 15, 18, and 37, as during those days they collected erroneous observations or the sensor was deactivated. We limited used days to nine days as a longer period would result in fewer sensors we could potentially use in our work. We split the data into two parts. The first part, i.e., the first six days, we use to train the Deep Reinforcement Learning (DRL) agent. The second part we leverage to validate the learned behaviour. The validation part of the data starts during the weekend (Sunday

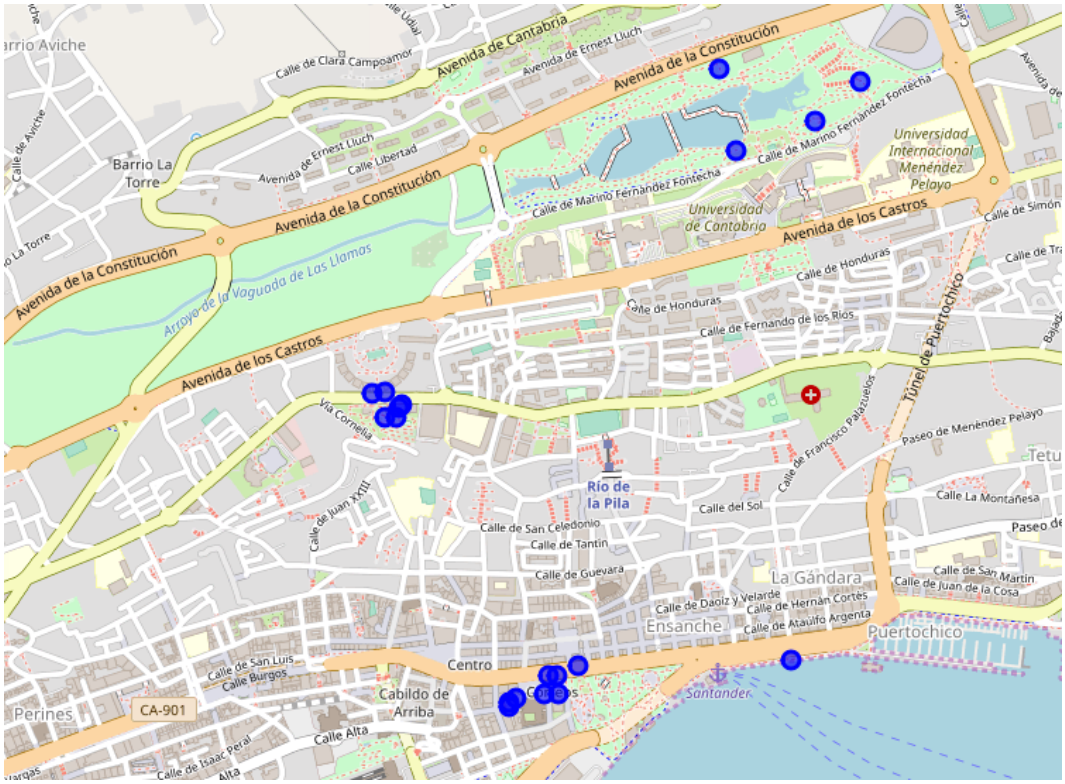


Figure B.2: Positions of temperature sensors in SmartSantander testbed

7th of March 2004) and lasts for two more working days. Following such an approach, we ensure that the mechanism is evaluated with observations collected on weekdays and during the weekend.

In general, the environment in which observations from 54 sensors were collected was very controlled and as such an ideal for the use in our work as it enables to know the ground truth(real observation values) very well. Additionally, the dataset contains a lot of additional information that can help in determining the correlation between sensors, e.g., office layout, work hours, etc. In contrast, the SmartSantander dataset allows us to evaluate our scheduling mechanism in a more realistic, less controlled environment.

B.2 SmartSantander Dataset

The SmartSantander dataset allows us to evaluate the scheduling mechanism performance in a smart city environment. In total, we use observations from twenty temperature, ten humidity, and eight ambient noise sensors. These sensors are deployed all over the city, typically on lamp posts. In Fig. B.2 we show the temperature sensor locations. Note that distances between sensors vary from a few meters to over 2 km. We focus on the observations that were collected between April 1st and April 19th, 2018. In contrast to intel data, the sensors did not collect observations very frequently; except for a few sensors, most have collected observation only every few hours. To that end, as we mentioned in chapter 4, we

use the Amelia II software program [104].

Amelia II software is a tool that leverages the imputation process to generate missing values. The authors describe the method Amelia II uses to generate missing data as follows: “Multiple imputation involves imputing m values for each missing cell in your data matrix and creating m "completed" data sets. (Across these completed data sets, the observed values are the same, but the missing values are filled in with different imputations that reflect our uncertainty about the missing data.)”[1]. To obtain the data for the evaluation, we, using Amelia, generate five new observation sets for a sensor with missing values. Then we average over the five datasets to obtain the end dataset. In the imputation process, we use observations from one sensor. We use observations from the sensor with the most collected observations. However, to avoid adding bias, we remove that sensor from the evaluation process.

[1]Software Amelia may be obtained at the following site: gking.harvard.edu/amelia

Acronyms

4G	4^{th} generation
5G	5^{th} generation
ACAP	Adaptive Compute Acceleration Platform
ANN	Artificial Neural Network
AoI	Age of Information
CPS	Cyber-Physical System
DDPG	Deep Deterministic Policy Gradient
DRL	Deep Reinforcement Learning
DQN	Deep Q-Network
GP	Gaussian Processes
IoT	Internet of Everything
IoNT	Internet of Nano Things
IoT	Internet of Things
IIoT	Industrial IoT
ITU	International Telecommunications Union
H2M	Human-to-Machine
LMMSE	Linear Minimum Mean Square Error
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-Power Wide-Area Network
LRP	Layer-wise Relevance Propagation
LTE-M	Long Term Evolution for Machines

M2M Machine-to-Machine

MARL Multi-Agent Reinforcement Learning

MDP Markov Decision Process

ML Machine Learning

MMSE Minimum Mean Square Error

MSE Mean Square Error

NB-IoT Narrowband IoT

RFID Radio-frequency Identification

RL Reinforcement Learning

SA Sensitivity Analysis

UAV Unmanned Aerial Vehicle

WSN Wireless Sensor Network

WWW World Wide Web

Bibliography

[1] Fortune Business Insights, "Internet of Things (IoT) Market Size, Share and Industry Analysis, 2019-2026," *Report ID: FBI100307*, Jul. 2019.

[2] IoT Analytics, "State of the IoT 2018:Number of IoT Devices now at 7B - Market Accelerating," Aug. 2018, Last Visited on February 15th, 2020. [Online]. Available: https://iot-analytics.com

[3] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A Comparative Study of LPWAN Technologies for Large-scale IoT Deployment," *Elsevier, ICT express*, vol. 5, no. 1, pp. 1–7, Mar. 2019.

[4] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[5] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, Jan. 2014.

[6] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big Data in Smart Farming - A review," *Elsevier, Agricultural Systems*, vol. 153, pp. 69–80, May. 2017.

[7] S. Kaul, R. Yates, and M. Gruteser, "Real-time Status: How Often Should One Update?" in *Proc. IEEE INFOCOM.* Orlando, FL, USA, Mar. 2012, pp. 2731–2735.

[8] M. Costa, M. Codreanu, and A. Ephremides, "On the Age of Information in Status Update Systems With Packet Management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, Feb. 2016.

[9] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or Wait: How to Keep Your Data Fresh," *IEEE Transactions on Information Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[10] R. D. Yates, "Lazy is Timely: Status Updates by an Energy Harvesting Source," in *Proc. IEEE ISIT.* Hong Kong, Jun. 2015, pp. 3008–3012.

[11] J. Yiu, "Cortex-M for Beginners an Overview of the ARM Cortex-M Processor Family and Comparison," *ARM Limited, White paper*, pp. 1–25, Sep. 2016.

[12] IoT Analytics, "LPWAN Emerging as Fastest Growing IoT Communication Technology," Sep. 2018, Last Visited on February 15th, 2020. [Online]. Available: https://iot-analytics.com

[13] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, Feb. 2017.

[14] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A Vision, Architectural Rlements, and Future Directions," *Elsevier, Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[15] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Elsevier, Computer networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[16] E. Borgia, "The Internet of Things Vision: Key Features, Applications and Open Issues," *Elsevier, Computer Communications*, vol. 54, pp. 1–31, Dec. 2014.

[17] International Telecommunication Union (ITU), "ITU Internet Reports 2005: The Internet of Things," *Geneva, Switzerland*, Nov. 2005, Last Visited on February 15th, 2020. [Online]. Available: http://www.itu.int

[18] J. B. Kennedy, "When woman is boss: An interview with Nikola Tesla," *Colliers, Seattle*, Jan. 1926, Last Visited on February 15th, 2020. [Online]. Available: http://www.tfcbooks.com/tesla/1926-01-30.html

[19] K. Ashton, "That 'Internet of Things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, Jun. 2009, Last Visited on February 15th, 2020. [Online]. Available: https://www.rfidjournal.com

[20] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A Review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT)," in *Proc. IEEE Internet Technologies and Applications*. Wrexham, United Kingdom, Sep. 2015, pp. 219–224.

[21] CMU SCS Coke Machine, "The "Only" Coke Machine on the Internet," *The Carnegie Mellon University Computer Science Department*, Jan. 2014, Last Visited on February 15th, 2020. [Online]. Available: https://www.cs.cmu.edu/~coke/

[22] M. Rosemann, "The Internet of Things : New Digital Capital in the Hands of Customers," *Business Transformation Academy, Business Transformation Journal*, no. 9, pp. 6–15, Jan. 2014, Last Visited on February 15th, 2020. [Online]. Available: http://eprints.qut.edu.au/66451/

[23] T. Mirlacher, R. Buchner, F. Förster, A. Weiss, and M. Tscheligi, "Ambient Rabbits Likeability of Embodied Ambient Displays," in *Proc. European Conference on Ambient Intelligence, Springer*. Salzburg, Austria, Nov. 2009, pp. 164–173.

[24] C. Zhu, V. C. Leung, L. Shu, and E. C.-H. Ngai, "Green Internet of Things for Smart World," *IEEE Access*, vol. 3, pp. 2151–2162, Nov. 2015.

[25] M. R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, and L. Ladid, "Internet of Things in the 5G Era: Enablers, Architecture, and Business Models," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[26] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A Survey on LPWA Technology: LoRa and NB-IoT," *Elsevier, ICT Express*, vol. 3, no. 1, pp. 14–21, Mar. 2017.

[27] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy Conservation in Wireless Sensor Networks: A Survey," *Elsevier, Ad hoc networks*, vol. 7, no. 3, pp. 537–568, May 2009.

[28] T. Rault, A. Bouabdallah, and Y. Challal, "Energy Efficiency in Wireless Sensor Networks: A Top-down Survey," *Elsevier, Computer Networks*, vol. 67, pp. 104–122, Jul. 2014.

[29] L. A. Villas, A. Boukerche, H. A. De Oliveira, R. B. De Araujo, and A. A. Loureiro, "A Spatial Correlation Aware Algorithm to Perform Efficient Data Collection in Wireless Sensor Networks," *Elsevier, Ad Hoc Networks*, vol. 12, pp. 69–85, Jan. 2014.

[30] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 828–854, 2nd Quart., 2017.

[31] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque, "Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 181–194, 1st Quart., 2014.

[32] M. C. Vuran, Ö. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Elsevier, Computer Networks*, vol. 45, no. 3, pp. 245–259, 2004.

[33] J.-F. Chamberland and V. V. Veeravalli, "How Dense Should a Sensor Network Be for Detection With Correlated Observations?" *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 5099–5106, Nov. 2006.

[34] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," in *Proc. IEEE ICDE*. Atlanta, GA, USA, Apr. 2006, pp. 1–12.

[35] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven Data Acquisition in Sensor Networks," in *Proc. VLDB*. Toronto, Canada, Aug. 2004, pp. 588–599.

[36] L. A. Villas, A. Boukerche, D. L. Guidoni, H. A. De Oliveira, R. B. De Araujo, and A. A. Loureiro, "An Energy-aware Spatio-temporal Correlation Mechanism to Perform Efficient Data Collection in Wireless Sensor Networks," *Elsevier, Computer Communications*, vol. 36, no. 9, pp. 1054–1066, May 2013.

[37] T. Yu, A. M. Akhtar, A. Shami, and X. Wang, "Energy-efficient Scheduling Mechanism for Indoor Wireless Sensor Networks," in *IEEE VTC*. Glasgow, Scotland, May 2015, pp. 1–6.

[38] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low Power Wide Area Networks: An Overview," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 2, pp. 855–873, 2nd Quart., 2017.

[39] B. Harrington, Y. Huang, J. Yang, and X. Li, "Energy-efficient Map Interpolation for Sensor Fields Using Kriging," *IEEE Transactions on Mobile Computing*, vol. 8, no. 5, pp. 622–635, May 2009.

[40] G. Hernandez-Penaloza and B. Beferull-Lozano, "Field Estimation in Wireless Sensor Networks Using Distributed Kriging," in *Proc. IEEE ICC*. Ottawa, Canada, Jun. 2012, pp. 724–729.

[41] W. Liu, Y. Shoji, and R. Shinkuma, "Logical correlation-based sleep scheduling for WSNs in ambient-assisted homes," *IEEE Sensors Journal*, vol. 17, no. 10, pp. 3207–3218, Mar. 2017.

[42] D. L. Zimmerman, "Optimal Network Design for Spatial Prediction, Covariance Parameter Estimation, and Empirical Prediction," *Wiley Online Library, Environmetrics: The official journal of the International Environmetrics Society*, vol. 17, no. 6, pp. 635–652, Aug. 2006.

[43] A. Krause, A. Singh, and C. Guestrin, "Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies," *Journal of Machine Learning Research*, vol. 9, pp. 235–284, Feb. 2008.

[44] F. Lindgren, H. Rue, and J. Lindström, "An Explicit Link Between Gaussian Fields and Gaussian Markov Random Fields: The Stochastic Partial Differential Equation Approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 4, pp. 423–498, Aug. 2011.

[45] N. Cressie, C. A. Calder, J. S. Clark, J. M. V. Hoef, and C. K. Wikle, "Accounting for Uncertainty in Ecological Analysis: The Strengths and Limitations of Hierarchical Statistical Modeling," *Ecological Applications*, vol. 19, no. 3, pp. 553–570, Apr. 2009.

[46] R. D. Yates and S. Kaul, "Real-time Status Updating: Multiple Sources," in *Proc. IEEE ISIT*. Cambridge, MA, USA., Jul. 2012, pp. 2666–2670.

[47] S. K. Kaul, R. D. Yates, and M. Gruteser, "Status Updates Through Queues," in *Proc. IEEE CISS*. Princeton, NJ, USA, Mar. 2012, pp. 1–6.

[48] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "Towards an Effective Age of Information: Remote Estimation of a Markov Source," in *Proc. IEEE INFOCOM*. Honolulu, HI, USA, Apr. 2018, pp. 367–372.

[49] A. Kosta, N. Pappas, A. Ephremides, and V. Angelakis, "Age and Value of Information: Non-linear Age Case," in *Proc. IEEE ISIT*. Paris, France, Jul. 2017, pp. 326–330.

[50] B. Yin, S. Zhang, Y. Cheng, L. X. Cai, Z. Jiang, S. Zhou, and Z. Niu, "Only Those Requested Count: Proactive Scheduling Policies for Minimizing Effective Age-of-Informatio," in *Proc. IEEE INFOCOM*. Paris, France, Apr. 2019, pp. 109–117.

[51] Y. Sun, Y. Polyanskiy, and E. Uysal-Biyikoglu, "Remote Estimation of the Wiener Process Over a Channel With Random Delay," in *Proc. IEEE ISIT*. Aachen, Germany, Jun. 2017, pp. 321–325.

[52] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of Information of Multiple Sources With Queue Management," in *Proc. IEEE ICC*. London, United Kingdom, Jun. 2015, pp. 5935–5940.

[53] R. D. Yates and S. K. Kaul, "The Age of Information: Real-time Status Updating by Multiple Sources," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1807–1827, Mar. 2019.

[54] J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Updating Strategies in the Internet of Things by Taking Advantage of Correlated Sources," in *Proc. IEEE Globecom*. Singapore, Dec. 2017, pp. 1–6.

[55] ——, "Using Correlated Information to Extend Device Lifetime," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2439–2448, Apr. 2018.

[56] Z. Jiang and S. Zhou, "Status from a Random Field: How Densely Should One Update?" in *Proc. ISIT*. Paris, France, Jul. 2019, pp. 1037–1041.

[57] S. Poojary, S. Bhambay, and P. Parag, "Real-time Status Updates for Correlated Source," in *Proc. IEEE ITW Workshop*. Kaohsiung, Taiwan, Nov. 2017, pp. 274–278.

[58] Q. He, G. Dan, and V. Fodor, "Minimizing Age of Correlated Information for Wireless Camera Networks," in *Proc. IEEE INFOCOM Workshop*. Honolulu, HI, USA, Apr. 2018, pp. 547–552.

[59] A. E. Kalor and P. Popovski, "Minimizing the Age of Information from Sensors with Common Observations," *IEEE Wireless Communications Letters*, Oct. 2019.

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

[61] R. A. Howard, *Dynamic Programming and Markov Processes*. John Wiley, Jun. 1960.

[62] M. A. Alsheikh, D. T. Hoang, D. Niyato, H.-P. Tan, and S. Lin, "Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 3, pp. 1239–1267, 3rd Quart., 2015.

[63] J. M. Zurada, *Introduction to Artificial Neural Systems*. West publishing company St. Paul, Jan. 1992, vol. 8.

[64] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[65] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari With Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, Dec. 2013.

[66] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the Game of Go With Deep Neural Networks and Tree Search," *Nature*, vol. 529, no. 7587, p. 484, Jan. 2016.

[67] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *Proc. ICML*. Beijing, China, Jun. 2014, pp. 1–9.

[68] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining Improvements in Deep Reinforcement Learning," in *Proc. Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, Feb. 2018, pp. 3215–3222.

[69] O. Simeone, "A Very Brief Introduction to Machine Learning With Applications to Communication Systems," *Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, Nov. 2018.

[70] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[71] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning That Matters," in *Proc. Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA, USA, Feb. 2018, pp. 3207–3214.

[72] A. Zappone, M. Di Renzo, M. Debbah, T. T. Lam, and X. Qian, "Model-Aided Wireless Artificial Intelligence: Embedding Expert Knowledge in Deep Neural Networks for Wireless System Optimization," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 60–69, Sep. 2019.

[73] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 1996–2018, 4th Quart., 2014.

[74] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 2923–2960, 4th Quart., 2018.

[75] F. Li, K.-Y. Lam, Z. Sheng, X. Zhang, K. Zhao, and L. Wang, "Q-Learning-Based Dynamic Spectrum Access in Cognitive Industrial Internet of Things," *Mobile Networks Applications*, vol. 23, no. 6, pp. 1636–1644, Dec. 2018.

[76] Y. Chu, P. D. Mitchell, and D. Grace, "ALOHA and Q-learning Based Medium Access Control for Wireless Sensor Networks," in *Proc. IEEE ISWCS*. Paris, France, Aug. 2012, pp. 511–515.

[77] J. Zhu, Y. Song, D. Jiang, and H. Song, "A New Deep-Q-Learning-Based Transmission Scheduling Mechanism for the Cognitive Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.

[78] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient Scheduling for Real-time Systems Based on Deep Q-learning Model," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 132–141, Jan. 2019.

[79] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 624–635, April 2018.

[80] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, and R. Y. Kwok, "Deep Reinforcement Learning for Intelligent Internet of Vehicles: An Energy-Efficient Computational Offloading Scheme," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[81] J. Du, H. Chen, and W. Zhang, "A Deep Learning Method for Data Recovery in Sensor Networks Using Effective Spatio-temporal Correlation Data," *Sensor Review*, no. 2, pp. 208–217, Mar. 2019.

[82] J. Zheng, Y. Cai, X. Shen, Z. Zheng, and W. Yang, "Green Energy Optimization in Energy Harvesting Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 150–157, Nov. 2015.

[83] F. A. Aoudia, M. Gautier, and O. Berder, "RLMan: An Energy Manager Based on Reinforcement Learning for Energy Harvesting Wireless Sensor Networks," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 2, pp. 408–417, Jun. 2018.

[84] A. Ortiz, H. Al-Shatri, T. Weber, and A. Klein, "Multi-Agent Reinforcement Learning for Energy Harvesting Two-Hop Communications with a Partially Observable State," *arXiv preprint arXiv:1702.06185*, Feb. 2017.

[85] M. K. Sharma, A. Zappone, M. Assaad, M. Debbah, and S. Vassilaras, "Distributed Power Control for Large Energy Harvesting Networks: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 4, pp. 1140–1154, Dec. 2019.

[86] J. Hribar and L. A. DaSilva, "Utilising Correlated Information to Improve the Sustainability of Internet of Things Devices," in *Proc. IEEE WF-IoT*. Limerick, Ireland, Apr. 2019, pp. 1–4.

[87] J. Hribar, A. Marinescu, G. A. Ropokis, and L. A. DaSilva, "Using Deep Q-learning To Prolong the Lifetime of Correlated Internet of Things Devices," in *Proc. IEEE ICC Workshops*. Shanghai, China, May 2019, pp. 1–6.

[88] N. Cressie and H.-C. Huang, "Classes of Nonseparable, Spatio-Temporal Stationary Covariance Functions," *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1330–1339, Dec. 1999.

[89] M. L. Stein, "Space-time Covariance Functions," *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 310–321, Mar. 2005.

[90] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel Lab Data," *Online dataset*, Mar. 2004, Last Visited on February 15th, 2020. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.html

[91] T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data," *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 590–600, Jun. 2002.

[92] Y. Chen and Q. Zhao, "On the Lifetime of Wireless Sensor Networks," *Communications letters*, vol. 9, no. 11, pp. 976–978, Nov 2005.

[93] V. Gutiérrez, E. Theodoridis, G. Mylonas, F. Shi, U. Adeel, L. Diez, D. Amaxilatis, J. Choque, G. Camprodom, J. McCann *et al.*, "Co-Creating the Cities of the Future," *Sensors*, vol. 16, no. 11, pp. 1971–1997, Nov. 2016.

[94] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Consensus in Ad Hoc WSNs With Noisy Links-Part II: Distributed Estimation and Smoothing of Random Signals," *Transactions on Signal Processing*, vol. 56, no. 4, pp. 1650–1666, Apr. 2008.

[95] A. V. Oppenheim and G. C. Verghese, *Signals, Systems and Inference*. Pearson, Mar. 2015.

[96] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," Internet Engineering Task Force, Tech. Rep., May 2014.

[97] G. Tsoukaneri, M. Condoluci, T. Mahmoodi, M. Dohler, and M. K. Marina, "Group Communications in Narrowband-IoT: Architecture, Procedures, and Evaluation," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1539–1549, Jun. 2018.

[98] SAMTECH Corporation, "SX1272/73 - 860 Mhz to 1020 MHz Low Power Long Range Transceiver Datasheet, Revision 4," Jan. 2019, Last Visited on February 15th, 2020. [Online]. Available: https://www.semtech.com

[99] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," in *Proc. NIPS*. Denver, CO, USA, Nov. 1995, pp. 345–352.

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

[101] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[102] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control With Deep Reinforcement Learning," *arXiv preprint arXiv:1509.02971*, Sep. 2015.

[103] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS-Workshop*. Long Beach, CA, USA, Dec. 2017, pp. 1–4.

[104] J. Honaker, G. King, and M. Blackwell, "Amelia II: A Program for Missing Data," *Journal of Statistical Software*, vol. 45, no. 7, pp. 1–47, Dec. 2011.

[105] M. Costa, T. Farrell, and L. Doyle, "On Energy Efficiency and Lifetime in Low Power Wide Area Network for The Internet of Things," in *Proc. IEEE CSCN*. Helsinki, Finland, Sep. 2017, pp. 258–263.

[106] J. Sorg, R. L. Lewis, and S. P. Singh, "Reward Design Via Online Gradient Ascent," in *Proc. NIPS*. Vancouver, Canada, Dec. 2010, pp. 2190–2198.

[107] J. A. Stankovic, "Research Directions for the Internet of Things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, Mar. 2014.

[108] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx Adaptive Compute Acceleration Platform: Versal TM Architecture," in *Proc. ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Seaside, CA, USA, Feb. 2019, pp. 84–93.

[109] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian Processes for Time-series Modelling," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 371, no. 1984, pp. 1–25, Feb. 2013.

[110] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," *arXiv preprint arXiv:1708.08296*, Aug. 2017.

[111] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, and W. Samek, "The LRP Toolbox for Artificial Neural Networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3938–3942, Jan. 2016.

[112] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Muller, "How to Explain Individual Classification Decisions," *Journal of Machine Learning Research*, vol. 11, pp. 1803–1831, Jun. 2010.

[113] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[114] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless Network Intelligence at the Edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, Nov. 2019.

[115] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series, and Products*. Academic press, May 2014.