Optimising Grid Topology Reconfiguration using Reinforcement Learning

Sai Medha Subramanian

Technische Universiteit Delft

TU Delft

TenneT

Optimising Grid Topology Reconfiguration using Reinforcement Learning

by

Sai Medha Subramanian

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 20th, 2020 at 03:00 PM.

Student number: 4852907
Project duration: December 1, 2019 – August 20, 2020
Thesis committee:

Dr.ir. J.L. (José) Rueda Torres,	IEPG, EEMCS, TU Delft, Chair
Dr. S.H. (Simon) Tindemans,	IEPG, EEMCS, TU Delft, Supervisor
Dr. J. (Jan) Viebahn,	TenneT TSO B.V., Supervisor
Dr. M.T.J. (Matthijs) Spaan,	ALG, EEMCS, TU Delft, External Member

TU Delft

Abstract

The de-carbonisation of the energy system, more commonly known as the 'Energy Transition' has a vital role to play in the pursuit of mitigating the climate emergency's impact. This transition calls for embracing new renewable energy technologies and integrating them at large scales into current power systems. Many challenges are becoming more and more evident with the increase in integration of renewable energy sources (RES) into power systems. There are difficulties involved in accurately forecasting the input of power from RES, in the increase in the use of power electronic converters and also due to a large number of prosumers. There is a global trend of moving toward making power systems more future-proof and this largely affects the roles and activities of a Transmission System Operator (TSO) such as TenneT.

The particular branch of power grid control and operation is one that is undergoing a massive overhaul. Control room operators are highly-skilled and need to be aware of all the physical processes involved in the power system in order to manually intervene when necessary. They are constantly monitoring the grid and are tasked with making quick decisions with a high frequency which rely on many factors such as the expected active power of conventional generator units, the expected demand and also the renewable energy forecasts for a given time instant. One important action that is often considered by the control room operators is reconfiguring the network topology. Making topological changes to the network offers a flexibility that is an under-exploited and low-cost alternative to maintain network security. These operational tasks are getting more complex and nuanced with the addition of newer technologies.

There is specifically an increasing reliance on Information, Communication and Technology (ICT) and newer smart grid technology which can assist in many ways such as higher levels of automation, increase in computation speed and so on. One particular field of study under the umbrella of ICT is the application of Artificial Intelligence (AI) solutions. These solutions have the potential to pave the path to better cyber-physical systems with which large strides can be made to cope with many challenges introduced by the 'Energy Transition'.

Various initiatives are being undertaken in this realm. One such initiative being Réseau de Transport d'Électricité (RTE)'s initiative of "Learning To Run a Power Network (L2PRN)" competition. This competition was conducted with the primary goal of introducing and recognising the potential of AI and machine learning-based tools in order to support control rooms and assist in making optimal decisions. One particular branch of AI that has shown great promise in the field of decision support is Reinforcement Learning.

This competition acts as a great starting point to bring together these two almost exclusive research communities. The research conducted in this thesis uses this competition as a stepping stone along with the toolchain developed for it, to investigate the use of an AI-based solution and test the behaviour of the agent, not just from a machine learning point of view, but also from a power system perspective. This thesis addresses the potential of machine learning as a decision support tool for power system control rooms by implementing a reinforcement learning algorithm to represent an artificial control room operator and assessing its performance on a particular IEEE test network. This thesis also hopes to provide some groundwork for TenneT and to contribute toward a 'Control Room of the Future' initiative which can incorporate such an AI-based decision support tool to assist grid operators in taking well-informed actions during the operation of the power systems.

Acknowledgements

I started at TU Delft as a wide-eyed student with big ambitions on synthesising two fields of interest: power systems and machine learning. I had wonderful support in addressing this goal. My first pillar of strength during this thesis, is in the form of Asst. Prof. Simon Tindemans. I am extremely glad to have met him, whose equal interest and enthusiasm in this field played a large role in bringing this work to fruition. I looked forward to our discussions every time owing to how insightful and encouraging they were. I am extremely grateful for his constant feedback and support during the course of the past 9 months.

None of it would have been possible without this opportunity from TenneT to conduct this research in the form of my Master thesis. My sincere gratitude to everyone at TenneT who made this possible, most importantly, my second pillar of strength, Dr. Jan Viebahn. Jan's infectious enthusiasm and ambition helped shape this project through and through, without which, this topic might not even have seen the light of day. The interest he placed in this work inspired me to work harder and reach farther. His constant encouragement was a large factor *behind* the successful completion of this thesis, and I am so thankful for it.

A special thank you to Antoine Marot and Dr. Benjamin Donnot from RTE in helping me see this project through by providing valuable insights and timely inputs that helped me adapt and realise the end goal.

TU Delft steered me toward my goals and many people helped me get here. Particularly, I would like to heartily thank Prof. Peter Palensky for giving me critical feedback during the course of this project, and along with Asst. Prof. Milos Cvetkovic, providing me a great chance to contribute in a small way, to another wonderful project, 'The Illuminator'. IEPG provided me a great platform to interact and share common interests with so many other researchers, particularly, Nidarshan, who gave me precise and sharp comments during this work. I would like to use this opportunity to also thank Dr. José Rueda Torres for chairing my thesis committee. I also thank Dr. Matthijs Spaan for being the external member in the committee and for taking the time out to review and assess my work.

I had great luck in the Netherlands, by stumbling upon a wonderful friend circle in Delft and Arnhem, who I am so glad to call family. A special thank you to Nived, Newman and also the rest of my 'TU Delft Survival Kit' who played such an important role in making these couple of years so memorable. I also would like to thank my Camelot family, who served as a lovely ray of sunshine, during this pandemic.

I am extremely grateful for some people who have been such a constant factor in my life, who took time out to understand this project and helped me learn more and adapt it. Neha, Deepti, Ann, Pauline, and my SNU family, I thank you all so much for always being around.

I cannot end this without acknowledging my parents' unwavering encouragement and equal excitement for my every milestone, no matter how small, which is something I am forever grateful for. And finally, my sister, who I am so proud of, who would always reach out to me, virtually, in every way possible.

Sai Medha Subramanian
Arnhem, August 2020

Contents

Abbreviations

AI Artificial Intelligence.

CE Cross-Entropy.

ICT Information, Communication and Technology.

L2PRN Learning To Run a Power Network.

MDP Markov Decision Process.

ML Machine Learning.

NN Neural Network.

PEC Power Electronic Converters.

RC Reference Configuration.

RES Renewable Energy Sources.

RL Reinforcement Learning.

RTE Réseau de Transport d'Électricité.

TSO Transmission System Operator.

1 Introduction

The impact that climate change has had around the world is seen unmistakably, especially in recent years. The phrase climate change becoming less common and 'climate emergency' gaining traction, compelled Oxford Dictionary to name it word of the year in 2019 [1]. One of the largest drivers has been the excessive dependence on burning fossil fuels and its contribution to the amount of CO_2 in the atmosphere. The de-carbonisation of the energy system, more commonly known as the 'Energy Transition' has a vital role to play in the pursuit of mitigating the climate emergency's impacts [2]. This transition calls for embracing new renewable energy technologies and integrating them at large scales into current power systems [3]. Doing so has introduced new challenges in the management of the power system in terms of its operation and planning [4]. An example of such a challenge in the use of wind turbines is that their operation will be stopped if the winds are too strong, thus this sudden drop in total capacity can become a large disturbance [5]. Additionally, photovoltaics can cause fast voltage fluctuations which affect power quality [6].

To overcome such difficulties, involving newer technologies in the sphere of 'Smart Grids' by incorporating Information, Communication and Technology (ICT) is becoming more and more ubiquitous [7]. Artificial Intelligence (AI)-based solutions have the potential to pave the path to better cyber-physical systems, which assists in making large strides toward coping with these challenges.

This thesis addresses one such AI-based solution to alleviate the burden on power systems, particularly in the branch of operation and control. The first chapter first briefly describes the motivation for this project, followed by the research questions that this thesis attempts to answer. Finally, the approach used in this thesis along with a short outline of the same is provided.

1.1. Motivation

For many decades, the function of a power grid has more or less remained unchanged; with reliable fossil-fuel based generation, uni-directional power flow and predictable power demands. Over the past decade, the "Energy Transition" has gripped the globe, particularly affecting the roles and activities of electric utility companies. This occurred due to many factors: reduced predictability caused due to larger Renewable Energy Sources (RES) penetration in the grid, bi-directional nature of power flow, a high increase in demand, an increase of prosumers, and the globalisation of energy markets [8].

The role of the control room operators is one of the many varied roles that is undergoing a massive overhaul. Figure 1.1 shows an example of a transmission grid's control room in one of the German TSOs 50Hz, in Neuenhagen, Germany. Currently, grid control rooms consist of highly-skilled employees who have knowledge of the entire grid and all its operations and a sound understanding of the physical processes in order to be able to manually intervene. But over time, due to many uncertainties, grid operation is becoming more complex and nuanced. The grid exhibits much faster and more intricate dynamics which come into play due to a large number of Power Electronic Converters (PEC) that are added to the mix [9]. There is an increasing reliance on ICT and newer smart grid technology is poised to have higher levels of automation, with some equipment having the ability to automatically open or close without operator input [10]. Making topological changes to the network offers a flexibility that is an under-exploited and low-cost alternative to maintaining network security. Unfortunately, existing software, computational methods and optimal power flow solvers available to the control room operators are insufficient and inadequate for real-time network operations in a reasonable computational time. Operators currently rely on their expertise to solve situations that arise on a daily basis. While this method of working has been sufficient, it does not fully account for the increasing RES and other factors that increase unreliability of standard grid operations [11].

Figure 1.1: The 50Hertz Transmission Control Centre in Neuenhagen near Berlin. Photo: Jan Pauls

With these recent changes, system operation is becoming a control problem that can benefit from a probabilistic system representation rather than a deterministic one. There is a need to prepare and adapt to the ever-changing circumstances that are introduced in the control and operation of the energy management system. The operation of control rooms needs to be made future-proof in order to continue the robust and reliable operation of the power grids. Various initiatives are being taken in this realm.

One such initiative is Réseau de Transport d'Électricité (RTE)'s initiative of "Learning To Run a Power Network (L2PRN)" competition. This competition was conducted with the primary goal of introducing and recognising the potential of AI and Machine Learning (ML) (particularly, Reinforcement Learning (RL)) based tools to support control rooms and assist in making optimal decisions. The first edition of this competition was held in 2019. For the first edition, a python package called 'Grid2Op' was developed to act as a framework which serves as the interface between a power system simulation tool and the developed RL-based solutions. A larger working group comprising researchers from a range of institutions has sought to build on the initial successes of L2PRN to create more sophisticated, difficult and practically relevant challenges. Researchers from the machine learning and power systems communities are encouraged to develop RL agents capable of operating a simulated power network securely and at low cost. There may be under-utilised, cost-effective flexibility in the power network that RL techniques can identify and capitalise on, that human operators and traditional solution techniques are unaware of or unaccustomed to. The goal is to try to identify potential RL agents that can act in conjunction, or in parallel with human network operators in the form of decision support tools that can optimise grid security and reliability, allowing more renewable resources to be connected while minimising the cost and maintaining supply to customers, and preventing damage to electrical equipment [11].

This competition acts as a great starting point to bring together the research communities of two otherwise almost exclusive fields. The research conducted in this thesis uses this competition as a stepping stone and the toolchain developed for it, to investigate the use of an RL-based solution and test the behaviour of the agent, not only from an ML point of view but also from a domain point of view, i.e., in terms of a power system perspective.

Thus, this thesis addresses the potential of RL as a decision support tool for power system control rooms by implementing a simple RL algorithm and assessing its performance on a particular IEEE test case.

1.1.1. The Dutch Transmission System Operator

TenneT is the Transmission System Operator (TSO) in the Netherlands with over 23,500 kilometres of high-voltage connections. The primary tasks that TenneT oversees are providing power transmission services, system services and facilitating the energy market. TenneT drives the energy transition by developing innovative instruments and establishing a key role in the energy data world. The TenneT Data Lab was set up to create new insights from data in a quick manner. Many new data-driven solutions have been tackled within the TenneT Data Lab, predictive maintenance of assets, forecasting of production of renewable energy, amongst others.

Control room operators of TenneT also foresee the use of such a tool, especially since in certain parts of the Dutch grid where there is an increasing in-feed of photo-voltaic based energy. This increase in RES is already evident in the northern part of the country. The dutch grid maintained by TenneT is seen in figure 1.2.

Figure 1.2: The Dutch network operated and maintained by TenneT [12]

This research was carried out in TenneT which contributes toward advanced analytics solutions to assist the TSO. A use-case for which this thesis acts as a preliminary step is to contribute toward a 'Control Room of the Future' which can incorporate such an AI-based decision support tool to assist in choosing actions during the operation of the network.

1.2. Research Questions

The main research objective of this thesis is:

Create a Reinforcement Learning agent that takes topology reconfiguring actions to successfully operate an IEEE test network for a duration of one week.

There are four key sub-research questions that this thesis addresses to achieve the above goal:

1. Can an IEEE test case be used to highlight the failures of using a default network configuration?
2. How can an action space, that consists of topology modifying actions, be defined while incorporating real-world constraints on the grid?
3. How can an agent be constructed that uses a simple reinforcement learning algorithm?
4. How does the agent perform for different settings?

1.3. Approach

This thesis work is carried out in the following manner:

1.4. Thesis Outline

This thesis is organised as follows: This chapter provides an introduction to the project and its intended outcomes. Chapter 2 discusses general relevant background to this thesis from both a power system point of view and a machine learning aspect. Chapter 3 discusses the framework utilised in this thesis. Following this, chapter 4 discusses the power grid and available data. It also describes the behaviour of the grid after deploying an agent that takes actions to retain the grid in its reference configuration. Chapter 5 describes the action space and the implementation of the agent using some prescribed specifications. Chapter 6 describes the performance of the agent for different test cases. It also provides a comparison between the agent developed here and another baseline agent. Finally, chapter 7 draws conclusions from this thesis and discusses the results and possible future work. Figure 1.3 describes this outline by highlighting how the chapters build on previous chapters' foundation.

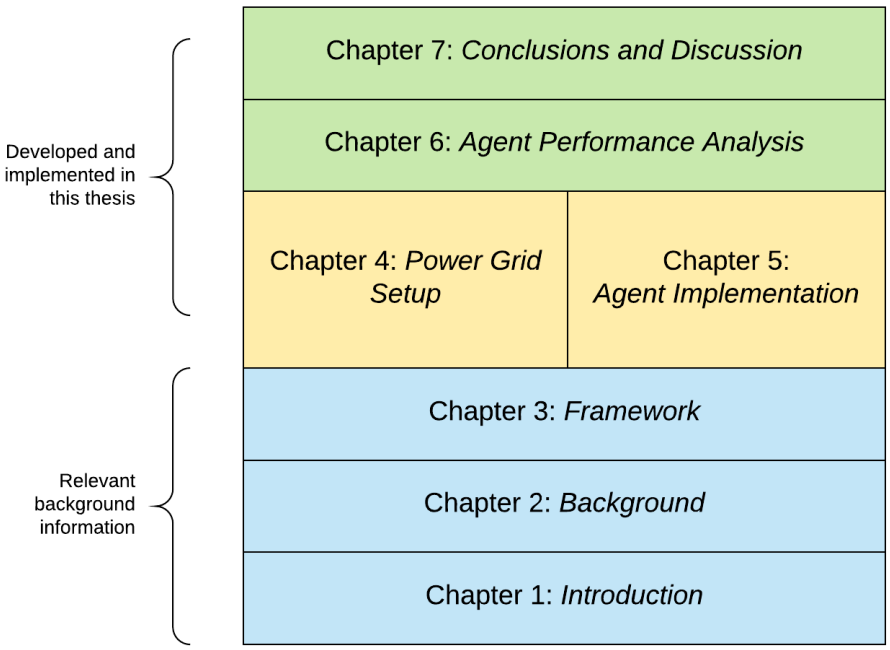


Figure 1.3: The outline of this thesis

2

Background

This chapter sets the stage for the thesis by describing some relevant background. Firstly, it covers the fundamentals of power system operation, stability and control. Following this, it briefly explains the types of switching actions and particularly the type of action relevant to this thesis. It then offers fundamentals from a Machine Learning (ML) perspective, briefly explaining the use of machine learning and then delves into Reinforcement Learning (RL) fundamentals. Lastly, it explains how network operation can be viewed as an reinforcement learning problem which lays the groundwork for the work undertaken in the remainder of this thesis.

2.1. Power System Domain

Electric power systems are considered the modern world's most intricate human-made network. They support a largely growing global power demand; currently it accounts for approximately 19% of all energy consumption and is expected to grow up to 24% by the year 2040 [13].

Power systems are also one of the most important factors for the growth of economies and improve the functioning of societies globally. During the emergence of electricity networks, though there was an initial widespread use of DC systems, soon after, they were almost completely replaced by AC systems. Their limitations became more apparent, such as how power could be delivered only over short distances, and these transmission power losses (I^2R) were quite large. To keep these losses low, voltage levels had to be made much higher. This led to the development of the transformer and subsequently, AC transmission systems. Nikola Tesla's development of multi-phase systems added to the increasing interest in widespread use of AC transmission systems. By the end of the 19th century, what is dubbed as the 'war of currents' was won over by AC power systems due to their technical superiority in terms of easily transforming between voltage levels which provides flexibility for different voltage levels. This led to all power networks across the world beginning to use AC principles.

While power systems vary largely in size and structural components, they have some consistent characteristics such as:

- they comprise of 3-phase AC systems operating at constant voltage
- they are utilised to transmit power over significant distances to reach out to wide area, which requires subsystems that operate at different voltage levels.

The basic structure and elements of a general conventional power system is visible in figure 2.1. It depicts the flow of power that is produced at the generators' end, and transmitted to consumers' end through a complex network of individual components, mainly, transmission lines, transformers, switching and protection devices. The general classification of the overall transmission network is as follows [14]:

- Transmission System: This interconnects all major generating stations (alternatively known as generators) and main load centres (or loads) in the system. It forms the backbone of the power system and operates at the highest voltage level.
- Sub-transmission System: In this stage, power is transmitted in smaller quantities from the transmission-side substations to the distribution-side substations. Larger industrial customers are directly supplied from this stage. This category is not always clearly established and is often grouped together with the transmission system.

- Distribution System: This represents the last stage of the transmitted power flow to the individual customers. The voltage level is on the lowest end of the range here.

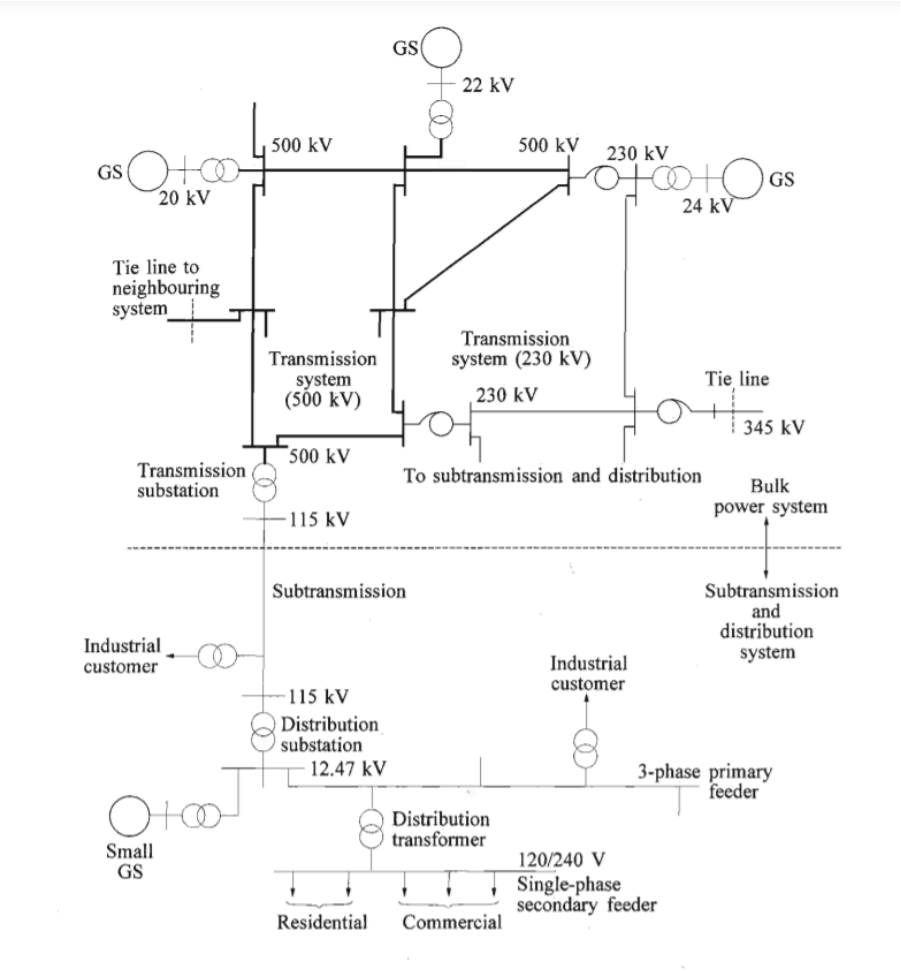
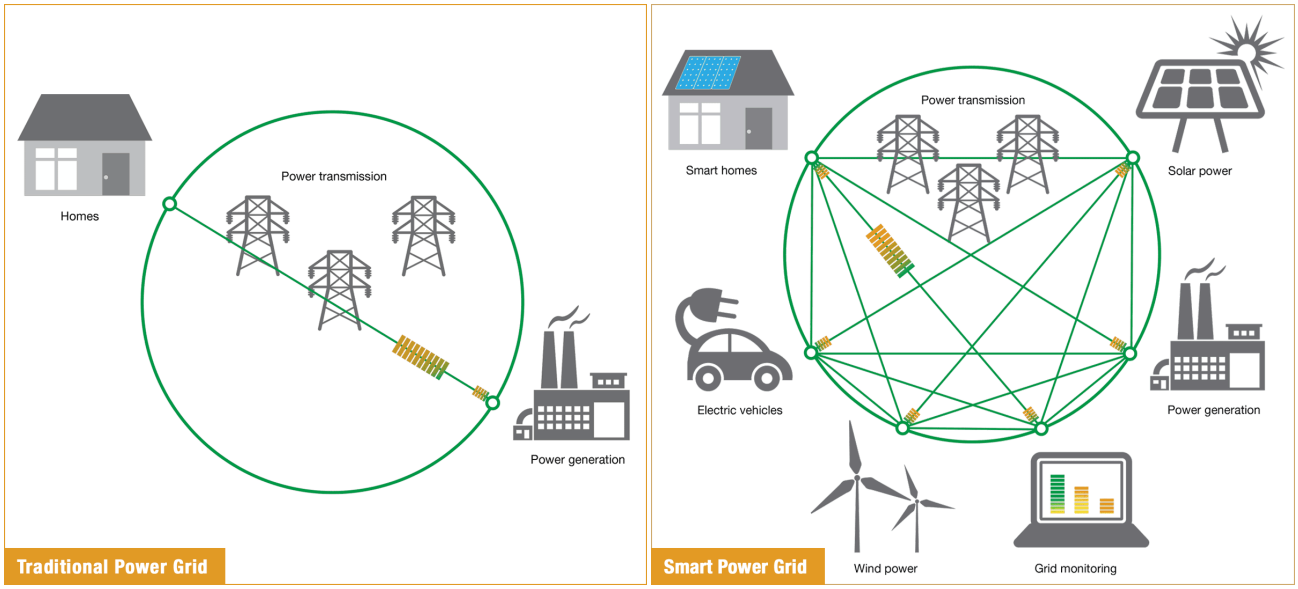


Figure 2.1: The basic elements of a conventional power system [14].

In the Netherlands, this classification is similar and straightforward. TenneT TSO B.V. is the sole TSO of The Netherlands. They are the only stakeholder responsible for managing the high-voltage grid (between a voltage of 110 kV to 380 kV) [15]. The distribution side is governed by a set of Distribution System Operators (DSOs) such as Liander N.V., Enexis B.V., Stedin Netbeheer B.V. and Westland Infra Netbeheer B.V. and more.

The overall system thus consists of multiple sources and several subsystems, with layers of transmission networks. This provides a high degree of necessary structural redundancy that assists the system in withstanding unusual contingencies without service disruption.

Until recent times, electrical power grids were dominated by conventional power plants that are fossil fuel based to cater to growing demand with centralised and uni-directional power flow (figure 2.2a). But, this is linked with a high emission of carbon by-products which exacerbates global warming effects. This has led to a world-wide trend in decommissioning of these conventional power plants. Modern power networks involve more RES penetration in the transmission side as generation sources, and also in the distribution side as typical small-scale RES power generation, called prosumers, such as seen in figure 2.2b [16]. These smart grids envision a more decentralised approach to allow for bi-directional power-flow and increasing demand, with newer technologies for monitoring and control of grids. Involvement of these aspects only adds a layer of unreliability and lack of robustness which makes the control and operation of such networks more difficult. Consistently supplying power safely and reliably is the primary responsibility of a TSO. In this realm, the most important tasks are to maintain the grid efficiently and also ensure power system stability.



(a) Conventional power grid

(b) Modern smart grid

Figure 2.2: Comparison between conventional and smart grids (Photos: Gatech.edu)

2.1.1. Power System Control

In order to ensure power system security, TSOs in general are required to meet some standards, such as:

1. **Constancy of frequency:** System frequency of the grid is a metric used to check and maintain the balance in between generation and load. In most parts of the world, the nominal system frequency is maintained at 50.0 Hz. This relation depicting the relation of frequency to the net power in the system can be approximated by equation 2.1, as:

$$P_g - P_d \propto \Delta f_s \tag{2.1}$$

Where:
P_d : Power Demand
P_g : Power generated
f_s : System frequency

This relation depicts that if there is a net power surplus, this causes an increase in frequency, and a shortage causes a decrease in the frequency. In the synchronous grid region of Continental Europe, a frequency deviation of upto ± 10 mHz is allowed before control mechanisms are activated. In Netherlands particularly, the normal frequency variation interval is 49.9 - 50.1 Hz and in critical or emergency situations the allowed variation is 47.5 - 51.0 Hz [17]. In the case that this delicate balance is disrupted, the first method of control is inertial system control followed by different levels of frequency control.

2. **Constancy of voltage:** Keeping voltage constant is a key factor in order to ensure the substation nodes' voltages are maintained at (near) nominal voltages of the transmission equipment. Not ensuring this could result in large-scale system blackouts. Voltage stability is controlled by the reactive power available in the system. This can be an input from generating sources, or power converter based generation, and rotating loads that assist by absorbing reactive power.

3. **Physical network constraints:** The existing impedance of the network is an important aspect that defines the constraints of the network. In accordance with Kirchhoff's and Ohm's laws, current flowing through parallel branches of the network will be inversely proportional to the impedance of that branch. While individual lines' impedance are generally considered to be constant and cannot be altered during operation; the overall path's impedance depends on the topology of the network at that given time. Thus, by varying the topology, the impedance of the path can be varied, thus, providing alternate paths for the flow of current. The impedance of the branches are defined by the physical parameters of the lines, such as the material, the area and length of the lines. Hence, together, these factors contribute toward defining upper limits to the amount of power that can flow through the lines; if breached, protection can step in and disconnect the equipment involved to prevent damage.

To analyse power system security and design appropriate control systems, it is useful to classify the system operating conditions. This can be mainly classified into 5 states: normal, alert, emergency, in extremis, and restorative [14]. The ways that transition can take place from one state to another can be seen in figure 2.3.

- In the *normal* state, all system equipment is being operated within a normal range. The system is secure and is able to withstand a contingency without any constraints being violated.
- The *alert* state is when the security level falls below a certain threshold of adequacy or if the probability of a disturbance increases due to unfavourable conditions (e.g.: worsening weather conditions such as approaching storms). Here, all system variables are still within operating limits; but the system's robustness has been reduced such that if a contingency occurs it could lead to overloading of equipment, which can place the system in an emergency state. It is also possible for a large disturbance to result into an extreme (in extremis) state directly from here. At this point, preventive actions such as security dispatch or use of reserve can be taken to restore the system back into a normal state.
- If a sufficiently large disturbance occurs, the system enters the *emergency* state. In this state, some system variables could exceed short-term ratings. Emergency control actions can help restore the system back to the alert state.
- If the above actions are ineffective (or not taken) the system enters the *in-extremis* state. The result here could be cascading outages and a possible shut-down of a major part of the system. Further control actions can be taken with the aim of saving the system from a widespread blackout.
- If effective control actions are being taken to reconnect all facilities and to restore system load, this is then the *restorative state*.

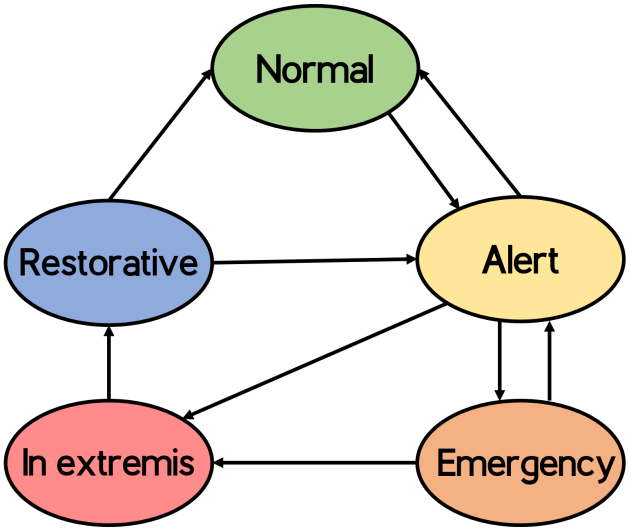
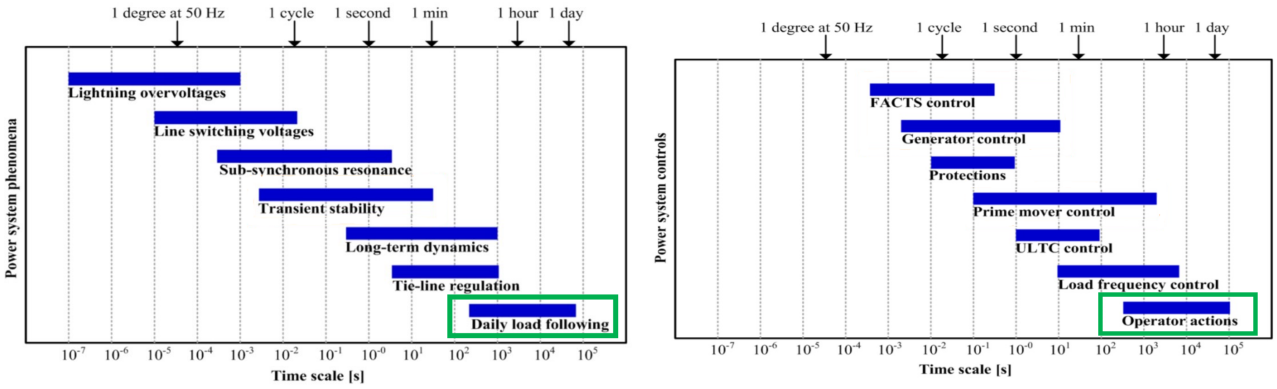


Figure 2.3: Power system operating states [14].

In this thesis the focus point is ensure that the power system remains in the normal state. The behaviour of the system in the other states are not examined here. If the system faces a contingency, it immediately goes into the emergency state and fails as there are no corrective or control mechanisms that are being studied or analysed here. Further, it helps to understand the timescale of operation and phenomena that occur. This can be visualised in figure 2.4a. The general time-scale that the changes in load profiles occur are in the range of hours. The highlighted portions of 2.4b indicate the time-scale of operation that control room operators work in.



(a) Time scale of various phenomena

(b) Time scale of control mechanisms

Figure 2.4: Time scale of occurrence of phenomena and respective control mechanisms (Adapted from [18])

2.1.2. Network Operation

Control room dispatchers(or operators) take decisions and perform actions during the time of operation of the grid. As described above, there are three main goals for power system operation and control: maintaining constant voltage and frequency, and ensuring that physical constraints of the network are not violated. In this thesis, the focus is on ensuring the power system does not breach its constraints (more particularly, the constraints imposed due to the thermal properties of the power lines) and that it can continue its steady state operation. The control of network to maintain voltage and frequency stability are not considered.

As described above in section 2.1.1, stochastic failures of lines, transformers and power plants can cause changes from normal to the vulnerable or emergency state of the network; this can be worse especially when the network is already weakened by scheduled outages and the overall security of the grid is compromised. Many such events that occur simultaneously can lead to large-scale blackouts. The most important task in correcting the state of the network is to return it from an emergency state to an alert state (figure 2.3). A further task is the restoration of the network, i.e. the return to normal after supply interruption [19]. Many actions can be undertaken to operate power system and to improve its state. Some actions are to vary the topology of the network by network switching or another is to change the power output of conventional generators (power re-dispatch) or also to vary the reactive power in-feed of generators. Network switching is an economical corrective control mechanism for power flows in order to relieve overloads in the system [20]. The application of corrective switching actions to improve the state of the network enlarges the scope of optimal load-flow by a further dimension. Network topology changing is a rather under-exploited and low-cost task considering it's binary nature of the control variables [19] [11].

The objective of corrective switching is usually overload reduction, but other objectives can also be achieved such as, improving the voltages at different nodes or reducing the short circuit power.

These actions can be conducted over a range of hours to minutes, depending on the operating state of the grid at a given point of time. Some of the switching actions that can be taken are [19]:

- Switching-on of open connections: Like lines, transformers and bus-bar couplings.
- switching-off of branches: This can also reduce a particular overload, even though it could weaken the overall network.

These possibilities are independent of the structure of the substations. A substation can be considered as a single node. But when the action of bus bar switching is wanted, this is no longer the case. A substation consists of one or more bus bars. Most common high-voltage power systems have arrangements which have the capability to split the substation to two separated bus-bars, which is known as bus splitting [21]. All incoming and outgoing branches, generators and loads are connected to one of the bus bars of a substation with separate circuit breakers in order to disconnect and reconnect the elements, as seen in figure 2.5. In this figure, a binary variable δ is introduced for each breaker, to represent the state of that circuit breaker: a closed breaker is represented by a $\delta = 1$ and an open breaker by $\delta = 0$ [22]. Using this capability of bus splitting, the system topology can be modified such that any congestion can be partially or also fully removed.

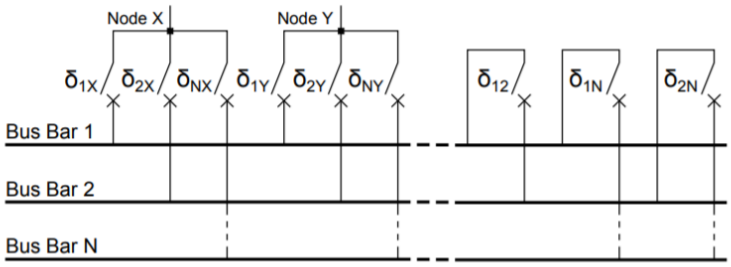


Figure 2.5: Switch gear arrangement in a substation with N bus bars [22]

The use of split bus-bars allows considerably more possibilities to be realised. By using split bus-bars, it is possible to achieve flow changes, first, by radial operation with the separation of injections and/or loads and, second, by enlarging the impedance caused by the opening of loops in a meshed network [19]. In figure 2.6, the split bus-bars for a single substation (here referred to as B5) in the IEEE 14 bus test case can be seen.

Bus-bar switching, as seen in this thesis, entails that an element (a generator, load or a line) is disconnected from one bus-bar and reconnected to another bus-bar of the same substation [22]. A possible issue of such actions is that a switching operation for reducing the overloading of a particular sub-network of the grid, could lead to overloading in other parts of the network [22]. Thus, multiple such actions may be required to provide relief to the overloaded areas of the network. In this thesis, switching between the bus-bars, post splitting of bus-bars for all substations in the given power system, is the main type of network topology related action that is considered. Figures 2.7a and 2.7b respectively show the particular choice of elements on each of the bus-bars, for before and after the split takes place [23].

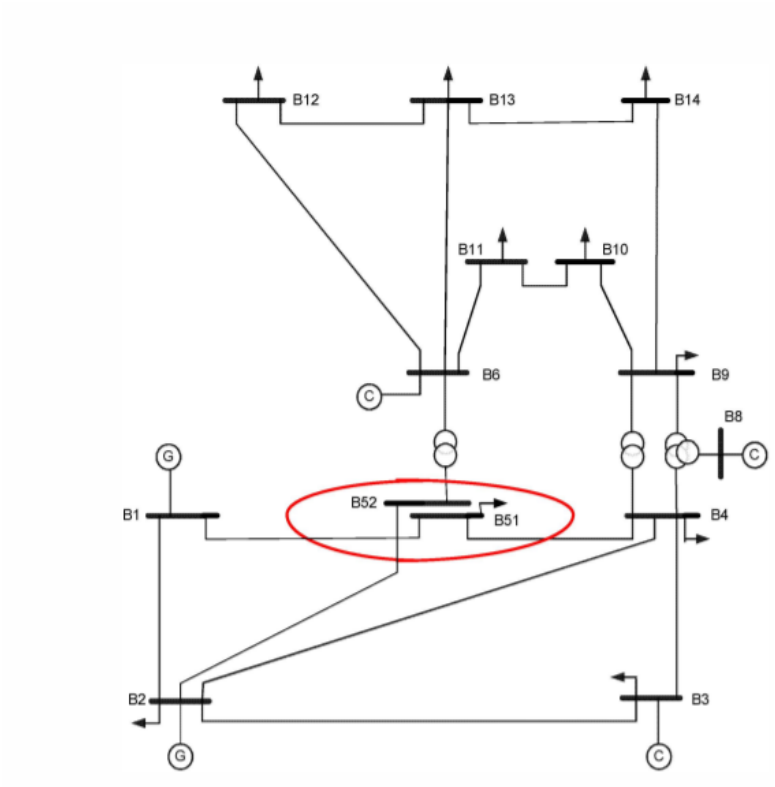
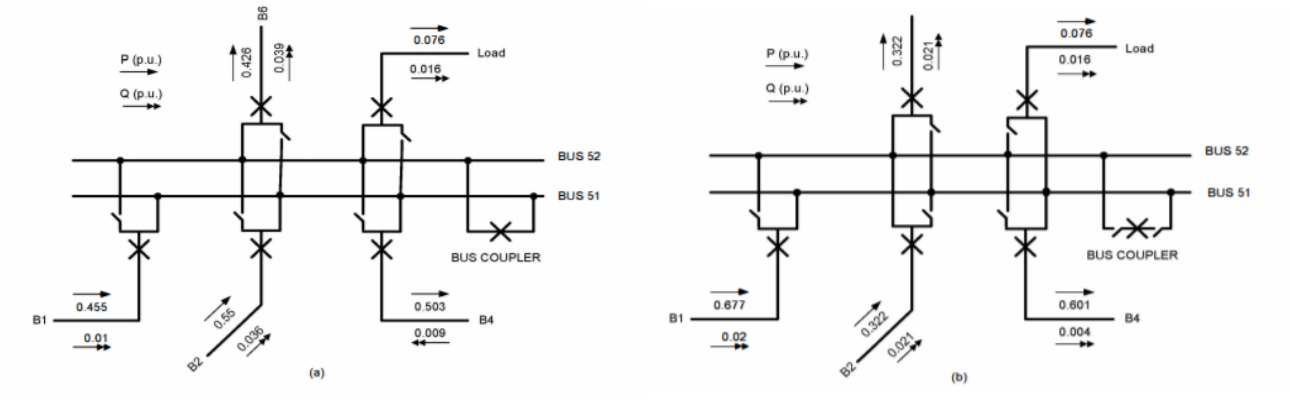


Figure 2.6: Split bus-bars of substation 5 [23]



(a) Before splitting action

(b) After splitting action

Figure 2.7: Substation configuration before splitting and after splitting [23]

In previous literature, corrective switching as a means to help mitigate congestion date back to the 1980s (in [20], [19] and [24]). Optimal bus-bar switching is also explored as different optimisation problems like in [22], [21], [23] and [25] using linear, mixed-integer linear programming, Benders decomposition for optimal power flow etc. Further advanced control using synchronised phasor measurement technology was explored in [26]. The exploration of this type of bus switching actions (also called substation reconfiguration in related literature) is mainly attributed to the fact that it is easier to implement when compared to other congestion mitigating tasks such as as generation re-dispatch and load shedding.

2.2. Machine Learning Applications

Over the past decades, the concept of machine learning has expanded so much into one of the mainstays of information technology and hence, a central (albeit usually hidden) part of daily lives. With the ever increasing amounts of data becoming available, smart data analysis has become ubiquitous as a necessary ingredient for technological progress [27]. A machine learning algorithm is an algorithm that is able to learn from data. In [28], a definition for learning is: " A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." The experiences E, tasks T and performance measures P can be substituted by a wide variety of entities. Based on the experience E they are allowed to have, machine learning algorithms can be broadly categorised into three categories, as shown below [29].

1. Supervised learning algorithms: Here, the algorithms experience a dataset containing features and each example of this dataset is associated with a label/target. For example, the famous Iris dataset [30] is supplemented with the species of each iris plant which serves as the labels of this dataset. There are a total of three species as per this dataset. A supervised learning algorithm can study this dataset and learn to classify plants into one out of the three species. Supervised learning algorithms can be largely categorised based on their application as either classification or regression problems as seen in figure 2.8.

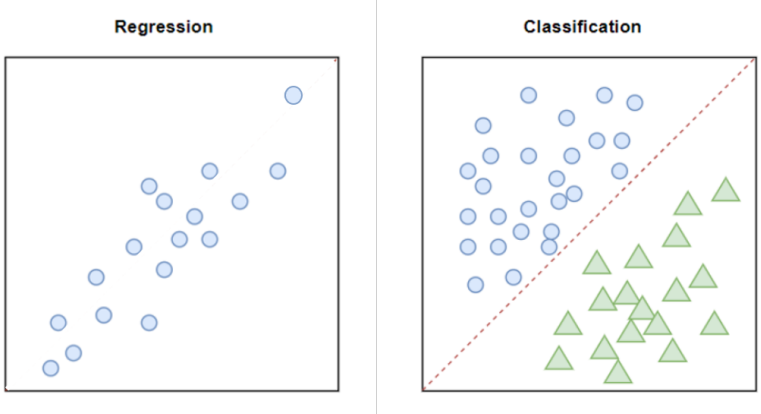


Figure 2.8: Supervised learning problems

2. Unsupervised learning algorithms: The algorithms experience a dataset containing a large number of features, from which useful properties about the structure of the dataset are learnt. A use of such an algorithm can be seen for the application of clustering, which is to divide a dataset into clusters of similar examples. This can be visualised as in figure 2.9.

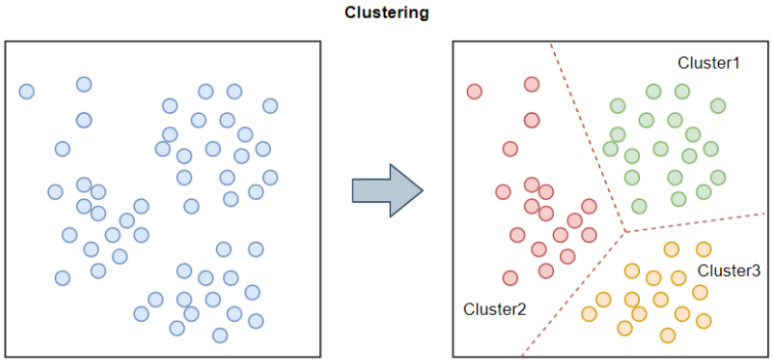


Figure 2.9: Unsupervised learning problem

3. Reinforcement Learning: These algorithms are utilised when the learning system can create datasets by actively interacting with an environment, so there is a feedback loop between the learning system and its experiences. This is further elaborated in 2.2.3.

With just these few machine learning categories, a large number of applications in diverse domains can be covered. This can range from weather forecasting to image classification, Big data visualisation to robot navigation. In figure 2.10, a more elaborate overview of the applications of machine learning can be seen.

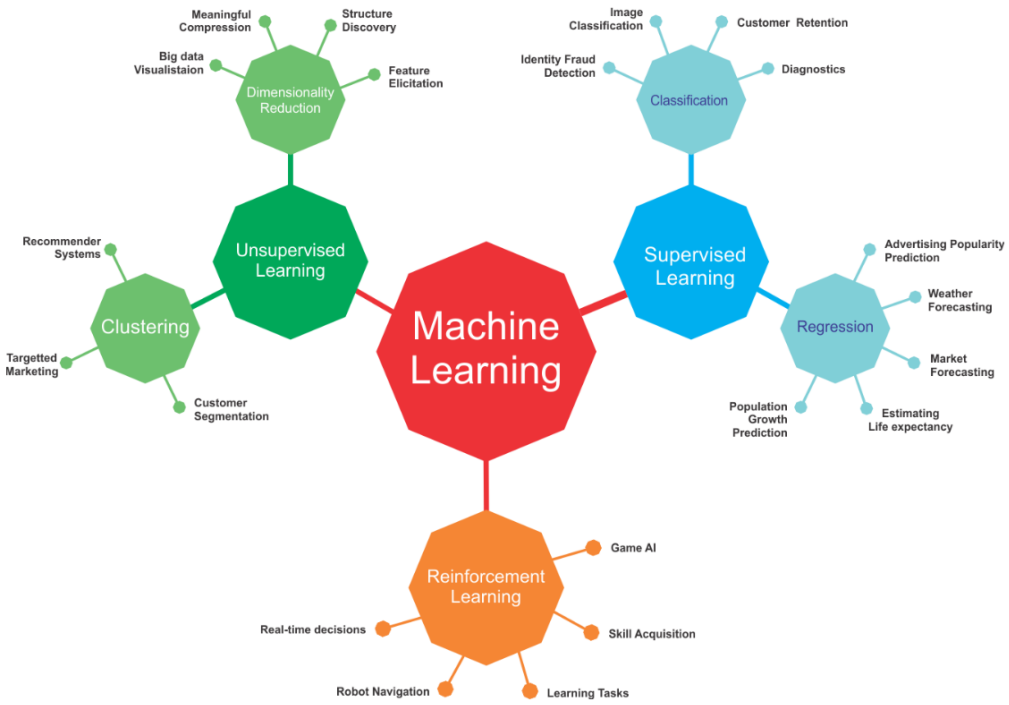


Figure 2.10: Machine Learning Applications [31]

2.2.1. Practical Approach

In order for the algorithm to create an effective model that can learn from the data it experiences or is fed, there is a series of steps involved. The first step is to appropriately define the problem, understand the main objectives expected from the model based on the inputs. After obtaining the required data for the model to

learn from, the overall data-set needs to be divided into two, namely: a “training set” and a “testing set”. A training data-set is what is used while the model is learning, and a testing set is used to verify the model after it has learnt/ been trained to evaluate the quality of learning.

Following this, an appropriate measure of performance needs to be chosen. This measure helps define the basis on which the model learns [32]. The model is initially fed a “training dataset” from which the model tunes its parameters in order to fit the “training dataset” appropriately. The method of tuning the model uses optimisation techniques such as gradient descent / stochastic gradient descent. The parameters (interchangeably used with the term *weights*) of the model of the model are those that when tweaked in an iterative manner, produce a model that is most closely giving a solution to the training data-set.

In this manner, the model is trained until some satisfactory criteria is reached. In order to verify the working of this model, the model is then fed the testing data-set using which the performance of the model can be realised [32]. This can also be visualised with respect to a particular model, a Neural Network (NN), in figure 2.11.

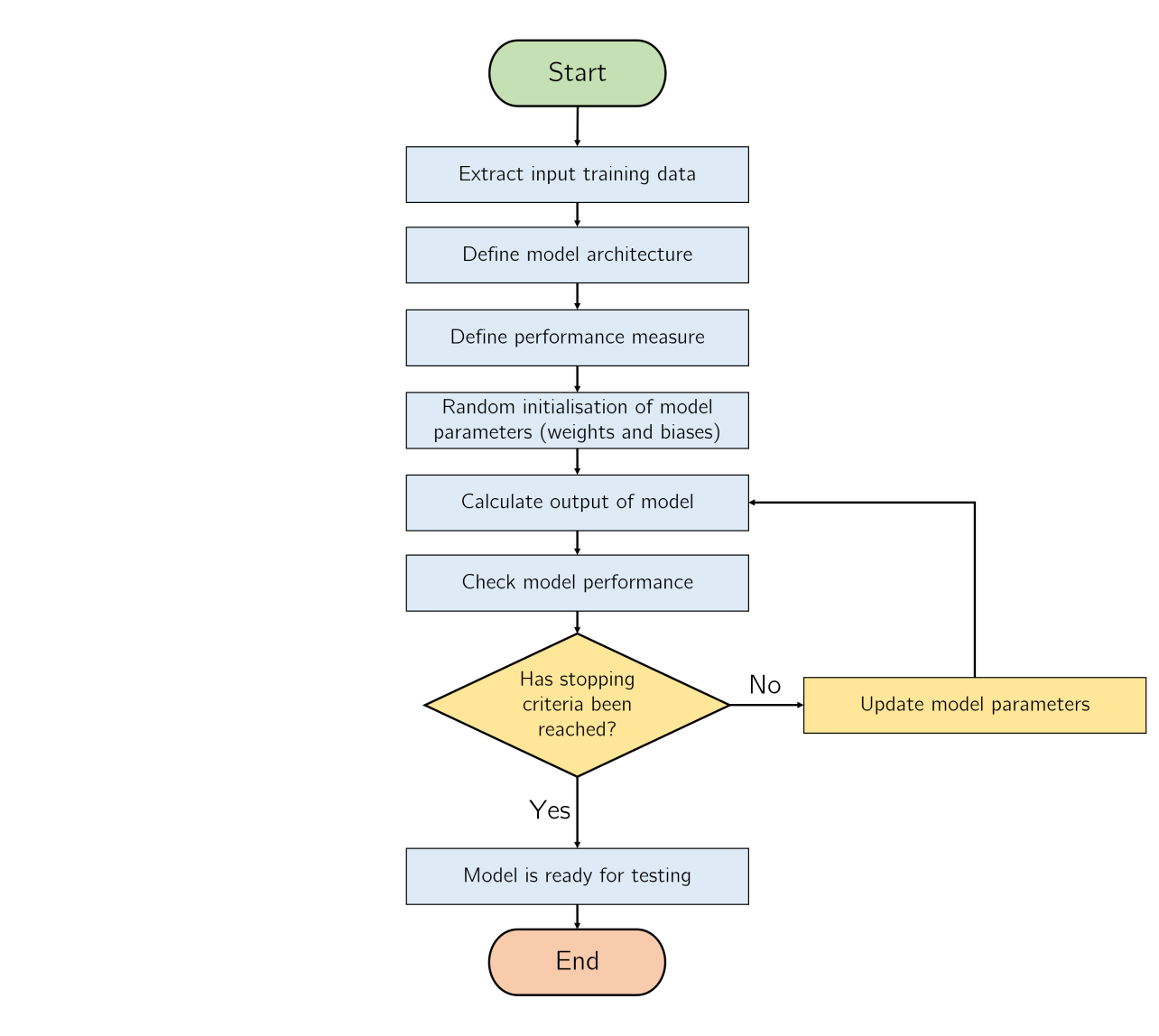


Figure 2.11: A generic flowchart for training a NN

2.2.2. Neural Networks and their architecture

The definition of a neural network, more properly referred to as an ‘artificial’ neural network (ANN) is: “A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs” [33]. The inspiration for an ANN to be used as a computational model, is the biological neural networks and the way they process information in the human brain.

The basic computational unit of a neural network is the neuron, also called a node, or unit, as seen in figure 2.12. A very simple example of such a node is one which takes several binary inputs x1,x2,…, and produces a single binary output. Each input is assigned a weight to indicate the importance of this particular input. The output of this neuron (0 or 1) depends on whether the weighted sum: $\sum_j w_j x_j$ is less than or greater than some threshold value [34]. Algebraically, this is seen by:

$$If \sum_j w_j x_j \geq threshold \Rightarrow y = 1$$

$$If \sum_j w_j x_j < threshold \Rightarrow y = 0 \tag{2.2}$$

Where y is the output of the neuron.

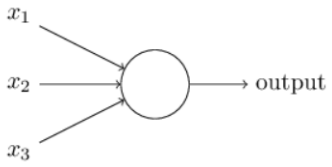


Figure 2.12: A single node in a neural network

A stack of such nodes make up a layer. A NN can have a range of layers. Each node in a layer receives inputs (either from other connecting nodes or an external source) and computes an output. Each input has an associated weight which is assigned on the basis of its relative importance to other inputs. The node then applies a function to the weighted sum of its inputs. Finally, an activation function is used to map the output of the nodes to another final outcome of each node, to the outcome of the neural network model. The output of the weighted sum (as in equation 2.2) is mapped to the final outcome using an activation function. An example of the activation function is a linear function, where the output of the activation is directly proportional to the output of the neuron by a constant factor c (as seen below).

$$f(y) = c \times y \tag{2.3}$$

Where y is the immediate output of the neuron. $f(y)$ is now the final output of the model.

The general architecture of a NN, as seen in figure 2.13, consists of:

- Input layer: the first layer of nodes which serves as the interface between external information and further layers in the NN by passing on information. The size (number of nodes) in this layer is usually equal to the number or input observations (or features).
- Hidden layer: In these layers, the intermediate processing and computation is done, after which, the weights are transferred to the following layer. The size of the hidden layers can be varied to vary the performance of the NN.
- Output layer: This is where an activation function is used in order to map the output of the nodes into a desired output format. The size of the output layer corresponds to the number of possible outputs.

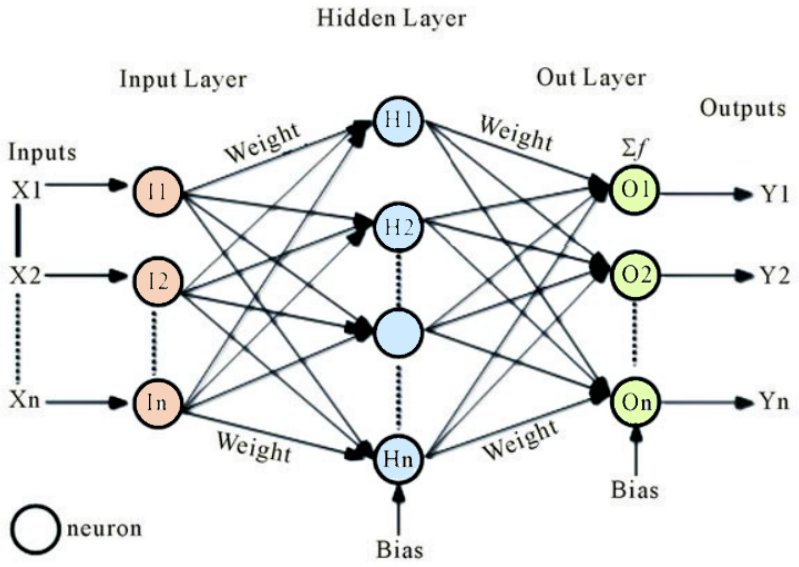


Figure 2.13: Architecture of a simple ANN

2.2.3. Reinforcement Learning Fundamentals

Reinforcement learning is one of three branches of machine learning paradigms besides supervised and unsupervised Learning. Reinforcement learning problems are concerned with learning what actions can be taken in situations in order to maximise a numerical reward signal. They are essentially closed-loop problems because of how the system's actions influence its later inputs. With increasing complexity in cases, actions may affect not only the immediate reward but also the next situation, and in turn, all subsequent rewards. The reinforcement learning system discovers what actions to take by trying them out, and hence there is no prior knowledge of how the actions influence the reward [35]. The area of reinforcement learning deals with how *agents* ought to take *actions* in order to maximise a *reward* which is done by taking the current reward and *state* provided by an *environment*. Figure 2.14 depicts this interaction.

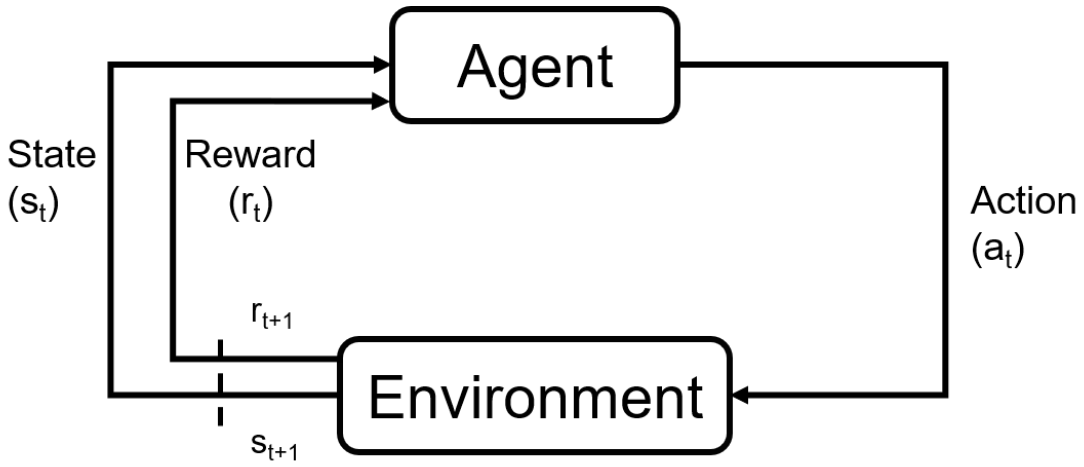


Figure 2.14: The interaction between an agent and its environment in RL [35]

Reinforcement learning differs from supervised learning problems since it doesn't rely on pre-defined and existing data with labelled input and output pairs for training. In the reinforcement learning domain, the learner has the capability to create its own data for training. Though reinforcement learning tends to be perceived as a type of unsupervised learning since it also doesn't rely on examples of outputs, unsupervised learning involves trying to find a hidden structure in the data. While, this can be useful in reinforcement learning problems as well, the goal here is to maximise a particular reward signal. Hence, reinforcement learning is a third and different branch alongside supervised and unsupervised learning under the umbrella of machine learning.

2.2.4. Reinforcement Learning Problem Definition

RL problems are closed-loop problems where the learning system's actions influence its later inputs. The agent (learner) is not fed with the "correct" actions that it needs to take; instead, it must discover which are the best actions it can take. This is done by learning which actions yield the highest reward, by merely trying out the variety of actions [35]. In order to understand the various aspects involved in setting up a reinforcement learning problem, the following definitions are presented [35]:

1. **Agent:** This is the decision or "action" taker that takes particular actions in order to maximise some reward.
2. **Environment:** This is the platform upon which the agent can take said actions. An environment can often be defined as an Markov Decision Process (MDP) which is defined in more detail in section 2.2.5.
3. **Action:** This is the output of the agent which is chosen from the available variety of actions (called the action space).
4. **State (or observation):** This is obtained from the environment and it describes the current situation of the system.
5. **Policy:** This defines the learning agent's behaviour for a given state. It is at the core of a reinforcement learning agent as, by itself, it is a sufficient way of determining the agent's output.
6. **Reward Signal:** It is a factor which contributes to the agent's learning. This is a numerical value provided by the environment to the agent at every time-step. This reward is commonly also indicative of the immediate problem and situation faced by the agent, that it will attempt to better.

2.2.5. Markov Decision Process

In addition to the above required aspects to set up a reinforcement learning problem, an important mathematical framework is the MDP. An MDP is used for modelling tasks that have the *Markov Property*; which states that a memory-less system in which the current state is sufficient to summarise all past experiences in a compact way so that redundant information to make decisions is not retained.

An MDP tries to capture the world (environment) that it is in, in the following way [35]:

1. **State/Observation**: (s) or $(s(t))$ describing the current situation of system.
2. **Model**: $T(s,a,s')$ which produces ($P(s'|s,a)$ which is the probability that the environment can transition to a state s' given that the the action a was taken from the state s. This is also the aspect that most encapsulates the *Markovian Property* as the probability of reaching state s' depends only on the current state s.
3. **Action**: $A(s)$, which is a set of all the possible decisions that can be taken by the agent.
4. **Reward**: $R(s)$ or $R(s,a)$ or $R(s,a,s')$. This aspect generally encompasses the domain knowledge of the particular world/environment.

These above four elements together form the MDP in most scenarios; which presents a problem. The question then follows, what is the solution of an MDP? The solution of an MDP is defined as the **Policy**. A policy can be deterministic or probabilistic. A deterministic policy takes in a state and returns an action that should (or will) be taken given this input state. A probabilistic policy uses the state and provides a probability distribution corresponding to all the possible actions. Thus a policy can be depicted with π and $\pi(s)$ returns either some action (a) or a probability distribution of actions.

There can be a certain *optimal* policy (π^*) that could produce the optimal long-term reward. A MDP is realised as a chain (or trajectory) of system state transitions and every transition is related to a tuple ($s(t)$, $A(s)$, $R(s)$). Consequently, a realisation of a MDP is related to a sequence of such tuples, which in RL, is also called an *episode*. In this study we focus on finite horizon environments such that the number of possible time steps for an episode is limited, say by t_{max}.Hence, the information related to an episode can be represented as a list of tuples

$$X = [(s_0, a_0, r_0), (s_1, a_1, r_1), \dots, (s_{t_{end}}, a_{t_{end}}, r_{t_{end}})] \,, \tag{2.4}$$

with $t_{end} \leq t_{max}$.

2.2.6. Types of Reinforcement Learning Algorithms

In figure 2.15, a (non-exhaustive) taxonomy of reinforcement learning algorithms can be found. This tree structure to describe reinforcement learning algorithms, while simplified, helps highlight large differences and similarities between algorithms.

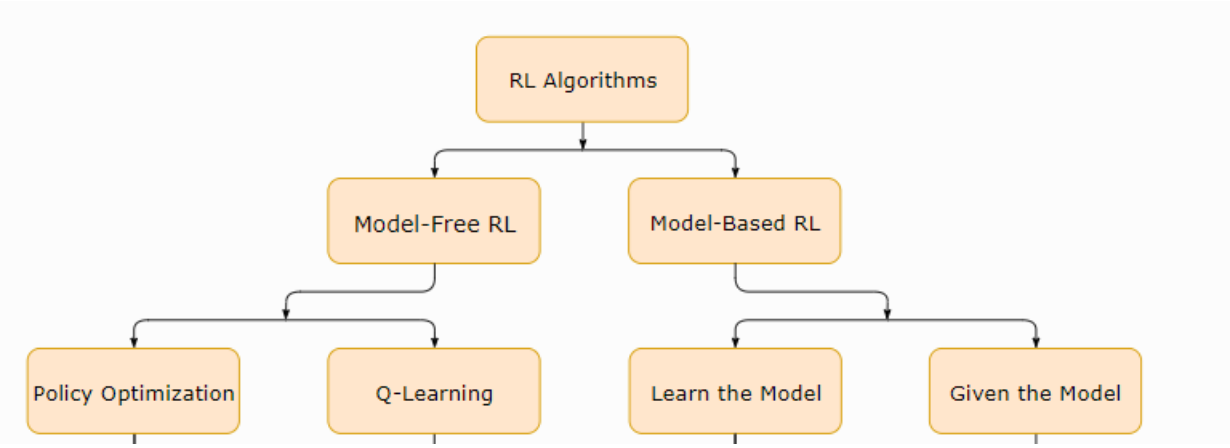


Figure 2.15: A Taxonomy of reinforcement learning algorithms [36]

The first branching point that arises while identifying reinforcement learning algorithms is whether the agent has access to a model of the environment that it can learn from. In many scenarios, ground-truth models are not available to agents. Currently, model-free methods are explored and developed more and tend to be easier to implement and tune. Next, the question arises on what the agent is required to learn. In model-free methods, the agent might need to learn particular policies or else value/Q-functions. In model-based methods, the learning goal would be to identify the structure of the model itself.

In this thesis, the focus will be on utilising model-free methods, a policy optimisation method called the cross-entropy method for reinforcement learning.

2.2.7. The Cross-Entropy Method for Reinforcement Learning

The Cross-Entropy (CE) method is a generic Monte Carlo technique. Such a Monte Carlo technique is one when the sampling relies on repeated random sampling to obtain numerical results. The underlying concept here is to use randomness to solve problems that might be deterministic in nature. This method was first proposed in [37] to act as an adaptive importance sampling procedure for the estimation of rare-event probabilities, that uses the cross-entropy or Kullback–Leibler divergence as a measure of closeness between two sampling distributions. It was then extended to solve both combinatorial and continuous problems. In simple terms, the CE method uses the idea that the probability of locating an optimal or near optimal solution using naive random search is a rare-event probability. The CE method can be used to gradually change the sampling distribution of the random search so that the rare-event is more likely to occur [38]. This method is also largely derived from the concept of cross entropy in the field of information theory where it is defined as the difference between two probability distributions or a given random variable or set of events. The output using CE method can be approximated by repeating two steps:

1. **Generate** a random sample from a search space X according to a specified probability distribution.
2. **Update** the parameters of the distribution based on the N^* best performing samples (elite samples) using cross-entropy minimisation.

From a CE point of view, the total reward criterion can be considered as a combinatorial optimization problem with the performance function

$$R(X) = \sum_{t=0}^{t_{end}} r_t \tag{2.5}$$

with X given by Eq. (2.4). In other words, R represents the cost of a trajectory/episode.

There can be many different methods of implementing the use of cross entropy for reinforcement learning. The method utilised here is derived from [39], and the core of this cross-entropy method is to throw away bad episodes and train on better ones. So, the steps of the method are as follows [39]:

1. Play N number of episodes using the current policy and environment.
2. Calculate the total reward for every episode and decide on a reward boundary. Usually, we use some percentile of all rewards, such as 50th or 70th.
3. Throw away all episodes with a reward below the boundary.
4. Train on the remaining "elite" episodes using observations as the input and issued actions on the desired output.
5. Repeat from step 1 until we become satisfied with the result.

The NN learns how to repeat actions, which leads to a larger reward, constantly moving the boundary higher and higher.

2.3. Use of Reinforcement Learning for Network Operation

Some of the tasks involved in power system operation and control are highly complex. Practical power system engineering require logic reasoning, heuristic search, perception, and/or the ability to handle uncertainties [40] [41]. AI-based solutions or intelligent systems (IS) are potentially powerful tools to support new solutions where other more traditional technologies cannot. Intelligent systems are natural decision support tools for planning, operation and maintenance of power systems [42]. Particularly, with respect to power system monitoring and control handled by control room operators, due to limited computing buget constraints, current optimal control methods are inadequate for real-time network operations on short temporal horizons in a reasonable computational time [8]. Especially, considering the largely changing landscape of generation and load patterns, newer computational methods are required [11]. Reinforcement learning has shown great promise in the realm of decision making tools and developing control algorithms. Deep reinforcement learning has made great strides particularly after breakthroughs with agents such as AlphaGo at Go [43]. Other recent work with deep learning and potential of reinforcement learning to power system related problems has been explored in [44], but this does not address continuous power system operation. Some initiatives are underway to bridge this gap, one such initiative is the L2PRN challenge.

2.3.1. L2PRN

In order to test the potential of reinforcement learning as a decision making tool in control rooms, representatives from RTE established a competition to test the potential of reinforcement learning in this area of controlling electric power transmission. To foster and promote advances within the two communities: the power system and reinforcement learning communities, a new platform was open-sourced to build and run simulated power system environments to create new baselines for controllers to perform continuous near real-time operations [8]. In 2019, the first edition of this competition was held, where a first IEEE14 bus environment test case was released, with an emphasis on using the topological flexibility of power systems. This challenge derives and takes inspiration from the Learning To Run competition [45], where the goal was to develop a controller for a human body to train it to walk and run. The goal of this competition was to operate the simulated power system continuously over several days at a 5 minute time-resolution. The aim was to manage the power flow at every time-step t, given injections, i.e. production and loads, and present grid topology denoted by τ.

The second edition of this competition is being held in the summer of 2020. There are many organisations involved in the development of this challenge in 2020 such as: RTE, INRIA, Turing Institute, EPRI, Google Brain, GEIRINA, TenneT, Pandapower and others. In [11], an overview of the contributors along with their representatives can be seen. This second edition also incorporates many other actions such as re-dispatch operation, maintenance scheduling, cyber-attacks and more complex situations for the agent to learn. This thesis, as previously discussed, focuses only on topology related actions, particularly the bus-bar switching actions.

2.3.2. Topology Reconfiguration as a Reinforcement Learning Problem

To explain the formulation of a reinforcement learning problem in the realm of network operation, it is important to understand which aspects represent which power system elements, in comparison to 2.14. This can be visualised in figure 2.16. The platform where the power system is simulated and the load flow can be computed, which is developed by RTE, serves as the environment. The action in our reinforcement learning problem is an action or a decision taken to change the topology of the grid. These actions are chosen by the agent, which in this context of this thesis, is an artificial control room operator. This agent takes decisions based on the state, or observations of the grid at every time-step that is, the active power, reactive power, voltage etc. at

every substation. Finally, since the agent aims to optimise something, this is represented by the reward. In this approach, the reward is represented by a function to maximise the overall residual capacity of the grid. In this thesis, the formulation presented here is the basis of developing and implementing the agent as discussed and elaborated on in the later chapters.

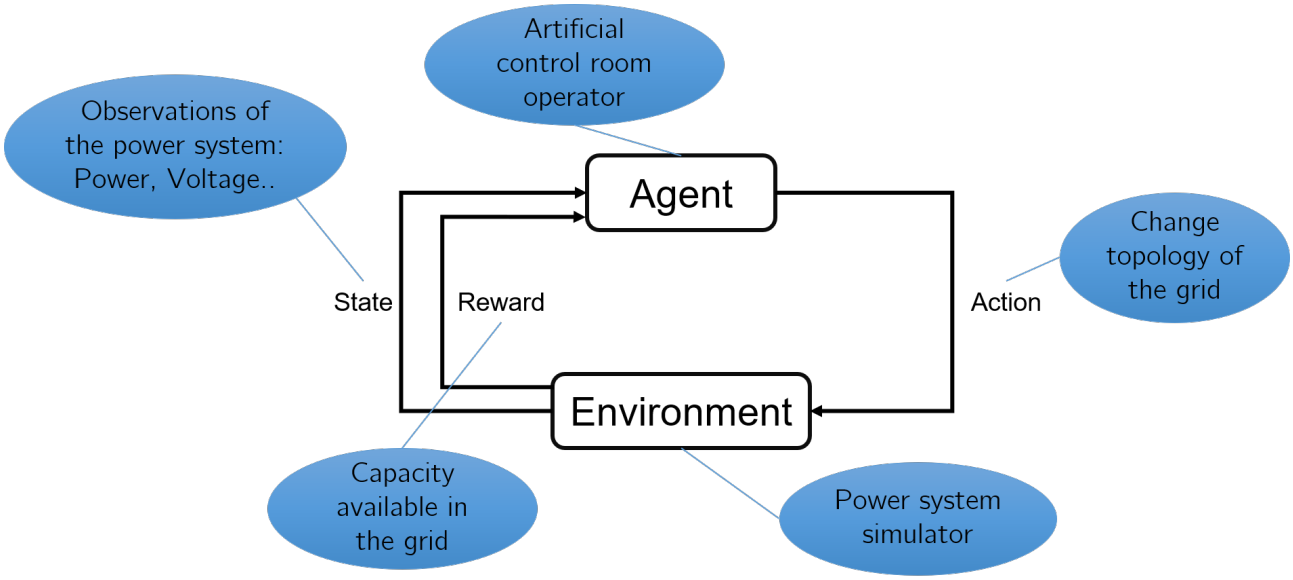


Figure 2.16: Formulation of the reinforcement learning problem with respect to network operation

3
Framework

In this chapter, the requirements and the basis of this project are discussed, such as the software involved and the important python packages used for this thesis, followed by a description of elements in the IEEE 14 bus test case. Finally, the rules that define the game of operating the power network are described.

3.1. Tools Description

This thesis was done entirely using the Python programming language, version 3.6. The specifications for the computer varied between 1 to 2 virtual CPUs and 4 to 32 GigaBytes of RAM; this varied depending on the computational requirement of the task being performed. A non-exhaustive list of packages used within Python are as follows *(and their respective versions)*:

- NumPy: To handle multi-dimensional arrays and matrices and perform mathematical functions *(1.18.2)*
- Pandas: To create data structures out of tables and numerical time-series *(1.0.3)*
- matplotlib: To create plots and visualisations *(2.0.0)*
- plotly: A second library to create more elaborate efficient visualisations *(4.5.0)*
- PyTorch: Used for the development of the reinforcement learning agent *(1.4.0)*
- Grid2Op: Serves as the environment for the reinforcement learning setup. *(1.0.0)*
  This package is described in further detail in the following subsection.

3.1.1. Grid2Op

Grid To Operate (or Grid2Op in short) [46] is an easy-to-use framework, that can be used for developing, training or evaluating performances of reinforcement learning agents which are deployed on the power grid to act in different ways. It assists in simulating a power grid (of any size or characteristics) subject to a set of temporal injections (production and consumption time-series).

The Grid2Op module allows to perform sequential actions on a power grid. It is a modular framework since it allows the use of different power-flow solver. Load flow computations are carried out using a backend available within this package. Grid2Op uses Pandapower as a default backend, which is an open source library that can be used to perform load flow computations. Pandapower proposes an internal representation of the available data required to construct the power grids. In their documentation, this package also provides compatibility with Open-AI gym, more commonly called Gym. Gym is a toolkit used for developing reinforcement learning algorithms. It supports teaching agents everything like playing games like Pong or Pinball [47]. In this toolkit, an interface is provided between the environment and the agent. This shifts the focus of the user onto only creating the algorithm of the agent. The goal of creating Grid2Op was to provide a Gym-like toolkit for power system applications. Grid2Op has classes that are implemented entirely using the Gym interface. An illustration of the architecture utilised in the Grid2Op package can be seen in figure 3.1. The numbers depicted in this figure show the sequence followed within the architecture. First, the agent gets the previous time-step's observations and reward. Following this, the agent submits an action to be taken. At the same time, the environment accesses the injection-related data for the current time-step. Using this, the environment calls the backend to perform load flow computations. This is the cycle followed within this architecture. More details regarding this package can be found in their documentation in [48].

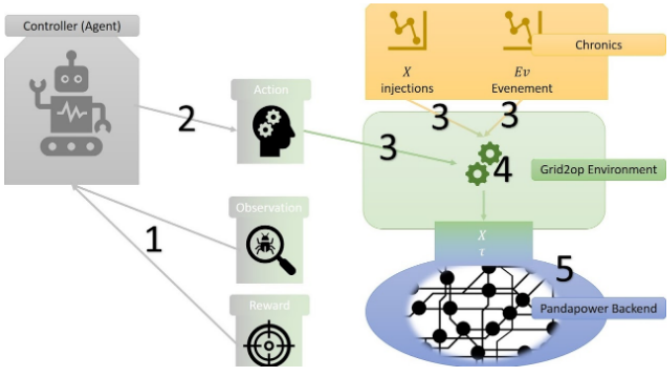


Figure 3.1: An illustration of Grid2Op's architecture [46]

This framework was developed with the purpose of serving as a computation engine for the L2RPN competitions. But it is fully customisable as it allows various aspects that can be changed; such as the backend, the power grid, the input data and so on. It has been built keeping in mind a need for flexibility so that it can serve different purposes that are not only restricted to reinforcement learning agents, or the L2RPN competition. Further, the time-series described in this chapter were developed for the L2PRN competitions and are specific to this Grid2Op package. This is the package that is used in this thesis, to serve as the environment for the reinforcement learning setup.

3.2. IEEE 14 Bus Test Case

The IEEE 14 Bus Test Case is the particular power grid used in this thesis. This test case is used since it is a realistic and commonly used IEEE power grid. It represents a portion of the American Electric Power System (in the Midwestern US) as of February, 1962. It serves its purpose as a minimalist grid to ensure feasibility and help analysis, yet a grid for which topological actions could still be useful. For this, it is useful to consider a grid which is meshed and contains several paths. This example is one such meshed grid with 2 West and East Corridors between a meshed South Transmission Grid and a meshed North Distribution Grid [49].

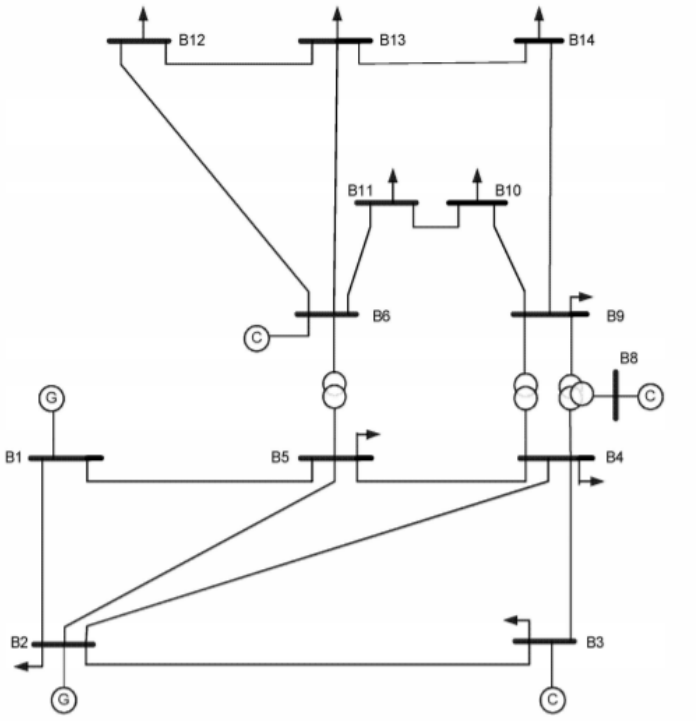


Figure 3.2: SLD of original IEEE 14 bus test case

The Single Line Diagram (SLD) of the original IEEE 14-bus standard test system is shown in figure 3.2. It consists of five synchronous machines in the form of only thermal production, including two generators, located at buses 1 and 2 as well as three synchronous compensators used only for reactive power support, located at buses 3, 6 and 8. The network used in this framework is slightly modified compared to the original case to better represent today's energy mix and eventually serve the purpose to manage resulting issues. This network consists of 14 substations, 5 generators, 11 loads, 20 lines. Occurrence of contingencies and maintenance were not considered in this test example, since they can very easily be the main cause of an infeasibility in the system. This means overloads and failures will be a result of peak production or peak load in certain areas of the grid. The base case topology T_{ref}, is fully meshed with all lines in service and one active electrical node/bus bar per substation [8].

Understanding the various assets of the grid is beneficial to implement and deploy an agent, mainly as it can help us understand the behaviour of the agent on this network. The assets involved in this 14 bus network are described below.

3.2.1. Substations / nodes

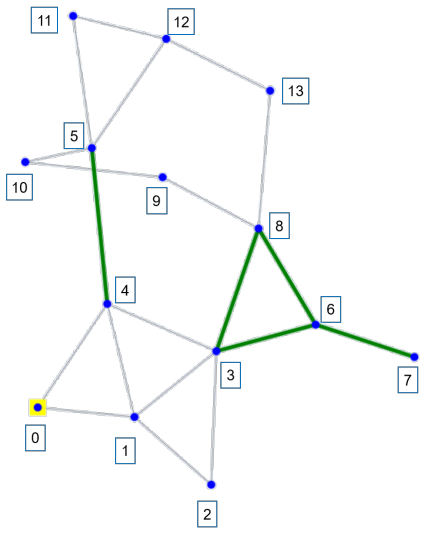


Figure 3.3: 14 bus network

This network consists of 14 nodes, as seen using a simplified way to depict the topology, in figure 3.3. Each of these nodes has equipment (hereinafter known as elements) connected to it, such as generators, loads and lines. Each of these nodes can be viewed as a substation which is modelled to have 2 bus-bars. Each bus-bar can be visualised as a sub-node of the substation which is main node. The elements connected to these substations can be connected to either one of these bus-bars; the default (or reference) setting is that they are all connected to bus-bar 1. In the real-world, the physical substation that exists varies from this modelled substation. A modelled substation relies on the fact that a particular substation has a corresponding nominal voltage level. If there exists multiple voltage levels within a particular physical substation, when this is modelled within a power flow solver, it will be modelled as multiple (2 or more) substations.

3.2.2. Transmission Lines

There are a total of 20 transmission lines in the system, including both the transmission and distribution grids as seen in figure 3.4. Each line acts as a connection between two buses, and hence these two connection points can be visualised as the line origin and the line extremity (or end). The power flow along each of these lines is visible as a form of net power flow, implying that the power flowing from bus 1 to 2 will have a positive value at line origin and negative value at line extremity. Transformers and phase shifters are also modelled as transmission lines. This is carried out such that a transformer is seen to be physically placed at the "from-bus" or line origin, and has a particular tap ratio value. Conventionally, both transmission lines and transformers are modelled in the same way. A value of zero for the tap ratio indicates that the modelled element is a transmission line and not a transformer, i.e. it is equivalent to a transformer with tap ratio set to 1. Hence, in this example, the lines marked in green are the transformers modelled as lines: line 17 and lines 15, 16, 18, 19 (which is in reality a three-winding transformer).

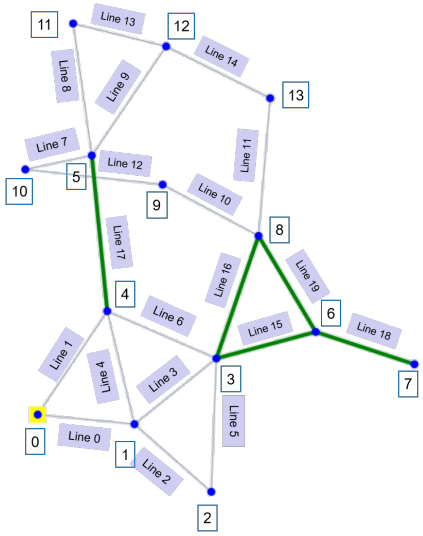


Figure 3.4: 14 bus network

3.2.3. Voltage Levels

This 14 bus network is a combination of a transmission grid and a distribution grid. This also helps justify the transformers in the system which divide the network into its two separate voltage levels. This is spatially visible in figure 3.5.

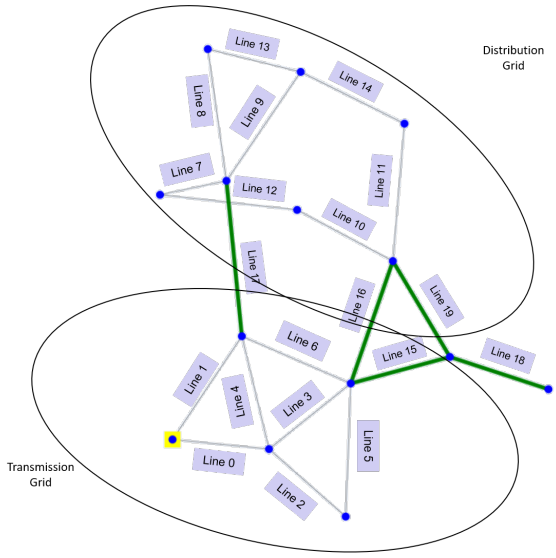


Figure 3.5: Two voltage levels in the 14 bus network

3.2.4. Generation Units

The original IEEE 14 bus test case had 5 conventional power plants which is modified here and instead, a few RES are introduced to the mix as can be seen in figure 3.6. There is also one nuclear power plant as part of the overall production in the grid. The specifications of the five generators are as follows:

- Generator 0: Nuclear Power Plant
- Generator 1: Thermal Power Plant
- Generator 2: Wind Power Plant
- Generator 3: Solar in-feed
- Generator 4: Thermal Power Plant

The generators are modelled as a complex sum of active and reactive power at a specific bus as follows:

$$S_g = P_g + (Q_g) \times j \tag{3.1}$$

Where the subscript 'g' implies generator and:
S_g : Apparent Power
P_g : Active Power
Q_g: Reactive Power

The magnitude of the apparent power can be seen from: $\left|S_g\right| = \sqrt{(P_g)^2 + (Q_g)^2}$

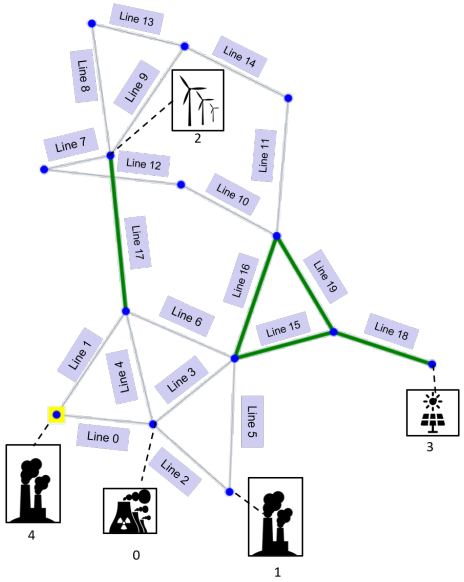


Figure 3.6: Generators in the 14 bus network

The time-series depicting the solar and wind energy generation are representative distributions for a French grid, provided as part of the Grid2Op package. The time-series of injections are chosen to be representative of winter months where peak loads are observed. The conventional power plants (the two thermal and one nuclear power plants) are treated as a baseload which then compensate for the remaining production required for overall balancing [8]. An example of the generation time-series can be seen in figure 3.7. This figure depicts the operation for the 14 bus power grid for 10080 minutes (1 week) or 2016 time-steps with each time step lasting 5 minutes. Hence, this example shown below provides the time-series for the duration of one week. Each of these generators have separate maximum power generation limits, as seen in table 3.1.

Table 3.1: Maximum allowed generation for each generator

Generator	Type	Pmax (in MW)
Generator 0	**nuclear**	**150**
Generator 1	**thermal**	**200**
Generator 2	**wind**	**70**
Generator 3	**solar**	**50**
Generator 4	**thermal**	**300**

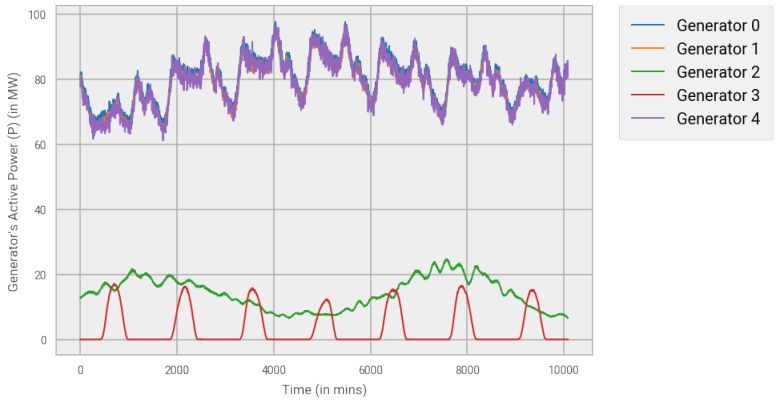


Figure 3.7: Example of generation time-series

3.2.5. Load Units

There are a total of 11 loads in the grid as seen in figure 3.8. the The time-series for the loads are generated to reflect typical operation of loads in France during winter months. Each of the 11 loads are computed, for these created time-series, using a constant key factor and the total load is divided proportionally amongst all 11 loads in the network. Upon each generated time-series, some noise has been added; this noise is built in a manner such that loads that are geographically closer to each other are more correlated compared to two loads that are far apart from each other [8] [50]. Similar to the generations, the loads are modelled as a consumption at a specific bus as a sum of complex quantities.
For the same sample generation time-series shown in figure 3.7, the load time-series can be found below in figure 3.9.

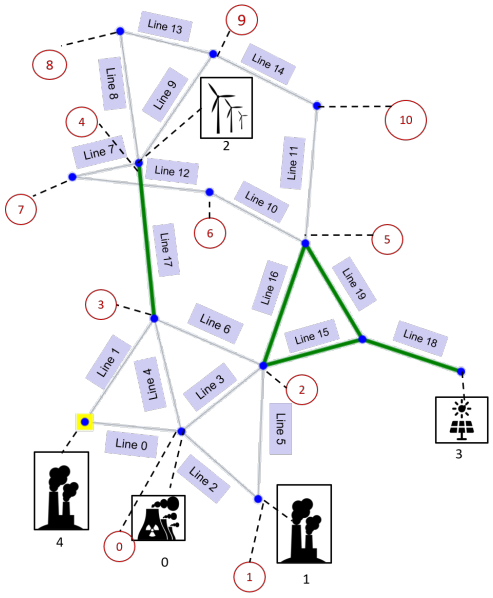


Figure 3.8: Load locations in the 14 bus network

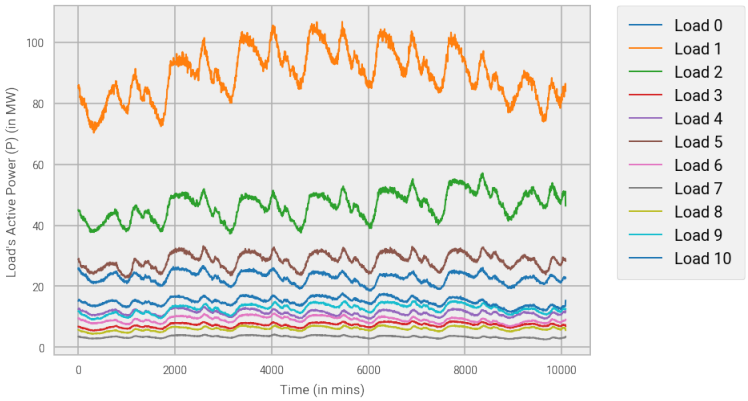


Figure 3.9: Example of load time-series

3.3. Game Rules and Game Over Situation

Grid operators need to consider various things in order to take optimal actions during the time of operation. There are various constraints that require to be taken into account. This can either be in the form of limitations of the equipment, and hence there are thresholds on the amount of allowed current flow, or also if there are planned outages (or maintenance) being conducted on parts of the grid that the operators need to work around. Further, there are also protection devices in place in the grid that get activated if current flow thresholds are crossed. To characterise all these nuances in the environment, they are represented as rules that the operator (in our case, the agent or artificial operator) need to follow, in a 'game' which is the operation of the power system. Below, the rules of the game are described, followed by a description of how a 'game over' situation can arise.

3.3.1. Game Rules

There are certain rules and constraints that are considered in the framework that can be customised. The parameters that can be customised help represent real-world problems that arise due to operational constraints. These parameters also help represent the protection equipment that exist in a power grid that will be automatically triggered based on some defined rules. They are [48]:

- **Reaction Time:** This is the number of timesteps allowed for a line to be overloaded before it gets disconnected. Each timestep as seen in this game, is 5 minutes long. (Default: 2 timesteps, or 10 minutes)
- **Line Status Remodification Time:** If a line was acted on by changing its status (connected / disconnected), this time indicates how many timesteps the agent will have to wait before modifying this line's status again. (Default: 0 i.e. deactivating this feature)
- **Topology Re-modification Time:** Similar to Line Status Re-modification, when the topology of a single substation is changed, this time indicates how long the agent will have to wait before modifying this particular substation's topology again. (Default: 0; deactivating this feature)
- **Maximum Substations Modification:** For two consecutive timesteps, this parameter indicates the maximum number of substations that can be modified by the agent. If the agent violates this rule, the chosen action will be replaced with an empty action (do-nothing action), i.e. an action which doesn't change the configuration of the network. (Default: 1 Substation)
- **Maximum Lines Modification:** Similar to Maximum Substations Modification, this parameter indicates the number of lines that can be modified by the agent for two consecutive timesteps. (Default: 1 line)

The rules help depict the real-world power systems and its constraints. Reaction Time is a parameter used to depict the behaviour of protection equipment such as circuit breakers. Line Status Re-modification Time is used to depict how much time some lines require to cool down before being brought back into operation. The other rules are used to restrict the complexity of the game.

3.3.2. Game Over Situation

There is a series of tasks that play a role in eventually causing failure in this simulated power grid. Based on a generation and/or load time-series, the power flow over the lines varies, and due to a possible excessive

amount of power in some lines, they face overloads. In reality, if there are many lines that get overloaded and then disconnected from the system, such a situation can result in a large congestions in the grid since there might not be a feasible way to transport power in the network. If such a situation is not handled operationally, the grid can face large failures that result in damaged equipment and also blackouts. In our game setup, this type of situation will result in a 'game over'. To describe the cause of 'game over', there are two important parameters:

- **Soft threshold**: If a line is overloaded, that is, ρ is above 1 for more than a pre-defined set of consecutive time-steps, the line is then automatically disconnected. The default value which is also used in this thesis is 2 time-steps.
- **Hard threshold**: If a line is highly overloaded and reaches a value of some factor $\times I_{max}$, then the line is automatically disconnected, regardless of the soft threshold. The default value of 2; hence 2 $\times I_{max}$ will make the line get disconnected.

If a line gets disconnected automatically, this then puts pressure on neighbouring lines, which might then get disconnected, leading to a cascading failure. This is detected by the backend of the environment which performs load flow calculations and it raises a flag.

The game over situation however, does not directly depend on these thresholds. The main criteria for the game to be over, depends on this flag provided by the backend which would indicate whether the load-flow solver is not converging at a solution. Another factor is, if any load gets disconnected from the grid, this will instantly cause the game to be over. The possibility of cascading failures play a role in this divergence of load flow. Hence, from this it is clear that the load-flow solver along with the occurrence of a load disconnection are the factors which cause a "Game Over" situation. This is also illustrated below in figure 3.10.

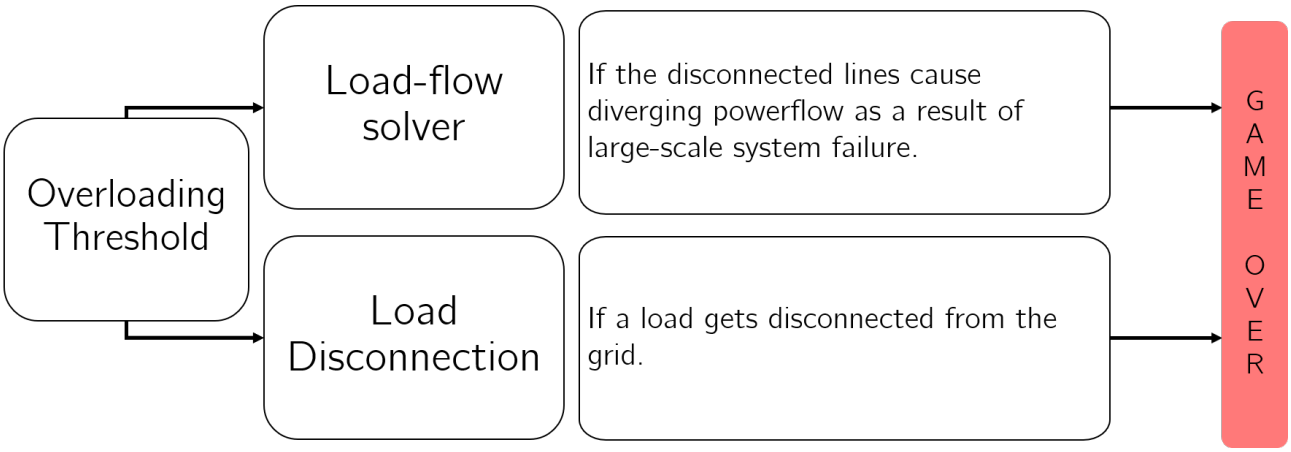


Figure 3.10: Illustration of the game over situation

It is good to also highlight the meaning of this situation from a reinforcement learning point of view. The reward during a game, can be accumulated by the agent at every time-step. This "Game Over" situation essentially means that the agent will not be able to collect reward any further, since the agent cannot 'play the game' any further. Thus, the total game's reward is restricted due to the game's failure.

4

Power Grid Setup

The platform and the power grid for which an agent will be developed have been described in chapter 3. Some preliminary work has to be carried out before a reinforcement learning agent can be developed. While building an agent, different conditions and data have to be experienced by it, from which it can learn. In this chapter, the power grid described in chapter 3 and its accompanying data is analysed and summarised. This helps us understand the behaviour of the grid when it is subject to this variety of inputs. This chapter first describes a vital parameter of in the grid (thermal limits) followed by an overview of different scenarios.

4.1. Thermal Limits

A parameter that plays an important role pertaining to the grid's behaviour is ρ (rho). It is defined as the current flow through a line over the maximum thermal current rating of the line, as seen below in equation 4.1.

$$\rho_l = \frac{\left|I_{flow}\right|}{I_{max}} \tag{4.1}$$

Where l is the line under consideration

Lines	Thermal Limits (A)	Lines	Thermal Limits (A)
Line 0	1000	Line 10	380
Line 1	1000	Line 11	380
Line 2	1000	Line 12	760
Line 3	1000	Line 13	760
Line 4	1000	Line 14	380
Line 5	1000	Line 15	760
Line 6	1000	Line 16	380
Line 7	760	Line 17	380
Line 8	450	Line 18	2000
Line 9	760	Line 19	2000

Figure 4.1: Thermal limits of the lines

For each line, there is a maximum current flow/rating of lines that can be defined to encapsulate the inherent thermal capacities of the lines. These maximum values (called thermal limits) are constraining factors which

cause overloads in lines. These, in turn, play a role in depicting the game over situation as seen in section 3.3.2. The values defined for the 14 bus network used in this thesis are as shown in 4.1. The motivation behind modifying these thermal limits is driven by the large influence they have on whether a 'game over situation' arises or not. They are adapted from the original Grid2op package's (section 3.1.1) limits in order to reflect the real-world grid as close as possible such that operating the grid doesn't become too easy or too difficult either. In the following sections, the behaviour of the grid is discussed using this setup of thermal limits.

4.2. Scenario Analysis

Here, we first understand the status of the grid on its own. For this, we must analyse the load and generation time-series available for different scenarios. A scenario is a set of load and generation time-series (also referred to as profiles) which are provided as input to the environment using which the observations of the grid will be obtained. There are a total of 1000 scenarios available, each having data for 8064 time-steps (i.e. 28 days). In this thesis, a time-step refers to a duration of 5 minutes. Analysing these scenarios also help establish the behaviour of the grid. The behaviour of the grid is represented by different parameters of the grid, such as the current flow through lines, overloading of lines, and congestion/bottleneck points in the grid, for various scenarios.

4.2.1. The 'Reference Configuration' Agent

To eventually create a reinforcement learning agent, it is useful to first understand the behaviour of a baseline agent. This particular baseline agent retains the network in a reference/ base case topology (T_{ref}) for the entire length of the scenario data. This baseline is called the 'Reference Configuration (RC)' agent.

This agent essentially does not take any actions on the grid and only retains the grid in its default or reference configuration. By using this agent, we can play the game and obtain corresponding observations and rewards. In this reference configuration, all elements are connected to bus-bar number 1 on each substation. The purpose of this agent is to showcase the operation of the power grid for a given duration of time when there is virtually no agent (or there is an agent performing no actions) and see when the power grid fails to continue operation and why this occurs.

This agent is mainly useful to help identify different types of scenarios. The scenarios can be classified using the results of this agent. The 1000 available scenarios are examined by deploying the RC agent on the grid and obtaining observations. By doing this for 1000 different scenarios, we obtain 1000 *episodes*. An episode is a run/play of the agent where it executes its policy on some scenario(s) and takes some actions on the network. Thus, the information contained in an episode can be represented as a list of tuples where each tuple consists of the observation, action and reward. The number of tuples in this list would then be the length or duration of an episode. An episode terminates when either the agent succeeds (wins the game) or fails to operate the grid for that scenario (loses the game). In our context, the agent's success (winning the game) is defined by whether it operates the grid for the entire duration as provided in the scenarios, so, the episode ends when available data in the scenario is exhausted.

In these 1000 episodes (or more explicitly, the RC episodes) there are cases where the agent succeeds, i.e., the power grid staying in the reference configuration is sufficient for the successful operation of the grid for the entire duration of the scenario.The rest of the episodes are those where this reference configuration is an insufficient topology to operate the grid for the provided input data, leading to infeasible grid operation, which caused the episode to end prematurely. The former type is thus named a "Successful Episode" and latter an "Unsuccessful Episode". In the overview of episodes provided here, a week (2016 time-steps / 7 days) of data available in the scenarios are explored. In the following sections, these two types are expanded upon after which a summary of this RC agent is discussed.

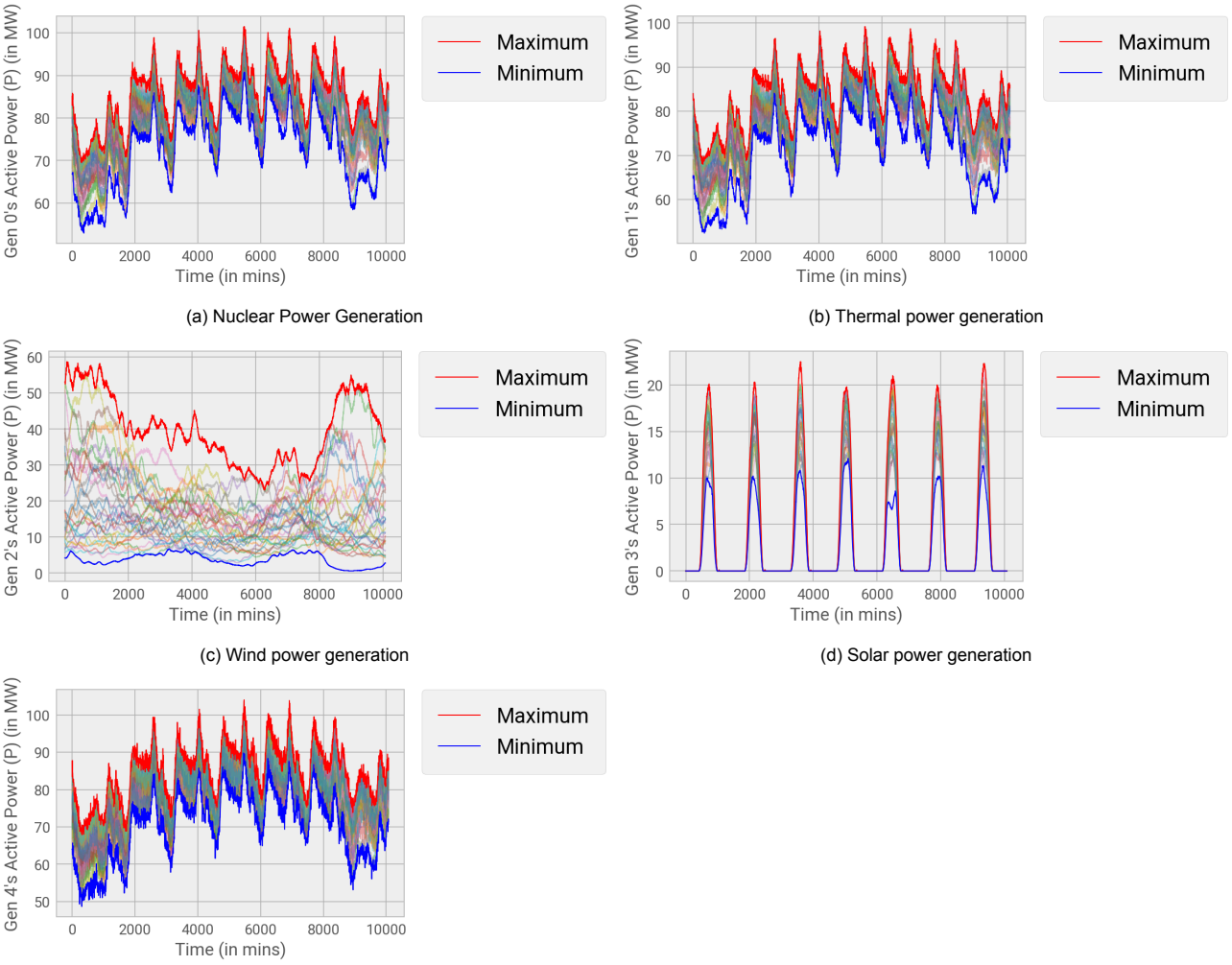
4.2.2. Successful Episodes

Out of 1000 episodes, 38% of them are successful, implying that the grid operates optimally for the entire duration of one week (2016 time-steps). As discussed in Chapter 3, observations such as Active Power (P), Current (I) and Voltage (V) can be extracted for all components in the power grid. To briefly describe these successful episodes, some of these parameters are visualised together for all these episodes. This method of visualisation, by using "envelopes", helps identify a general range of variations of these observations. The main idea behind creating these envelopes is to use them as a means of comparison for episodes that are unsuccessful, such that the envelopes can indicate how "normal" or "abnormal" the episodes are, compared to the successful ones. This will be explored in the following section on "Unsuccessful episodes".

In figure 4.2, the envelopes created for the Active Power of all the generators can be seen. The top-most curve (in red) marks the maximum value reached in any of the time-steps for all these successful episodes,

and similarly, the bottom-most curve (in blue) marks the minimum values. The load profiles and their envelopes can also be visualised in the same manner; a subset of the 11 loads' active power profiles are visible in figure 4.3.

An important parameter to visualise how "stressed" or overloaded the system is, is "rho" (ρ) as discussed in section 4.1 and equation 4.1. ρ is a ratio of I_{flow} over ${}_{I}max$ that is determined by the maximum permissible current flow due to thermal limits of the line. An inherent maximum and minimum 'envelope' already exists in the form of the range of this ratio; i.e. between 0 and 1. Nevertheless, in figure 4.4, the loading of all 20 lines for a single successful episode can be seen.



(a) Nuclear Power Generation

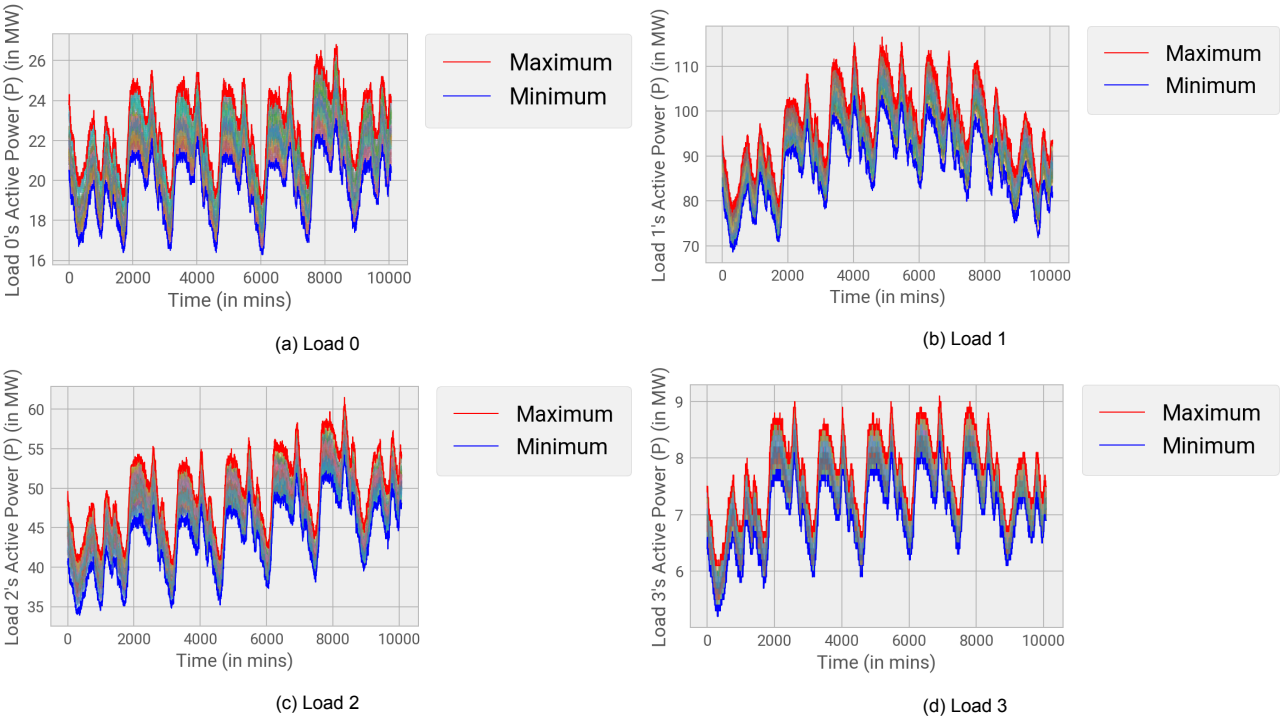
(b) Thermal power generation

(c) Wind power generation

(d) Solar power generation

(e) Thermal Power Plant

Figure 4.2: Generators' Active Power Profiles for Successful Episodes



(a) Load 0

(b) Load 1

(c) Load 2

(d) Load 3

Figure 4.3: Loads' Active Power Profiles for successful episodes

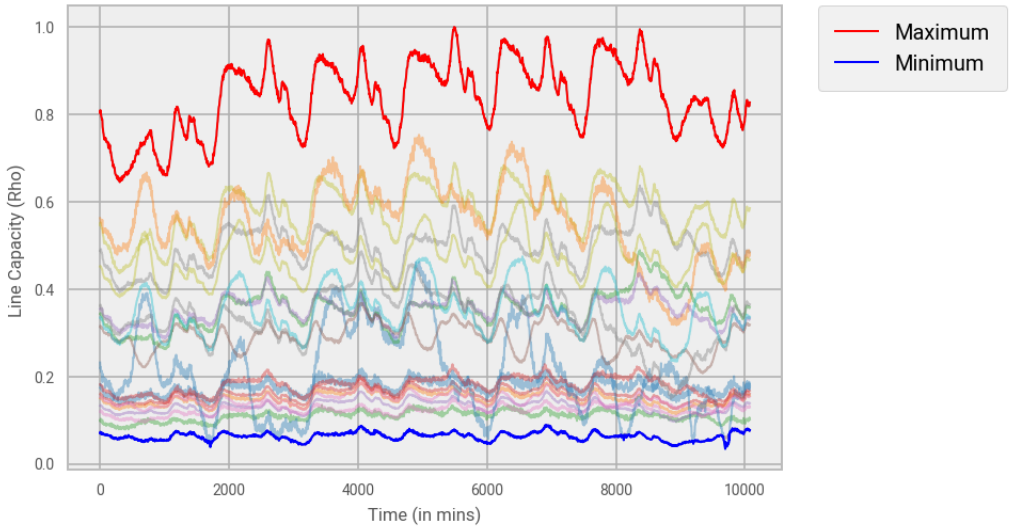
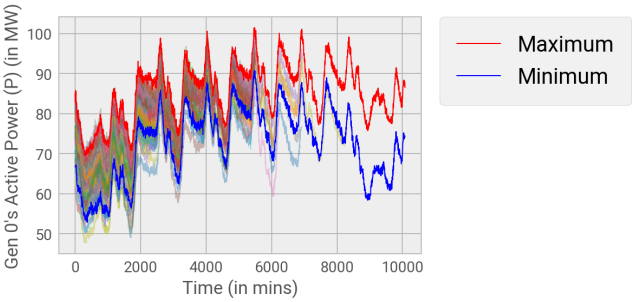


Figure 4.4: Loading of all 20 lines for a single successful episode

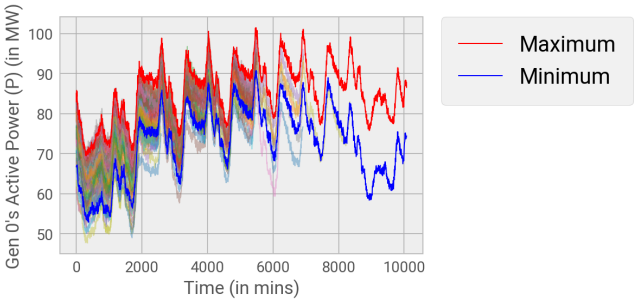
4.2.3. Unsuccessful episodes / Failure episodes

Out of the 1000 RC episodes that are available for analysis, 62% of them are unsuccessful. This means that the episodes terminated prematurely due to an inconsistent or infeasible situation that occurred in the power grid. This happens mostly in the form of overloads which trigger a "game over" situation (section 3.3.2).

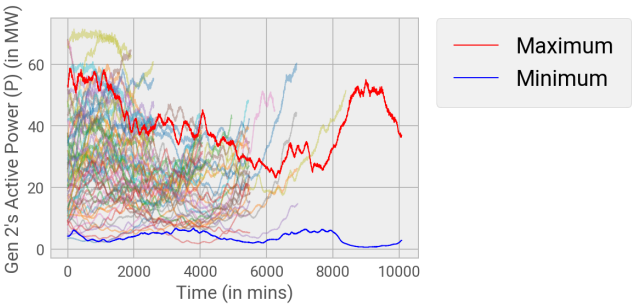
Using the envelopes that indicate the maximum and minimum values of the successful episodes and overlaying them onto the unsuccessful episodes, we hope to identify how different (or similar) unsuccessful episodes are in comparison to the successful ones. By doing this, there can be some direct distinctions made between the different episodes. In figure 4.5, for each of the generation units, deviation from the envelopes for the successful episodes can be seen. While in the solar in-feed, there is not a large deviation from the envelopes of the successful episodes, the deviations are more noticeable in the rest, most prominently in that of the wind generation. Similarly, this can be done for the load profiles as well. A sample of the curves is presented in figure 4.6. In this case, the deviations are negligible.



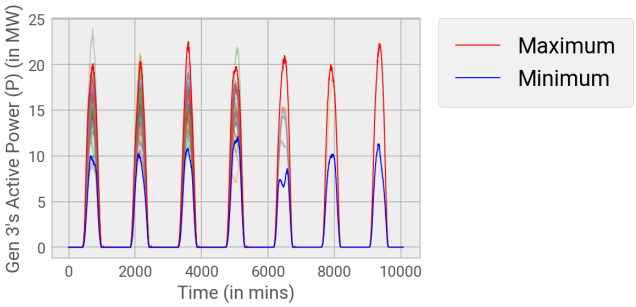
(a) Nuclear Power Generation



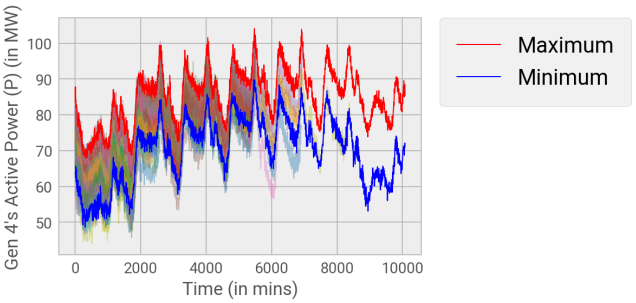
(b) Thermal power generation



(c) Wind power generation

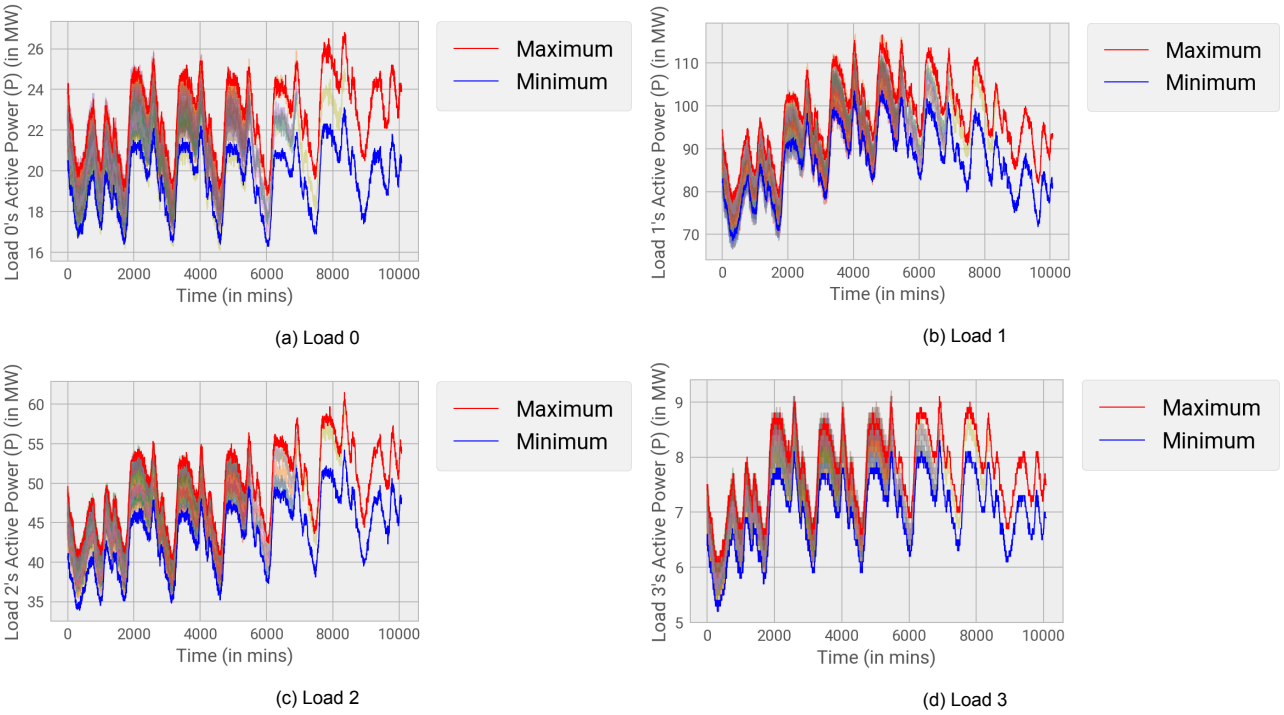


(d) Solar power generation



(e) Thermal Power Plant

Figure 4.5: Generators' Active Power Profiles for unsuccessful episodes



(a) Load 0

(b) Load 1

(c) Load 2

(d) Load 3

Figure 4.6: Loads' Active Power Profiles for unsuccessful episodes

The other discussed parameter is that of ρ. Again, it is more useful to see all 20 lines' loading together, as seen in 4.7. For this particular episode, the failure occurs at toward the end of the week (over 8000 time-steps), that is around four days. This reflects the large impact of ρ on the game over situation. The lines getting overloaded nearing the time of failure all cross a ρ value of 1. This is further explained in the next section.

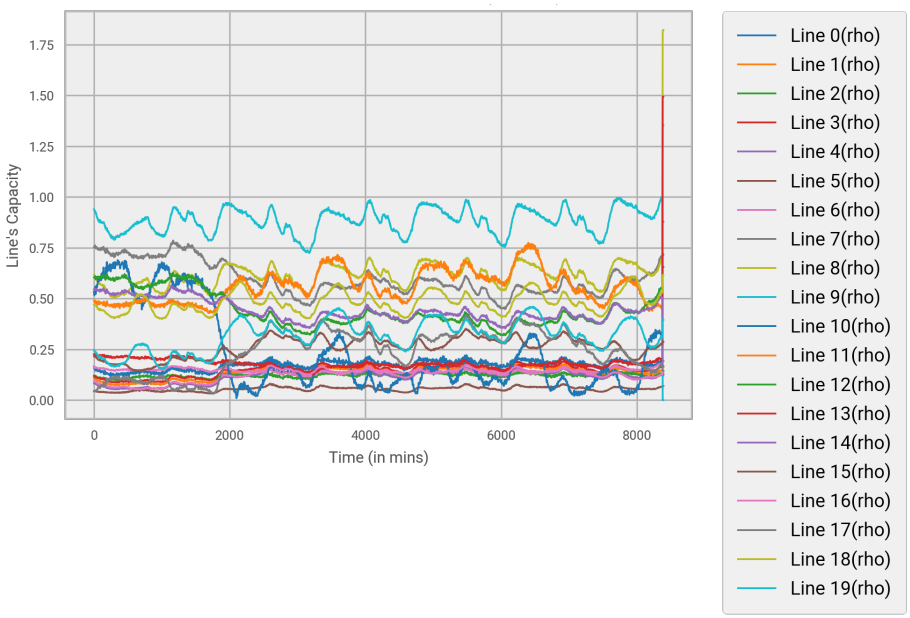
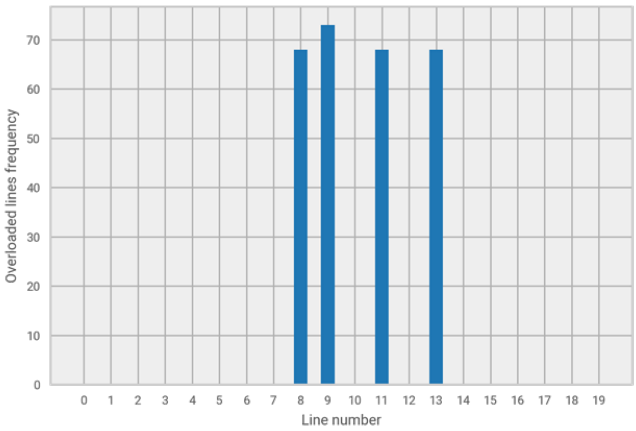


Figure 4.7: Lines' loading for all 20 lines for a single unsuccessful episode

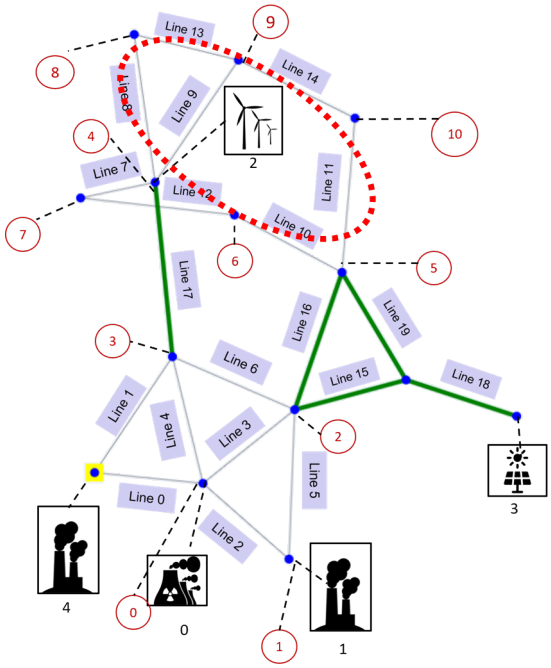
The envelopes described here do not indicate a hard-and-fast rule for failure, rather act as a visualisation tool to summarise these episodes. Note that there are episodes for which the load and generation profiles fit into the same envelope as that of the successful episodes. However, they lead to "game over" situation. This signifies that the failures caused are a combined effect of the injection profiles as well as this reference topology of the system. In the following section, a brief analysis on what leads to the game over situation in specific episodes can be seen.

4.3. Failure analysis

To understand how the game over situation arises in all the unsuccessful episodes, the overloading of lines can be analysed. A certain trend was noticed: the lines that got overloaded frequently and led to the failure of the episode, were all in the distribution side of the grid. The lines that get overloaded are: 8, 9, 11 and 13 (as seen in figure 4.8a). With this, a bottleneck of the grid can be identified (figure 4.8b), which essentially encapsulates the impact on lines 8, 9, 11 and 13. At this point, it is useful to analyse how a failure occurs due to the the overloading of these lines. To do so, a single episode is analysed. By doing this, we can understand how this overload and consequent bottleneck led to a game over situation.



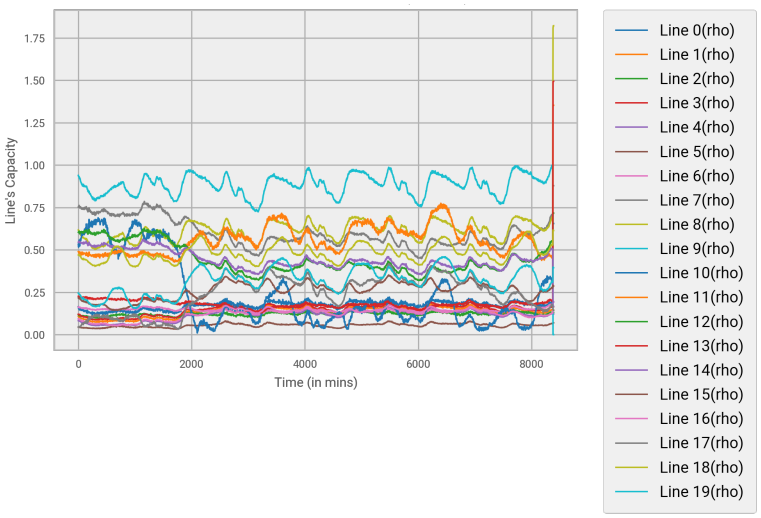
(a) Frequency of line overloading



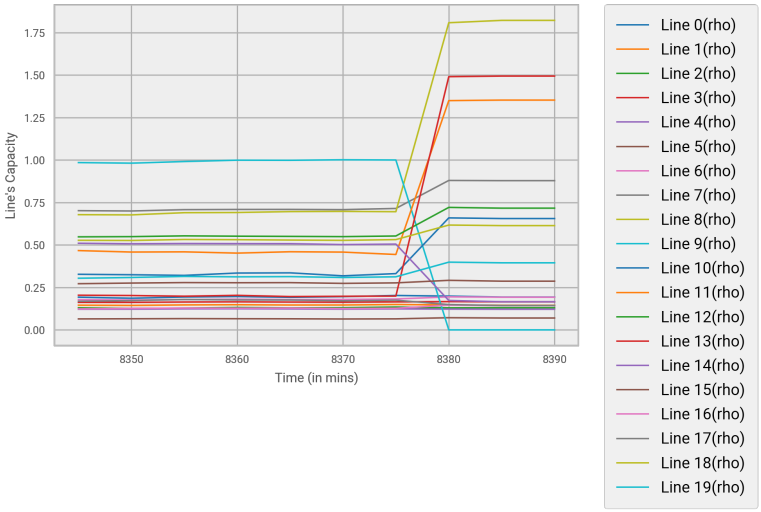
(b) Depicting the bottleneck in the grid

Figure 4.8: Line overloading and bottleneck in grid

The ρ for all 20 lines for a single episode is seen in figure 4.9. As mentioned in the section on Game Over Situation (section 3.3.2), we know that for the soft threshold criterion, if a line is overloaded ($\rho > 1$) for more than a pre-defined value of consecutive time-steps (here: 2 time-steps), the line is then automatically disconnected. So, looking at the ρ of all 20 lines to see when overloads occur (figure 4.9a), the lines that get overloaded can be noticed towards the end of the episode. A magnified version of this is depicted in figure 4.9b. From here, we can clearly see the step-by-step process, which leads to the end of the episode. We see that line 9 gets overloaded for more than two consecutive time-steps (8370 till 8375 minutes). This causes it to get disconnected due to the rules of the environment by the next time-step. As this happens, this simultaneously stresses neighbouring lines (8,11, and 13), which get overloaded for more than two time-steps as well. Consequently, this causes all three lines to be disconnected. This leads to grid situation where four lines are out of order. If we look at the grid, we see that lines 8, and 13 are directly connected to a single load (load number 8). From the game over rules, we know that disconnection of a load triggers the game over flag. This behaviour, where line 9 gets overloaded and then disconnected, which leads to lines 8, 11, 13 getting disconnected as well, was verified for all the failure episodes.



(a) Overloading of 20 lines



(b) Overloading of 20 lines (zoomed)

Figure 4.9: Overloading of 20 lines for a failure episode (episode 15)

While, in the above manner, the behaviour of the grid for each episode can be depicted, an attempt is made to generalise episodes. An important factor which is related to the location of a bottleneck in the grid is the location of the wind power generator at substation 5. This generator has the largest variability compared to all the generators. A comparison can also be drawn between the RES and conventional power generators in the network. To see the effect of RES with respect to the power generated by conventional generators, figure 4.10 can be referred. Figure 4.10 depicts the maximum of RES power generated for all episodes against the maximum power generated by conventional generators for all episodes. Compared to the wind power, the impact of solar power on the overall RES power is relatively lower due to a smaller magnitude. With this, a stark differentiation between success and failure episodes can be made. For any episodes that have a generated renewable source of power of larger than 30 MW, there have been no successful RC episodes. This highlights the effect of the wind power on the overloading of lines in close proximity to it. This further clarifies the trend depicted for the above discussed episode (figure 4.9). Another interesting aspect that can be noticed in figure 4.10 is that while the impact of higher wind power generation helps categorise a scenario on whether it will result in a failure episode, we cannot say the same for whether it will be a successful episode. Though there is a high generation from all other generators, this does not necessarily result in a failure episode. This is justified largely by the location of the wind generator. It is located at the distribution side, which has lower line ratings compared to that of the lines in the transmission side. So, the impact due to the large amounts of wind power is more evident than the impact due to the sum of all other generators since their location is at the transmission end, which has lines with higher ratings. Even though the successful episodes all appear on the larger side of the range of total generated power, the result that there are successful episodes, once again is combined with the fact that wind power generation is low for these episodes. Hence, the impact of other generators is not large

enough to decide the result of the episode. It is important to note here again that in the setup for this thesis, the only allowed actions involve changing the topology of the grid and other solutions such as redispatch are not considered.

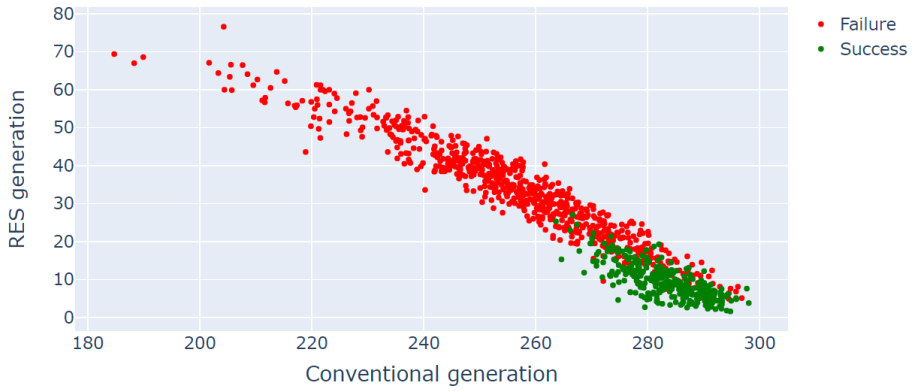


Figure 4.10: Maximum RES power vs maximum conventional power generation

4.4. Levels of difficulty

Another important factor that plays a role for developing an agent, is allowing agents to experience different available scenarios. One way to categorise these different experiences, is by establishing levels of difficulty for the 'game' of network operation. One of the best ways of determining these levels of difficulty, is related to the ρ values for lines. Since, line loading plays a large role in determining how stable the grid is, the higher the loading of lines, the more difficult that episode is. These loaded lines are an effect of the overall production in the grid. To visualise this, all the episodes with overloading ($\rho > 1$) against the total power generation in the network can be seen in figure 4.11. Each point in this figure represents a single failure episode. This figure once again describes the four lines that create the bottleneck (lines 8, 9, 11 and 13). In this figure, we can see that of the four lines that face overloading, the line that has the highest variability of overloading is line 11. This helps in determining the levels of difficulty, such that an episode with higher loading on line 11 is more difficult that an episode with lower loading.

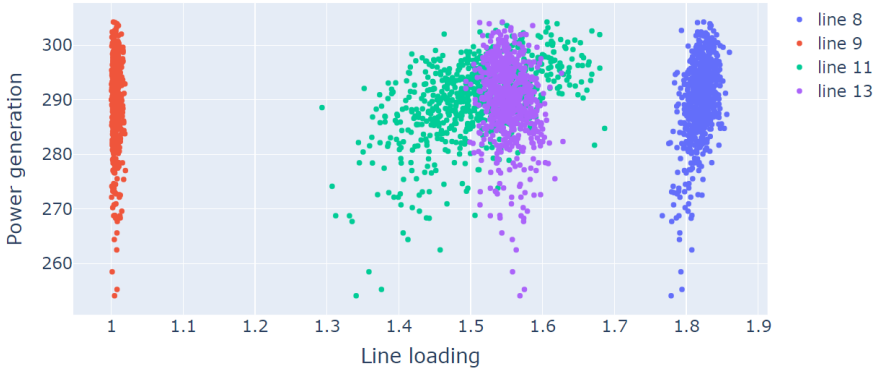


Figure 4.11: Comparing total generation with lines' loading

Thus, we establish three levels of difficulty to categorise all the 1000 available scenarios:

1. *Easy*: These are those scenarios for which the RC episodes faced no overloading at all, hence, the successful RC episodes (the green region of figure 4.12)
2. *Medium*: These are those scenarios for which the RC episodes face lower overloading on line 11, i.e. in the range of 30 % to 50 % overloading (the yellow region of figure 4.12)
3. *Difficult*: These are those scenarios for which the RC episodes faced overloading on line 11 in range of 50 % to 70 % (the orange region of figure 4.12).

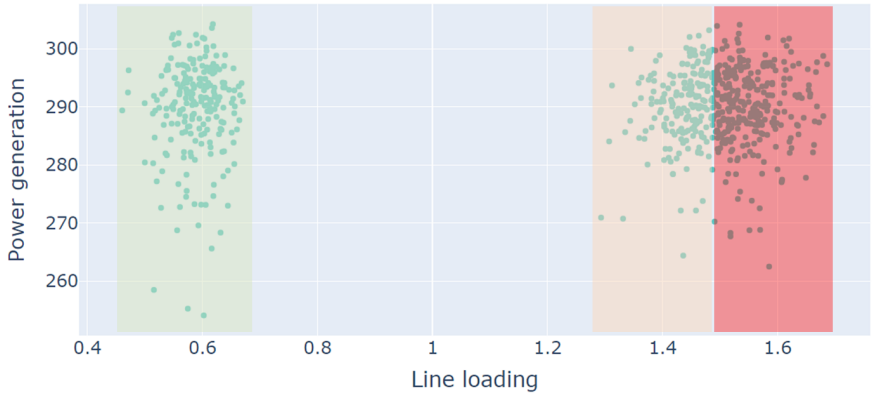


Figure 4.12: The three difficulty levels depicted determined by line 11's loading

In figure 4.13, the overloading of lines can be seen compared for both types of power: renewable and generation. From this it is clear that even though the increase in loading of line 11 is inversely dependent on RES, the impact of RES is smaller compared to conventional, as line 11's loading increases with increasing total (RES + Conventional) power. The impact of RES is lower, mainly due to a much lower magnitude when compared to the magnitude of conventionally generated power.

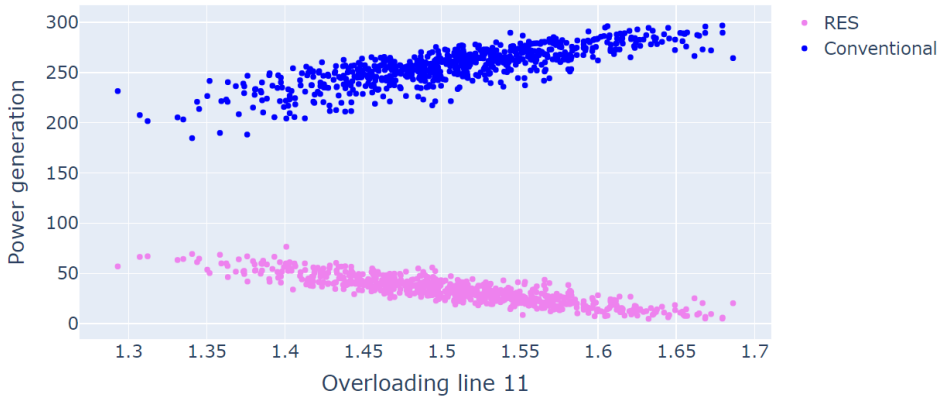


Figure 4.13: Comparing both types of generation with line 11's loading

Another interesting aspect to see is how long different episodes are. The variation of episode duration with the operation of the RC agent, compared with line 11's loading is presented in 4.14. There is no directly visible trend in the lengths of different episodes. This shows that the overloading of the line does not directly play a role in how long the episode lasts; the duration of the episode is dependent only on when the particular situation that leads to a failure occurs (as described in section 4.3). Similarly, in figure 4.15, the lengths of episodes are compared with the total mean power generation of the episodes. This once again shows that the episode duration is not directly dependent on the power generation values, as episodes face a failure situation for a range of power generation.

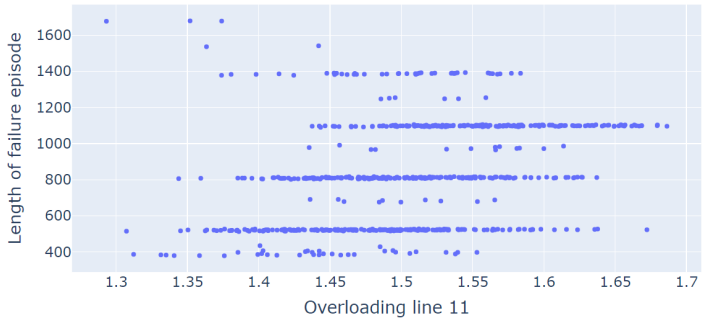


Figure 4.14: Comparing episode lengths with line 11's loading

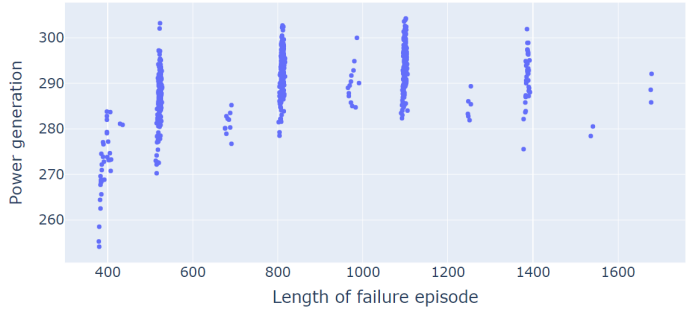
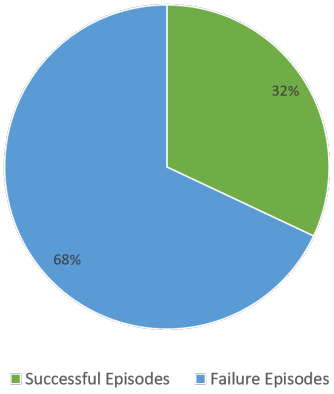


Figure 4.15: Comparing total generation with episode lengths

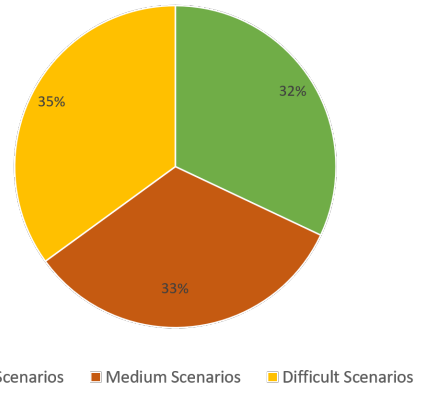
4.5. Summary

In this chapter, we discussed the behaviour of the power system by using an RC agent to operate the grid. Different scenarios result in different behaviour which show whether the RC agent for this scenario would succeed or fail. We see that out of the 1000 available scenarios, 32 % of the corresponding RC episodes result in successful episodes and the rest 68% lead to failure episodes. Figure 4.16a depicts this share of success and failure episodes.

Along with this, using the results of the RC agent, we also created levels of difficulty, using line 11's variability of loading. These levels of difficulty help categorise the scenarios, and using this categorisation we see an almost equal number in each of the categories, as seen in figure 4.16b.



(a) Split-up of successful vs failure episodes



(b) Split-up of scenarios for the three levels of difficulty

Figure 4.16: Summarising the episodes and scenarios

5

Agent Implementation

5.1. Action Space and State Space

This section describes the way actions are defined within this study, including the relevant constraints, as well as the total number of configurations that are possible on the IEEE 14 bus grid. An action in this context, is the decision that will be taken by the agent given a set of observations and reward in a particular environment. The reinforcement learning problem formulation can be found in section 2.2.4. Though it contains only 14 substations, the number of overall configurations that exist, and hence, the total number of potential actions that can be taken become very large.

The type of action applied in this study is the so-called *bus-bar switching* action. It essentially means that after a single electrical node can be split into 2 separate nodes, which in the power system context corresponds to using two separate bus-bars, the action is choosing to place a combination of elements on the two bus-bars. This is further described in section 2.1.2. In reality, a series of actions is necessary in order to carry out bus splitting and to obtain a required substation configuration. However, in this RL setup, we focus only on the final required substation configuration. Consequently, the actions of the agent are simply represented by choosing specific substation configurations directly. The overall number of configurations of a specific substation is dependent on the number of elements connected to the substation. Mathematically, this is the combinatorial problem of determining the number of ways elements can be placed on one bus-bar (and all other elements on the other bus-bar) and taking into account that the order of these elements does not matter. Not taking any constraints into account, the solution is directly related to the binomial coefficient

$$b(n,k) \equiv \frac{n!}{k!(n-k)!}\,, \tag{5.1}$$

which gives the number of possibilities to connect k out of n elements to a single bus-bar. That is, n is the total number of elements connected to a sub-station and consequently $n-k$ is the number of elements connected to the other bus-bar.

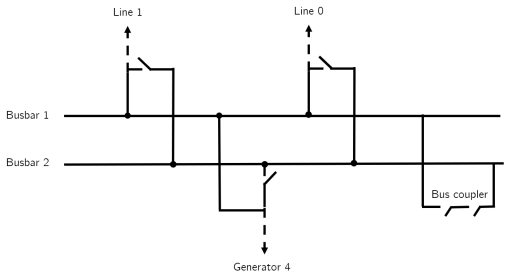
However not all actions are valid (i.e. consistent with a power grid), but the following constraints need to be satisfied:

1. **Minimum Element Constraint:** At a single bus-bar, there needs to be a minimum of two elements connected to it (or 0 elements).
2. **One-Line Constraint:** Out of the elements connected to a bus-bar, at least one of them has to be a line. Hence, this means that there can never be only non-line elements (generators and loads) connected to a bus-bar.
3. **Connex Constraint:** A subset of the power grid should not get isolated from the rest of the power grid due to any action. The grid should remain ”connex” at all times.

Note that constraints 1-2 are *local* in the sense that they only focus on the configuration of each substation on its own whereas constraint 3 is *non-local* since it is related to the overall grid topology induced by the configuration of 2 or more sub-stations.

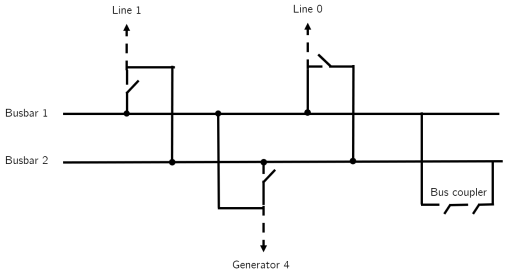
In figure 5.1, we can briefly see how configurations will be eliminated based on constraints, with respect to a single substation (substation 0). For substation zero which has a 3 elements connected, using equation 5.1,

we know that the total number of combinations possible are 8. We consider that bus-bars are indistinguishable (or symmetrical), thus, placing 3 elements on bus-bar 1 is symmetrical to placing the 3 elements on bus-bar 2. Thus, we reduce the number of configurations due to the symmetrical nature of the bus-bars, by halving the number of possible combinations. So, the number of configurations will be reduced to 4. Figure 5.1 shows these 4 configurations. First, in figure 5.1a, the reference configuration of the substation can be seen with all elements connected to bus-bar 1. Figures 5.1b, 5.1c and 5.1d depict the three other configurations for this substation (beside the reference configuration). Now, immediately, we see that in any of the other combinations, there ends up being only a single element connected to the other bus-bar. This directly violates the **Minimum Element Constraint**. In addition, figure 5.1d would also be eliminated due to the **One-Line Constraint**. This reduces the number of possible configurations from a total of 8 -> 4 (removing actions due to symmetrical bus-bars) -> 1 (removing actions for violating constraint no. 1). Thus, the single configuration that is allowed for substation 0 is the reference configuration (figure 5.1a). It is good to note at this point, that these figures are only representative and are largely simplified to show the difference in configurations. In this manner, the configurations are eliminated for each of the substations. In the following section, how the number of configurations for each substation can be calculated after incorporating the constraints is discussed.



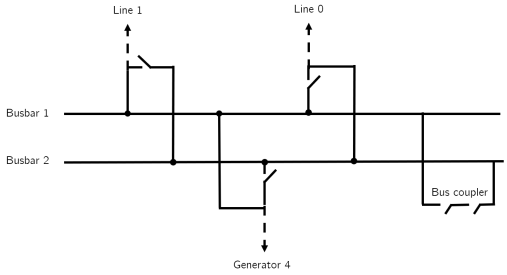
Reference configuration
(All elements connected to busbar 1)

(a) Reference configuration (configuration 0)



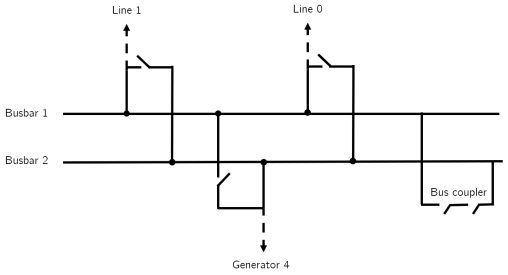
Configuration 1
(Line 0, Generator 4 at bus-bar 1;
Line 1 at bus-bar 2)

(b) Configuration 1 (Not a valid configuration)



Configuration 2
(Line 1, Generator 4 at bus-bar 1;
Line 0 at bus-bar 2)

(c) Configuration 2 (Not a valid configuration)



Configuration 3
(Line 0, Line 1 at bus-bar 1;
Generator 4 at bus-bar 2)

(d) Configuration 3 (Not a valid configuration)

Figure 5.1: All configuration for substation 0 *without* incorporating the constraints

5.1.1. Mathematical approach

For constructing the action space for the agent we focus only on constraints 1-2 in this study, as described above. The total number of configurations τ of a substation satisfying the first two constraints is given by

$$\tau = \alpha(n) - \beta(n) - \gamma(n') \, . \tag{5.2}$$

with

$$\alpha(n) \equiv \frac{\sum_{k=0}^{n} b(n,k)}{2} \tag{5.3}$$

$$\beta(n) \equiv \frac{\sum_{k\in\{1,n-1\}} b(n,k)}{2} \tag{5.4}$$

$$\gamma(n') \equiv \begin{cases} 0 & n' = 0,1 \\ \alpha(n') - \beta(n') & n' \geq 2 \end{cases} \tag{5.5}$$

Where:
n is the total number of elements connected to a substation
k is the number of elements connected to a single bus-bar
n' is the number of non-line elements connected to a substation.

The number of ways of choosing k out of n elements for each substation can be determined by summing the binomial coefficients. Since the bus-bars are indistinguishable, the configurations of one of the bus-bars are symmetrical to the other, so to remove duplication of configurations, the sum should be halved. This is because, for example, if there are a total of 5 elements at a sub-station, and 2 specific elements are chosen out of the 5 to be placed on one bus-bar, automatically the other 3 elements are to be placed on the other bus-bar. There is also the symmetrical configuration possible, namely, when choosing the other 3 elements, but this is already covered by the previous configuration when choosing 2 elements since the bus-bars are indistinguishable. The term that represents this is $\alpha(n)$.

Now, the constraints need to be taken into consideration. First, to incorporate the "Minimum Element" constraint which says there can never be only a single element at a bus-bar. Due to this constraint, in all the choices, anytime that 1 element or $n-1$ elements are chosen, these choices should be discarded. $\beta(n)$ is the term used to depict this constraint.

Next, the "One-Line" constraint states that a substation configuration has to have at least one line connected to it, thus, no configuration can have only non-line elements at a bus-bar. If a substation has only 1 non-line element connected to it, any resulting configuration will have this element in at a bus-bar along with the other line elements. If the substation has 0 non-line elements, the "One-Line" constraint will not pose any issues. So, for all substations being connected to 2 or more non-line elements, there exists one configuration each where a choice of only non-line elements is made. Thus, this configuration needs to be dropped. This is then once again represented as the number of ways the non-line elements can be chosen, i.e. $\alpha(n')$. But, to ensure that the "Minimum Element" constraint isn't violated again, the choices of 1 or n-1 non-line elements needs to be discarded from $\alpha(n')$. This is done by excluding $\beta(n')$ from it, resulting in the term $\gamma(n')$. Putting the constraints together, results in equation 5.2.

Using equation 5.2, the number of configurations for each substation can be calculated as seen in table 5.1. At each instant of time each substation always exhibits a specific configuration. Consequently, in terms of actions each number in Table 6.1 also includes one reference configuration action (namely the case when the default configuration of that substation is chosen). Hence, the total number of actions including only a single reference configuration action, which encapsulates one single reference configuration per substation results in a total number of configurations as: $112 - 14 + 1 = 99$. In this setup, at any given point of time, only one substation can be acted on. Thus, at a particular time instant, only one of these 112 substation configurations can be chosen. Thus by using each of these 112 configurations, we can traverse the entire action space.

Table 5.1: Possible configurations for each substation

Substation Number	Number of elements connected	Number of lines connected	Number of substation configurations
1	3	2	1
2	6	4	25
3	4	2	3
4	6	5	26
5	5	4	11
6	6	4	25
7	3	3	1
8	2	1	1
9	5	4	11
10	3	2	1
11	3	2	1
12	3	2	1
13	4	3	4
14	3	2	1
		Total number of Substation Configurations:	**112**

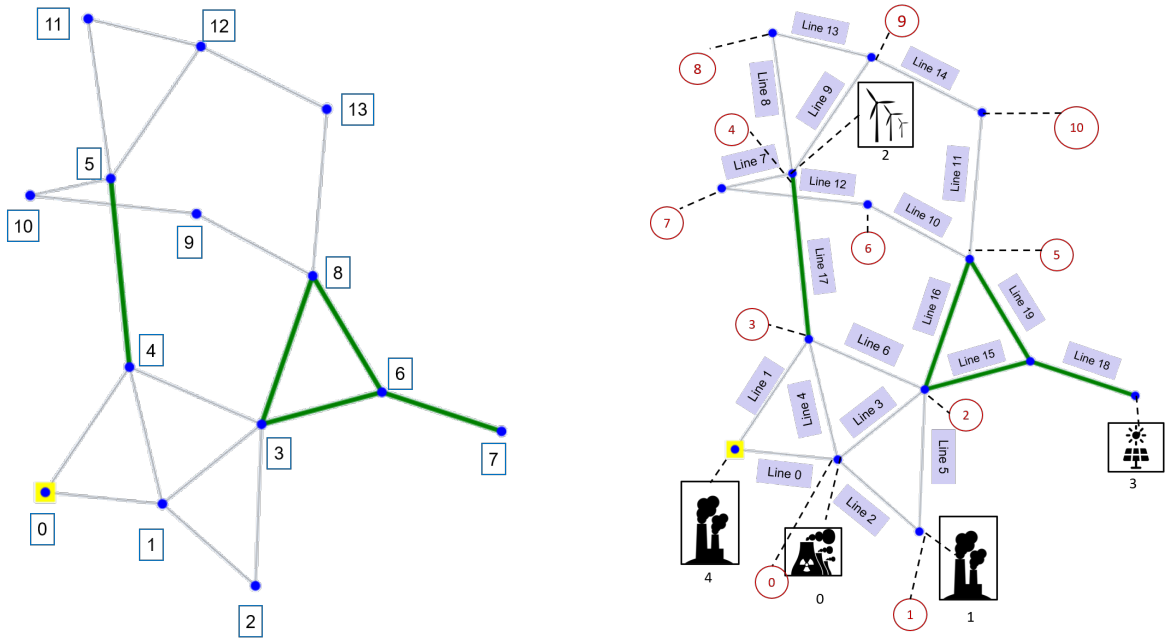


Figure 5.2: Reference Power Grid

The formulae developed in equations 5.2 to 5.5 depend on these particular constraints, but can be generalised for other networks of different sizes as well.

From table 5.1, it is also possible to compute the total number of topological states of the IEEE14 power grid subject to only bus-bar switching actions. For each single substation configuration, there are a large number of configurations possible at all the other substations. Thus, the total number of topological states is the product of all the individual substation configurations, which amounts to **23595000**. This is an upper-bound on the total number of states possible on this power grid, for bus-bar switching actions, keeping in mind the fact that the connex constraint is not considered here.

5.1.2. Implementation

In order to implement the above method and create a set of substation configurations which realise the physical constraints involved while choosing a configuration, the outline of the methodology used is visualised in figure 5.3.

Firstly, a list of all possible combinations of elements connected to a substation can be made with regard to a single bus-bar (out of two bus-bars) at a substation. This can be done by using equation 5.1. Using this equation, depending on the number of elements connected to the substation and the number of ways a subset of the elements can be chosen, a total number of possible configurations can be found. This results in what is more commonly known as the Pascal's triangle (figure 5.4). With reference to our grid model, using equation 5.1, a total number of all possible combinations of elements for a single substation can be found out by mapping n,k in figure 5.5. Further, the total sum of all combinations is calculated as seen here, to be **340**. It is important to note, that this is the total number of combinations of elements that can be chosen across all substation to create substation configurations *without* incorporating the physical grid constraints.

After finding the upper limit in terms of the total number of possible element combinations without incorporating constraints, duplicate configurations (due to symmetrical bus-bars) need to be removed, following which the combinations that violate the constraints needs to be removed (as per the flowchart in figure 5.3).

This will then result in the table as seen in table 5.1.

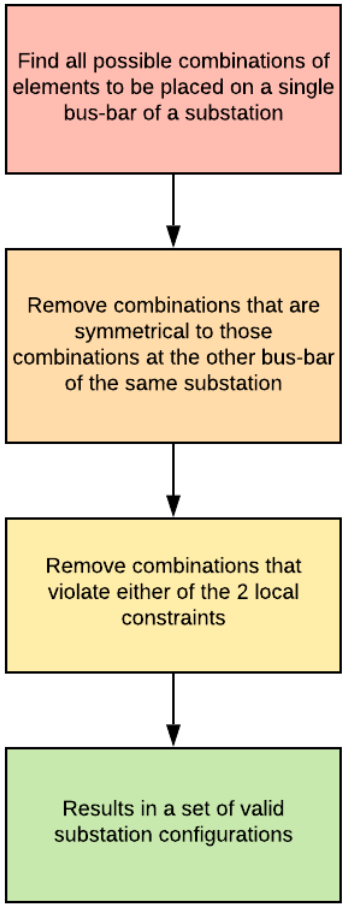


Figure 5.3: Steps involved in creating a valid set of substation configurations

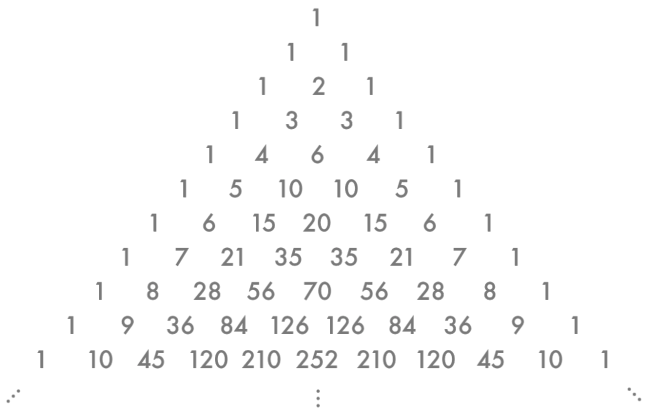


Figure 5.4: Pascal's triangle

Number of ways to choose elements to be placed on a bus-bar (k)

Number of elements connected to a substation (n)

	0	1	2	3	4	5	6	Total for a single substation	No. of substations with this no. of elements connected	Total number of combinatons
0	1							1	0	0
1	1	1						2	0	0
2	1	2	1					4	1	4
3	1	3	3	1				8	6	48
4	1	4	6	4	1			16	2	32
5	1	5	10	10	5	1		32	2	64
6	1	6	15	20	15	6	1	64	3	192
									Total:	**340**

Figure 5.5: Total number of element combinations *without* checking constraints

5.2. Training/Development Specifications

In order to develop the agent, a list of specifications (also referred to as parameters) can be utilised, and changed to observe differences in the working of the agent with respect to varying specifications. Below listed is an explanation of the different specifications used in this thesis. A brief list can also be seen in figure 5.2 that will then be used as the framework alongside the results of the agents. The specifications listed in violet are related to the power system aspect, whereas the the elements listed in green, are related to the the training and reinforcement learning aspect. In the following subsections, detailed descriptions of these specifications can be found, along with the different values that can be assigned to each specification.

Table 5.2: All specifications used for agent development

Parameters
Environment
Scenario(s)
Observation input
Action Space
Reward function
NN architecture
Learning Rate
Batch Size
Loss Function
Percentile for CEM
NN weights initialisation

5.2.1. Power System Specifications

Described here are the power system related specifications that can be assigned while training the agent.

1. **Environment**: As previously described in section 2.2.4, an environment is the platform using which an agent can take actions and train on. It is higher level representation of the world with which an agent will interact. In our setup, this environment is defined in the package Grid2Op (section 3.1.1) where the environment receive a chosen action from the agent and will then return a set of observations that the agent can use to perform the next action, at the next time-step. Grid2Op allows for many different environments that can be utilised which are created mainly for different L2PRN editions. The environment used here the simulated IEEE 14 bus network, (termed 'rte_case14_realistic' within Grid2Op). The only change made to this network are that the thermal limits are adapted as seen in section 4.1.

2. **Scenario(s)**: For training the agent, it should experience a large variety of scenarios with varying difficulty levels. In this thesis however, we focus on creating multiple variants of an agent that have each been exposed to a scenario with a different difficult level. Hence, for training the agents, a single scenario out of the 1000 available scenarios is used.

3. **Observation**: In chapter 3, a brief overview of how the observations can be visualised can be seen. These observations are available at every step of the 'game', hence, they are available for every time-step (5 minutes duration). The complete list of observations available in the framework are as described below, and the length of the observation is provided in (*corresponding brackets*):

    - Generators' active power: The active power produced by each of the generators at this particular timestamp. This is expressed in MW. (*number of generators, i.e. 5*)
    - Generators' reactive power: The reactive power produced by each of the generators at this particular timestamp. This is expressed in MVar. (*number of generators, i.e. 5*)
    - Generators' voltage: The voltage magnitude of the bus to which each of generator is connected to. This is expressed in kV. (*number of generators, i.e. 5*)
    - Loads' active power: The active power drawn by each load. This is expressed in MW. (*number of loads, i.e. 11*)
    - Loads' reactive power: The reactive power value of each of the loads. This is expressed in MVar. (*number of loads, i.e. 11*)
    - Loads' voltage: The voltage magnitude of the bus to which each load is connected. This is expressed in kV. (*number of loads, i.e. 11*)
    - Lines' active power: The active power flow at the origin and extremity of each of the lines. This is expressed in MW. (*number of lines, i.e. 20 ×2 (origin, extremity) = 40*)
    - Lines' reactive power: The reactive power flow at the origin and extremity of each of the lines. This is expressed in MVar. (*number of lines, i.e. 20 ×2 (origin, extremity) = 40*)
    - Lines' voltage: The voltage magnitude of the bus to which the origin as well as the extremity of each line is connected. This is expressed in kV. (*number of lines, i.e. 20 ×2 (origin, extremity) = 40*)
    - Lines' current: The current flow at the origin and extremity of each line. This is expressed in A. (*number of lines, i.e. 20 ×2 (origin, extremity) = 40*)
    - Lines' loading: This represents the present loading of each of the lines. This is also called $Rho(\rho)$ and it is defined as the observed current flow divided by the thermal limit of the line. It has no units. (*number of lines, i.e. 20*)
    - Topology: For each element (line origin, line extremity, load or generator), there is a corresponding number to represent which bus-bar this element is connected to at its substation at the present time-step. This will essentially be an array of 1s or 2s, depending on which bus-bar the element is connected to out of the two bus-bars at each substation. (*Total number of elements:* $5+11+40 = 56$)
    - Status of the line: Represents whether each line is connected (True) or disconnected (False). This is will be an array of Boolean values. (*number of lines: 20*)
    - Overload timesteps: Gives the number of time-steps since each line has been overloaded (if at all). (*number of lines: 20*)

    This overall list of observations results in a total size of 324 available observations. From this, either a subset of observations or all available observations can be used while training the agent.

4. **Action Space**: In section 5.1, the action space has been described in detail. In table 5.1, a consolidated list of number of possible actions substation-wise can be seen. There are a total of 112 actions possible. Either all 112 can serve as the action space for training, or else some subset of this list can be used.

5. **Reward Function**: Within the Grid2Op platform, multiple reward functions are available. A few are as follows:

    - Flat Reward: This reward function acts adds a flat value every time-step. There is no negative reward, only a positive reward every time-step that the agent survives.
    - Increasing Flat Reward: Here, the reward function returns a value depending on the count of time-steps that the agent has successfully managed to operate the grid. It adds a constant value for each time-step.

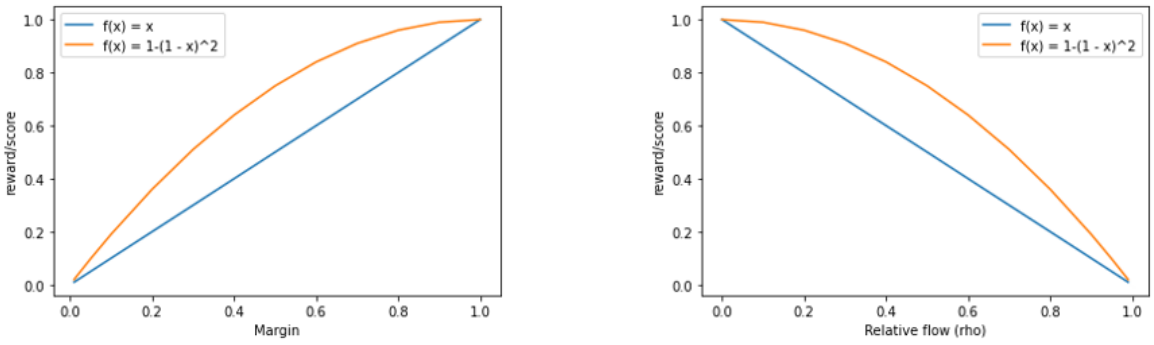
- Close to overflow Reward: This reward checks for all the lines close to overflowing. It then returns the maximum allowed reward if there is no overloading in the lines, and a minimum allowed reward if more than one line is close to overloading. If one line is close to overloading, the mean reward is returned for that time-step.
- L2PRN Reward / Residual Capacity Reward: This is the reward used for the first L2PRN competition. The goal with this reward is to return the overloading of the grid. This is the standard reward used in the implementation of an agent in this thesis. While returning a reward, the function used here, compares the $\rho(rho)$ value or relative flow through each of the lines with 1 (indicating that the line is at 100 % capacity [8]). The minimum of these 2 values are returned, as seen in equation 5.6. Using this, margins or the 'residual capacity' of each of the lines are calculated. The $Margin_l$ is essentially $1-output_l$. Margins for each line are calculated as seen in equation 5.7. Finally, a summation of all this value for all lines in the network are returned as the reward for this particular time-step (equation 5.8).

$$output_l(t) = min(1, \frac{i_l(t)}{Imax_l(t)}), \forall l \in L \tag{5.6}$$

$$Margin_l(t) = max(0, 1 - \frac{i_l(t)}{Imax_l(t)}), \forall l \in L \tag{5.7}$$

$$Reward(t) = \sum_{l=0}^{N_l-1} f(Margin_l(t)) \tag{5.8}$$

Where:
$Margin_l$ is the residual capacity for each line, l
i_l is current flowing through the line, l
$Imax_l$ is the maximum allowed current through line l or the thermal limit of the line,
L is the total set of all lines in the network,
N_l is the total number of lines in the set L, the lines are indexed from 0 to N_l - 1.
In equation 5.8, the function $f(x)$ is chosen to be $1-(1-x)^2$. This is done, in order to encapsulate the concept that: higher reward will be given for reducing the loading of lines that are closer to their limit, than the reward given for reducing the loading of lines that already have low loading. This can also be seen in figure 5.6 which compares the reward and margin (figure 5.6a) and also the reward and relative flow (figure 5.6b) for the lines with two different functions used. The slope of the line for $f(x) = x$ is constant for any value of rho. But, for $f(x) = 1-(1-x)^2$, the slope is larger for a change of loading from a large value of rho (say, 0.9 or 1) to a value lower than that, implying that the reward assigned is larger. Hence, for a line with already low rho (say, 0.3), the reward for reducing this to an even lower value of rho will be lesser than the previous case.



(a) Reward assigned vs margin

(b) Reward assigned vs relative flow (ρ)

Figure 5.6: Reward assigning for 2 different functions

5.2.2. Algorithm Specifications

Described here are the reinforcement learning related specifications that can be assigned while training the agent.

1. **NN Architecture**: As discussed in section 2.2.2, the machine learning model used for training the agent is a neural network. The architecture of this neural network can be changed in order to test the impact

of varying architectures on the accuracy of the model. As discussed in section 2.2.2, the input layer of the NN will have the size of the corresponding observation space as described above in section 5.2.1. Hence, the default size is 324. There are 2 hidden layers in this NN, each with a size of 300 neurons. Finally, the output layer will correspond to the size of the action space, and the overall size of the action space is 112. A reference architecture is provided in figure 5.7.

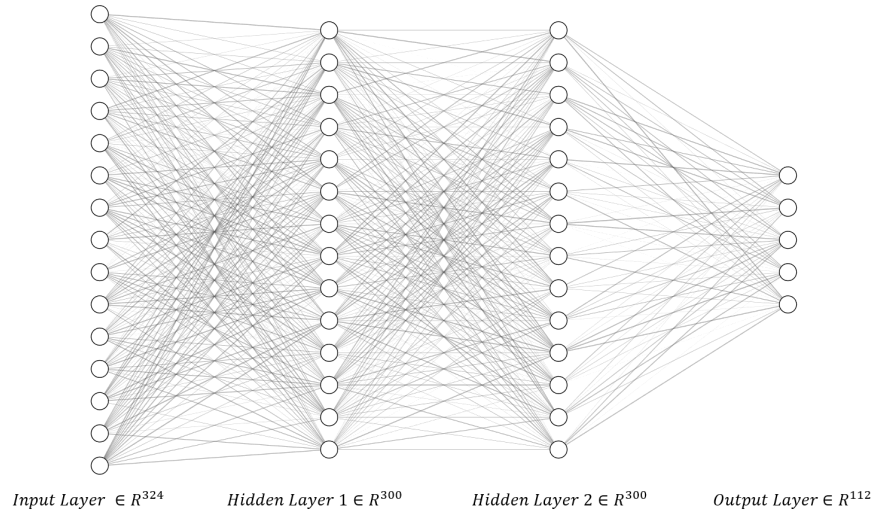


Figure 5.7: The architecture of the Neural Network

2. **Learning Rate**:The learning rate is a hyper-parameter of the NN that controls how much to change the model in response to the estimated error each time the model weights are updated. The learning rate is generally a value in the range of 0 - 1. Choosing a value on the lower end of this range can result in a long training process that may get stuck. On the other hand, a value too large may result in learning a sub-optimal set of weights too fast [51].

3. **NN weight initialisation**: At the start of training of the NN, the weights need to be initialised. During the course of training the NN, the weights get updated in order to achieve the optimal set of weights. Typically, the weights are initialised to small random numbers. However, some Python packages have the capability of initialising weights differently, such as [52]:

   - Zeros: Generates arrays of weights initialised to 0.
   - Ones: Generates arrays of weights initialised to 1.
   - RandomNormal: Generates arrays of weights initialised with a normal distribution.
   - RandomUniform: Generates arrays of weights initialised with a uniform distribution.
   - Orthogonal: Generates a random orthogonal matrix of weights.
   - Identity: Generates the identity matrix for the weights.

   In this thesis, the standard method of initialisation of weights is the random normal.

4. **Batch Size**: The batch size is a hyper-parameter that defines the number of samples that the model goes through before the internal model parameters are updated[53].

5. **Loss Function**: The problem of learning of a NN can be treated as an optimisation problem or a function approximation where an algorithm is used to navigate the space of possible sets of weights that can be used in the model in order to make good predictions. Training of a NN is usually done by using the stochastic gradient descent optimisation algorithm and weights are updated using the back-propagation of error algorithm. The “gradient” used here, refers to an error gradient. The NN with its set of assigned weights is used to make predictions and the error is calculated for these predictions. The gradient descent

algorithm changes the weights so that the next evaluation reduces the error. This implies that the optimisation algorithm is navigating down the gradient (or slope) of error. The function used to find an optimal or candidate solution (i.e. a set of weights) is referred to as the objective function. The goal is generally to maximise or minimise the objective function. In NNs, the goal is to minimise the error. The objective function described here is also called the loss function and the value calculated by this loss function is known as the “loss.”The choice of the loss function is dependent on the configuration of the output layer of the NN. Another factor is the type of problem we have, such as: classification or regression. Two most common loss functions are [54]:

- Mean Squared Error (MSE): This is calculated as the average of the squared differences between the predicted and actual values. The result will always be positive and a perfect value is 0.0.
- Cross Entropy Loss: Also known as Logarithmic Loss, here each predicted probability is compared to the optimal output and a score is calculated that penalises the probability based on the difference from the expected value. The penalty is logarithmic, assigning smaller scores for smaller differences and vice versa. Cross-entropy loss is also minimised, meaning that smaller values of this loss depict a better performing model. A model that predicts perfect probabilities has a cross entropy or log loss of 0.0.

  In this study, the loss function used is Cross Entropy loss. This cross entropy loss is calculated as follows:

$$H(P,Q) = -\sum P(x) * log(Q(x)) \tag{5.9}$$

  Where H(P,Q) is the cross entropy loss function
  P(x) is the probability distribution of the output of the model
  Q(x) is the distribution of the distribution of the output for the played episodes.

6. **Percentile for CEM**: The CE method is the algorithm used to create this RL-based agent, as discussed in 2.2.7. As discussed here, a percentile is calculated for the reward for each batch of episodes. Further, only episodes above this reward value are used for training. This percentile value is another specification that can be varied in order to implement the RL agent. This becomes clearer from the following explanation of the CE method algorithm.

5.3. CEM Implementation

The algorithm used to develop an RL-based agent in this thesis is called the ‘Cross Entropy Method’. The basis of this algorithm is explained in section 2.2.7. The algorithm is implemented in the following manner, as seen in figure 5.8. To start with, in the environment a starting policy is provided. The internal model parameters of the neural network describe the policy; at the beginning of the training process, these parameters are randomly defined. Using this kind of stochastic policy, ‘N’ episodes are played to create a single batch. Thus, such a stochastic policy can produce randomised episodes for a single fixed scenario. Each episode is played in the following manner (also in figure 5.9):

- Calling the required scenario into the environment
- Obtaining the first time-step of observations from the environment
- Feed the observations to the model to get a probability distribution of output actions.
- Using this probability distribution, choose an action
- Provide this action to the environment to obtain the next observation and reward
- Repeat until the episode ends.

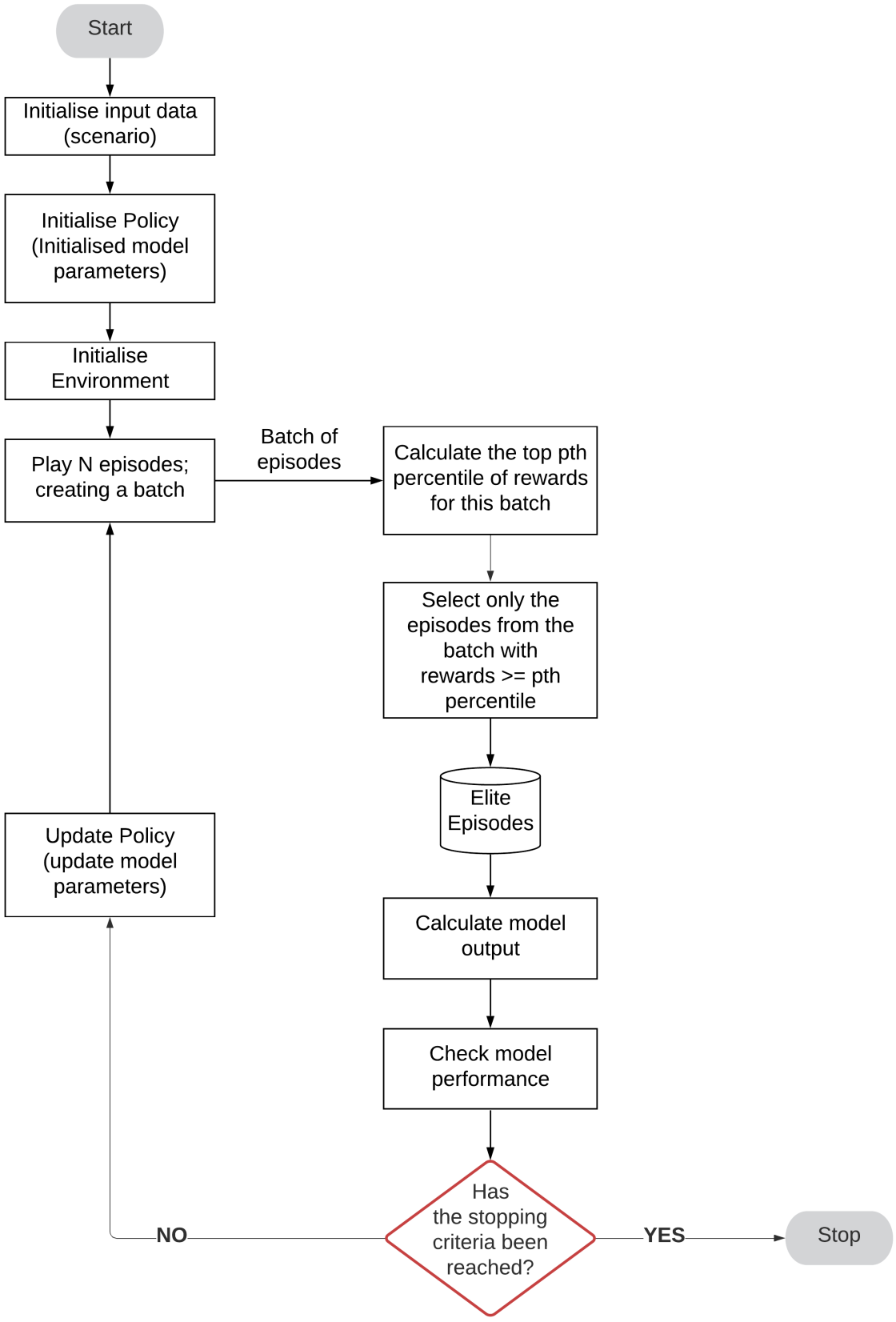


Figure 5.8: Implementation of the CEM algorithm

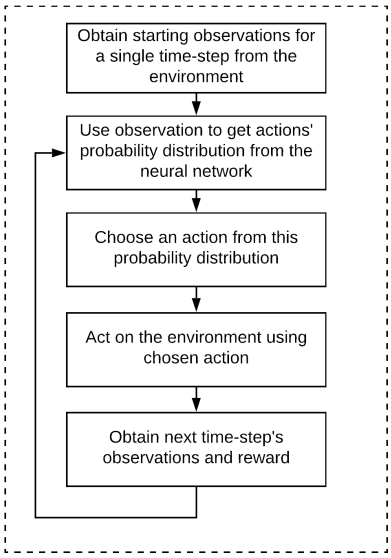


Figure 5.9: Playing an episode

This batch is then sorted in a descending manner by each episode's reward. Using a percentile (p) input, we then calculate the top p^{th} percentile of episodes, which are called the 'elite' episodes. Hence, any episodes below this threshold will not be considered further for training. An important point to note here, is that the reward for an episode is the sum of all rewards for every time-step in that episode. While equation 5.8 is used to calculate the reward for a single time-step, this is then summed for an entire episode, as seen below in equation 5.10. In this CEM implementation, the reward that is monitored and using which episodes are considered elite or not, is the total episode reward (or simply, an episode reward).

$$\text{Episode Reward} = \sum_{k=0}^{N_t-1} Reward(k) \tag{5.10}$$

Where:
N_t = Number of time-steps in the episode
$Reward(k)$ is the reward for a single time-step

These 'elite' episodes are then used for training. The training takes place by calculating the loss between the output of the model (neural network) and the mean output of the elite episodes. To elaborate on how this loss is calculated, a list of the observations of all elite episodes, along with a list of the actions taken for all elite episodes are used. The observations are used as input to the neural network and actions are obtained as the output of the neural network. The cross entropy between the actions in the neural network's output and the actions taken in the elite episodes is calculated.

Using this, the performance of the model is checked, by checking the mean reward of the batch or boundary reward of a batch. The boundary reward is the p^{th} percentile reward for a particular batch. Then, we check if a certain stopping criteria has been reached, for example: 'has reward mean become x?' or 'has reward boundary reached y?' If this stopping criteria has been reached, then the model is considered ready for use or deployment. If not, the iterative process restarts by creating a new batch of episodes that are created with the newly updated model parameters (hence, an updated policy). This is done over and over again, till the required stopping criteria is reached.

5.3.1. Training the CEM-RL agent

In total, 2 models are developed. These two models are trained on two separate scenarios one from the Easy category, and one from the Medium category respectively. When played by the RC agent, the easy scenario is one that resulted in a successful episode and the medium scenario resulted in a failure episode. This is also elaborated in section 4.4. For each agent, the specifications as provided in table 5.2 will be seen along with two metrics that are monitored during the time of training: the reward and the loss.

Each point in figures 5.10 and 5.11 depict a single scenario. Both these figures are described in sections 4.3 and 4.4 using figures 4.10 and 4.12 respectively. In figure 5.10, the characteristics (the maximum RES and

maximum conventional power generation) of the two scenarios which are used to train and develop the two models respectively can be seen by the two yellow squares. The success and failures of episodes shown here correspond to the success and failure episodes of the RC agent.

In figure 5.11, the factor used to define the difficulty level of episodes, the loading of line 11, can be seen for the two scenarios used for training. The choice of these scenarios, is to ensure that scenarios for training are chosen from two different levels of difficulty, namely, easy and medium level. Thus, this is visible, in figure 5.11, the different levels of difficulty are highlighted, to indicate that the scenarios chosen are from two different levels of difficulty.

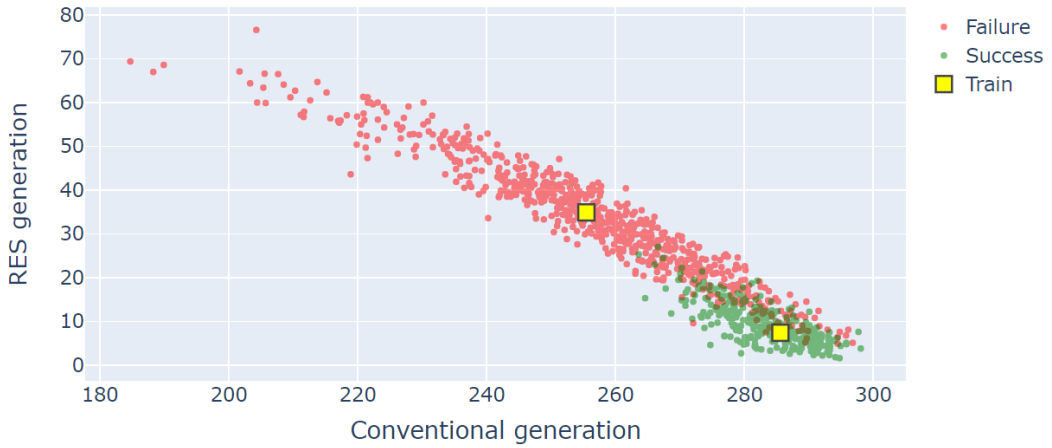


Figure 5.10: Scenarios chosen for training the 2 agents (Comparing maximum generation values)

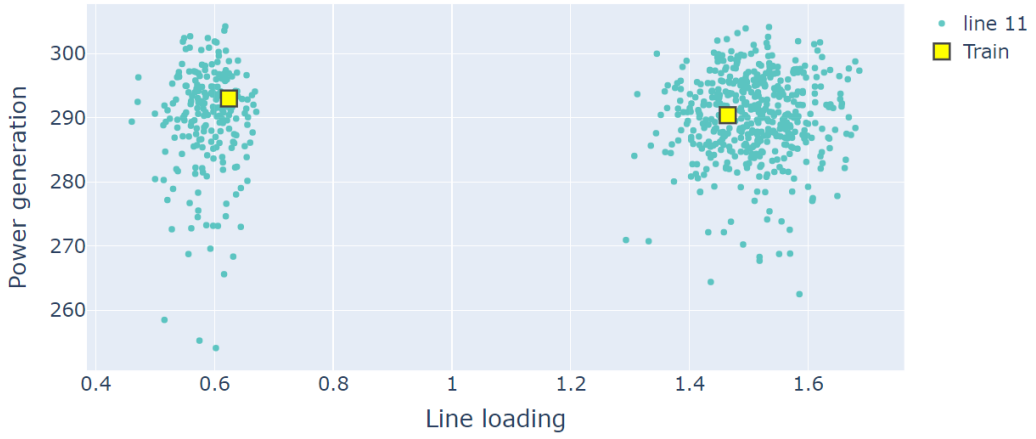


Figure 5.11: Scenarios chosen for training the 2 agents (Comparing generation with lines 11's loading

5.3.2. Agent 1: The *Easy* Agent

The specifications used for training this agent are seen in table 5.3. Agent 1 is trained on a scenario which falls in the 'Easy' category and hence is hereafter also called the Easy Agent.

Table 5.3: Training specifications for Agent 1

Category	Agent 2 Specifications
Environment	IEEE 14 bus network
Scenario(s)	1 scenario (Easy)
Observation input	Default set (size: 324)
Action space	All available actions (size: 112)
Reward function	L2PRN reward
NN architecture	Input layer + 2 hidden layers + 1 output layer
Learning rate	0.0001
Batch size	100
Loss function	Cross-Entropy loss
Percentile for CEM	95%
NN weights initialisation	Random - Normal

For Easy Agent, we see the corresponding loss and reward values for each batch against the number of batches in figure 5.12. We see that for the agent to reach stable reward and loss values, it takes over 80 batches (or 80 iterations). Reward mean as seen here is the mean reward of all 5 elite episodes, and reward boundary is the reward for the episode that had the p^{th} percentile reward. During the course of training, the goal is to the ensure that such a policy is learnt or developed by the model such that it continuously churns out episodes with high reward, and hence, 'successful episodes'. The stopping criteria used here is if the mean reward of the batch has a difference of less than 5% with the reward boundary consecutively for three batches. This is to ensure that the a large majority of episodes have a reward close to the reward boundary itself, implying that there are more episodes with higher rewards.

We also note the reward for the RC agent for this same particular scenario as a dotted line in this figure. Using this, we can highlight the fact that the reward boundary has reached the RC agent's reward. Further, the fact that the mean reward also falls within a 5% range of this boundary, implies that the majority (if not all) of 'elite episodes' have rewards in a 5% range of the RC agent's reward for this scenario.

Thus, after these 80 batches, we can now consider the agent to be trained and that the policy learnt by this last batch is the policy that will be used by this agent.

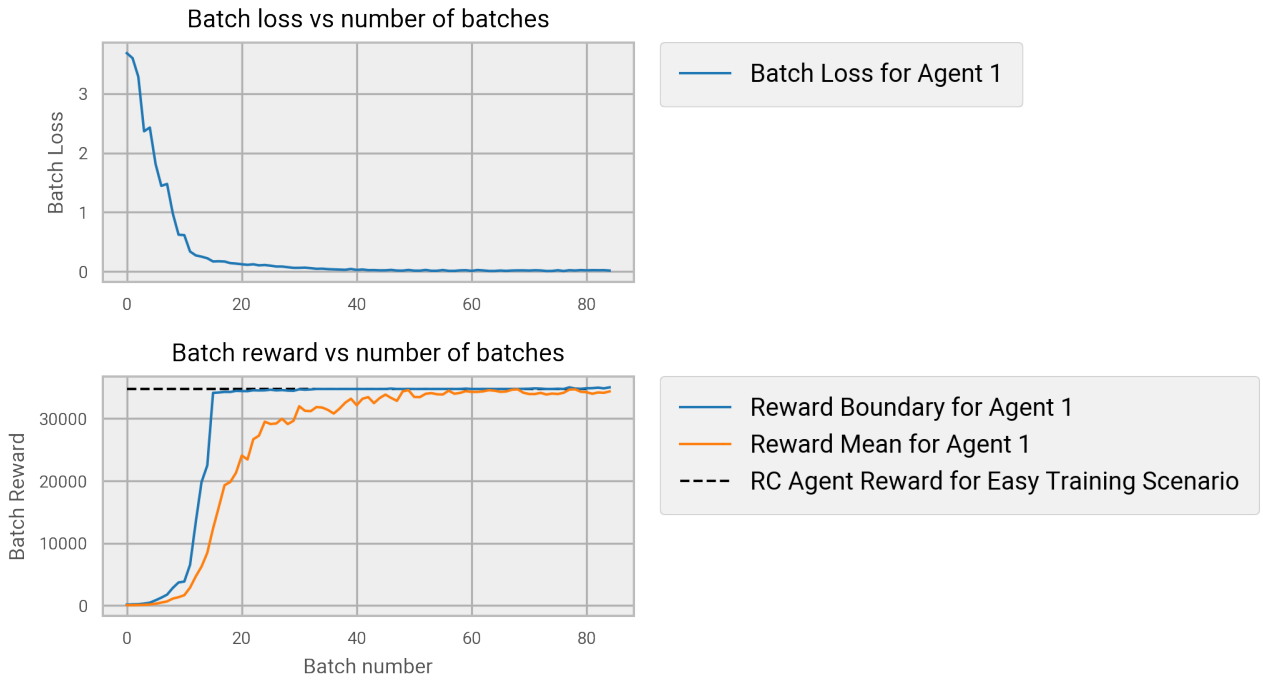


Figure 5.12: Training Results for Agent 1

In order to understand the policy that has been learnt better and to highlight how exactly the network is changing for this particular policy, this policy is used to play a 1000 episodes for this easy scenario, more

explicitly, the easy scenario that this agent was trained on. All 1000 episodes played can be summarised as follows in figure 5.13. 96.0 % of the episodes are successful episodes, that is, the agent operated the network successfully for a duration of 1 week. Of this, 70.2 % of the time, the agent behaved like the RC agent thus resulting in a reward same as that of the RC agent. The rest 12.5 % of the episodes, the agent took actions that resulted in better topological configurations and had better performance than the RC agent, that is, higher reward. 4% of the episodes resulted in failures.

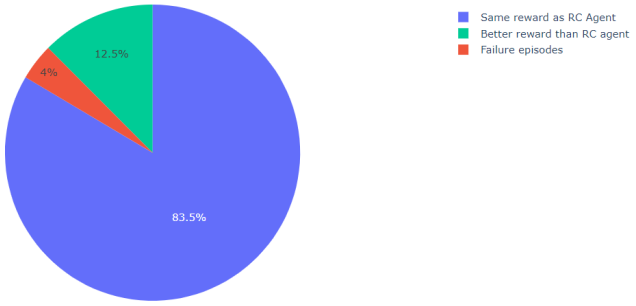


Figure 5.13: Split-up of the 1000 episodes played by Easy Agent for the corresponding easy training scenario

In figure 5.14, a time-series depicting the changes of topology in the episode with the highest reward out of the 1000 episodes is seen. Here, the difference (or distance) from the reference topology that the network starts in, (in terms of number of elements changed) is shown. This helps highlight the number of times a change in topology is seen, and also how far the new topology moves away (or not) from the reference topology. Thus, in this episode, the network topology changed at just under the 20^{th} time-step by changing the configuration of 2 elements in the network. After this point, there are no further actions taken that further increased or decreased the difference from the reference topology.

In figure 5.15, a similar plot for this episode is seen, but for each substation. Thus, this helps highlight the substation at which the action that caused to deviate from the reference topology (in terms of number of elements). For this episode, it is visible that this change occurs due to action at substation 3.

In both these figures, the dotted black line depicts the duration that the RC episode ran for the corresponding scenario. Since this scenario is from the easy category, it implies that the corresponding RC episode is a successful episode, thus the length of the RC episode is a week (2016 time-steps).

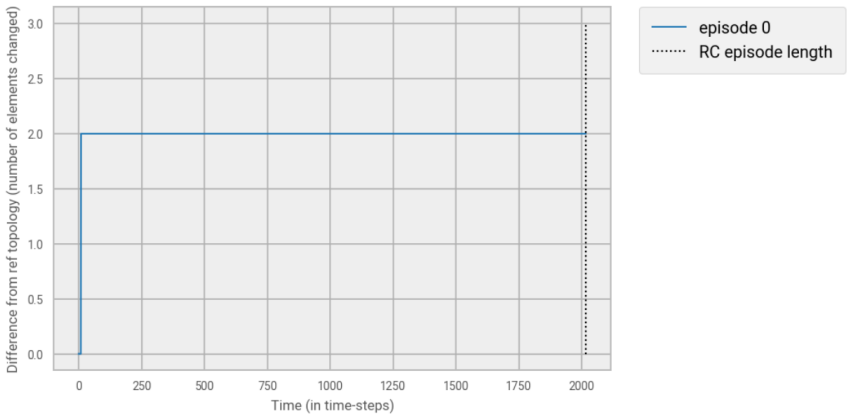


Figure 5.14: Best episode's change in topology time-series

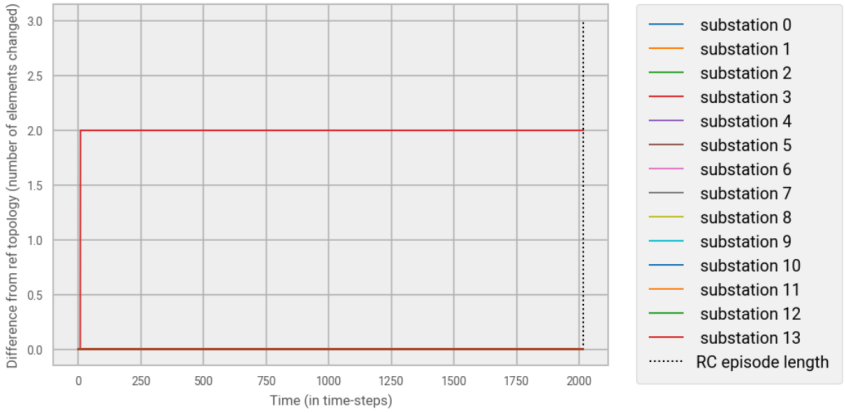


Figure 5.15: Best episode change in topology time-series (substation-wise)

To show all the successful episodes from the 1000 episodes played by this agent, aggregated views are presented. Upon analysing the topology time-series for multiple episodes, the main distinguishing factors for the episodes are found to be:

- The first action that is taken to produce a new topology different from the reference topology
- The number of actions taken in an episode
- The time-step when the first action is taken.

It is good to note that topology changes as mentioned here refer to the changes in topology by calculating the number of elements whose connections change with respect to the way the elements are connected in the reference topology. Different actions taken result in different distances (also referred to as differences) from the reference topology as shown in figure 5.14. There is a possibility of two actions making the same number of element changes, hence, 'side-stepping' in the difference, which will then go unnoticed. For example, an action (or rather, a chosen configuration) may result in 2 elements whose connections are changed with respect to the reference topology. A configuration chosen at a later time-step that puts one element back to the way it is connected in the reference topology and changes 2 other elements, the net change of elements is still 2, though, in reality, 3 elements have been changed. This is not considered mainly because this happens rarely (if at all) and hence, the substation-wise topological time-series (like in figure 5.15) are a sufficient representation of the changes in topology that occur.

In figure 5.17, we see all the successful episodes out of the 1000 episodes, that have rewards higher than the corresponding RC agent's reward. While there are certain episodes that took other actions first, the most common behaviour seen is to take a particular action (number 50) at substation 3. We see a trend for the episodes that use this behaviour, such that the reward largely depends on the time-step of when the action is taken. The earlier the action is taken, the larger the reward. This larger reward implies that the grid has higher residual network capacity from an earlier point of time in the episode rather than later. Hence, we see that the better episodes make network changes early on in the episode and retain the network in this configuration for the rest of the duration of the episode, resulting in a 'successful' episode.

Besides these 1000 episodes that are essentially created in a probabilistic manner, a single episode is played using a deterministic policy. In this manner, to create episodes using such a policy, instead of choosing an action from a probability distribution, the action with the highest probability is chosen at every time-step instead. The difference from the probabilistic way of creating an episode is highlighted in figure 5.16. Episodes created using this deterministic policy are hereon also referred deterministic episodes.

By using this deterministic policy, it resulted in an episode that did not take any actions, hence, stayed in the reference configuration and consequently having the same reward as that of the RC episode for this scenario. This is also marked in figure 5.17 by the pink square.

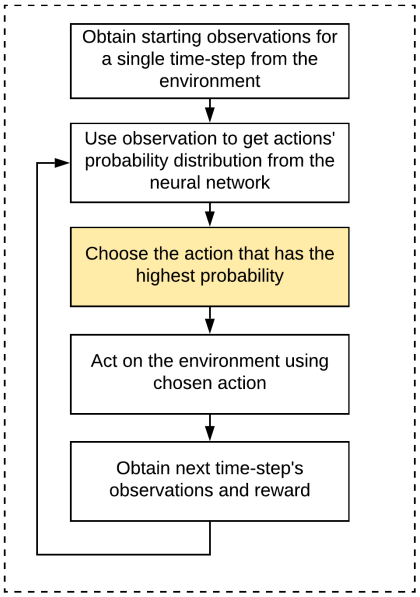


Figure 5.16: Playing an episode in a deterministic manner

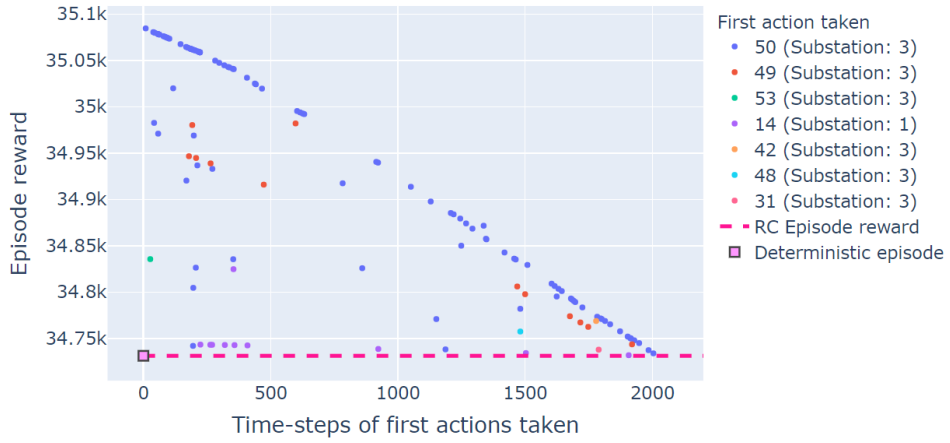


Figure 5.17: Episode rewards compared with the time-step of first action (highlighting the first action taken)

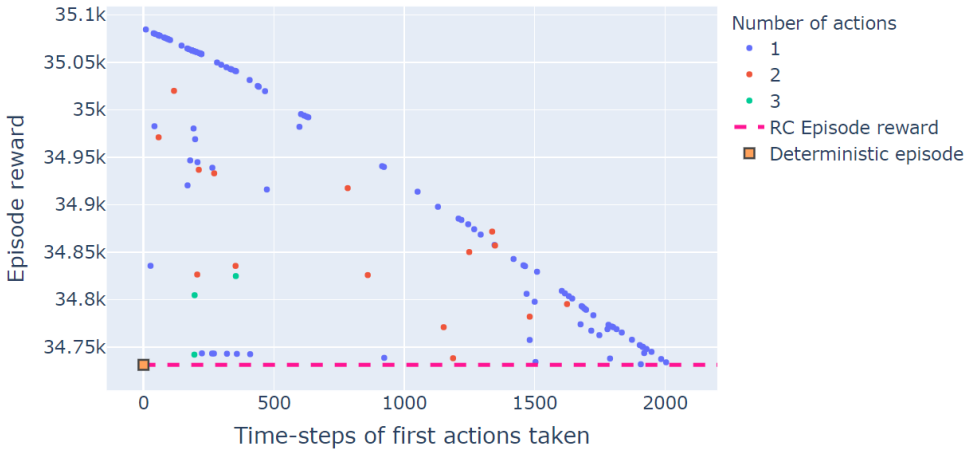


Figure 5.18: Episode rewards compared with the time-step of first action (highlighting the number of actions taken in the episode)

Similarly, the other parameter that can be indicated in all the successful episodes is the number of topological changes (or actions) that took place for all of them. This is highlighted in figure 5.18, where we once again see that while there are episodes where 2 or more actions are taken, the majority of them have only a single action taken.

In table 5.4, the actions along with their descriptions are provided for Easy Agent are provided. Out of the 7 actions that are chosen by this agent (out of 112), 6 of these actions are from a single substation: substation number 3.

Table 5.4: Preferred actions and corresponding substations for Easy Agent

	Easy Agent	
Substations	**Action number**	**Action description**
1	14	Bus-bar 1: Load 0, Generator 0, Line 3, Line 4 Bus-bar 2: Line 0, Line 2
3	31	Bus-bar 1: Load 2, Line 6, Line 3 Bus-bar 2: Line 15, Line 16, Line 5
	42	Bus-bar 1: Load 2, Line 6, Line 16, Line 3 Bus-bar 2: Line 15, Line 5
	48	Bus-bar 1: Load 2, Line 16, Line 3, Line 5 Bus-bar 2: Line 6, Line 15
	49	Bus-bar 1: Line 3, Line 6, Line 15, Line 16 Bus-bar 2: Load 2, Line 5
	50	Bus-bar 1: Line 6, Line 15, Line 16, Line 5 Bus-bar 2: Load 2, line 3
	53	Bus-bar 1: Line 3, Line 5, Line 15, Line 16 Bus-bar 2: Line 6, Load 2

Thus, all the actions preferred by this agent that resulted in successful episodes are seen in this table. The episodes that resulted in failures all took actions that outside of this list of actions and these actions and ended up as failures.

5.3.3. Agent 2: The *Medium* Agent

The specifications for training this agent are as follows in table 5.5. Agent 2 is trained using a scenario which falls in the 'Medium' category, and hence hereafter known as the Medium Agent.

Table 5.5: Training specifications for Agent 2

Category	**Agent 2 Specifications**
Environment	IEEE 14 bus network
Scenario(s)	1 scenario (Medium)
Observation input	Default set (size: 324)
Action space	All available actions (size: 112)
Reward function	L2PRN reward
NN architecture	Input layer + 2 hidden layers + 1 output layer
Learning rate	0.0001
Batch size	100
Loss function	Cross-Entropy loss
Percentile for CEM	95%
NN weights initialisation	Random - Normal

Similar to Easy Agent, for Medium Agent as well, we can also see the corresponding loss and reward values for each batch against number of batches in figure 5.19. Compared to Easy Agent, this agent takes a larger number of batches before stabilising. It takes over 220 batches before the reward values stabilise. As described previously, the mean reward for all 5 elite episodes, and reward boundary is the reward for the episode that had the p^{th} percentile reward. During the course of training, the goal is to the ensure that such a policy is learnt or developed by the model such that it continuously churns out episodes with high reward, and hence, 'successful episodes'. The stopping criteria used here is again to ensure the mean reward of the batch is within a 5% range of the reward boundary for these batches consecutively for three batches. This is to ensure that the a large

majority of episodes have a reward close to the reward boundary itself, implying that there are more episodes with higher rewards. The black dotted line in the figure represents the RC agent's reward for the corresponding medium scenario that it was trained on. Since it is a medium scenario, it implies that the RC episode resulting from this scenario was a failure. Since this means that the episode ended in lesser than 1 week, the reward accumulated by this time is also quite low.

Thus after these 220 odd batches, the agent is now considered to be trained and the policy learnt by this last batch is the policy that will be used by this agent when utilised.

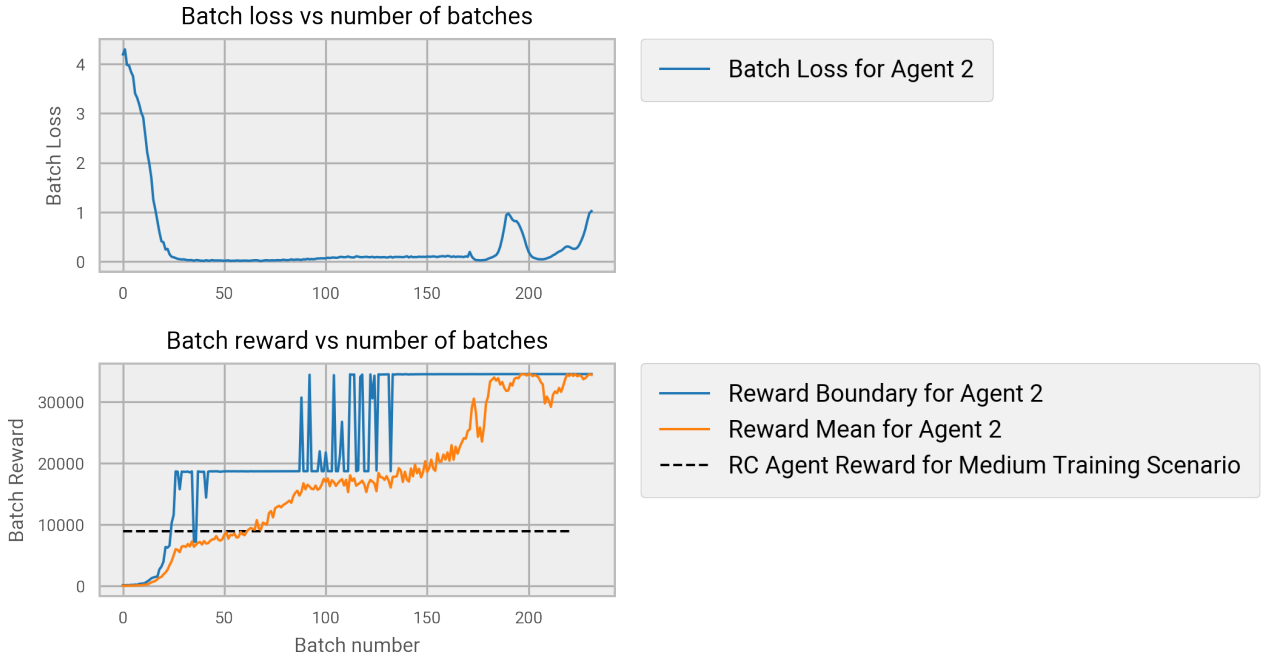


Figure 5.19: Training Results for Agent 2

To understand the policy better, we once again understand how the network changes for the particular policy by playing a 1000 episodes for the same 'medium' scenario which is used for its training. All 1000 episodes played in a probabilistic manner can be summarised as seen in figure 5.20. Since, here, the RC episode itself is one that is a failure, the categories shown here are only success and failure episodes. Thus, we see that of the 1000 episodes, 98.4 % of them resulted in successful episodes, and only 1.6 % are failure episodes.

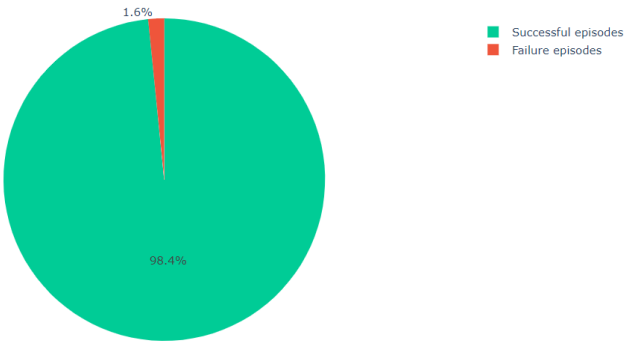


Figure 5.20: Split-up of the 1000 episodes played by Medium Agent for its corresponding medium training scenario

In figure 5.21, the episode with the highest reward out of the 1000 episodes is seen. Here, the difference (or distance) from the reference topology that the network starts in, (in terms of number of elements changed) is seen again. With the Medium Agent, we see that in this episode, the network topology undergoes multiple changes very early on in the episode; within the first 6 time-steps. The changes that happen are clearer in the

inset graph in figure 5.21. These changes result in a total of 7 elements' positions being changed compared to the reference topology. After this point, there is a change back and forth for a single element at past the 1500^{th} time-step.

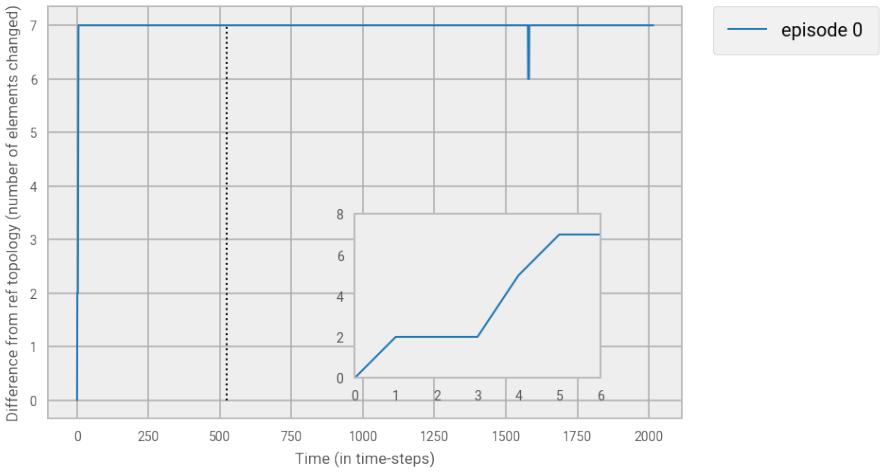


Figure 5.21: Best episode's topological time-series

For a more detailed view, figure 5.15 is provided which is a similar plot for this episode is seen, but for each substation. Thus, this helps highlight the substation at which the action that caused to deviate from the reference topology (in terms of number of elements). For this episode, it is visible that this changes occur at multiple substations, namely, substation 3, 4 and 8, as can be seen from the inset graph visible.

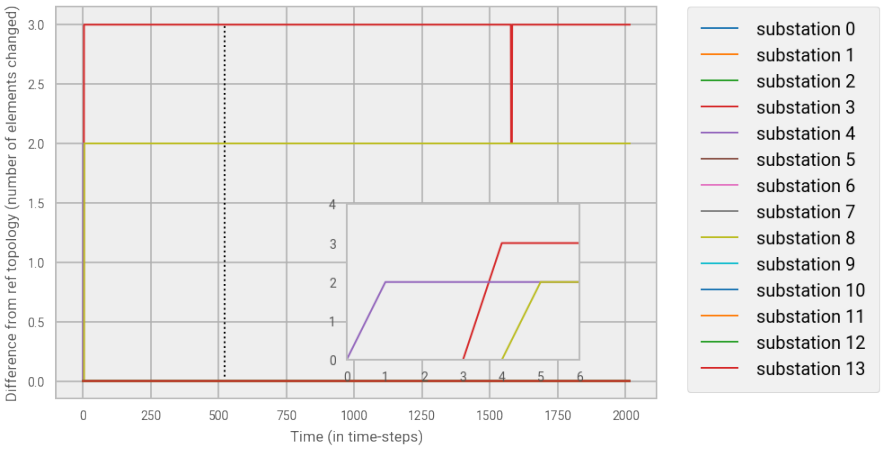


Figure 5.22: Best episode (substation-wise)

In both the figures 5.21 and 5.22, the vertical dotted line seen is the time-step at which the RC episode ended. This helps show that this Medium Agent takes actions that result in shifting the position of 7 elements before the time-step where the RC episode ended, which is what ensures the agent's survival for the same corresponding episode.

An aggregation of all the 1000 episodes played with the Medium Agent for corresponding scenario can be seen in the following figures. Unlike for the Easy Agent, where the factor which affected the reward the most is the time-step of first action taken, here the combination of factors is more evident. In figure 5.23, we can see each episode's reward vs the number of actions taken in an episode (in a zoomed in manner). There is a variation in the episode rewards even if the number of actions taken in each episode are the same. The rewards also vary for different substations where actions are taken.

In figure 5.24, the same plot is provided but here, instead the time-step of first actions is highlighted. This shows that the Medium Agent always takes actions very early on in the episode, in either the first or the second time-step, unlike the Easy Agent which takes its first action at any point in time of an episode.

Figure 5.23: Episode rewards compared with the number of actions taken in each episode (highlighting the first action taken)

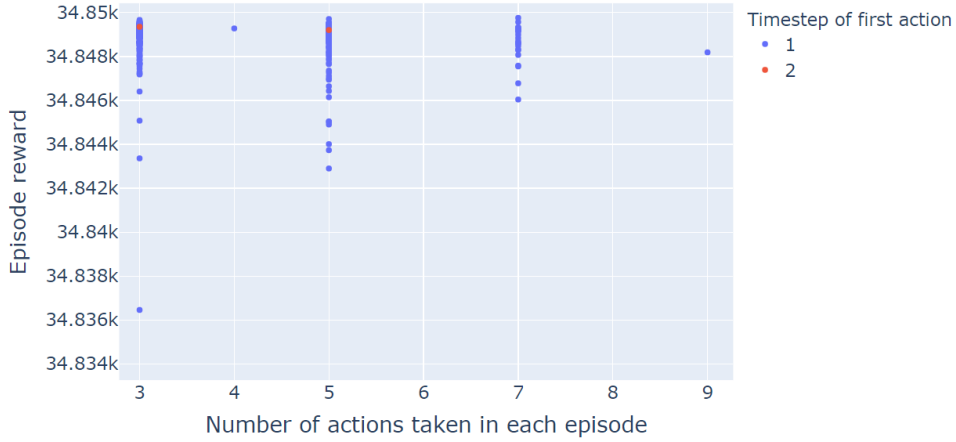


Figure 5.24: Episode rewards compared with number of actions taken in each episode (highlighting the time-step of the first action taken)

In these figures, the RC episode's reward is not shown since we know that the for a scenario from the 'Medium' category, the corresponding RC episode would be a failure episode, implying that it would end before the 7 day mark, and hence have a much lower reward not in the range of the rewards shown in these figures. The deterministic episode played with the Medium Agent made some changes in the topology but this resulted in a failure episode and is not shown in this figure.

An important point here is, the distinguishing factors listed above (the first action, the number of actions, the time-step) are not always a sufficient set of factors to explain all the episodes. As seen in these figures, there are episodes where the agent takes a large number of actions, 5 or more. The difference in reward can depend on other factors as well, such as the time-step of when each of these other actions are taken, which actions they are and so on. This is because, each action and time-step of when the action is taken, can result in a new set of observations, and thus tweaking the reward. Nevertheless, the factors used here to provide an overview and highlight the variety in successful episodes.

In table 5.6, a table is provided describing the preferred actions for the Medium Agent. For this agent, 4 actions are listed, of which two are from the same substation, substation 3.

Hence, the successful episodes all prefer actions from this table. The 1.6% of episodes that ended up as failure episodes either didn't make topology changes at all and hence, ended up failing.

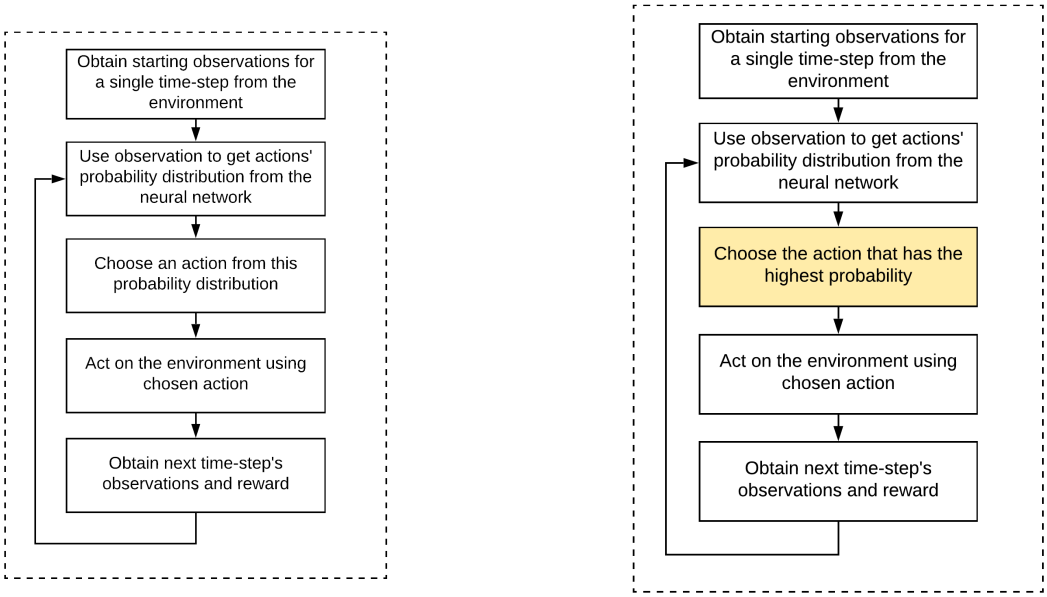
Table 5.6: Preferred actions and corresponding substations for Medium Agent

	Medium Agent	
Substations	**Action number**	**Action description**
3	36	Bus-bar 1: Load 2, Line 3, Line 16 Bus-bar 2: Line 5, Line 6, Line 15
	43	Bus-bar 1: Load 2, Line 5, Line 6, Line 16 Bus-bar 2: Line 3, Line 15
4	62	Bus-bar 1: Line 1, Line 6, Line 17 Bus-bar 2: Load 3, Line 4
8	100	Bus-bar 1: Line 10, Line 11, Line 16 Bus-bar 2: Load 5, Line 19

6

Agent Performance Analysis

Now that two agents have been trained, they need to be tested and evaluated. To do this, scenarios are chosen to test from. Each agent is tested on each test scenario. In this chapter, the evaluation of the two agents : Easy Agent and Medium Agent will be presented. Each agent is evaluated on each of the chosen test scenarios, and is evaluated in two ways. First, using a probabilistic policy, of playing the game a 1000 times using the method shown in figure 6.1a. Secondly, the game is played using a deterministic policy, using the method in figure 6.1b. Following this, an overview of the two agents' performance for varying scenarios is illustrated.



(a) Playing an episode in a probabilistic manner

(b) Playing an episode in a deterministic manner

Figure 6.1: Two ways of evaluating an agent

6.1. Test Scenarios

Similar to how scenarios are chosen for training, based on their difficulty levels, scenarios are also identified for testing. 3 scenarios are chosen from each of the difficulty scenarios: Easy, Medium and Difficult. Each point in figures 6.2 and 6.3 depict a single scenario. Both these figures derive from sections 4.3 and 4.4 using figures 4.10 and 4.12 respectively. The points used for training and testing are emphasised and shown with different shapes. The points representing the scenarios used for training are shown by the yellow squares, and the ones used for testing are shown by purple triangles.

Once again, the choice of these scenarios, is to ensure that scenarios for testing are chosen from each level of difficulty. Thus, this is visible, in figure 6.3, the different levels of difficulty are highlighted, to indicate that the scenarios chosen are from different levels of difficulty.

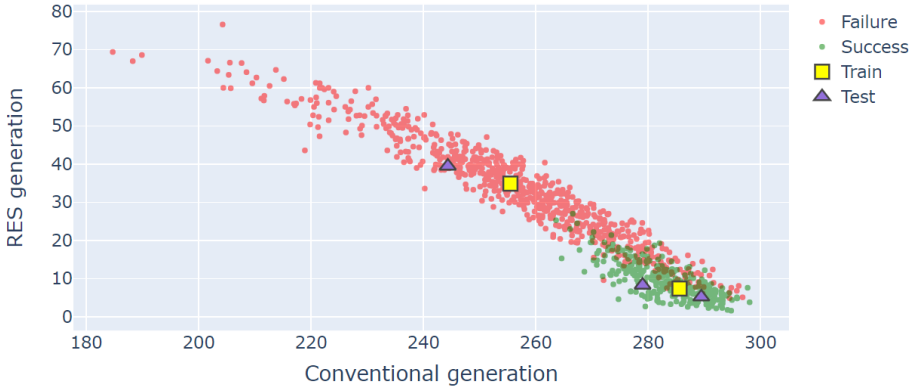


Figure 6.2: Scenarios chosen for evaluating the 2 agents (Comparing maximum generation values)

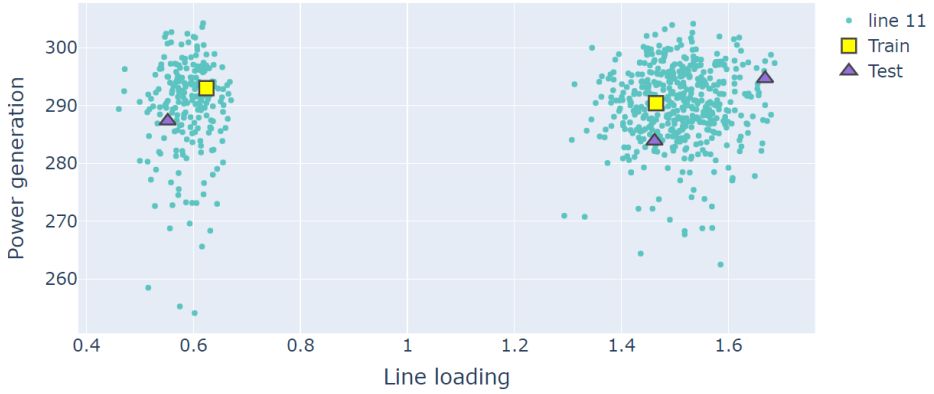


Figure 6.3: Scenarios chosen for evaluating the 2 agents (Comparing generation with line 11's loading)

6.2. Easy Agent evaluation

The easy agent is tested on three separate scenarios, one from each level of difficulty, namely, easy, medium and difficult. For each evaluation, 1000 episodes are played by Easy Agent on this scenario in a probabilistic manner. A combined view of all 1000 episodes are presented to highlight three factors:

- The first action that is taken to produce a new topology different from the reference topology
- The number of actions taken in an episode
- The time-step when the first action is taken.

Another way of evaluating the agent, is by playing an episode by this agent in a deterministic manner.

6.2.1. Evaluating an easy test scenario

The evaluation of the Easy Agent for an easy test scenario is presented here. All the 1000 probabilistic episodes have a split-up as seen in figure 6.4. 88% of them are successful, meaning that the Easy Agent traversed the episode for a duration of a week. Out of this 88%, 74% of the time, the agent used the reference configuration which in itself is a sufficient topology to use for this scenario. The other 14% of the episodes have a reward larger than the RC episode's reward.

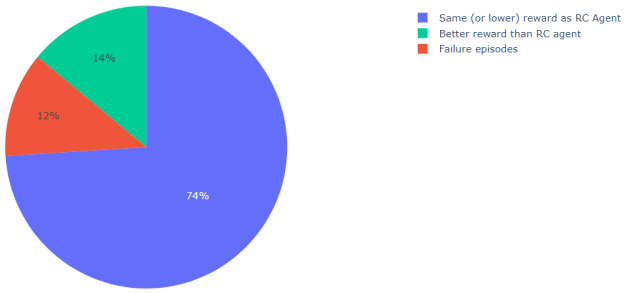


Figure 6.4: Split-up of the 1000 probabilistic episodes played by Easy Agent for this easy test scenario

In figure 6.5, we see all the successful episodes out of the 1000 episodes, that have rewards higher than the corresponding RC episode reward (highlighted by the pink dotted line), that is, the 14% of episodes from the above chart. While there are certain episodes that took other actions first, the most common behaviour seen in these 1000 episodes is to take a particular action (number 50) at substation 3. Similar to the episodes used to describe the policy of the Easy Agent (section 5.3.2), we see a trend for the episodes where the episode reward largely depends on the time-step of when the action is taken. The earlier the action is taken, the larger the reward. This larger reward implies that the grid has higher residual network capacity from an earlier point of time in the episode rather than later. Hence, we see that the better episodes make network changes early on in the episode and retain the network in this configuration for the rest of the duration of the episode, resulting in a 'successful' episode.

The other parameter that can be indicated in all the successful episodes is the number of actions that took place for all of them. This is highlighted in figure 6.6, where we see that for a majority of episodes, only a single action is taken.

The other way to evaluate the episode, is to play it in a deterministic manner. This is shown by a square, which has the same reward as that of the RC episode for this corresponding scenario. Hence, the deterministic policy replicated the policy of the RC agent.

Figure 6.5: Episode rewards compared with the time-step of first action (highlighting the first action taken)

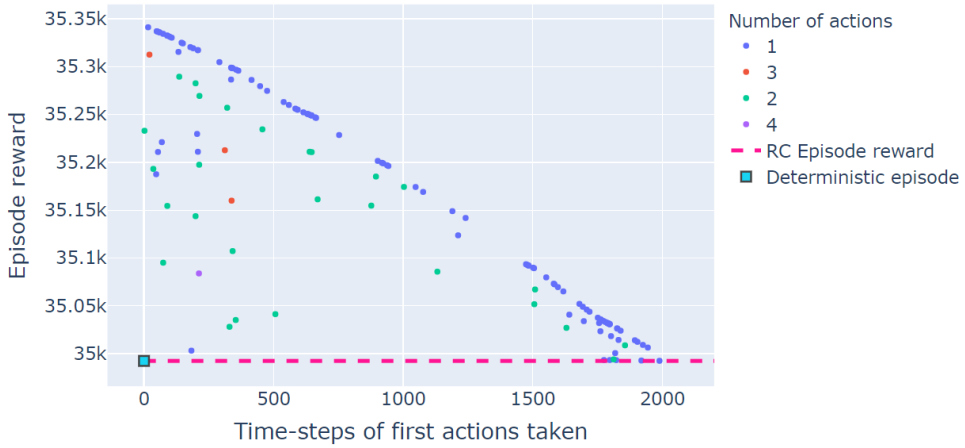


Figure 6.6: Episode rewards compared with the time-step of first action (highlighting the number of actions taken)

6.2.2. Evaluating a medium test scenario

The evaluation of the Easy Agent for a medium test scenario is presented here. All 1000 probabilistic episodes is summarised in figure 6.7. Here, only 8.2% of them are successful, meaning that the Easy Agent traversed the episode for a duration of a week. The rest 91.8% of the episodes, the agent used a sub-optimal configuration which is not a sufficient topology to use for this scenario.

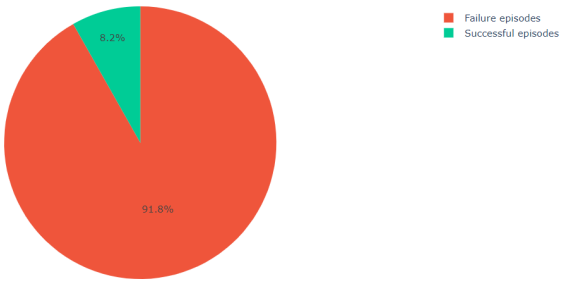


Figure 6.7: Split-up of the 1000 probabilistic episodes played by Easy Agent for this medium test scenario

In figure 6.8, we see all the successful episodes out of the 1000 episodes (the 8.2% of episodes). The time-step of when the first actions are taken are shown, and further, the actions that are taken at these time-steps are highlighted. The policy of this Easy Agent is to generally favour a particular action (number 50) at substation 3. While, in these episodes we see other substations which begin to crop up, the frequency of choosing this particular action is still much higher.

The trend where the episode reward largely depends on the time-step of when action number 50 is used and only a single action is taken in the episode is still seen here, however not as evidently and also the number of episodes that are successful are fewer. The time-step until which actions are taken, is also lesser compared to the previous test scenario, where actions are taken all the way until the end of the episode. This is justifiable by the fact that this is a medium scenario which implies it is the RC agent used on this scenario resulted in a failure scenario, meaning that the episode ended prematurely and did not last a duration of one week. Thus, if this agent takes its first action very late in the episode, the length of the episode could reach the length of the RC episode, and hence, will meet the same fate, and end up as a failure.

The other parameter that can be indicated in all the successful episodes is the number of actions that took place for all of them. This is highlighted in figure 6.9. Here, the number of episodes that take either 1 or 2 actions in the episode are comparable. This indicates, that since this episode is of a higher difficulty level, a single change may not be sufficient to operate the grid successfully for an entire week. The other way to evaluate the episode, is to play it in a deterministic manner. This is not depicted here, since the deterministic episode resulted in a reward which is the same as the RC episode's reward, which is of a much lower range and hence is not shown in this figure.

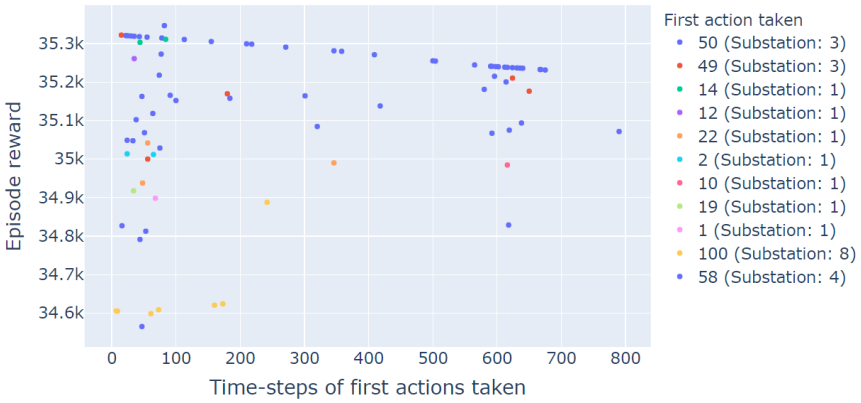


Figure 6.8: Episode rewards vs the first time-step an action is taken (highlighting the first action taken)

Figure 6.9: Episode rewards vs the first time-step an action is taken (highlighting the number of actions taken in the episode)

An interesting aspect to see now, is the topological time-series of the episodes. In figure 6.10, the best episode's (out of the 1000 probabilistic episodes) topological time-series is seen. Here, we see the number of elements that are varied to result in a new topology that is now sufficient to traverse the situation that arises in the reference topology that results in a failure. The time-step of when the RC episode fails, indicated by the dotted line, is around 800 time-steps. Which also justifies in the previous figures, why there are no successful episodes where the first action is taken after 800 time-steps, because the episode would have already ended, since it is using the reference topology. Similar to this plot, in figure 6.11, the same time-series can be seen, but to indicate the changes of elements substation-wise. This shows that two actions are taken to move 2 elements each, consecutively in substation 3 and 1 respectively, resulting in a change of position for a total of 4 elements for the overall network.

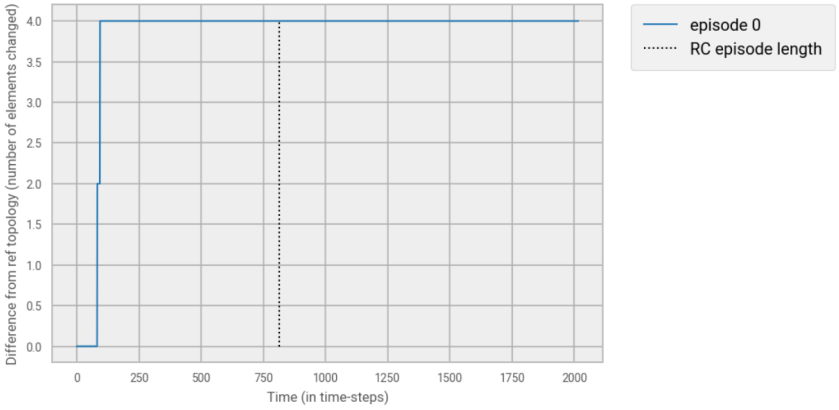


Figure 6.10: The best episode's topological time-series

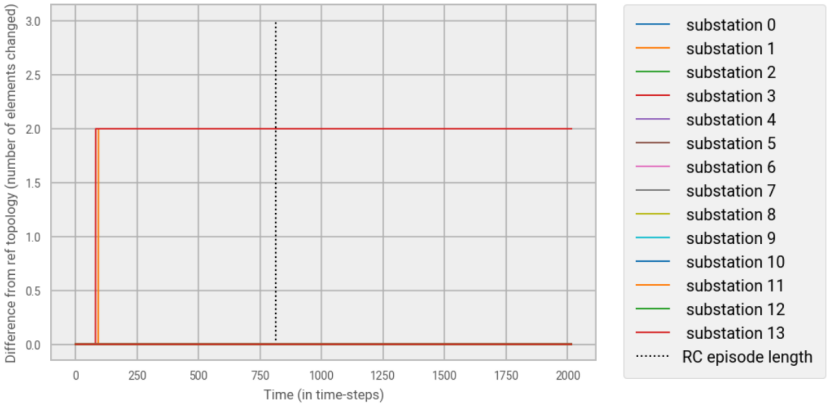


Figure 6.11: The best episode's topological time-series (substation-wise)

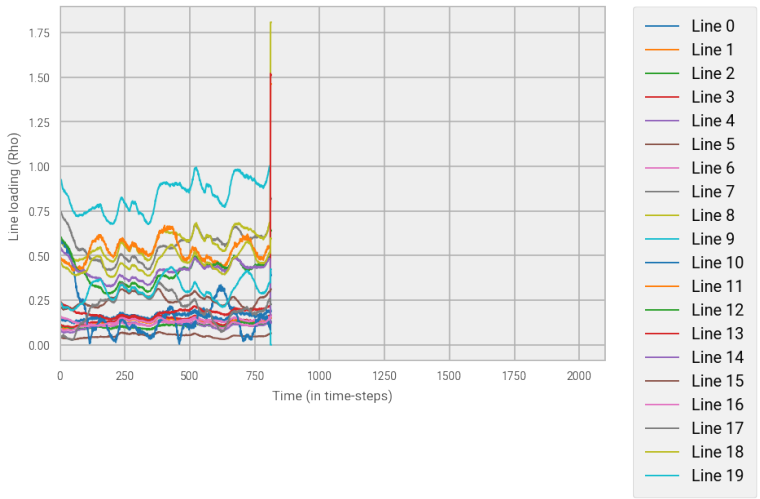


Figure 6.12: Loading of all 20 lines during the RC episode

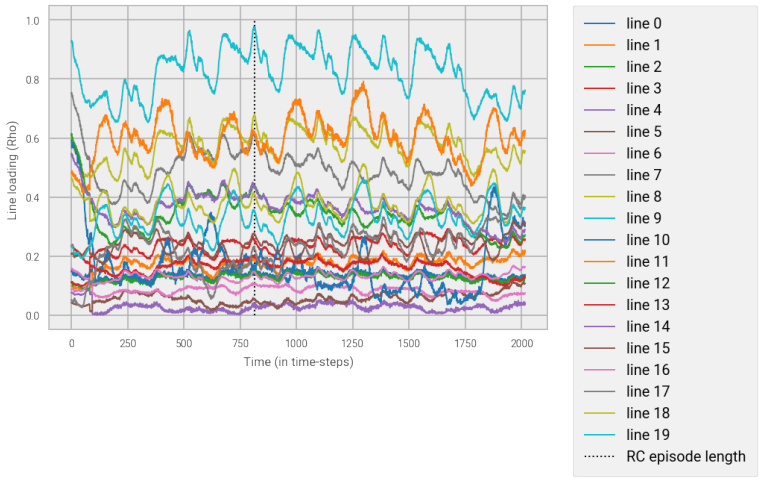


Figure 6.13: Loading of all 20 lines during the best episode

Another way to see how the best episode compares to the RC episode, is by looking at the loading of lines over the duration of both the episodes. Since this test scenario is a medium scenario, it implies that the RC agent for this scenario resulted in a failure episode. Thus in figure 6.12, we see the loading of lines up until the

episode's failure at over the 750th time-step. Whereas, for the episode where the Easy Agent was used on this scenario, the loading of any lines do not exceed a ρ value of 1 at any point, and the episode lasts the entire duration of 1 week / 2016 time-steps. This is seen in figure 6.13.

6.2.3. Evaluating a difficult test scenario

The evaluation of the Easy Agent for a difficult test scenario is presented here. To summarise all the 1000 probabilistic episodes figure 6.14 is provided. Here, only 4.7% of them are successful, meaning that the Easy Agent traversed the episode for a duration of a week. The rest 95.3% of the episodes, the agent used a sub-optimal configuration which is not a sufficient topology to use for this scenario.

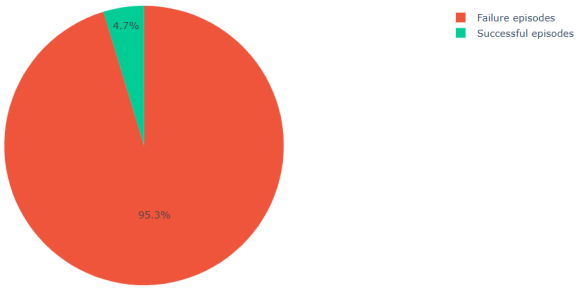


Figure 6.14: Split up of all 1000 probabilistic episodes played by Easy Agent for this difficult test scenario

In figure 6.15, we see all the successful episodes out of the 1000 probabilistic episodes (the 4.7% pf episodes). The time-step of when the first actions are taken are shown, and further, the actions that are taken at these time-steps are highlighted. The policy of this Easy Agent is to generally favour a particular action (number 50) at substation 3. Which also results in this trend where the episode reward largely depends on the time-step of when action number 50 is used and only a single action is taken in the episode is still seen here. Here, the number of episodes that are successful are quite few. The time-step until which actions are taken, is also shorter than the total length of a successful episode. Once again, this is due to the fact that this is a difficult scenario which implies it is a failure scenario, meaning that the episode ended prematurely and did not last a duration of one week. Thus, if this agent takes its first action very late in the episode, the length of the episode could reach the length of the RC episode, and hence, will also end up as a failure. As seen earlier, the other parameter that can be indicated in all the successful episodes is the number of actions that took place for all of them. This is highlighted in figure 6.16. Here, the number of episodes that take only 1 action are more than the episodes that take 2 actions. The other way to evaluate the episode, is to play it in a deterministic manner. This is not depicted here, since the deterministic episode resulted in a reward which is the same as the RC episode's reward, which is of a much lower range and hence is not shown in this figure.

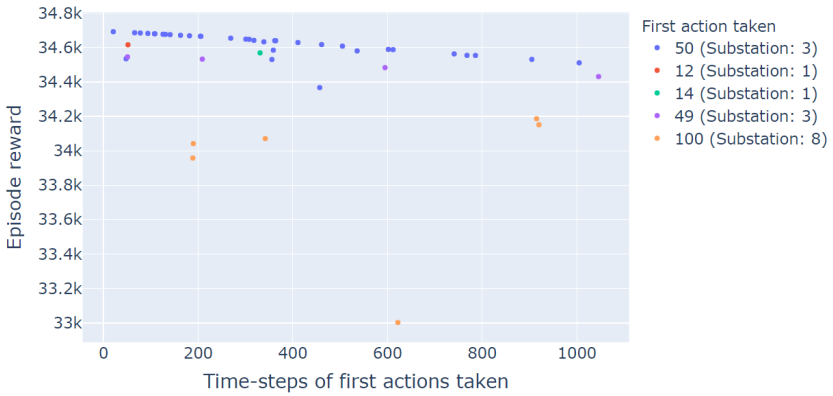


Figure 6.15: Episode rewards vs the first time-step an action is taken (highlighting the first action taken)

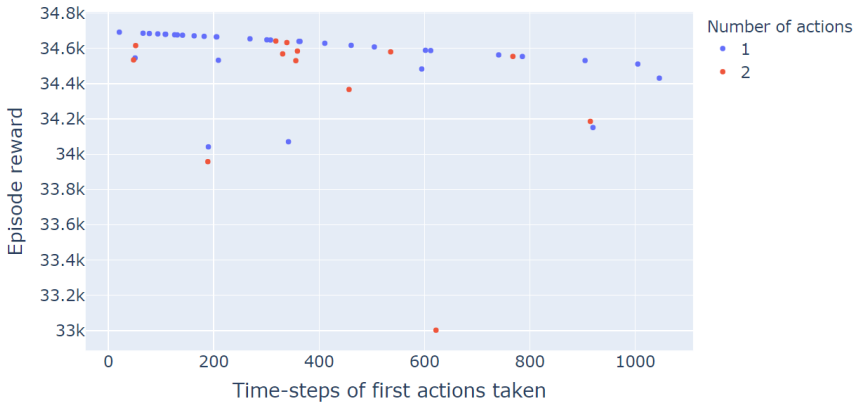


Figure 6.16: Episode rewards vs the first time-step an action is taken (highlighting the number of actions taken in the episode)

The topological time-series of the episodes can be seen for this scenario as well. In figure 6.17, the best episode's (out of the 1000 probabilistic episodes) topological time-series is seen. Here, we see the number of elements that are varied to result in a new topology that is now sufficient to traverse the situation that arises in the reference topology that results in a failure. The time-step of when the RC episode fails, indicated by the dotted line, is around 1100 time-steps. Which also explains why there are no successful episodes where the first action is taken after 1100 time-steps, because the episode would have already ended, since it is using the reference topology. Similar to this plot, in figure 6.11, the same time-series can be seen, but to indicate the changes of elements substation-wise. This shows that a single action at substation 3 is taken which resulted in a change of location for two elements and hence resulted in a new network topology.

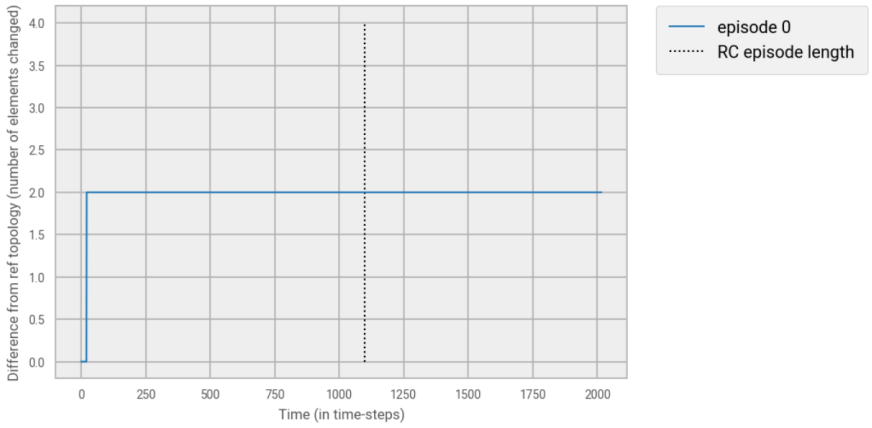


Figure 6.17: The best episode's topological time-series

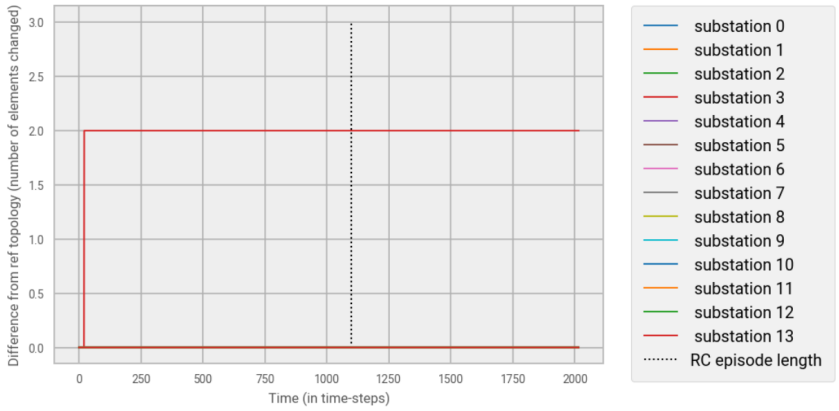


Figure 6.18: The best episode's topological time-series (substation-wise)

We can also see how the best episode compares to the RC episode, by looking at the loading of lines over the duration of both, the RC episode and the best episode played by this agent. Since this test scenario is a difficult scenario, it implies that the RC agent for this scenario resulted in a failure episode. Thus in figure 6.19, we see the loading of lines up until the episode's failure at over the 1000th time-step. On the other hand, in figure 6.20, the episode where the Easy agent was used on this scenario, the loading of any lines do not exceed a ρ value of 1 at any point, and the episode lasts the entire duration of 1 week / 2016 time-steps.

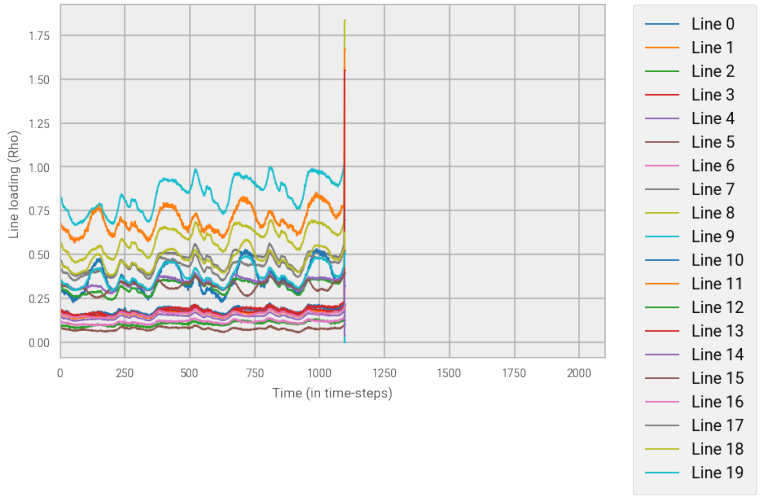


Figure 6.19: Loading of all 20 lines during the RC episode

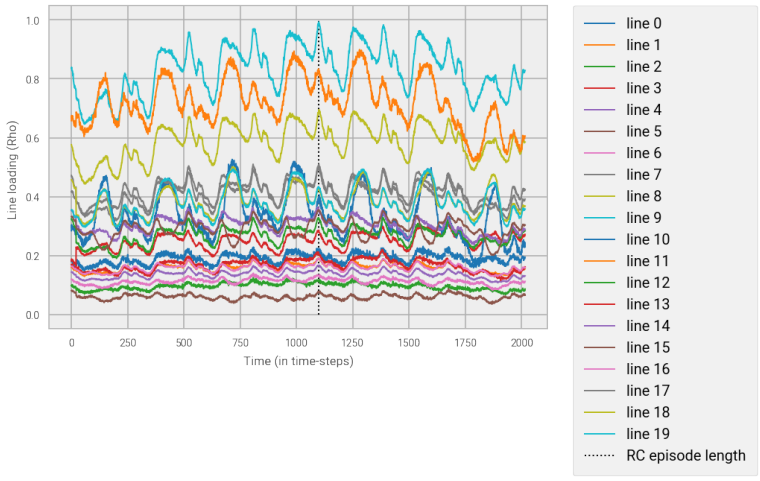


Figure 6.20: Loading of all 20 lines during the best episode

6.3. Medium Agent evaluation

Similar to the Easy Agent, the Medium Agent is tested on the same three scenarios, one from each level of difficulty, namely, easy, medium and difficult. For each evaluation, 1000 episodes are played by Easy Agent on this scenario in a probabilistic manner. A combined view of all 1000 episodes are presented to highlight three factors:

- The first action that is taken to produce a new topology different from the reference topology
- The number of actions taken in an episode
- The time-step when the first action is taken.

An episode played using a deterministic policy for each of the three scenarios is also discussed.

6.3.1. Evaluating an easy test scenario

The evaluation of the Medium Agent for an easy test scenario is presented here. All the 1000 probabilistic episodes can be summarised in 6.21. 98.4% of them are successful, meaning that the Medium Agent traversed the episode for a duration of a week. Though successful, these episodes resulted in slightly lower rewards compared to the RC episode's reward. Out of the 1000 episodes, only 1.4 % of them resulted in failure episodes.

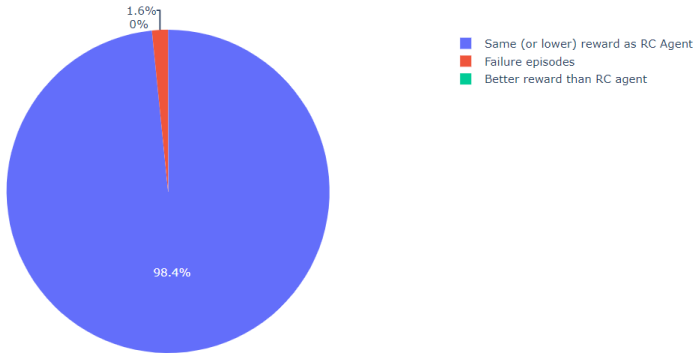


Figure 6.21: Split-up of all 1000 probabilistic episodes played by the Medium Agent on this easy scenario

In figure 6.22, we see 98 % of the episodes (the successful episodes) out of the 1000 episodes played by the Medium Agent.

As seen while understanding the behaviour of the Medium Agent (by evaluating the agent on the same scenario it was trained on, as described in section 5.3.3), looking at how the episode rewards vary with the time-step of first action is not the most appropriate measure. In all these episodes, we see that the agent takes its first action within the first two time-steps (as evident from figure 6.23). Thus, to see the spread, we look at the number of actions taken in an episode on the x axis instead. In figure 6.22, we see the policy explained for Medium Agent evidently, by the actions and substations that are preferred. This implies that Medium Agent is operating the grid with the lines a little closer to full capacity compared to the RC agent. Thus, the Medium Agent does not find a policy which results in topologies that are better (in terms of reward) than the reference topology for this easy scenario. In figure 6.23, it is evident that majority of episodes take an action immediately in the first time-step, and some take actions in their second time-step. In both these figures, the number of episodes are quite concentrated at these points, meaning the agent behaved identically for a large number of episodes resulting in identical rewards for these episodes. Thus, many of the points are overlapping in these figures.

The other way to evaluate the episode, is to play it in a deterministic manner. For this scenario, the deterministic episode resulted in a much lower reward, and resulted in a failure and hence, is not presented here.

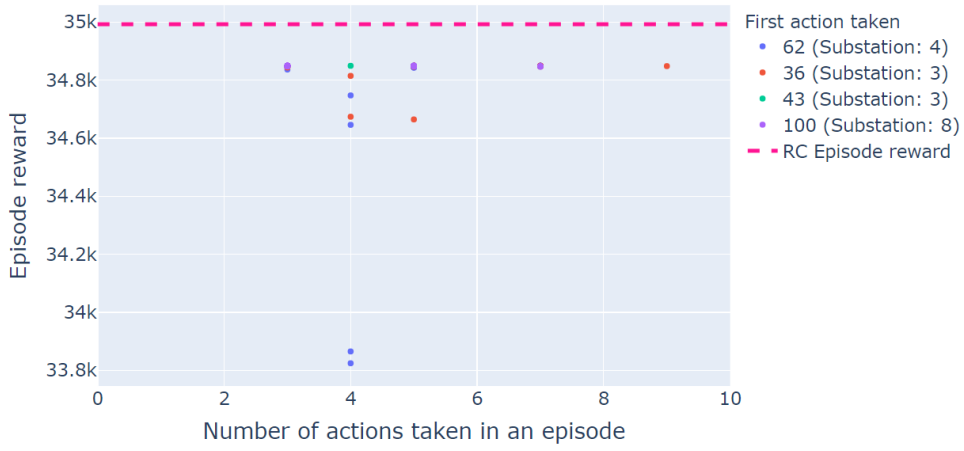


Figure 6.22: Episode rewards vs the number of actions taken (highlighting the actions taken at the first time-step)

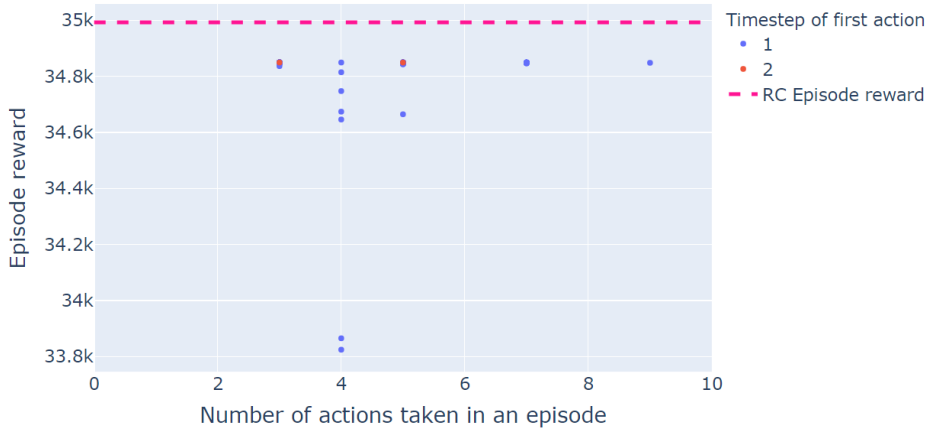


Figure 6.23: Episode rewards vs the number of actions taken (highlighting the time-step where the first action is taken)

The topological time-series of the episodes can be seen for this scenario as well. In figure 6.24, the best episode's (out of the 1000 probabilistic episodes) topological time-series is seen. Here, we see the a total of seven elements' positions are varied to result in a new topology. Since, this is an easy scenario, the RC episode's length is a week long, (2016 time-steps). In figure 6.25, the same time-series can be seen, but to indicate the changes of elements substation-wise. This shows that actions at 3 substations are taken in in quick succession at substation 4, substation 8 and substation 3 respectively.

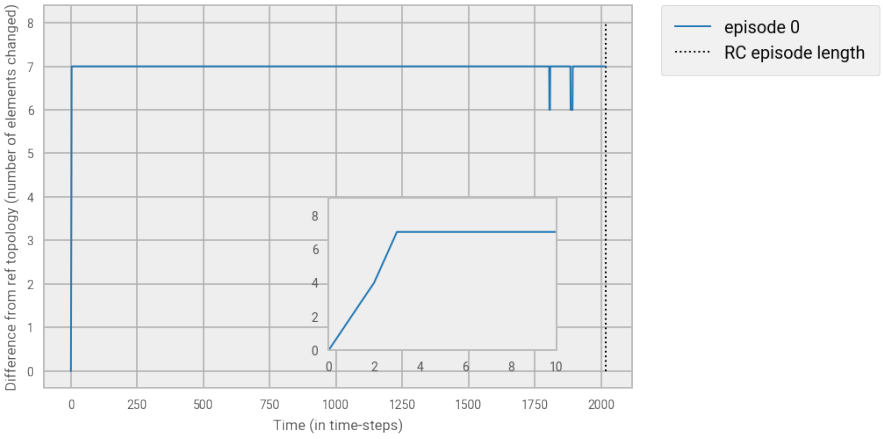


Figure 6.24: The best episode's topological time-series

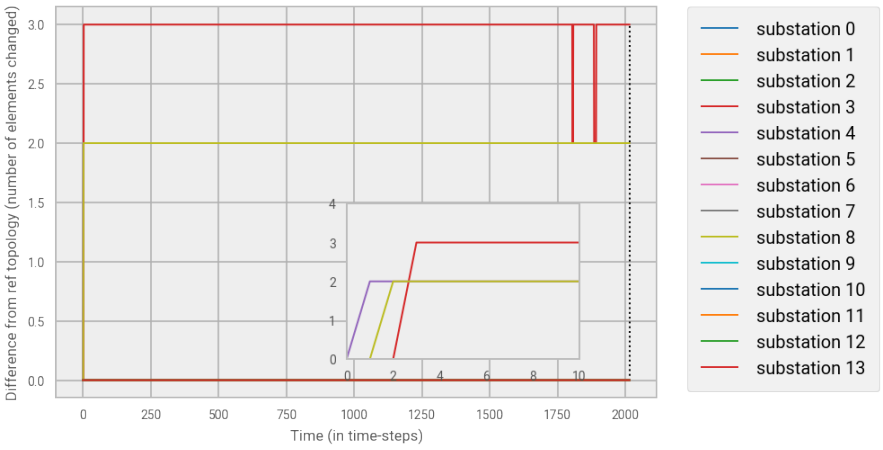


Figure 6.25: The best episode's topological time-series (substation-wise)

6.3.2. Evaluating a medium test scenario

The evaluation of the Medium Agent for a medium test scenario is presented here. All the 1000 probabilistic episodes are summarised in figure 6.21. 98% of them are successful, meaning that the Medium Agent operated the network successfully for a duration of a week, with only 2% of the episodes resulting in failures.

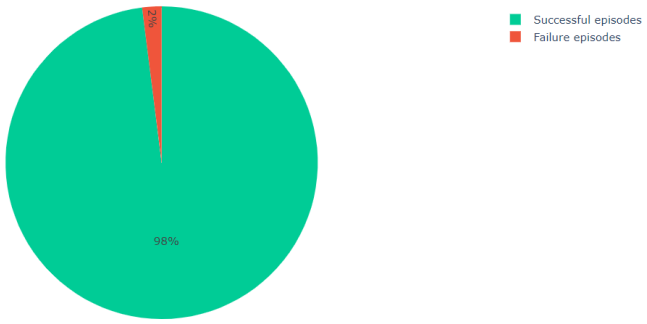


Figure 6.26: Split-up of all 1000 probabilistic episodes played by the Medium Agent on this medium scenario

In figure 6.27, we see all the successful episodes out of the 1000 episodes, (98% of episodes) for the Medium Agent. The episode rewards against the number of actions taken are presented here. Once again,the substations where actions are taken are seen here to be substations 3,4 and 8. The corresponding RC episode's reward is not seen here because this is a medium scenario and hence, is a failure episode. Thus, the range of this episode's reward is in a much lower range. In figure 6.28, the policy of Medium Agent to take its first actions very early on in the episode is seen once again. In both these figures, the number of episodes are closely concentrated at , meaning the agent behaved identically for a large number of episodes resulting in identical rewards for these episodes.

Figure 6.27: Episode rewards vs the number of actions taken (highlighting the actions taken at the first time-step)

Figure 6.28: Episode rewards vs the number of actions taken (highlighting the time-step where the first action is taken)

The other way to evaluate the episode, is to play it in a using a deterministic policy. For this scenario, the deterministic episode resulted in a higher reward than those of the probabilistic episodes, as seen by the square points in both these figures. Thus, it is also interesting to see the topological time-series for this deterministic episode, as presented in figure 6.29. Here, we see that 2 elements are changed early on in the episode, which in itself resulted in a topology which is sufficient to cross the RC episode's length (around 800 time-steps). But an additional change is also made later on in the episode, at around the 1250^{th} time-step. In figure 6.30, we see that the first action is taken at substation 4, and the additional change happens at substation 3.

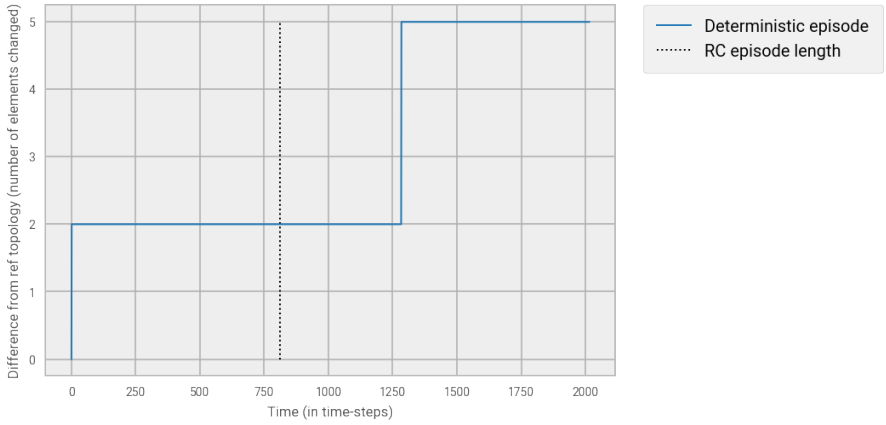


Figure 6.29: The deterministic episode's topological time-series

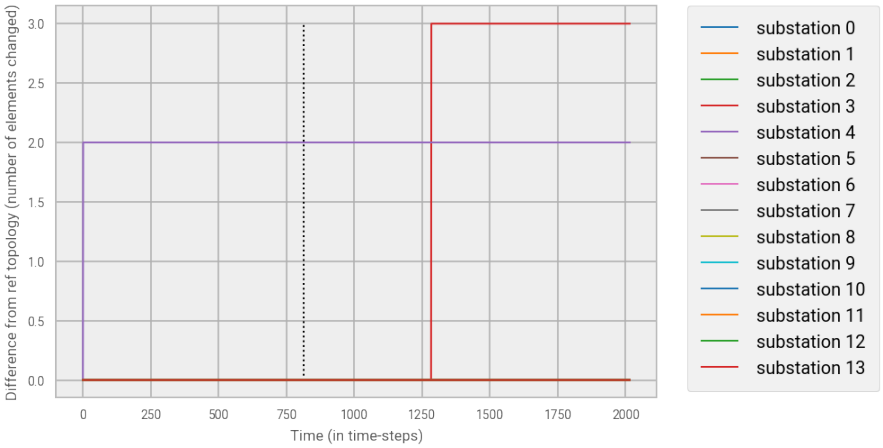


Figure 6.30: The deterministic episode's topological time-series (substation-wise)

The topological time-series of the probabilistic episode with the highest reward is seen in figure 6.31. Here, we see the a total of seven elements' positions are varied to result in a new topology. In figure 6.32, the substation-wise changes in the number of elements is indicated. This shows that actions at 3 substations are taken in in quick succession at substation 4, substation 8 and substation 3 respectively.

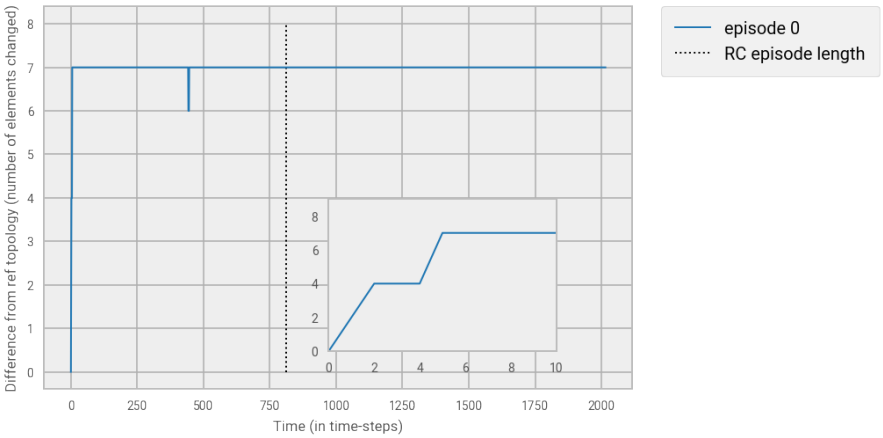


Figure 6.31: The best episode's topological time-series

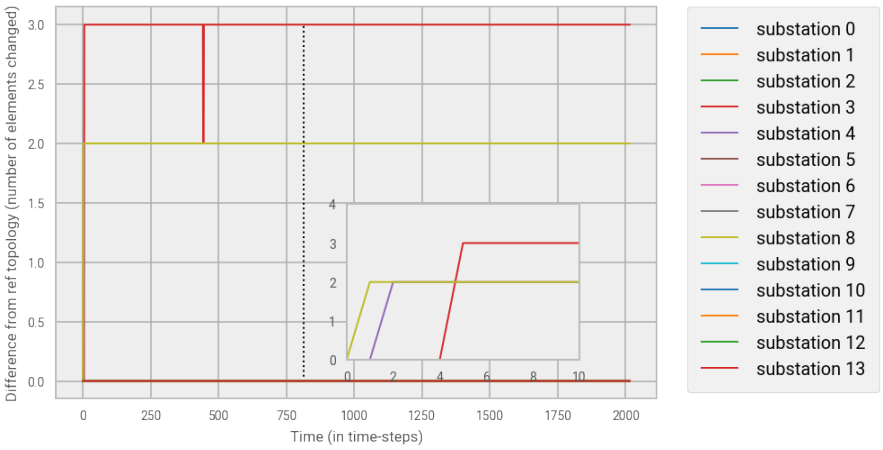


Figure 6.32: The best episode's topological time-series (substation-wise)

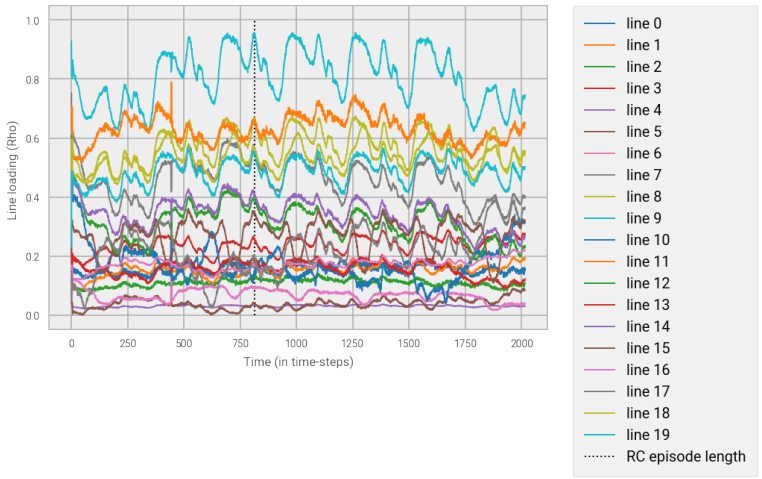


Figure 6.33: Loading of all 20 lines during the best episode

To also see how the best episode compares to the RC episode by looking at the loading of lines for the best episode, figure 6.33 is provided. Since this test scenario is a medium scenario, it implies that the RC agent for this scenario resulted in a failure episode. From the dotted line in figure 6.33, we see that the episode ends at around the 750th time-step. On the other hand, in figure 6.33, the episode where the Medium agent was used on this scenario, the loading of any lines do not exceed a ρ value of 1 at any point, and the episode lasts the entire duration of 1 week / 2016 time-steps.

6.3.3. Evaluating a difficult test scenario

The evaluation of the Medium Agent for a difficult test scenario is presented here. To summarise all the 1000 probabilistic episodes, 6.21 is provided. 97.9% of them are successful, meaning that the Medium Agent operated the network successfully for a duration of a week, with only 2.1% of the episodes resulting in failures.

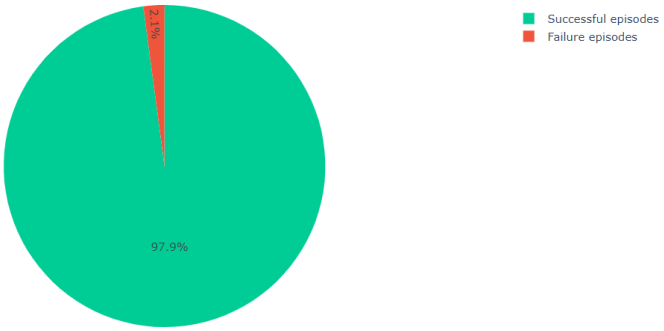


Figure 6.34: Split-up of all 1000 probabilistic episodes played by the Medium Agent on this difficult scenario

In figure 6.35, we see all the successful episodes (97.9% of episodes) out of the 1000 probabilistic episodes for the Medium Agent. The episode rewards against the number of actions taken are presented here. Once again,the substations where actions are taken are seen here to be substations 3,4 and 8. Though, compared to the medium scenario, there is one less action that is taken by the agent at the first time-step. The corresponding RC episode's reward is not seen here because this is a medium scenario and hence, is a failure episode. Thus, the range of this episode's reward is in a much lower range. Similarly, the deterministic episode's reward is also not shown here for the same reason. In figure 6.36, the policy of Medium Agent to take its first actions very early on in the episode is seen once again. In both these figures, the number of episodes are closely concentrated, meaning the agent behaved identically for a large number of episodes resulting in identical rewards for these episodes.

Figure 6.35: Episode rewards vs the number of actions taken (highlighting the actions taken at the first time-step)

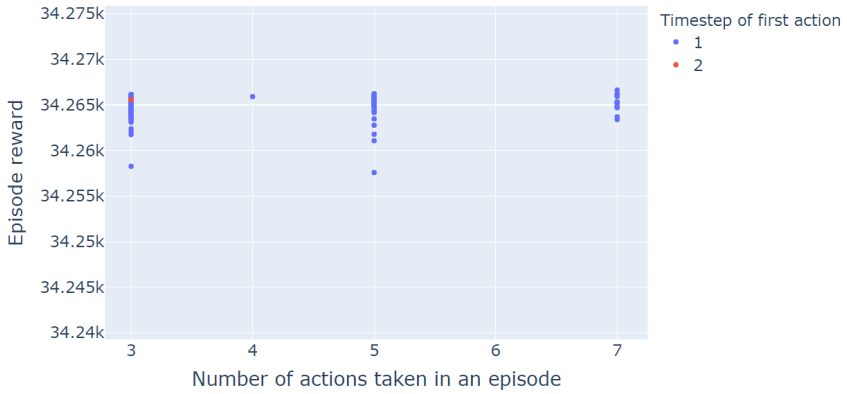


Figure 6.36: Episode rewards vs the number of actions taken (highlighting the time-step where the first action is taken)

The topological time-series of the probabilistic episode with the highest reward is seen in figure 6.37. Here, we see the a total of seven elements' positions are varied to result in a new topology. In figure 6.38, the substation-wise changes in the number of elements is indicated. This shows that actions at 3 substations are taken in in quick succession at substation 3, substation 8 and substation 4 respectively.

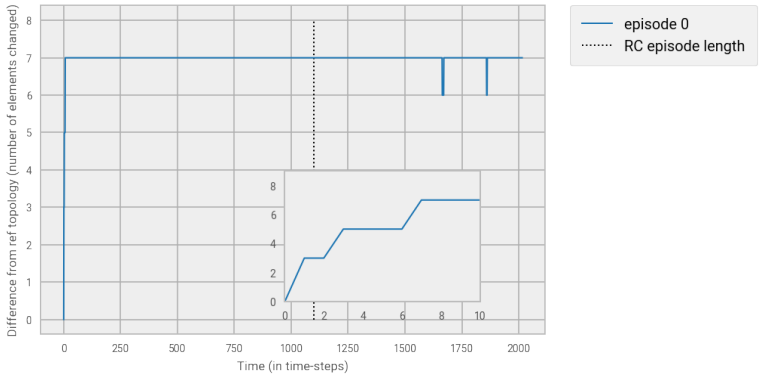


Figure 6.37: The best episode's topological time-series

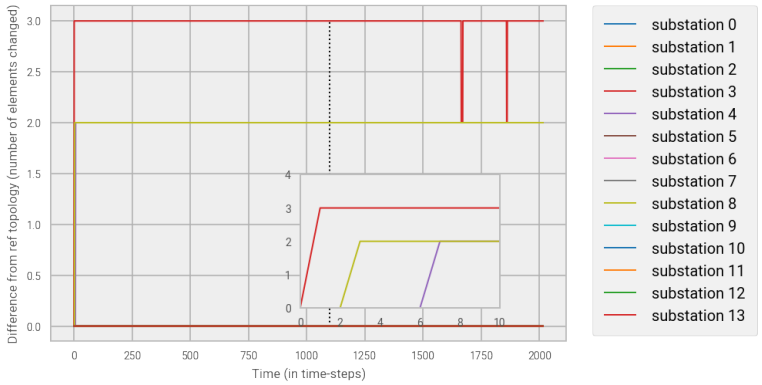


Figure 6.38: The best episode's topological time-series (substation-wise)

We can see the loading of lines over the duration of this best episode in 6.39. Since this test scenario is a difficult scenario, it implies that the RC agent for this scenario resulted in a failure episode. We know that the episode fails at over the 1000th time-step. In figure 6.39, we see in the episode where the Medium agent was used on this scenario, the loading of any lines do not exceed a ρ value of 1 at any point, and the episode lasts the entire duration of 1 week / 2016 time-steps.

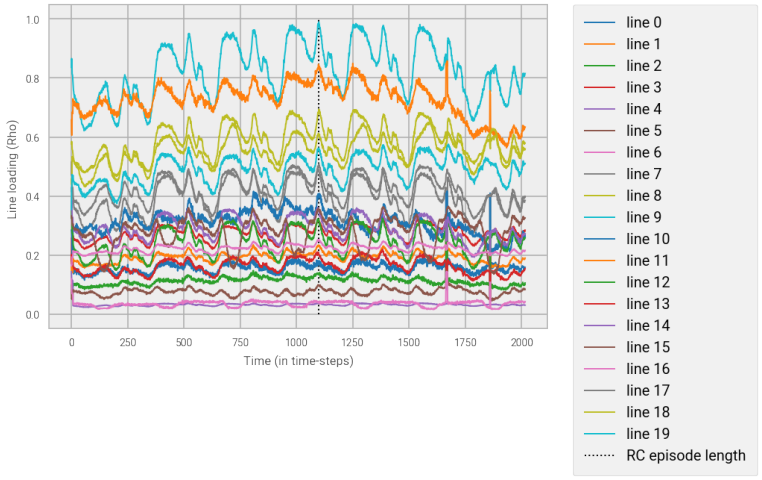


Figure 6.39: Loading of all 20 lines during the best episode

6.4. Overview of agent performance

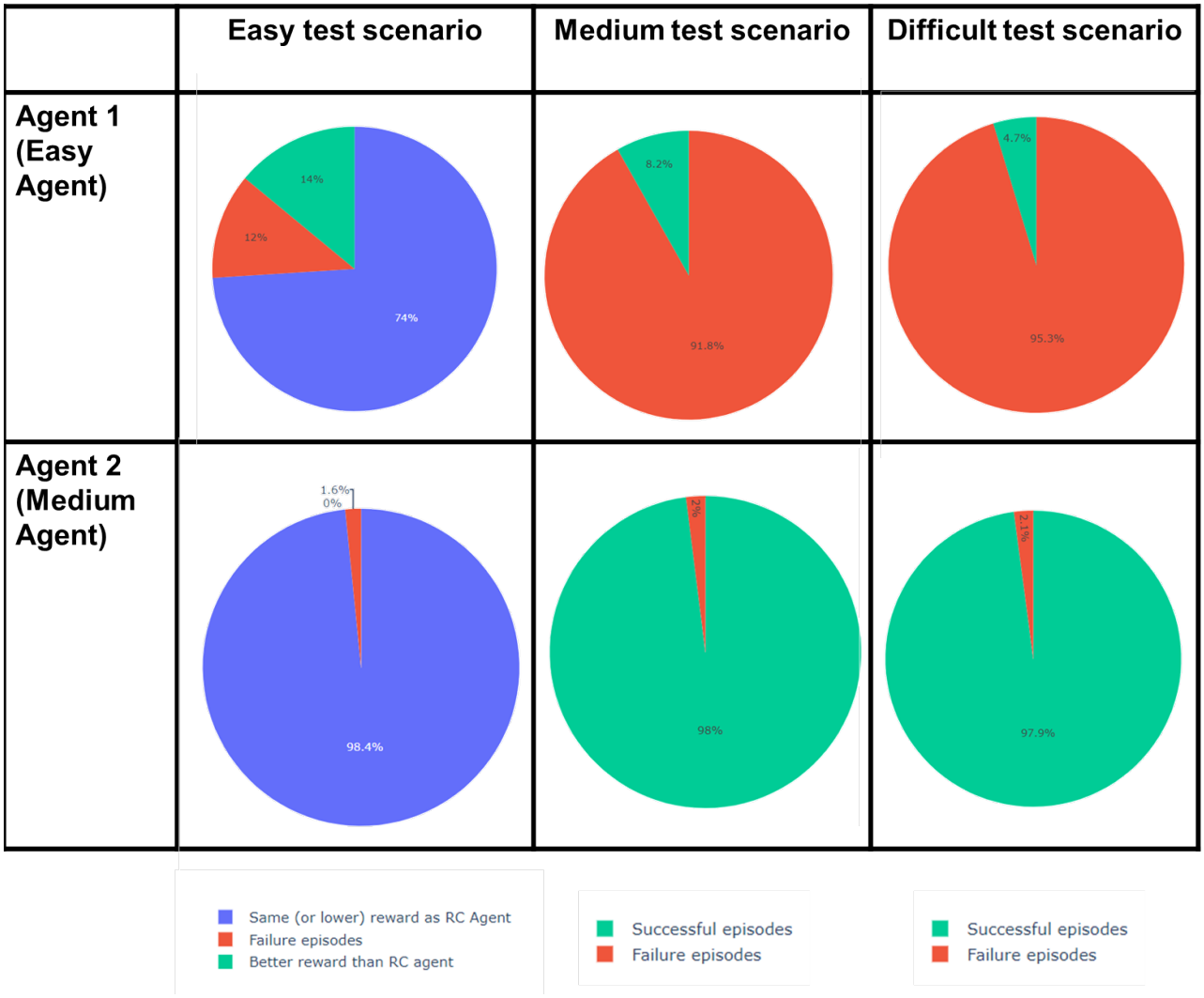


Figure 6.40: Overview of the two developed agents by playing the game in a probabilistic manner

In figure 6.40, an overview of the performance of both agents for the three test scenarios, one from each level of difficulty, is presented. It is immediately evident, that Easy Agent's policy is not one that is sufficient to tackle scenarios of other difficulty levels, as its performance on the medium and difficult scenarios is quite poor. The Easy Agent's performance is low for the medium test scenario, and dropped further for the more difficult scenario. However, the Medium Agent's policy is one that performs quite well for all three scenarios from varying difficulty levels. It produced successful episodes around 98% of the time for all three test scenarios. The only drawback, with its performance for the easy test scenario is that its reward is lower than that of the corresponding RC episode, nevertheless, successful episodes are produced.

A comparison can also be drawn for the performance of the agents for these test scenarios presented here and also for the scenarios that it was trained on. In figure 5.13 and 5.20 the performance of Easy and Medium agent for the corresponding scenarios it was trained for are presented. When comparing them with the overview presented in 6.40, we see that the percentage of success is slightly higher for both agents for the scenarios they were trained on.

Additionally, figures 6.41 and 6.42 show the varying episode rewards for each of the agents to highlight the difference between three types of policies:

- The policy used by the RC agent: Where the RC agent is used and hence, no topological changes take place.
- The deterministic policy: Where the policy of the agent is used to play the game in a deterministic manner (refer to figure 5.16).

- The probabilistic policy: The policy of the agent is used, to play the game in a probabilistic manner (refer to figure 5.9). Here, since there are a total of 1000 episodes, the mean of these episodes and the maximum of these episodes are shown.

For Easy Agent, in the figure 6.41, we see that though the maximum probabilistic reward for the easy scenario is higher, the mean is lower than the deterministic and RC episode rewards. Similarly, for the other two scenarios, the maximum reward for the probabilistic episodes is high, because the agent does produce some successful episodes. But, as we know from figure 6.40, the percentage of successful episodes for both the medium and difficult scenarios is very low. Hence, this results in a low mean probabilistic reward as well.

For Medium Agent, we see comparable mean and max rewards since the percentage of successful episodes is comparatively very high. Only in the case of the medium test scenario, the deterministic episode reward is higher than the probabilistic (mean and maximum) rewards, in all other cases, the probabilistic mean and maximum rewards are higher.

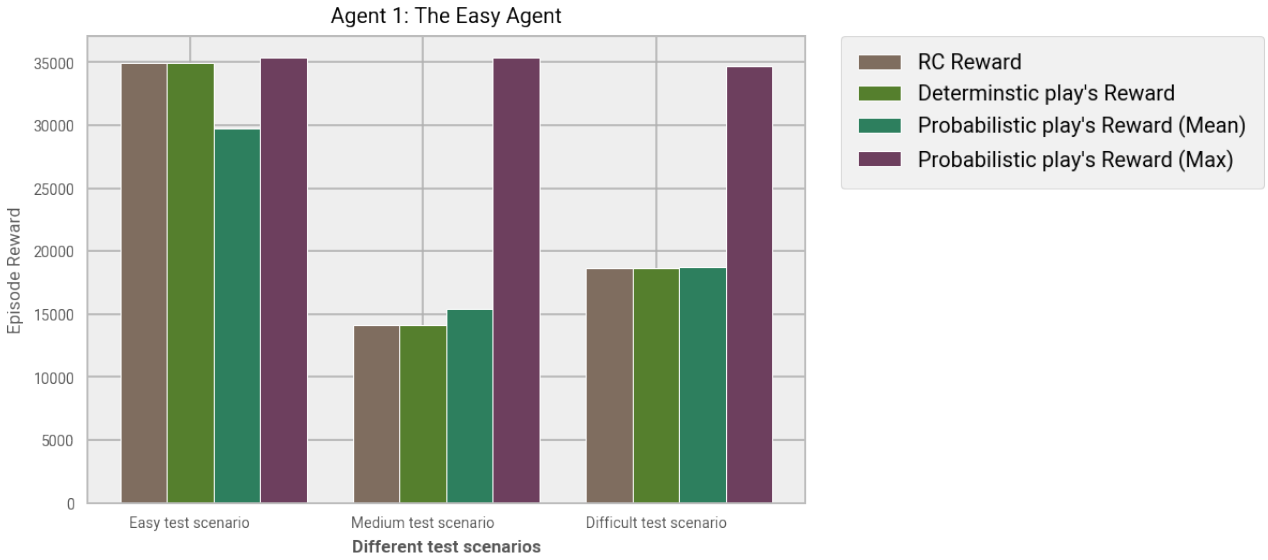


Figure 6.41: Overview of Agent 1's performance for different test scenarios

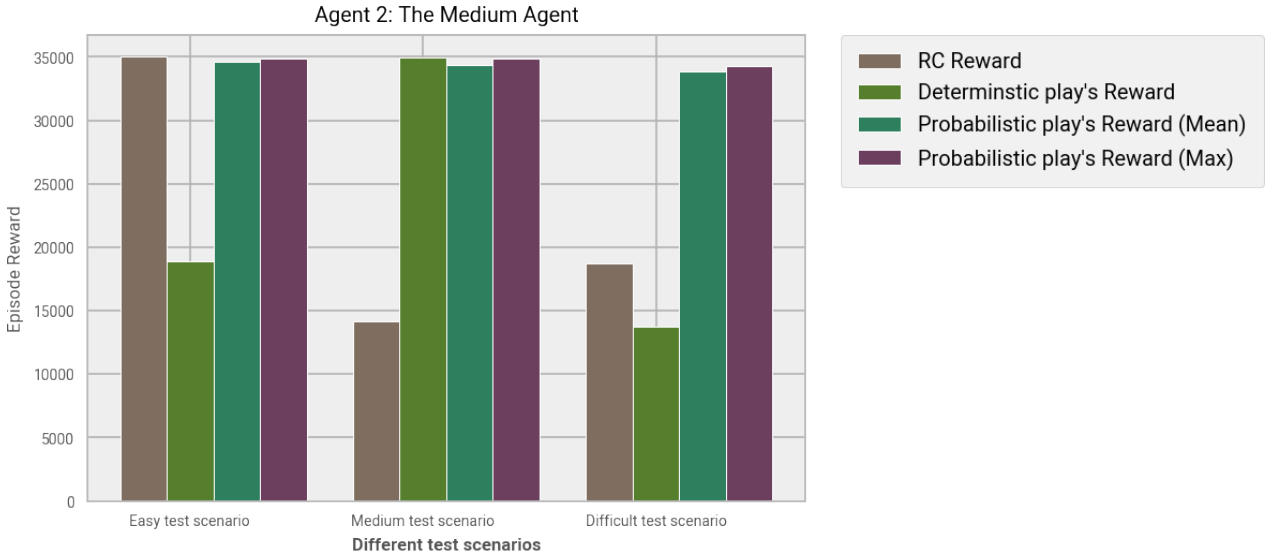


Figure 6.42: Overview of Agent 2's performance for different test scenarios

When drawing a comparison for the two agents, we see that the policy used by Medium agent performs better for scenarios from all three difficulty levels. Thus, this policy is more generalised compared to the policy

of Easy agent which does not perform well for scenarios outside of the same difficulty level used for its own training. Particularly when looking at the different kinds of policies, probabilistic and deterministic, Easy Agent's two policies are comparable in that they both produce good results only for a scenario from the easy category. But, on the other hand, for Medium agent, the probabilistic policy performs better for a test scenario from all three categories. The deterministic policy for the Medium Agent produces a good result only for the medium scenario.

The important difference between the probabilistic and deterministic policies lies in how the action is chosen. In the probabilistic policy, the action is chosen from the probability distribution. This means that even if the probabilities for certain actions are low, during the course of an episode, these actions can get chosen. But, on the other hand, in the deterministic policy, for every time-step, only the action with the highest probability will get chosen. Thus, if even only of them always has the largest probability then the other actions never get chosen. By seeing that the probabilistic policy tends to perform better, it can imply that multiple actions matter to successfully traverse the episode.

An interesting result from all the best episodes for the Medium agent evaluation can be noticed. The best episode played by Medium agent for all three test scenarios, used a similar topology. This topology was a result of a combination of three actions: action number 36 at substation 3, action number 62 at substation 4 and action number 8 at substation 8. The descriptions of these actions can be seen below in table 6.1. This topology proves to be a sufficient one to operate the grid successfully for all three test scenarios. This indicates that this topology is optimal, for these three test scenarios, even when each scenario is from a different difficulty level.

Table 6.1: Actions taken to result in a generalised topology

	Medium Agent	
Substations	**Action number**	**Action description**
3	36	Bus-bar 1: Load 2, Line 3, Line 16 Bus-bar 2: Line 5, Line 6, Line 15
4	62	Bus-bar 1: Line 1, Line 6, Line 17 Bus-bar 2: Load 3, Line 4
8	100	Bus-bar 1: Line 10, Line 11, Line 16 Bus-bar 2: Load 5, Line 19

7

Conclusions

This final chapter discusses the results obtained in the thesis and highlights the main conclusions. It describes the approach followed in answering the research questions addressed at the start of the thesis. It also draws some comparison with the agents developed by the previous year's L2PRN winners. Then, some brief challenges are described and it finally concludes this thesis with some recommendations for future work.

7.1. Discussion

The main research objective of this thesis is to create a reinforcement learning agent that takes topology reconfiguring actions to successfully operate an IEEE test network for a duration of one week. In order to accomplish this larger goal, this objective is broken down into a number of sub-research questions as seen in section 1.2. In chapter 2, some requisite background is provided. In this chapter, firstly, necessary information of the power system context is provided, following which, a brief introduction to machine learning and particularly reinforcement learning is provided. Chapter 3 describes the grid and the general framework used in this thesis.

In chapter 4, the 1000 available scenarios are analysed by deploying an RC agent on this particular IEEE 14 bus test case. This agent maintains the grid in a single default configuration or as more commonly referred to in this thesis, as the reference configuration. By doing so, 1000 corresponding episodes were obtained, known as the RC episodes. The episodes can result in either successes or failures. By understanding the different ways these episodes could be categorised, based on how the scenario failed or succeeded, three difficulty levels were identified. This highlights how the reference configuration is an insufficient configuration to use for a majority of the available scenarios. This helps address research question 1.

Following this, the first half of chapter 5 addresses the method used in this thesis to create an action space with bus-bar switching actions by also incorporating the real-world constraints described. With this, research question 2 is addressed. The second half of this chapter discusses the implementation of the Cross-Entropy Method based reinforcement learning algorithm used to develop our agents. Two agents are trained for different input settings. This implementation helps answer research question 3.

Chapter 6 discusses the results of these trained agents by evaluating them against different scenarios. Finally, the results are presented and discussed. This describes the answer to research question 4 and is also summarised in the following subsection.

7.1.1. Summary of results

In this thesis, two agents were developed, namely, Easy Agent and Medium Agent and are called so, based on the difficulty levels of the scenarios they were respectively trained on. Each of the agents are then evaluated on three other scenarios: one from each level of difficulty. The overview of results provided for both these agents can be seen in section 6.4. First, the probabilistic method of evaluation is discussed here.

Easy Agent is trained on a scenario from the easy category. From the results presented for this agent, we see that while this agent has a high success rate (88 %) for a scenario from the easy category, it performs very poorly scenarios from the other categories. This can be attributed to two things. Firstly, the topology that the agent steers toward is not a sufficient topology for scenarios that are more difficult. Secondly, this agent, while being trained could have been overfit for the scenario it was trained on, and thus only performs well for a scenario of the same level of difficulty, and does not achieve a good success rate for the other scenarios. This topology and the series of actions preferred by this agent, is thus, not one that can be generalised for different scenarios.

Medium Agent is trained on a scenario from the medium category. The results presented for this agent, on the other hand, prove more satisfactory. This agent has a high success rate for all three test scenarios, of 97.9% and over. For the easy test scenario however, the topology that the agent chose is slightly sub-optimal compared to that of the reference topology itself since the resulting episodes' rewards is marginally lower. But, for both the medium and difficult scenarios, the agent produced high rewards.

For the second type of evaluation, deterministic policies are used. For Easy Agent, the deterministic policy produced good results only for the easy test scenario, which is the same level of difficulty of the scenario used to train this agent. Similarly, for Medium Agent as well, the deterministic episode performs well only for the medium test scenario. Thus for both agents, the deterministic episodes produced high reward only when the test scenarios are from the same difficulty level as the scenario that is used to train the respective agent. This indicates that the agents can perform well in a deterministic manner for other scenarios only from the same level of difficulty, but they are not generalised enough to produce good results for scenarios from other levels of difficulty.

An important thing to note here, is that both the agents are trained on only a single scenario. This is a sufficient method for testing and evaluating results and this approach. We see that for these agents, the performance varies (or reduces) for other scenarios. This is because, there could be situations in the grid that haven't been experienced by the agent, since it was exposed to only one scenario. Thus, for deployment and use of such an agent, it is important that a single agent has experienced a variety of situations and events. For this, the agent will have to be trained using a large number of scenarios. An agent that has 'seen' a variety of situations, has the potential of reacting more accurately and precisely to a given set of observations as compared to the agents trained on a single scenario. The main reason for not following such an approach in this thesis is due to the increased computational complexity involved in exposing the agent to a large number of scenarios.

7.1.2. Comparison with previous L2PRN results

As mentioned in section 2.3.1, the first edition of 'Learning to Run a Power Network' was held in 2019 and the second edition is in progress now in 2020. The competition setup utilised a setup that is comparable to the one incorporated in this thesis, to an extent. The IEEE 14 bus test case was used and similar 1000 scenarios were used. The reward function used in this competition is also the same as the one used in this thesis. But there are some crucial differences in the two methods. A non-exhaustive list of differences is discussed here.

- The backend used to compute powerflows was PyPownet in the competition. This year, the backend utilises pandapower instead.
- The defined thermal limits of the lines are different. The thermal limits incorporated for L2PRN were done so to ensure a particular amount of overloading (3%) over a single corridor in the grid, while using a baseline agent such as the RC agent. This is discussed further in [8].
- The action space used in their setup allowed more types of actions and was not constrained to bus-bar switching. The number of actions that are allowed were more and included line disconnections and re-connections. This was not considered here, as this involves dealing with a dynamic action space that changes at every time-step compared to the static one incorporated in this thesis.
- The length of the scenarios varied in both the setups as each scenario had a duration of 1 month for the competition setup as compared to 1 week as seen in this thesis.

The top winners of this competition (Geirina and Iowa State University) used other reinforcement learning approaches such as actor-critic, and dueling DQN respectively. The actor-critic approach learns a policy function directly, through the 'actor' module. Then, the 'critic' module criticises the actor by providing a new state after taking an action, in order to adjust and improve its behaviour. The DQN approach also utilised some other enforced rules such "don't do anything if all your margins are good enough, below a threshold of 80%". In fig 7.1, a comparison of the different policies and topology changes used by these agents can be seen in the form of a combined topological time-series. The 'DN GameOver' as seen in this figure is similar to the RC episode length as discussed in this thesis. The best performing agent (called LB in the figure), produces a topological time-series similar to the ones present in this thesis. Some actions get taken early on in the episode resulting in the change of a few elements (and hence causes a deviation from the reference topology), and stays in such configuration for the rest of the time. Some agents, like Easy and Medium Agent, find an optimal configuration early on in the episode and stay in this configuration which proves sufficient to operate the network and survive the entire target duration. Other agents continue to make a lot of changes during the course of the episode, while still traversing the entire duration of the episode.

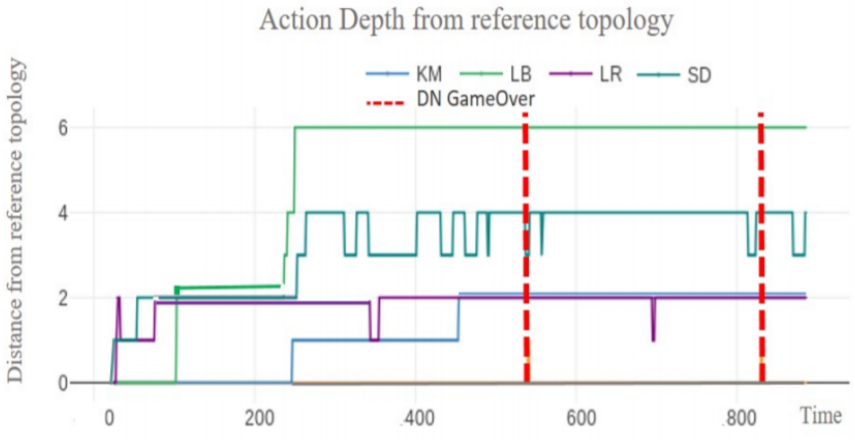


Figure 7.1: Results of the top 4 participants from the L2PRN competition [8]

7.2. Recommendations for future work

- A detailed sensitivity analysis can be conducted to see the direct influence of different parameters on the policy and training of the agents. The specifications used to train the agents in this thesis can be tested further to identify most influential parameters. A list of possible sensitivities is provided in table 7.1.

Table 7.1: A non-exhaustive list of parameters that can be used for sensitivity analysis

Parameters
Environment
Scenario(s)
Observation input
Action Space
Reward function
NN architecture
Learning Rate
Batch Size
Loss Function
Percentile for CEM
NN weights initialisation

  Some preliminary observations on which parameters can be influential show that learning rate and the way the neural networks are initialised can play an important role. Further, short-listing features that are most important can be crucial, and they can also could help vary the neural network architecture. Changing the reward function will play a role in how the agent learns, and different reward functions will produce agents that develop different policies.

- In this thesis, for input to the agent, the observations for only the current time-step were provided. Thus, the agent took actions for the next time-step only knowing the observations of the previous time-step. The model or agent can be provided with observations from multiple time-steps, say, the last 5 time-steps, to explore how the agent can sense a temporal change. Further, forecasts of RES and market-based output of conventional generation can be used for this.

- To create more generalised agents, the agents can be trained on a large number of scenarios by using a multi-environment approach. This was not done here, mainly since it can be computationally quite expensive. An intermediate step to training an agent on a large number of scenarios can also be to train an agent on 3 scenarios, one from each of the levels of difficulty.

- Another interesting area to explore is the dependence of an agent's behaviour on the particulars of a single power grid. A question that can arise is: Can an agent also be trained and generalised on not only different scenarios for a single power grid, but also be trained on different and arbitrary power grids?

- In this thesis, the duration of scenarios was restricted to a week. This was done mainly since the time required to train an agent can become quite large for longer durations. But other methods can be explored to successfully operate the grid for longer durations.
- Action space can get quite large for larger networks. Ensemble learning approaches could be used to combine the knowledge of differently trained models/agents. These agents could not just be trained using different settings, could also be trained for different subsets. For example, modes can be trained for different type of actions.
- This similar approach can be carried out for larger IEEE cases, and also be tested on subsets of the real-world Dutch grid.

Bibliography

[1] N. Zhou, "Oxford Dictionaries declares 'climate emergency' the word of 2019," *The Guardian*, Nov. 2019. [Online]. Available: https://www.theguardian.com/environment/2019/nov/21/oxford-dictionaries-declares-climate-emergency-the-word-of-2019

[2] A. Grübler, "Diffusion: long-term patterns and discontinuities," in *Diffusion of technologies and social behavior*. Springer, 1991, pp. 451–482.

[3] A. Grubler, C. Wilson, and G. Nemet, "Apples, oranges, and consistent comparisons of the temporal dynamics of energy transitions," *Energy Research & Social Science*, vol. 22, pp. 18–25, 2016.

[4] E. Espe, V. Potdar, and E. Chang, "Prosumer communities and relationships in smart grids: A literature review, evolution and future directions," *Energies*, vol. 11, no. 10, p. 2528, 2018.

[5] N. Voropai and D. Efimov, "Operation and control problems of power systems with distributed generation," in *2009 IEEE Power & Energy Society General Meeting*. IEEE, 2009, pp. 1–5.

[6] N. Brinkel, M. Gerritsma, T. AlSkaif, I. Lampropoulos, A. van Voorden, H. Fidder, and W. van Sark, "Impact of rapid pv fluctuations on power quality in the low-voltage grid and mitigation strategies using electric vehicles," *International Journal of Electrical Power & Energy Systems*, vol. 118, p. 105741, 2020.

[7] "Securing the Electric Grid with Common Cyber Security Services Jeff Gooding - PDF Free Download," library Catalog: docplayer.net. [Online]. Available: https://docplayer.net/10771861-Securing-the-electric-grid-with-common-cyber-security-services-jeff-gooding.html

[8] A. Marot, B. Donnot, C. Romero, L. Veyrin-Forrer, M. Lerousseau, B. Donon, and I. Guyon, "Learning to run a power network challenge for training topology controllers," *arXiv preprint arXiv:1912.04211*, 2019.

[9] A. M. Prostejovsky, C. Brosinsky, K. Heussen, D. Westermann, J. Kreusel, and M. Marinelli, "The future role of human operators in highly automated electric power systems," *Electric Power Systems Research*, vol. 175, p. 105883, 2019.

[10] S. Stevens-Adams, K. Cole, M. Haass, C. Warrender, R. Jeffers, L. Burnham, and C. Forsythe, "Situation awareness and automation in the electric grid control room," *Procedia Manufacturing*, vol. 3, pp. 5277–5284, 2015.

[11] A. Kelly, A. O'Sullivan, P. de Mars, and A. Marot, "Reinforcement learning for electricity network operation," *arXiv preprint arXiv:2003.07339*, 2020.

[12] "Grid maps - TenneT." [Online]. Available: https://www.tennet.eu/company/news-and-press/press-room/grid-maps/

[13] M. Naglic, "On power system automation: Synchronised measurement technology supported power system situational awareness," 2020.

[14] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.

[15] CBS, *Elektriciteit in Nederland*. CBS, 2015, vol. ?

[16] "Applying Lessons Learned from One of the Biggest Blackouts in History | School of Electrical and Computer Engineering at the Georgia Institute of Technology." [Online]. Available: https://www.ece.gatech.edu/news/272591/applying-lessons-learned-one-biggest-blackouts-history

[17] A. Q. Al-Shetwi, M. Z. Sujod, and N. L. Ramli, "A review of the fault ride through requirements in different grid codes concerning penetration of pv system to the electric power network," *ARPN journal of engineering and applied sciences*, vol. 10, no. 21, pp. 9906–9912, 2015.

[18] F. Milano, *Power system modelling and scripting*. Springer Science & Business Media, 2010.

[19] H. Koglin and H. Müller, “Corrective switching: a new dimension in optimal load flow,” *International Journal of Electrical Power & Energy Systems*, vol. 4, no. 2, pp. 142–149, 1982.

[20] A. A. Mazi, B. F. Wollenberg, and M. H. Hesse, “Corrective control of power system flows by line and bus-bar switching,” *IEEE Transactions on Power Systems*, vol. 1, no. 3, pp. 258–264, 1986.

[21] M. Heidarifar and H. Ghasemi, “A network topology optimization model based on substation and node-breaker modeling,” *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 247–255, 2015.

[22] T. Van Acker, D. Van Hertem, D. Bekaert, K. Karoui, and C. Merckx, “Implementation of bus bar switching and short circuit constraints in optimal power flow problems,” in *2015 IEEE Eindhoven PowerTech*. IEEE, 2015, pp. 1–6.

[23] E. Nasrolahpour, H. Ghasemi, and M. Khanabadi, “Optimal transmission congestion management by means of substation reconfiguration,” in *20th Iranian Conference on Electrical Engineering (ICEE2012)*. IEEE, 2012, pp. 416–421.

[24] R. Van Amerongen and H. Van Meeteren, “Security control by real power rescheduling network switching and load shedding,” in *Proc. 1980 CTGRE Report*, vol. 32, no. 2, 1980.

[25] W. Shao and V. Vittal, “Corrective switching algorithm for relieving overloads and voltage violations,” *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1877–1885, 2005.

[26] Z. Zbunjak and I. Kuzle, “Sips development method and busbar splitting scheme supported by pmu technology,” 2018.

[27] A. Smola and S. Vishwanathan, “Introduction to machine learning,” *Cambridge University, UK*, vol. 32, p. 34, 2008.

[28] T. M. Mitchell *et al.*, “Machine learning,” 1997.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[30] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[31] “10 Companies Using Machine Learning in Cool Ways,” library Catalog: www.wordstream.com. [Online]. Available: https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications

[32] V. Roman, “How to develop a machine learning model from scratch,” *Medium. April*, vol. 2, 2019.

[33] M. Caudill, “Neural networks primer, part i,” *AI expert*, vol. 2, no. 12, pp. 46–52, 1987.

[34] M. A. Nielsen, *Neural networks and deep learning*. Determination press San Francisco, CA, 2015, vol. 2018.

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

[36] “Kinds of RL algorithms,” library Catalog: ychai.uk. [Online]. Available: https://cyk1337.github.io/notes/2019/04/02/RL/SpinningUp/RL-taxonomy/index.html

[37] R. Y. Rubinstein, “Optimization of computer simulation models with rare events,” *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997.

[38] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L’Ecuyer, “The cross-entropy method for optimization,” in *Handbook of statistics*. Elsevier, 2013, vol. 31, pp. 35–59.

[39] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[40] I. Dahhaghchi, R. D. Christie, G. W. Rosenwald, and C.-C. Liu, “Ai application areas in power systems,” *IEEE expert*, vol. 12, no. 1, pp. 58–66, 1997.

[41] M. Kezunovic, “A survey of neural net applications to protective relaying and fault analysis,” *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 5, pp. 185–192, 1997.

[42] C. Ramos and C.-C. Liu, “Ai in power systems and energy markets,” *IEEE Intelligent Systems*, vol. 26, no. 2, pp. 5–8, 2011.

[43] 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, pp. 484–489, 2016.

[44] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement learning for electric power system decision and control: Past considerations and perspectives,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6918–6927, 2017.

[45] Ł. Kidziński, S. P. Mohanty, C. F. Ong, J. L. Hicks, S. F. Carroll, S. Levine, M. Salathé, and S. L. Delp, “Learning to run challenge: Synthesizing physiologically accurate motion using deep reinforcement learning,” in *The NIPS’17 Competition: Building Intelligent Systems*. Springer, 2018, pp. 101–120.

[46] “L2RPN challenge - Grid2Operate,” library Catalog: l2rpn.chalearn.org. [Online]. Available: https://sites.google.com/chalearn.org/l2rpn/grid2operate

[47] OpenAI, “Gym: A toolkit for developing and comparing reinforcement learning algorithms,” library Catalog: gym.openai.com. [Online]. Available: https://gym.openai.com

[48] “Welcome to Grid2Op’s technical documentation! — Grid2Op 1.2.0 documentation.” [Online]. Available: https://grid2op.readthedocs.io/en/latest/

[49] “DR POWER | Data Repository for Power system Open models With Evolving Resources.” [Online]. Available: https://egriddata.org/dataset/ieee-14-bus-power-flow-test-case

[50] “rte-france/Grid2Op,” Aug. 2020, original-date: 2019-10-10T17:04:46Z. [Online]. Available: https://github.com/rte-france/Grid2Op

[51] J. Brownlee, “Understand the impact of learning rate on neural network performance,” *Mach. Learn. Mastery. URL https://machinelearningmastery. com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/(accessed 12.12. 19)*, 2019.

[52] ——, “Why initialize a neural network with random weights,” *Machine Learning Mastery*, 2019.

[53] ——, “What is the difference between a batch and an epoch in a neural network?” *Machine Learning Mastery*, 2018.

[54] ——, “Loss and loss functions for training deep learning neural networks,” *Machine Learning Mastery*, vol. 23, 2019.