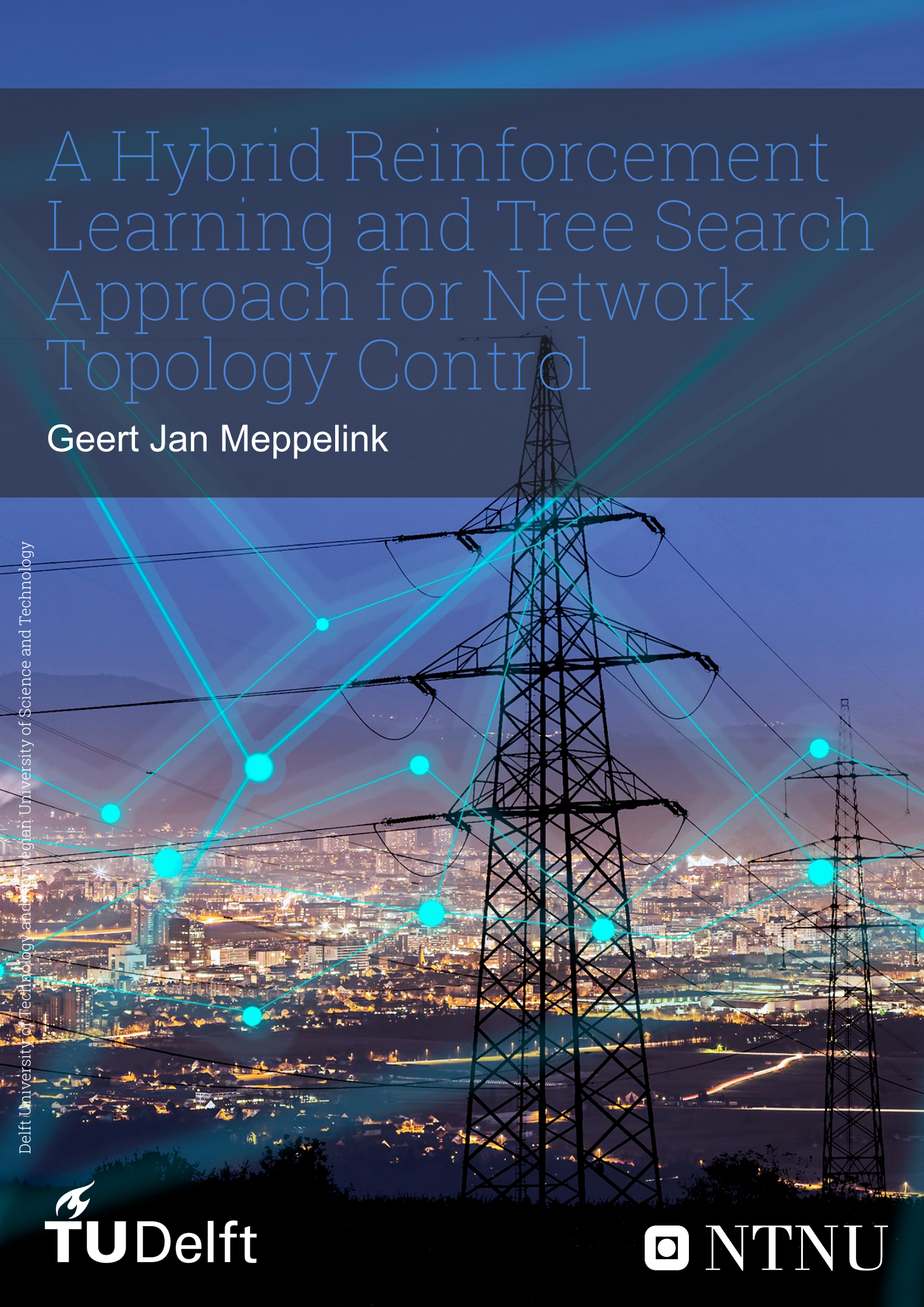
# A Hybrid Reinforcement Learning and Tree Search Approach for Network Topology Control

Geert Jan Meppelink



Delft University of Technology and Norwegian University of Science and Technology

TU Delft

NTNU

# A Hybrid Reinforcement Learning and Tree Search Approach for Network Topology Control

by


## Geert Jan Meppelink

| TU Delft | 4692810 |
|---|---|
| NTNU | 582216 |


to obtain the degree of MSc. Electrical Engineering
at the Delft University of Technology,
and the degree of MSc. Wind Engineer,
at the Norwegian University of Science and Technology,
to be defended publicly on 22 December, 2023 at 10:30 AM.

| | | |
|---|---|---|
| Project duration: | January 2023 - December 2023 | |
| Thesis committee: | Prof. Dr. Ir. Han La Poutré, | TU Delft, Chair |
| | Dr. Ir. Jochen L. Cremer, | TU Delft, Supervisor |
| | Ir. Ali Rajaei, | TU Delft, Daily Supervisor |
| | Prof. Dr. Ir. Olav B. Fosso, | NTNU, Supervisor |


Cover: Shutterstock

An electronic version of this thesis is available at http://repository.tudelft.nl/.


TU Delft NTNU

# Acknowledgements

With great pride and joy I can look back on my student life over the past years, this thesis marking the end of it. Following my studies throughout Delft, Budapest, Copenhagen and Trondheim has allowed me to follow interesting courses and explore great cities, enriching my academic and personal life. I would like to start with a word of thanks towards the people helping me along the way.

First of all, I would like to give great thanks to Jochen Cremer, Ali Rajaei and Olav Bjarte Fosso for their help and supervision during my thesis, aiding me in exploring the subject and an approach for a solution. I want to thank you for your professionalism and optimism during the project, guiding me in the right direction while giving me the needed confidence to finish this thesis. Furthermore I would like to thank Han la Poutré for taking the time to chair my thesis committee and providing me with valuable feedback during the final part of my thesis.

I am thankful for the people that enabled me to participate in the EWEM studyline, and I am most of all thankful for all the friends I have met along the way, making the past few years so memorable. Finally, I am very thankful for my parents and family, whose unwavering support and confidence helped me throughout my thesis and my student life.

Geert Jan Meppelink
*Delft, December 2023*

# Abstract

The escalating demand for electricity, fuelled by the widespread adoption of heat pumps, electric vehicles, and industrial electrification, is placing immense pressure on power grids, pushing them to their limits. This surge in demand, coupled with the integration of intermittent renewable energy sources, introduces challenges for ensuring a consistent, reliable, and secure electricity supply. Previous research has highlighted an underutilised flexibility in power grids through network topology control, altering connections to redirect power flows and mitigate transmission line overloads. However, conventional computational methods struggle to overcome the complexity arising due to the countless configurations. To tackle this issue, the French system operator, RTÉ, has initiated the 'Learning to Run a Power Network' competitions, urging participants to explore AI-driven solutions to allow future network operators to make informed decisions, ensuring economic, safe and sustainable power grid operations.

The proposed approach merges a curriculum-trained machine learning agent with a Monte Carlo Tree Search (MCTS) to enhance power network action security. The MCTS guides the simulation of potential actions, considering future outcomes for improved long-term performance identification. The methodology employs curriculum learning to generate labelled training data, with supervised imitation learning training a neural network to replicate optimal actions experimentally validated. MCTS is then used to secure these actions, leveraging outcomes in the training algorithm for enhanced sample efficiency and reduced training times. The approach uses MCTS-verified, simulation-tested actions for immediate training feedback, eliminating the need to wait for scenario completion, enhancing sample efficiency. An electrically distance-guided search in the MCTS improves convergence by prioritising actions closer to overflows, often found to be most influential in reducing violations.

The proposed approach shows the effect of the MCTS in promoting beneficial action trajectories, surpassing the brute-force baseline. Utilising the robust MCTS-based actions, a sample-efficient and stable training regime is created, utilising direct training, while the usage of the proximity factor shows potential in increasing convergence during simulation and training. However, increased research into optimising and setting training and testing parameter is needed to better the performance of the machine learning (ML) methods, especially before application within real-sized grids. While improved performance was shown through the usage of the applied method, optimal performance is hindered by the usage of solely topological reconfigurations. Further research is needed to combine the approach with methods for determining redispatch actions to reduce operational costs while keeping the system optimally safe and secure.

# Contents

# Nomenclature

## Abbreviations

| Abbreviation | Definition |
|---|---|
| AC | Alternating Current |
| AAT | Agent Action Threshold |
| AI | Artificial Intelligence |
| BFS | Breadth-first search |
| BSDF | Bus Split Distribution Factor |
| CSR | Compressed Sparse Row |
| DC | Direct Current |
| DDQN | Deep Duelling Q-Network |
| DER | Distributed Energy Resources |
| DCPF | Direct Current Power Flow |
| DQN | Deep Q-Network |
| DRL | Deep Reinforcement Learning |
| ESS | Energy Storage Systems |
| FACTS | Flexible Alternating Current Transmission Systems |
| L2RPN | Learning To Run a Power Network |
| LODF | Line Outage Distribution Factor |
| ML | Machine Learning |
| MCTS | Monte-Carlo Tree Search |
| MCTS-SG | Monte-Carlo Tree Search Safeguard |
| MDP | Markov Decision Process |
| OTS | Optimal Transmission Switching |
| PBT | Population Based Training |
| PPO | Proximal Policy Optimisation |
| PTDF | Power Transfer Distribution Factor |
| ReLU | Rectified Linear Unit |
| RL | Reinforcement Learning |
| RTÉ | Réseau de Transport d'Électricité |
| TPU | Tensor Processing Unit |
| UCB | Upper Confidence Bound |

## Symbols

| Symbol | Definition | Unit |
|---|---|---|
| $P$ | Power | [MW] |
| $\boldsymbol{v_\pi(s)}$ | State-value function | [-] |
| $\boldsymbol{q_\pi(s,a)}$ | Action-value function | [-] |
| $P(s,a)$ | Prior probability | [-] |
| $N(s,a)$ | Visit count | [-] |
| $\rho$ | Line loading | [%] |
| $\gamma$ | Discount factor | [-] |
| $\boldsymbol{\pi^*}$ | Optimal policy | [-] |

# List of Figures

# List of Tables

# 1

# Introduction

Power grids are the backbone of our electricity system, facilitating the transfer of electricity from the generation points to consumption locations. The rapid increase in electricity demand, driven by factors such as heat pumps, electric vehicles, and the electrification of industry, is putting immense pressure on power grids, pushing them to their maximum capacities [1]. Ensuring a constant, reliable, and safe electricity supply is a challenge for power system operators, particularly with the growing integration of intermittent renewable energy sources. These sustainable energy sources, such as wind and solar, address our critical need of reducing greenhouse gas emissions but bring additional uncertainty and limited flexibility. In previous studies researchers have shown an under-exploited flexibility of the power grid in the form of grid or network topology control [2, 3]. This network topology control provides a possibility to enhance the power grid's flexibility by altering the connections of its lines and configurations of its buses to redirect power flows and alleviate overloads in transmission lines.

In the face of increasing uncertainties within an expanding grid with myriad configurations, conventional computational methods are struggling with the complexity of finding optimal solutions for control. Network operators often default to experiential intuition or predefined manuals, potentially resulting in sub-optimal operation and, in extreme cases, blackouts. To address this challenge, the French system operator, RTÉ, have set up a series of annual competitions, called 'Learning to Run a Power Network', or L2RPN. Competitors are encouraged to combine their expertise in machine learning (ML) and power grids to investigate artificial intelligence (AI) driven solutions for network topology control, with a focus on a particular branch of ML, called reinforcement learning (RL). AI agents, using either RL, its advanced version, called deep reinforcement learning (DRL) or other ML methods are trained to control the power grid. The agents try to operate the power grid by altering generation set-points, changing the topology, charging/discharging storage units or curtailing renewables. The agents follows multiple objections: balancing generation and consumption, minimising losses, supplying all loads safely and especially preventing cascading failures which could lead to blackouts. The aim of these competitions is to offer a basis of investigation towards AI based techniques that could assist future network operators in making informed decisions to keep the power grid operated economically, safely and sustainably.

## 1.1. Conventional Approaches & Motivation

Transmission switching as a means of grid control was first put forward in the eighties by Koglin & Müller [4]. However, grids growing in size and complexity have made conventional computational methods struggle to find optimal control solutions in real time, forcing grid operators to rely on their experience or predefined manuals. This reliance can lead to sub-optimal outcomes or even blackouts.

In this context, the L2RPN competitions have been held by the French network operator, RTÉ, since 2019 [5–8]. In this competition competitors have explored the use of machine learning agents, particularly deep reinforcement learning agents, to control the grid using topological actions. These agents aim to use the yet unused, cost-effective flexibility inherent in interconnected power systems, utilising inexpensive topology changes instead of more costly redispatch actions [8].

The L2RPN competition brought the development of many various agents employing diverse combinations of machine learning techniques, simulation tactics, and heuristic rules. Notable methods include curriculum learning [9], which involves training agents in progressively challenging steps, and model-based approaches similar to AlphaZero, where promising actions are simulated to ensure their future impact [10]. While these agents have achieved excellent improvements in performance, they still encounter difficulties rising from the vast complexity of the action space.

The large number of potential topological rearrangements in a power system results in an expansive action space, hindering reinforcement learning agents in converging upon an optimal policy. Current approaches address these challenges through brute force action space reduction methods. The competitors from enliteAI, which have won the most recent L2RPN competition, reduce the original action space of 72.958 topological actions to the 2000 most frequent actions through a brute-force search [11]. This heuristic creates an action space which may not be best-suited for optimising redispatch actions. While this action space reduction method has been proven to work well, a clear answer for the optimal size of the reduced action space has not been presented, as a sufficient amount of actions should be available to keep the power system intact.

The consequences of actions can be drastic, as topological modifications often deteriorate the grid's ability to withstand disturbances or failures. During stable states, when supply meets demand with the grid components all operating safely within its limits, not changing the grid topology often proves to be the best course of action, whereas unsuitable topological actions during critical grid states can trigger cascading outages and blackouts. Currently, heuristic expert rules are in place that prevent the agent from taking actions during stable operations and aim to restore the grid to its optimal, fully connected state. Remedial actions taken during critical states should therefore not put the system at risk.

## 1.2. Proposed Contributions

This thesis introduces a hybrid approach, combining a curriculum-trained machine learning agent with a specialised MCTS, to improve the security of proposed actions in the power network. By incorporating future outcomes through selective simulation of projected suitable actions, the MCTS aids in selecting actions that show better long-term performance. The expert rules frequently employed by L2RPN competition agents are elaborated upon and evaluated. The implications of using a reduced action space is investigated, identifying the trade-off between performance and complexity. Such an optimal reduced action space ensures both flexibility for addressing overloads and fast computational speeds, to adhere to operational time constraints.

The proposed method utilises curriculum learning to generate labelled training data for a neural network. Through supervised imitation learning, the neural network is trained to mimic the optimal actions in specific states, the data of which are experimentally realised. The MCTS is used to improve the security of its actions. A training algorithm based on MCTS outcomes is used to improve sample efficiency and reduce training times. The rationale is that MCTS-verified actions, being simulation-tested, offer immediate training feedback, eliminating the need to await scenario completion, and improve sample efficiency, as better actions can be found directly.

Additionally, an electrically distance guided search is added to the MCTS to improve the convergence process. Electrical distance, measured using a breadth-first search (BFS), measures the lowest line count between an overload and a potential action, termed as 'hops'. This BFS metric remains consistent even when topological actions have significantly altered grid connections. By prioritising actions based on electrical distance, the MCTS is guided towards actions closer to overloads, often the most influential, thereby streamlining its selection process.

The performances of the different agents are compared using two baselines, do-nothing and brute-force. The performance, training time and sample efficiency are analysed for different training regimes, and the increased security of the MCTS is quantified.

## 1.3. Research Questions

This sections outlines the primary academic contributions of this thesis, along with their associated research questions:

**Objective 1:** Develop a hybrid network topology control agent, utilising a curriculum-learning pre-training regime and a Monte-Carlo Tree Search during operation.

- Q1 How well can a hybrid agent leverage both the pre-training sample-efficiency from the curriculum-learning regime and the improved security of the Monte-Carlo Tree Search?
- Q2 To what extent is the agent capable of maintaining network operations within limits using only network topology control actions?

**Objective 2:** Utilise the capabilities of the Monte-Carlo Tree Search to generate training samples for the agent to optimise its performance.

- Q1 Can the robust actions proposed by the Monte-Carlo Tree Search, tested through simulation, be utilised as viable training samples for the agent?
- Q2 How significantly do the Monte-Carlo Tree actions improve the sample efficiency and performance stability during training?

**Objective 3:** Incorporate the proximity of actions to the overflow as additional information in the Monte-Carlo Tree Search.

- Q1 How can the proximity be considered for estimating electrical distance and enhancing Monte-Carlo Tree Search convergence?
- Q2 How effectively does the inclusion of a proximity factor improve the convergence of both the Monte-Carlo Tree Search and the agent action policy during training?

## 1.4. Thesis Outline

Chapter 2 discusses the necessary background for the thesis. This includes a concise description of network operation, network security and transmission switching, as well as a foundational level clarification of ML, RL and the usage of the MCTS in the AlphaGo/Zero algorithms. Additionally, this chapter describes the details regarding to the L2RPN competition. After this Chapter 3 will establish an overview of conventional and state-of-the-art methods for network topology control, and the barriers it still faces. After this the two approaches, a curriculum-learning and a MCTS agent, are outlined, which form the basis of the proposed method. The chapter concludes with a review and discussion on the different methods for determining electrical distance within the power grid. Chapter 4 sequentially outlines the proposed approach for combining curriculum-learning and a MCTS, its training regime and how electrical proximity can be used to improve the convergence during operation and during training. Chapter 5 presents the case studies. Firstly, the hyperparameter analyses used for determining optimal operating settings are described, after which two sets of case studies highlight the performance of the MCTS security, and electrical distance factor, respectively. Finally, Chapter 6 provides a discussion on the results of the method, drawing a conclusion on the research questions and providing suggestions for future work. A schematic of the thesis outline can be found in Figure 1.1.

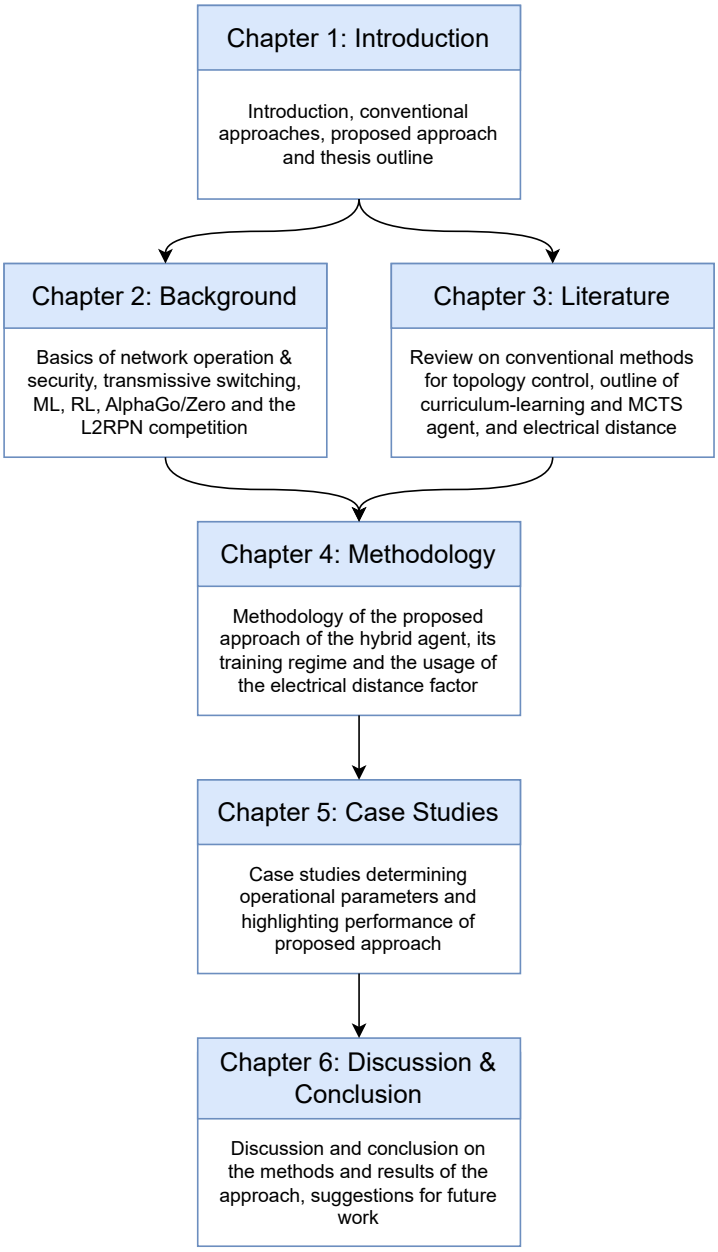



**Figure 1.1:** Flowchart illustrating the thesis outline

# 2

# Background

## 2.1. Network operation

Power networks have undergone steady developments to become the structures that they are today, the backbones of societies. Where historically power was predominantly generated by large thermal units, burning fossil fuels such as coal, oil and gas, currently the drive for decarbonisation has led to an increased share of renewable and variable sources such as wind, solar, hydro and distributed energy resources (DER). The fossil fuels which were used to power the large thermal units were able to be transported over land, allowing the power plants to be situated closely to the hubs of consumption. The shift towards more carbon-free sources sees generation harnessing the power of natural resources, which are frequently located in more remote areas. Combined with a larger share of smaller, dispersed DER, the power generation distribution has changed in many ways over the last century, necessitating more flexible operation to account for increased variability.

At their core, power networks are governed by three fundamental constraints, which need to be met to ensure their seamless operation.

- **Thermal limits**. Upper limit to the amount of power that can be transferred. Surpassing these limits for extended periods of time will likely amount to the failure and/or disconnection of the equipment.
- **Voltage regulation.** The voltage needs to be kept within a defined range to ensure secure operation, where large, unexpected deviations can cause severe disturbances in the system. A larger share of renewable energy integration adds to the challenge by introducing more voltage variability [12].
- **Generation - load balance.** Every power network operates on, or as closely around as possible, the equilibrium between the power generated and the power consumed, as prolonged or significant deviations from the nominal values can destabilise the entire grid.

In modernising grids with increased levels of variable sources, congestion management needs to be applied more often to reduce the violations within the grid. Historically, congestion management is done by changing the active and reactive power output of different conventional power generators at different points in the system, called redispatching. Redispatching conventional generators is still considered to be the main source of flexibility in a system to cover the balance between supply and demand. Using redispatching often requires using less cost-effective power generation units or utilising excessive operational reserves which both make for a higher cost for operating the system. Per illustration, the yearly redispatching costs in the German transmission grid went from 13 million € in 2010 to 220 million € in 2016 [13]. These development within power network underscore the necessity for alternative methods for congestion management. Network topology control, including line switching or node switching, can offer flexible, cost-effective methods for congestion management. However, due to the computational complexity of the countless possible switching topologies and the lack of qualified topology optimisation methods, it is still underutilised within current grids.

### 2.1.1. Network Security

For the secure operation of power networks it is imperative that stability within the system is maintained both under regular operating conditions and during unforeseen contingencies. This means that:

- **During normal operations:** all parameters, including power flows on equipment, voltage, and frequency, should remain within predetermined real-time limits.
- **During contingency operations:** all parameters, including power flows on equipment, voltage, and frequency, should remain within predetermined limits after the loss of any one single element in the network. This is called N-1 security [14].

The essence of secure operation lies in adhering to the three fundamental requirements for normal operations; ensuring no breach of thermal limits on equipment, keeping voltage levels stable, and balancing power generation and consumption. However, the integration of renewable resources presents a new set of challenges for network operators. Given the intermittent nature of renewables, such as wind and solar, maintaining the networks three primary constraints becomes increasingly complex.

The requirement to adhere to all thermal limits in the system becomes increasingly difficult with a higher concentration of fluctuating inputs from renewable sources, which lack the inherent stability found in large thermal generation units, due to power electronic at the grid interface the fact that renewable energies do not often comprise of large spinning units, which can counter grid variations due to its large inertia [15]. Network operators need to use the right tools at their disposal to ensure thermal limit constraints during routine operations. These tools also need to used preemptively to ensure that the network operates within its limit after an expected or unexpected outage of any element in the system. Network limits also possess a temporal element. Lines or cables within a grid can tolerate temporary overflows while a solution is found to revert the grid back within its limits. An immediate overflow does not typically result in an instantaneous disconnection of an element. However, correct mitigation has to be undertaken such that a single overflow does not cause other components to malfunction. Constraining to the thermal limits during normal operations and for the N-1 contingency operational cases is vital in preventing simultaneous outages which can disturb the security of the entire grid. If the thermal limits are not adhered to, highly loaded lines might have to be disconnected for security. The power that was flowing through that connection will be rerouted, possibly raising nearby power flows past their thermal limits, which can cascade a string of outages in a highly loaded network, ultimately leading to a complete blackout [16].

The shift of intermittent renewable energy generation in the power grid does not only affect the stability of the thermal limits within the network, but also poses new challenges for balance control between the generation and the loads. Balancing power generation with consumption, which is needed to keep the frequency stability of the system intact, has become progressively more challenging due to the intermittent nature of many renewable energy sources. Renewable energy sources such as wind and solar are expected to dominate the European power production in the coming decades [17]. Even though the predicted production of these intermittent sources can be estimated, the variability throughout longer periods of time such as days or seasons can establish more difficult production patterns for the entire power mix. As production of wind and solar does not line up with general consumption patterns for users, other measures, such as demand response or temporary energy storage, have to be put in place such that the consumption in the grid can be met at all times [18].

In real-time, network operators employ a hierarchy of remedial actions to counteract system constraints, such as line thermal overflows:

- **Line switching:** connecting or disconnecting lines in the network.
- **Node switching/splitting:** splitting or coupling busbars at a substation, or changing which busbar an element is connected to. This groups or separates the connections within a busbar, or the connections through the node.
- **Generation redispatching:** increasing or decreasing the generation at certain locations in the network to increase or reduce the flows on the lines.
- **Load curtailment:** disconnecting a certain amount of load from the system which matches the lack of generation in the system to keep the frequency balance intact.

- **Generation curtailment:** similar to load curtailment, where an excess of power generation can be disconnected to retrieve a frequency balance. Can sometimes be necessary if renewable sources are producing too much power during periods of lower consumption [19].

Load or generation disconnection, while effective, is often a last resort due to its societal impacts and potential business disruptions. Similarly, redispatching generation, governed by market dynamics, can be cost-intensive. Using the inherent design flexibility of substations, which allows for partitioning or coupling, can be an efficient way to manage line overflows and increase the system security and operational efficiency.

### 2.1.2. Transmission switching

Transmission switching, which entails both line switching and node splitting/switching, is the act of changing the network topology by switching line or busbar connections in or out. By switching on/off of transmission network elements, such as transmission lines or substation connections, the network topology is altered, which then sees the active and reactive power flows rerouted as per the Kirchhoff laws. This switching on/off is done by closing/opening the circuit breakers already present in the current network topology. By rerouting the active and reactive power flows congestion management can be done in a cost-effective way, which keeps the system secure and economically operated. An example of using node splitting can be seen in Figure 2.1. In the example the overflowed line from the current state is switched to the second bus of the node, changing the flows in the network and reducing the overflow.

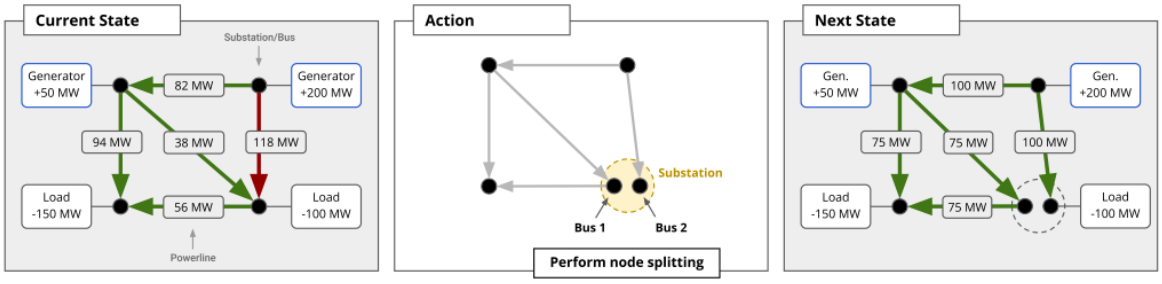



**Figure 2.1:** Example of node splitting to manage an overflow (Figure taken from Dorfer *et al.* [11])

Using the circuit breakers already in the system to control the network topology reduces operational costs as no investment on additional assets are needed. Transmission control switching can not only be used to try and tackle thermal flow limits, but can also be used as a way of coping with voltage violations. By rerouting the reactive power or switching off the line that produces the reactive power, often in lightly-loaded systems, the voltage violations can be helped. This also helps in reducing system operational costs compared to installing costly FACTS (Flexible AC Transmission Systems) devices.

With more intermittent renewable energy coming into the system more flexibility is needed. Currently limited network topology switching is used, only as a corrective mechanism by system operators based on their own operator knowledge or on predetermined look-up-tables. This is only done is contingency states to try and alleviate the thermal or voltage violations. These pre-determined switching actions can be based on assumed steady-state system conditions which can cause unwanted consequences during real-time operation. Reliable and large-scale system applicable simulators or frameworks are needed which can simulate the real-time effect of these switching actions before network operators can feel confident in applying network transmission control on a larger scale. It is expected that in the future network operators are assisted by software selecting possible options for resolving thermal flow or voltage violations, as can be seen in Figure 2.2.

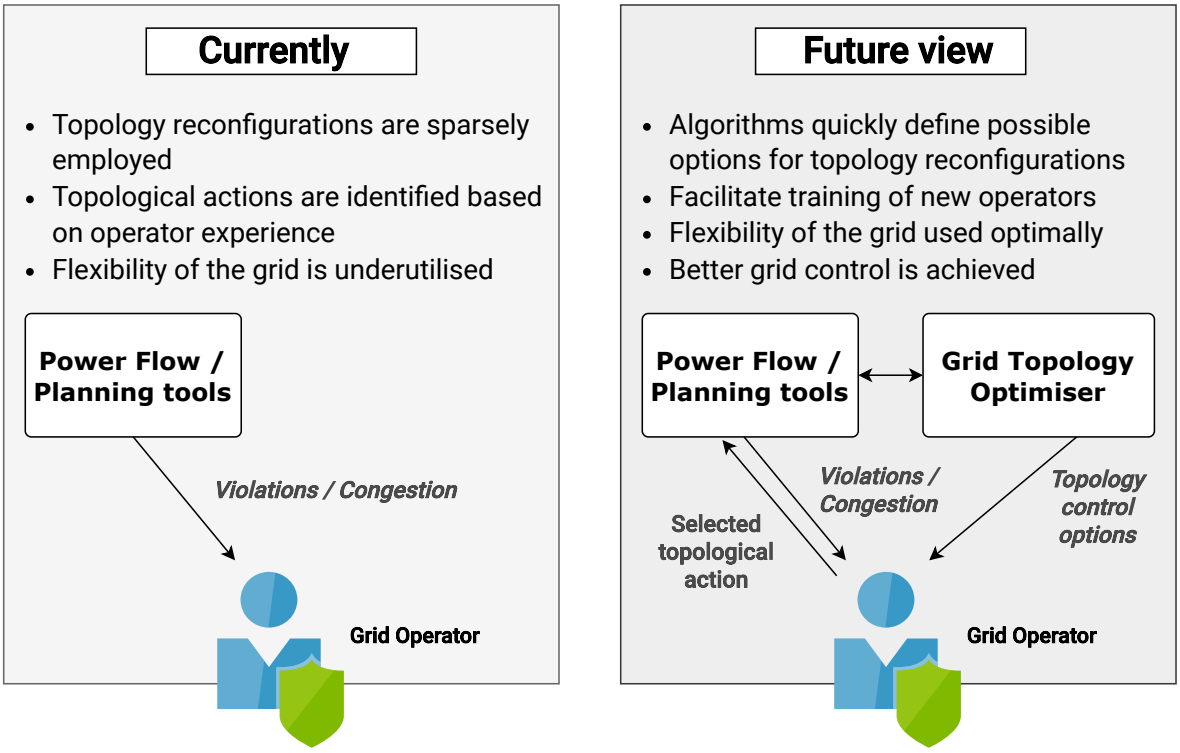



**Figure 2.2:** Future view of topology control integrated with system operator decision making (Adapted from Ruiz & Caspary [20])

## 2.2. Machine Learning

In global power networks, the flexibility from topological changes remains an underutilised, cost-effective solution for network security. The challenge lies in simulating the vast range of options in real-time with the current operator model and toolkit. Historically, operators relied on their experience for network issues, suitable for a static, predictable system. However, the evolving network, with its unpredictable resources altering flow patterns, demands more advanced control solutions. In this new control scenario ML agents can propose high probability switching or other control actions which can be simulated and checked by the system operator. ML can offer a good alternative when the computational complexity becomes too large so that a deterministic optimal solution cannot be found. ML is a subset of AI that focuses on building systems that can learn from data. Rather than being explicitly programmed to perform a certain task, these systems are trained using large amounts of data and algorithms that give them the ability to learn how to perform the task [21]. By using function approximations properties of neural networks high-dimensional action patterns, called policies, can be approximated and employed [22]. The goal of this thesis is to advance node-switching control based on combining ML and domain knowledge to offer a good alternative in situations where computational complexity becomes too substantial.

### 2.2.1. Neural Networks

A neural network is a computational ML model inspired by the way biological neural networks in the human brain work [23]. It consists of interconnected nodes, called neurons, organised into layers. A neural network can be used to learn and approximate complex, non-linear relations by adjusting the weights of the interconnections between the nodes based on training data. An example of a fully connected neural network can be seen in Figure 2.3. A neural network often comprises [24]:

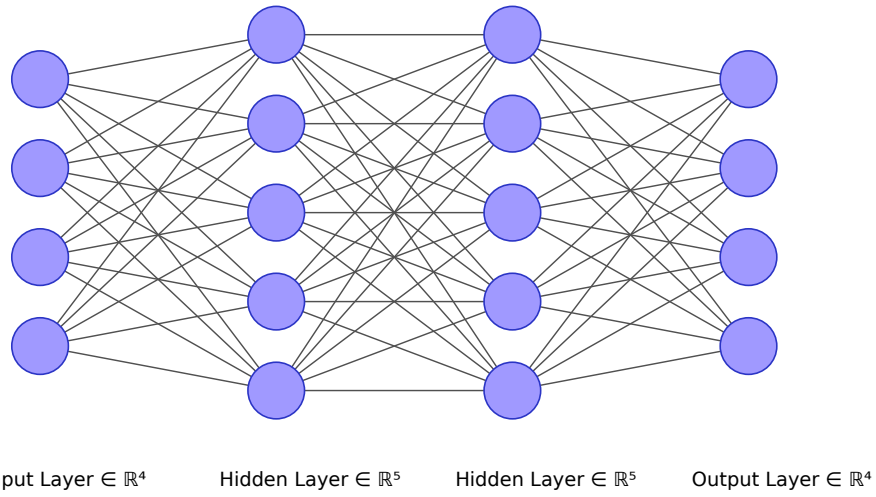



**Figure 2.3:** Schematic representation of a fully connected linear neural network, with one input layer, two hidden layers and one output layer

**Linear transformations:** $h_{k+1} = Wh_k$

**Non-linear activation functions:** $h_{k+2} = f(h_{k+1})$

**A loss function on the output, e.g.**

**Mean-squared error:** $l = \|y^* - y\|^2$

**Log likelihood:** $l = logP[y^*]$

Each node in the neural network propagates its input to the following layer using the linear transformation and non-linear activation functions. The weights of the nodal interconnections, which determine the impact the propagated values, are iteratively trained during the training process using back-propagation. This tuning algorithm calculates the gradient of the loss function with respect to each weight using the chain rule, and performs gradient descent on the outcome to converge the model towards the best-mapping solution [25]. Neural networks using these methods for training have shown impressive approximation capabilities [26, 27].

### 2.2.2. Training Neural Networks

Neural networks can be seen as the mechanism used for representation learning. For a task with a certain objective, the neural network can be trained to learn an approximate representation required to achieve the objective of the task, directly from the raw or pre-processed inputs.

#### Supervised Learning

One common approach to training ML models is supervised learning. In this training strategy, an algorithm is trained on labelled data, comprising input data and their associated correct outputs. Consider a set $\{(x, y)\}, x = 1 \dots M$, where $x$ is the input and $y$ is the corresponding correct output for $x$. The objective is to teach an algorithm to determine a function approximation $f(x)$ that can capture the relationship between the input and the output, such that $f(x) \approx y$. This learned function can then be utilised to accurately predict outputs for unseen input data [28].

Imitation learning is a type of ML where models learn by observing and imitating the behaviour of an expert. Various ML training regimes can be used to learn to the desired behaviour by mimicking the actions of an expert agent in similar situations. This approach can also be seen in the frame of a supervised learning regime, where labelled data of an agents optimal actions for different situations can be used as training examples. In cases where labelled training data is not readily available another form of training can be used, called RL, where an agent learns an optimal action policy by interacting with an environment.

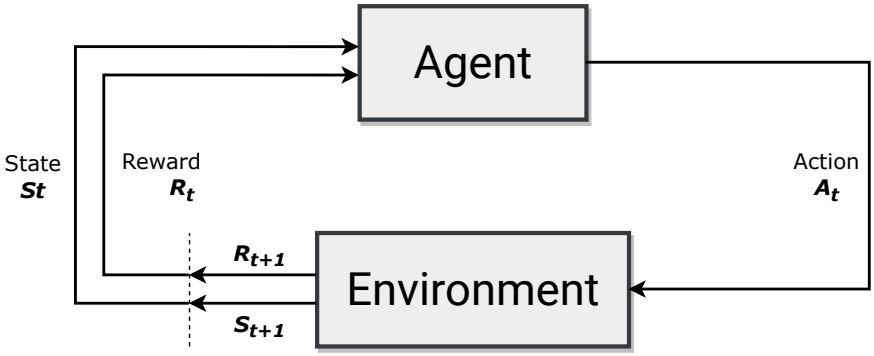



**Figure 2.4:** The agent-environment interaction in an MDP, showcasing states, actions and rewards (Adapted from Sutton & Barto [29])

### 2.2.3. Reinforcement Learning

In cases where labelled training data is not readily available another form of training can be used, called RL. RL is a type of machine learning where an agent learns an optimal action policy by interacting with an environment. The agent takes actions, receives feedback in the form of rewards or penalties, and adjusts its strategies to maximise cumulative rewards over time. This trial-and-error learning approach allows agents to discover optimal strategies even in complex, unknown environments.

The core approach of RL is learning through interaction. As the agent engages with its environment, it observes changes and the rewards it obtains. Actions taken might influence not only immediate rewards but also shape future outcomes, affecting subsequent rewards. Thus, an agent may opt for an action with a lesser immediate reward, anticipating to be rewarded more in the future. Based on action outcomes, the agent refines its strategy for future similar situations, which is similar to behaviourist psychology observed in humans [29].

The RL environment can be described as a Markov Decision Process (MDP), which is characterised by the following components:

- **States**: Observable conditions of the environment by the agent.
- **Actions**: Operations the agent can perform, influencing the environment and the received reward.
- **Reward function**: Specifies the reward based on the action taken in the current state.
- **Transition function**: Determines the resulting state from the action taken.

As the agent acts upon the environment, the state and reward are changed according to the reward and transition functions. The RL agent's task is to derive a policy (action strategy) that maximises the total anticipated reward:

$$\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \tag{2.1}$$

The variable $\gamma$ in this formula is called the discount factor, and is typically $\gamma < 1$. The reasons range from mathematical considerations, as to prevent cumulative infinite returns, to behavioural tendencies, where immediate rewards are favoured [30]. The RL process can be visualised as in Figure 2.4.

Generally, two primary strategies are considered to address reinforcement learning problems. The first strategy comprises methods utilising a value function, while the second strategy comprises methods centred on policy optimisation. For methods using value functions, the value or expected reward of occupying a specific state is calculated. The state-value function, represented as $v_\pi(s)$, indicates the anticipated reward for a state $s$ when following a certain policy $\pi$. Conversely, the action-value function, $q_\pi(s, a)$, predicts the reward for executing action $a$ in state $s$ and subsequently following policy $\pi$. To

estimate the optimal values for these functions, policy iteration is used. This involves a dual process: initially refining the value function using the policy, termed policy evaluation, and subsequently refining the policy with this enhanced value function, termed policy improvement. Policy improvement involves a 'greedy' approach, constantly selecting the option promising the highest expected reward. On the other hand, policy optimisation techniques strive to directly pinpoint the optimal policy, denoted as $\pi^*$. Here, a policy is established that directly correlates actions to input states, guided by specific parameters $\theta$. These parameters are updated in response to observed outcomes of interaction with the environment, refining the policy.

A crucial distinction in reinforcement learning refers to the use of model-based versus model-free methods. In model-free approaches, no system model is derived from interactions with the environment. However, in model-based methods, such a model is constructed based on interactions with the environment, or a physics-based model for the environment is constructed or available. Using this model the transition function allows the prediction of a subsequent state from the current state and a proposed action, allowing the need for actual action execution. Notably, several groundbreaking machine learning achievements, like the AlphaGo/Zero algorithms, have relied on parts of both of these methodologies [10, 31, 32].

#### Deep Reinforcement Learning

RL experiences many of the same hurdles that other algorithms face, in memory storage and computational complexity [22]. These hurdles can be tackled using DRL. DRL merges the trial-and-error methodology of traditional RL with the capabilities of deep neural networks. These networks, characterised by numerous hidden layers, adeptly address challenges like memory storage and computational complexity that RL faces. Leveraging the non-linearity of neuron activation functions, DRL can approximate abstract continuous functions, especially when the neural network has a sufficient number of neurons. By recognising and abstracting low-dimensional features from high-dimensional data, deep neural networks can effectively estimate optimal policies and value functions without depending on manually crafted features.

### 2.2.4. AlphaGo/Zero Algorithms

In recent years a breakthrough paper was released by researchers from the company DeepMind which introduced an algorithm called AlphaGo [31]. Employing RL and deep neural networks, this algorithm achieved the milestone of defeating a professional human player in the strategy game called 'Go', a feat which was thought to still be a century away. This influential algorithm forms the basis for the approach taken in this thesis, and will therefore be outlined in the following section.

While other perfect information games such as chess and backgammon had already found their machine learning breakthrough in the late 90's of the last century [33, 34], the game of Go was still thought of as too complex to be solved to a near optimal extent. The initial 2016 algorithm of AlphaGo, trained on expert human play and enhanced via reinforcement learning, combined two deep neural networks. One neural network for the policy and another for the value network, augmented by a specific tree search technique known as a MCTS. In a subsequent paper released in 2017 an updated algorithm, called AlphaGo Zero, was defined [32]. This updated algorithm was notable for three main refinements: firstly, no prior human knowledge was assumed, secondly, the value and policy network were combined into a single deep neural network, and finally, a simpler tree search was used, which beat the prior algorithm 100-0. This approach used by the AlphaGo Zero algorithm will be outlined here as its combined use of model-free (deep neural networks) and model-based (MCTS) ML approaches offers promising starting points for AI approaches for network topology control.

In the AlphaGo Zero algorithm a combined deep neural network is used for both the value network and the policy network. It outputs a value evaluation $\boldsymbol{v}$ and move probabilities $\boldsymbol{p}$, estimating the probability of winning and which actions to take from a specific state, respectively. Using this integrated network a MCTS is executed from the current position at each state, which outputs tree search move probabilities $\pi$ after completion of the simulation. This guided search outputs stronger actions compared to only the neural network, and can therefore be used to improve the action policy. The algorithm then plays against itself using this approach, and is able to use the outcome of the game at the end to improve

its value function. This can be seen as policy iteration through iterative policy improvement and policy evaluation to guide the neural network toward its optimal value.

In the MCTS, extensive game simulations from the current state shape the action policy, based on the most valuable actions observed. Each simulation starts from the starting state, called the 'root node'. Each state-action pair, ***(s, a)***, indicates a certain action taken from a specific state, and is also defined as an 'edge'. Each edge contains several parameters:

- ***P(s, a)***: the prior probability of taking action ***a*** from state ***s***;
- ***N(s, a)***: the visit count, equal to the amount of times the action ***a*** has been traversed from state ***s*** in the tree, and;
- ***Q(s, a)***: the average action value, equal to the average value of all nodes below this edge

Each simulation fully traverses down the tree until a state is found which has not been discovered yet, which is called a 'leaf' state. Action selection at each step is guided by a so called upper confidence bound (UCB), which is composed of the average action value ***Q(s, a)*** and a term based on prior probability and visit count. The chances to take an action increase with high prior probability and low visit count, to encourage exploration of possible high value actions. When a leaf state is encountered, its values get initialised using the neural network, which estimates a value for and the move probabilities from that position. After such a leaf state is reached the parameters ***N(s, a)*** and ***Q(s, a)*** in the subtree which got to that state are updated. After enough simulations the most visited actions are deemed to be the strongest, and the action policy is based hereafter. The MCTS algorithm is laid out in Figure 2.5.

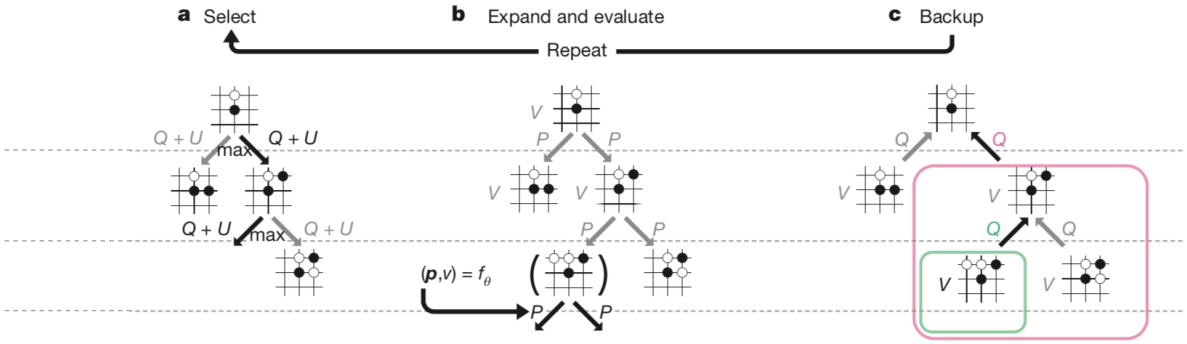



**Figure 2.5: Monte Carlo Tree Search in AlphaGo Zero.** a) Select: each action is taken based on the maximum upper confidence bound. b) Expand and evaluate: the neural network initialises the action probabilities and value for a leaf state. c) Backup: the visit count and average action value are updated throughout the subtree. (Taken from Silver *et al.* [32].)

Using the deep neural network and the MCTS the algorithm trains by playing full games against itself. Post-game, the network is optimised to better predict the value according to the final outcome of the game, and better align the predicted action probabilities with those derived from the tree search, constructing a reinforcement learning algorithm that surpasses human performance without pre-existing human knowledge.

In the following years up to now two more papers were published, updating the AlphaGo algorithm. AlphaZero, detailed in a 2018 paper [10], extended the approach to function solely on the game rules without any domain-specific knowledge, and was applicable across different games. The 2020 paper introduced MuZero [35], which, while still reliant on MCTS and built upon the AlphaZero framework, generalises to single-player games and non-terminating challenges. Furthermore, MuZero incorporates a reward function that allows scenarios with intermediate rewards, aligning closely with the dynamics of network topology control where such considerations are crucial for optimal functionality.

## 2.3. Learning To Run A Power Network (L2RPN) Competition

The application of ML and RL techniques in the realm of power networks has been seeing an increase in interest in recent years. This is highlighted in the series of competitions organised by RTÉ, the French transmission network operator, called 'Learning to Run a Power Network' (L2RPN). This initiative seeks to explore the usage of machine learning in enhancing the efficiency and reliability of power network operations [8, 36]. Their competition not only offers a simulator and baselines but also provides educational resources. While power systems experts may lack knowledge of machine learning, and ML experts might be unfamiliar with power systems, it is expected that a cooperative relationship between the two domains can yield significant benefits. Having started in 2019, this annual competition shifts its focus each year, addressing distinct challenges relevant to the future of power grids.

The competition models the real-time decision-making environment of transmission system operators, emphasising the secure and efficient management of the transmission network. Here, machine learning agents play a pivotal role. They have the capacity to reconfigure the network, through line and substation switching, to optimise line flows and prevent blackouts. Such agents can strategically use these switching actions to reduce operational costs, given that these switching options are substantially more economical than adjusting power plant outputs or resorting to curtailment. An illustration of this operational approach is depicted in Figure 2.6.

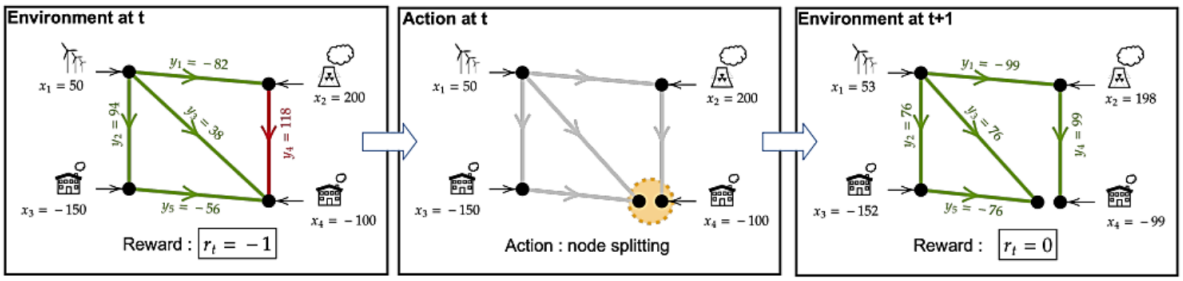



**Figure 2.6: L2RPN overview example:** An agent observes the power network at timestep ***t***, and sees that an overflow ***y4*** is present, and receives a negative reward. The agent takes the action to split a substation (node). The resulting topology does not experience any overflows and the reward increases. (Taken from Marot *et al.* [36].)

The open-source framework developed for the competition, called 'Grid2Op', lets users create and simulate through realistic environments of sequential operation scenarios. Grid2Op is based on existing non-linear physical power grid simulators and is able to simulate challenging but realistic scenarios representative of current and future problems in the power grid:

- The uncertainty resulting from the increased penetration of renewable energies, which will increase in the future.
- Static stability and robustness during line contingencies.

Their most recent framework offers a 118-node grid and realistic stochastic time series based on varying amounts of renewable energies. At the base of the control is the action space, consisting of cost-effective topology actions altering network connections and more expensive generator redispatch actions. Depending on the challenge, the action space can span up to 70,000 discrete actions and a 40-dimensional continuous action space.

Following a couple of feasibility trials, the competitions addressing challenges centred on real-world power grid issues started in 2020. That year, the competition tackled robustness and adaptability, separated in two distinct tracks. The following years saw an evolution of these tracks. For instance, the 2021 competition emphasised the development of reliable agents capable of issuing alerts during low-confidence scenarios. Meanwhile, the 2022 edition, which is most focused on the hurdles future grids will face, utilised grids characterised by high renewables penetration, thereby introducing increased variability and uncertainty.

### 2.3.1. Earlier competitions

#### L2RPN 2020

The 2020 competition featured two distinct tracks. The robustness track utilised a smaller 45-node grid where an opposing agent could initiate grid attacks through line disconnections, simulating potential outages due to weather anomalies or targeted assaults. This mirrors the real-world challenges of ageing grids, increasing extreme weather events due to climate change, and the growing need to factor in worst-case scenarios within computational limits. Subsequent competitions integrated this adversarial agent to better equip systems for such challenges. The goal of this track is to develop a method resilient in real-time under these adverse conditions.

The adaptability track, on the other hand, is set on a larger 118-node grid. Here, changing energy distributions test system flexibility limits. As the transition towards decarbonised power and more renewable energy sources advances, the system encounters increased unpredictability and variability in production. However, grid expansion lags. Agents in this track must manage energy compositions with varying amounts of renewable energy integration, delving into the extent of flexibility AI can bring amid slow grid developments.

Evaluation of participants was grounded in total operational costs per scenario, encompassing loss costs, redispatching costs, and blackout costs if not completed. Scores, for clarity, were normalised between [-100, 100]. An immediate blackout would score -100, doing nothing yielded a median score, while a score of 80 was achieved by safely navigating all scenarios. The ultimate 20 points were reserved for minimising losses while ensuring safety across all scenarios.

#### L2RPN 2021

The 2021 competition expanded on its 2020 predecessor, emphasising the trust dimension in AI agents. Given the utmost of importance of reliability, AI will not gain full autonomy rapidly. To thrive in a human-in-the-loop system, an AI agent must propose control actions, transparently indicating any low-confidence decisions. Trust, as articulated by the Trust Equation by Charles Green, depends on credibility, reliability, and intimacy, combined with the impression that the AI represents human's best interests. Through consistent transparency, predictable responses, and discerning alerts, trust is built, paving the way for future reliable autonomous systems. This edition also included the adversarial opponent that unpredictably disconnects lines, compelling agents to adapt to unexpected outages.

The performance of competing agents was calculated by weighing operating cost and alarm scores, as described by:

$$\mathit{Score} = 0.3 \times \mathit{Score}_{\mathit{alarm}} + 0.7 \times \mathit{Score}_{\mathit{Operation\ Cost}} \tag{2.2}$$

The alarm score evaluates timely and relevant alerting prior to a failure. With a constrained "attention budget", agents must be thoughtful in sending alarms. Optimal alerting occurs 35 minutes before a failure, but alarms within a 10-60 minute range also receiving lesser rewards. The grid has three distinct areas, and alarms specific to the correct malfunction zone earn higher rewards. An optimal alarm score is achieved without failures. If an alert precedes a failure, scores range between [0, 100], but a -200 penalty is applied for un-alerted failures. This competition iteration probes the cooperation between AI and human operators, emphasising the perception of AI on its own short-comings.

### 2.3.2. 2022 L2RPN competition

This thesis focuses on the 2022 L2RPN competition, where its 118-node grid is used for the development and examination of the proposed agents, and for performing case studies. A brief overview of the power grid environment and its rules is provided below. Please refer to the official documentation for all details [37].

#### Focus of the competition

In the 2022 L2RPN competition the emphasis is put on the uncertainties and challenges anticipated in future power grids [8]. Currently, research is being done to explore the feasibility of a 100% renewable

energy-powered network, dominated by fluctuating sources like wind and solar. While sustainable, these intermittent energies pose integration challenges in the power grid due to their unpredictability, stressing the necessity for some controllable, rotating-mass production in the grid. Additionally, distributed energy sources often reduce local power flows, elevating the risk of over-voltage issues.

This conceptual view of the future is modelled on a comprehensive 118-node grid, employing an RTÉ developed library called Chronix2Grid to produce energy mix time series. The tools from this library function as a sophisticated solver, matching required generation with diverse energy outputs, factoring in weather conditions, and minimising carbon emissions. Consequently, the competitive energy mix features only 2% carbon-emitting energy. Figure 2.7 depicts a sample time series for diverse energy sources, arranged by Chronix2Grid.

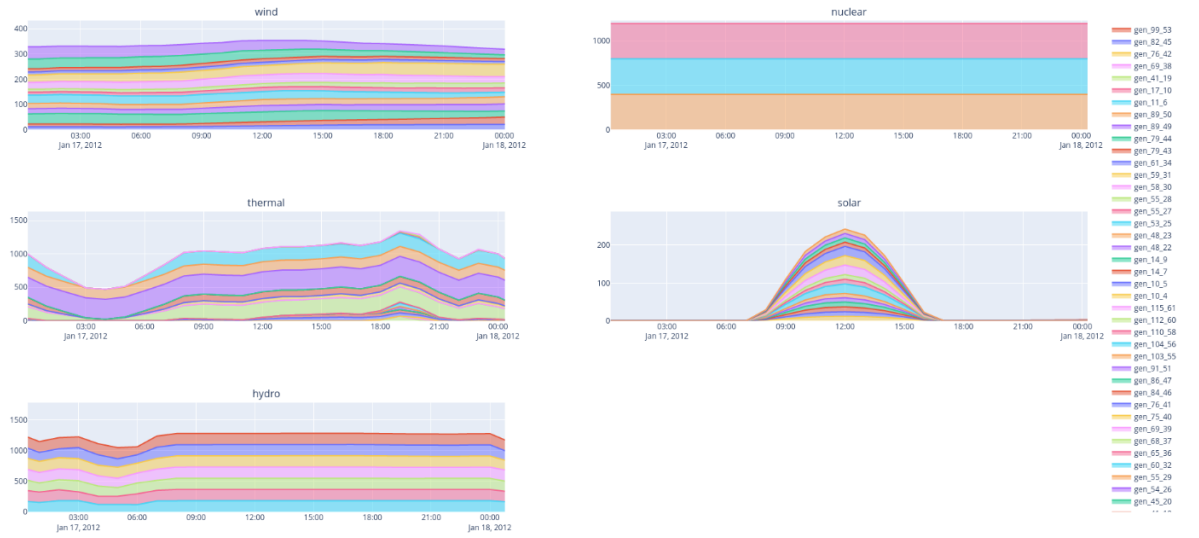



**Figure 2.7:** Example of synthetic time series for varying energy sources (Taken from Serré *et al.* [8].)

Combining elements from the 2020 competition's adaptability and robustness tracks, this edition introduces a high renewable energy quotient and retains the adversarial disruptions. Enhancements towards a more realistic grid include seven batteries for improved control and the ability to curtail renewable sources. The competition is reshaped into a new problem by having an updated simulator and data which represents future zero carbon scenarios. This competition tries to advance AI in the department of varying and unpredictable power system control.

### L2RPN environment design

The power grid consist of generators, load, power lines, substation and storages, which follow predetermined output curves as defined by the Chronix2Grid library for each scenario. A layout of the 2022 grid can be seen in Figure 2.8.

**Scenarios.** Scenarios in the Grid2Op environment consist of week-long simulations, built up out of 2016 timesteps of 5 minutes. All grid data is provided by a chronic, structured data that changes the input parameters of the system between one time step and another. A chronic provides data for generator setpoints, load consumption and structural information such as planned outages or unplanned hazards, the adversarial attacks. 32 years of power grid data in 5 minute intervals is available for developing and testing the agent.

**Generators, loads & storages.** Generators produce power, with renewables (sun, wind, hydro) influenced by weather simulations and thermal generators (coal, gas, nuclear) following set output curves. The outputs of generators can be adjusted: thermal power can be redispatched, and renewables curtailed. Loads consume power and adhere to an pre-defined load pattern, unknown to the agent. Storages can store power over time, allowing for charging or discharging.

**Substations and power lines.** Substations link elements (generators, loads, lines) and have two buses. Elements must be assigned to a bus, determining the grid configuration. Lines, connecting substations, possess thermal limits. Exceeding these limits for more than a brief period of time, two timesteps, leads to an auto-disconnect, with substantial overflows causing immediate disconnects and potential cascading failures.

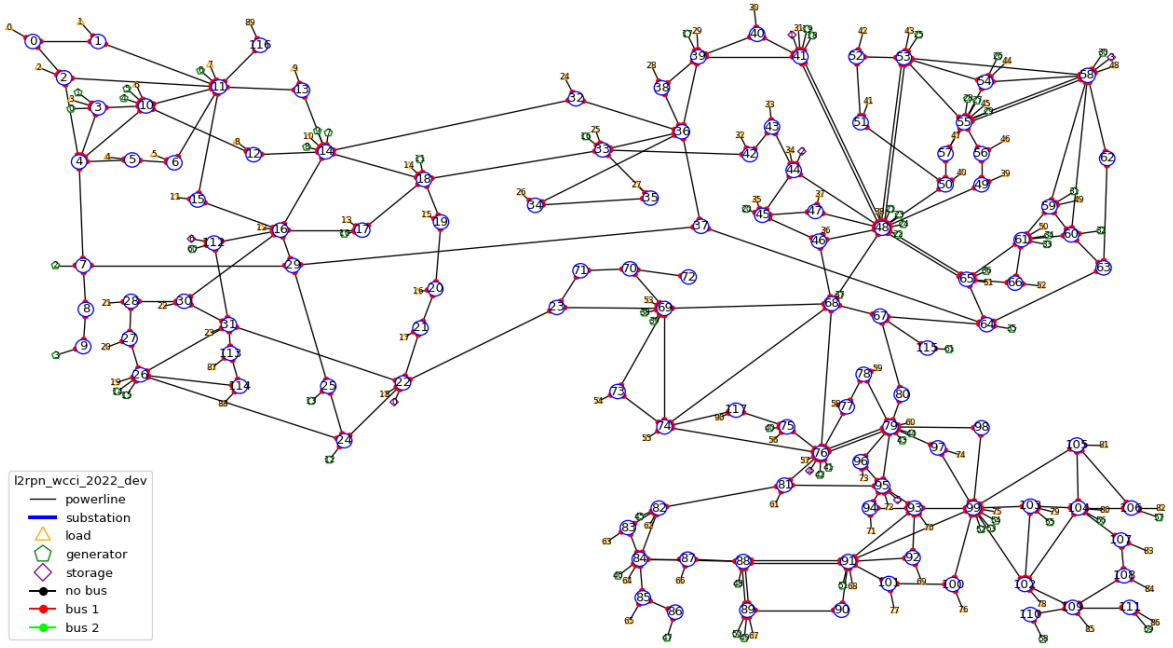



**Figure 2.8:** Schematic layout of the 2022 grid

The agents are tasked with managing the grid defined by the following:

- **Action space.** The algorithms can perform two main actions. They can modify the grid's topology or adjust power production. Specifically, there are over 70,000 discrete actions available for topology adjustments and a 40-dimensional continuous space for altering power production.
- **Observation space** At every interval, the algorithm can access a complete snapshot of the power grid status. This encompasses data on power nodes (both production and consumption), power line flows, and other relevant metrics. After each action, the simulated environment updates, providing the algorithm with a fresh observation.
- **Reward mechanism.** While participants have the liberty to customise their reward functions, the final standings of the competition were determined based on the cumulative operational cost of the network.
- **Simulation.** The agent has access to limited simulation for planning. Unlike the complete observation, it features forecasts of future load, generators, weather patterns and planned outages. Therefore, while the agent can use this to test the general impact of an action, precision is imprecise.
- **Game over.** The simulation scenario concludes if the grid fails to meet the total power demand.

# 3

# Literature review

## 3.1. Conventional Methods for Topology Control

Topology control has been a recent topic of interest for research as its applications can advance power system security and its ability to operate in a cost-effective manner. Increased interest has come from advancements in direct computational abilities or the advancement of good approximation methods based on ML. There are however still hurdles to overcome before topology control as an optimal control solution can be applied (semi-)autonomously by transmission system operators, and it is still an ongoing field of research.

### 3.1.1. Network Topology Control

The advantages of the power grid which can be found in the form of versatile transmission topology control has been shown in previous studies, for which its use is emphasised to currently remain under-exploited [2, 3]. Substantial challenges remain to be addressed and overcome in order for topology control switching to be feasibly implemented in large-scale systems. Existing research has been confined to small-scale models that used the needed approximations to facilitate computational manageability. The primary obstacle to scaling up these optimisation models is their computational complexity, which arises from multiple factors.

Transmission switching optimisation can be shaped as a mixed-integer programming problem, inherently non-linear due to the non-linear dynamics of the power system. Binary variables are employed to represent the switching status of transmission lines and substation links, while continuous variables denote the operational parameters of the system. In expansive networks with nodes numbering in the thousands, the computational complexity quickly escalates, aggravated by the need to incorporate additional security and power generation constraints. Present studies have primarily focused on the binary decision-making involved in transmission line switching, which already poses computational challenges for larger systems. Optimal transmission switching (OTS) using a DC (Direct Current) approximation of the power flow was used by a 2008 paper to convert the challenge to a mixed-integer linear problem [38]. Recent state-of-the-art studies since then have focused on approximating the AC non-linear problem to explore the flexibility to be gained by using OTS combined with other flexibility methods, such as a dynamic thermal rating (DTR) or increased energy storage systems (ESS) [39].

Little research has yet been done into substation switching, despite its potential for re-configuring active and reactive power flows [40]. The inclusion of substation switching would further amplify the number of variables and the associated computational complexity. State-of-the-art methods use various approximating methods to approach the massive solution space [41, 42]. ML methods, such as RL, are used to explore the flexibility of using substation bus-bar splitting, indicating efficient and diverse agent behaviour [41]. Further research is being done to explore the effectiveness of ML methods when dealing with real-sized grids. Deterministic methods for computing optimal topologies are held back by the complexity introduced by the bus-bar switching variables. A recent method introduces Bus Split

Distribution Factors (BSDFs), allowing rapid computation of the effects of busbar splitting on the DC load flow [42]. By modelling the busbar coupler as a branch of vanishing reactance and using transformed line outage distribution factors (LODFs) faster screening of proposed topological actions can be achieved, allowing for rapid N-1 checks, necessary for secure operation. These methods indicate the usage of smart formulas during DC approximations of the power flow to improve topology control. However, more research needs to be done to improve the methods towards actual grid use, including scalability towards multiple busbars in substations and the added complexity of larger, real-sized grids.

System security is another critical aspect that requires attention, particularly as the grid integrates more renewable energy sources, the variable nature of which introduces further uncertainty. Accounting for such unpredictability necessitates additional computational resources. The variable generation from renewables may also induce rapid changes in power flows, posing additional risks to system stability. Incorporating robust security protocols, such as the N-1 criterion ensuring that the failure of any single component does not endanger the system security, significantly intensifies computational demands. Inter-temporal constraints, which become relevant in control systems where switching is instantaneous but the ramping of generation or storage devices is not, have not been fully addressed in existing studies. Furthermore, while the DC Power Flow (DCPF) model is frequently used in power system simulations, the ACPF model may be necessary for comprehensive network topology control, or at the very least for validating the feasibility of solutions proposed by the DCPF model. This transition introduces a further increase in computational requirements.

### 3.1.2. Current usage and research barriers

The concept of transmission corrective switching emerged in the early 1980s, introduced by Koglin & Müller [4], characterised as a strategy to rapidly identify improved network configurations despite a vast array of switching possibilities. Their approach rests on the finding of potential switching actions through reduction methods and the rapid assessment of the proposed actions. Nowadays, developed topology control methodologies stemming from recent studies can be classified into three principal categories [20]:

- **Optimisation** of network topology under normal operational conditions without accounting for contingencies;
- **Prevention** of violating post-contingency states through preventive topology control during normal operational states, for planned outages and for co-optimising resource schedules for day-ahead planning, and;
- **Correction** during violating post-contingency states through corrective topology control to mitigate overflows.

Currently, protocols for system operators are in place to assist with voltage problems or transfer capabilities. Lightly loaded lines might be taken out of service to mitigate the capacitive effects on the voltages, while sometimes switching is done to increase capacity on other lines [39]. These switching actions and protocols are however based on operator experience and down-scaled simulations, and are not used for optimal transmission switching. While the under-exploited efficiency of busbar splitting has been shown in research, the lack of computational methods to propose optimal topology control actions has deterred current usage.

The objective of research across these domains is to alleviate existing or potential future violations and to reduce overall operational costs. The strategies proposed employ computational shortcuts, relaxations, or sophisticated heuristics to accelerate processing, as overcoming computational hurdles remains central to enabling the widespread application of transmission switching.

The issue of computational complexity is a recurrent theme, presenting a significant obstacle for real-time implementation in real-size extensive power systems. Investigations have tackled these challenges individually, but a comprehensive approach of all hurdles faced is necessary. The aforementioned elements: mixed integer programming, renewable integration and associated system security, temporal control constraints, and compatibility with ACPF, are integral to the evolution of power system management. These optimisation issues could be resolved using hypothetical, infinitely capable

mixed integer programming solvers, yet such advanced solvers are not currently available. Current research is increasingly focused on the potential of machine learning to address these complexities, with initiatives like the L2RPN competition playing a pivotal role in advancing this field.

## 3.2. Earlier competitors of the L2RPN competition

During the L2RPN competitions, numerous participants have attempted to achieve optimal performance by applying a diverse set of decision-making objectives, ML algorithms, domain expertise, or a union of these components. A thorough literature review of past contestants was conducted to identify shared strengths and common weaknesses. The aim is to propose a methodology that not only leverages their collective competencies but also addresses the persistent challenges that remain unmitigated.

In the 2019 L2RPN feasibility challenge, agents were tested on a small 14-bus grid, presenting a full action space encompassing 3,120 possibilities. This scenario, while not fully indicative of performance scalability to larger grids, revealed constraints and common challenges inherent even at this minimal level. Insights into prevalent difficulties and strategies emerged from this exploration. The most successful agent experienced several issues in its preliminary development phase. Initial attempts utilising a Deep Reinforcement Learning (DRL) framework with a Deep Q-Network (DQN [43]) that used the entire action space failed to converge, attributed to the vast dimensionality of the action space [44, 45]. Their final strategy implemented imitation learning to initialise the parameters of a Deep Duelling Q-Network (DDQN [46]). This approach, along with a reduced action space and the simulation and verification of the top-N actions deduced by the DDQN, underscored the value of combining machine learning techniques with domain-specific knowledge. Other methodologies, such as the one by Ramapuram A. et al. [47], applied similar techniques, opting for a reduced action space restricted to node-switching activities after identifying that line switching adversely affected grid stability. Additionally, a novel training regimen using curriculum learning was deployed, structuring the learning process of the agent through progressively complex stages to improve algorithmic convergence and reduce the need for vast environment interaction [48].

In the competition of the subsequent year, competitors refined their approaches to accommodate more complex systems in the robustness or adaptability track, working with a 45-bus and 118-bus system, respectively. The winning competitor used an innovative genetic algorithm to refine their policy network [49]. They, along with other contestants, adopted expert rules, such as simulating top action predictions from the policy network and applying domain-specific expertise to narrow the action space, to enhance stability [50, 51]. The runners-up integrated domain-knowledge informed expert rules with curriculum learning and Deep Reinforcement Learning (DRL), specifically employing the Proximal Policy Optimisation (PPO [52]) algorithm for the latter stages of neural network training.

Building on the robustness and adaptability tracks of the 2020 competition, the 2021 edition moved towards gaining trust in autonomous AI systems by integrating an alarm mechanism. This feature enabled agents to signal about upcoming decisions of low confidence. The highest-scoring agent of this competition deployed various modules tailored to specific operational goals. A low-level expert system was programmed to re-establish the grid back to its default, fully connected configuration when no overflows were present in the system, a strategy proven to be resilient to unanticipated contingencies. The core module executed single-step actions suggested by the policy network or devised multi-step strategies through a combination of the policy network and forward planning to propose sequences of actions. In scenarios where the primary agent module failed to resolve overflows, an emergency module was used to attempt alternative mitigation tactics such as power redispatch or the execution of pre-set emergency procedures. Additionally, an alarm module was incorporated to notify human operators during those critical states [53]. Deviating from most approaches, the second-place finisher did not use machine learning altogether in its control policy. Instead, it employed a brute-force methodology to derive a action space, simulating all possible actions during overflows to select the one minimising the system maximum loading [54]. The third-place participant, a company called enliteAI, adopted a Monte Carlo Tree Search (MCTS) approach similar to the one used in AlphaGo, as seen in section 2.2.4, complemented by a low-level expert rule system that restricted actions to critical situations, thereby enhancing operational efficiency and decision-making precision [55].

#### Earlier competitor review summary

The competitor review highlights a variety of strategies employed to address the topology challenge, yet similarities are found across the methodologies that underscore their critical nature:

- **Reduced Action Space.** Although not utilising the full action space may introduce a degree of bias in control policies, competitors universally adopt reduced action spaces. This is primarily because the machine learning or computational methods at their disposal are not able to handle such expansive computational dimensions.
- **Expert Rules.** A majority of agents rely on low-level expert rules to enhance system stability and resilience against unforeseen outages. The three principal expert rules involve: a) re-establishing connection of disconnected lines whenever feasible, b) resetting the grid to its default, fully interconnected state in non-critical situations, and c) requesting an action from the agent only when the maximum grid load exceeds a specified threshold.
- **Safety Check for Top-N Actions.** Rather than executing the highest-ranking action suggested by the policy, most agents introduce a layer of safety checking by simulating potential actions. This ensures the selection of the safest course of action.

In the subsequent sections, we delve deeper into two distinct competitor methodologies, as they form the foundational framework upon which this thesis proposes to build and enhance. These approaches have been selected due to their usage of the premises found in literature, while innovating their approach to the challenges inherent in topology control. The ensuing description aims to explore the strengths and limitations of these strategies, setting the stage for the introduction of the proposed method which seeks to incorporate the strengths of both approaches.

### 3.2.1. CurriculumAgent

Curriculum learning is the act of presenting tasks organised in order of complexity to a learning algorithm. Humans and animals are know to learn much better when the information they receive is not random but throughout progressively complex concepts, and research has shown that neural networks exhibit advanced behaviour, achieving improved generalisation and quality of local minima [56]. Leveraging such a training regime, in which the agent was build up using increasingly complex training tasks, Binbinchen's CurriculumAgent was able to achieve the second place during the 2020 L2RPN competition. Their '*Teacher-Tutor-Junior-Senior*' framework, in which a RL agent is trained after earlier imitation learning steps of a rule-based agent, is outlined. Initially, the *Teacher* employs brute force to evaluate all 66,918 possible actions topological actions, reducing the action space to a set of 208 impactful actions. These actions are used by the *Tutor* in creating observation–action pairs for the *Junior* model, a feed-forward network, to learn from. Finally, the *Senior* RL agent is trained within a Grid2Op training setting, initiated with the weights from the *Junior* to quicken learning and convergence. The complete framework will subsequently be discussed per module.

#### Teacher

While there are a total of 66,918 topological actions in the grid, not all actions are needed, as many action are found at the larger substations, where a significant number of actions results in similar topologies. This introduces unneeded complexity, which needs action reduction. The *Teacher* model undertakes this by exhaustively evaluating all the actions. This process involves simulating various scenarios, brute-force simulating through all actions and selecting the action that minimises the maximum loading present in the simulated next time step, and recording the results. Following extensive simulations, a reduced action space can be created by frequency, where the top-N most prevalent actions are to be induced in the reduced set. A schematic overview of the *Teacher* module can be seen in Figure 3.1.

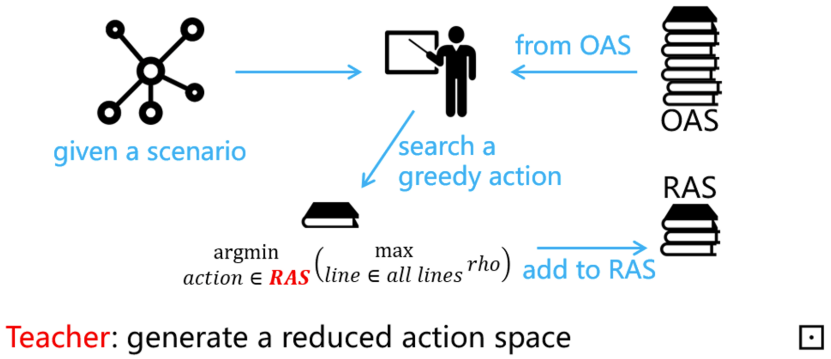



**Figure 3.1:** Schematic overview of the *Teacher* module (Taken from AsprinChina [57])

Tutor

Using the selected action set from the *Teacher*, a rule based *Tutor* module is constructed to generate training data for the *Junior* stage of the framework. This *Tutor* operates as a functional agent within the Grid2Op environment, capturing observations and corresponding actions to build its experience archive. The *Tutor* is also rule-based, where it will not interact with the grid if the defined threshold for the maximum loading in the system is not breached. If the maximum loading in the system does exceed the limit, the *Tutor* evaluates each of the actions from the reduced action set, greedily choosing the action that minimises the projected next-step maximum loading. Another expert rule is found in this module, which sees the agent automatically reconnect disconnected power lines if possible. A schematic overview of the *Tutor* module can be seen in Figure 3.2.

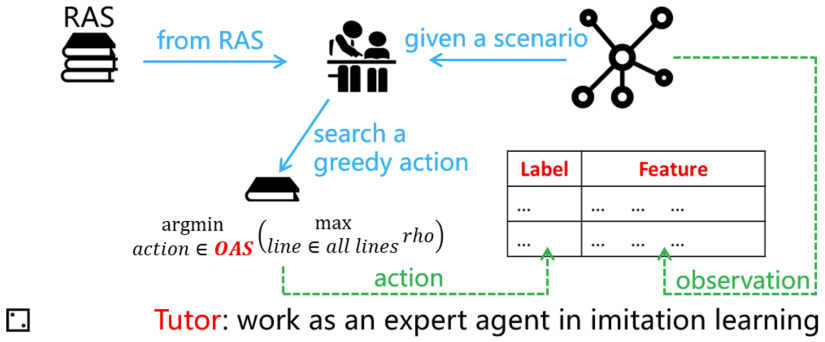



**Figure 3.2:** Schematic overview of the *Tutor* module (Taken from AsprinChina [57])

Junior

The *Junior* agent, a straightforward feed-forward neural network, tries to mimic the decisions by the *Tutor*. Its architecture is simple: the input layer accepts observations shaped like those from the *Tutor*, and the output layer corresponds to the topological actions of the reduced action set. In Binbinchen's design, the network features four layers, each with 1000 neurons using ReLU (Rectified Linear Unit) activation, to handle the data processing. The neural network is imitation learned to match the labelled one-hot encoded data from the *Tutor*, signifying the optimal action per observation. Its weights are thereafter used as a starting point for the *Senior* training regime. A schematic overview of the *Junior* module can be seen in Figure 3.3.

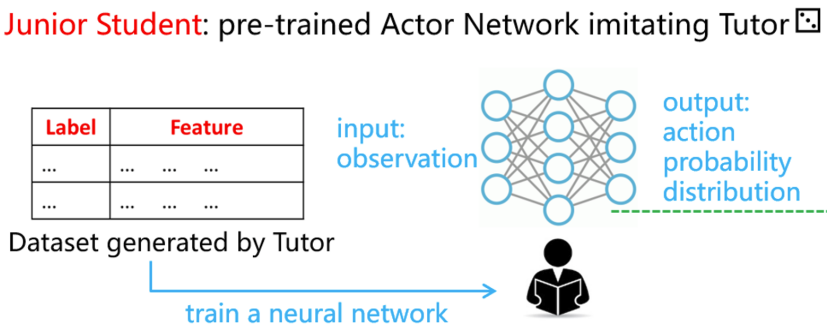



**Figure 3.3:** Schematic overview of the *Junior* module (Taken from AsprinChina [57])

### Senior

The *Senior* agent, developed as the final component in this framework, adopts the same neural network structure as the *Junior*. It is trained using the advanced Proximal Policy Optimisation (PPO) algorithm, and the initial weights of the *Senior* neural network are inherited from the *Junior* to facilitate a quicker learning curve. It is programmed to act only when critical conditions are met, specifically when the maximum loading of the system exceeds the threshold, except for necessary line reconnections. During its training, the *Senior* collects the standard Grid2Op rewards across steps. Upon reaching a stable performance, the RL agent is integrated with heuristic tactics, such as the operation limit and line reconnection, forming a sophisticated final agent. In this final stage the RL model is only asked to act when the set threshold is surpassed, then generating a sorted probability list of actions from the RL policy. This list is then methodically examined to select an appropriate action. A schematic overview of the teacher module can be seen in Figure 3.4.

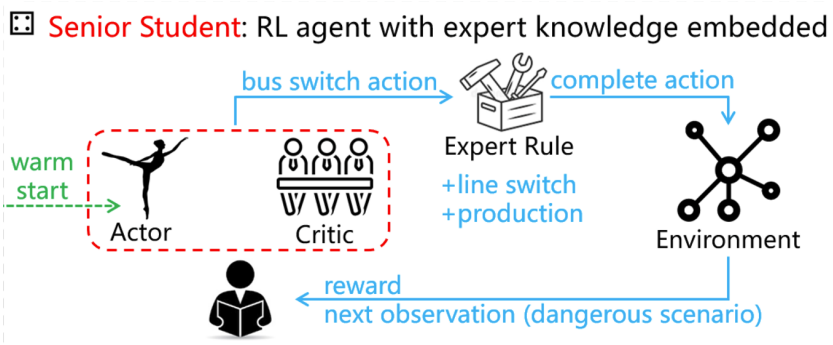



**Figure 3.4:** Schematic overview of the *Senior* module (Taken from AsprinChina [57])

## 3.2.2. enliteAI Agent

Building on their 2021 submission, as seen in section 3.2, enliteAI has developed a refined version of their L2RPN submission agent, which achieved the first rank during the 2022 version of the competition [11]. Their approach used an AlphaZero-based grid topology optimisation agent, leveraging its selection mechanism to streamline the simulation process such that control solutions are found, which are optimal considering future operation of the power system as well. Their MCTS algorithm and the expert rules that the agent adheres to are outlined in the following sections.

### Reduced action space

The expansive action space in the L2RPN 2022 grid, totalling 72,958 possible topological changes, excludes the direct application of AlphaZero given realistic compute constraints. Both the output layer of the policy network, and the MCTS branching factor would need to handle this vast number of possibilities. To address this, a brute-force search was performed to reduce the action space down to the 2,000 most common actions.

### Expert rules

The enliteAI agent also leverages some low-level expert rules to increase security, by ensuring agent inaction during non-critical states and having a default, most resilient topology. Their module called '*Grid State Observer*' determines whether or not a state is safe, defined using the maximum line loading in the system. Agent remedial actions are skipped during these states. During these non-critical states topology recovery is used as well to return the grid topology to its fully connected state.

### Monte Carlo Tree Search

At the core of the control algorithm stands a MCTS algorithm based on Google Deepmind's AlphaZero algorithm [32]. Each search through the search tree consists of a certain numbers of iterations, traversing the tree from the root node each time until a leaf state is encountered and added. The action selection at each node in the tree is governed by a smart selection process based on the UCB, that promotes exploration of high probability and little visited actions. Once the set amount of iterations has been reached, an action is taken based on long-term power system stability.

### Action selection process

The AlphaZero algorithm is trained tabula rasa, which necessitates massive data during training. 5,000 Tensor Processing Units (TPUs) for self-play game generation and 64 TPUs for neural network training in games like Chess, Shogi, and Go. To limit the resource demands, enliteAI introduced an MCTS early stopping protocol that stops the search when a sufficiently good solution emerges. The early-stopping protocol is based on two criteria: the count of unique recovery nodes in the search tree, or the existence of a node the end of a training episode. Recovery nodes are states in the simulation that were deemed stable by the *Grid State Observer* for a pre-determined amount of steps before requiring agent intervention again. The methodology stops the simulation when the number of recovery states hits a defined threshold, beginning the action selection algorithm. The action selection step would commence at the state of the tree in Figure 3.5, for the example with parameters of $t_{skipped} = 10$ and $t_{stopping} = 6$.

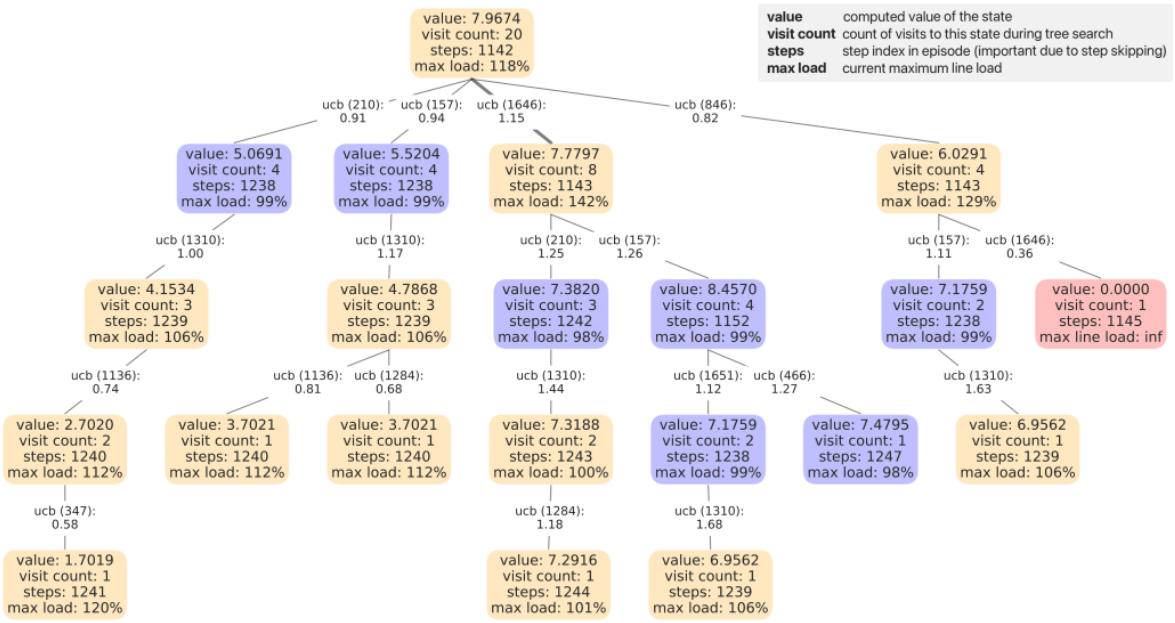



**Figure 3.5:** Example of a grid topology MCTS tree. Nodes represent states (root is the current state); edges are simulated actions. Yellow nodes are critical states, red nodes are black-out states and blue nodes are critical states that internally skipped steps because they were deemed safe by the Grid State Observer. Edges correspond to topology actions with the action ID (in brackets) and the corresponding upper confidence bound. (Taken from Dorfer *et al.* [11])

Due to this early termination of MCTS, the traditional action AlphaZero selection metric, visit count, cannot be used. Instead, the action leading to the child node with the highest number of survived steps in the tree is chosen. This coincides with selecting the action with action id 1646 in Figure 3.5, as.

#### Heuristic Value Function

The AlphaZero algorithm is modified by substituting a learned neural network with a heuristic value function derived from domain expertise. This adaptation speeds up training and yields more consistent learning outcomes, facilitated by the straightforward rewards structure of the power grid environment. Similar to AlphaZero, values for nodes in the MCTS tree are averaged from the branch values below, normalised by visitation frequency. New leaf nodes get an initial heuristic value rather than one predicted by a neural network, enhancing search stability in the variable-reward landscape of power grids. Their heuristic assumes rewards remain consistent in the short term, a reasonable expectation for power grids where significant changes are rare outside of unpredictable events.

## 3.3. Impact of Electrical Distance

The effective management of power systems involves a comprehensive understanding of the operational dynamics of the system, including how electrical distance impacts the reduction of line overflows. Electrical distance, defined through various metrics such as admittance and hop count, significantly influences the redistribution of power flows and the strategic interventions for overload mitigation. In this section the definition of electrical distance and its influence on mitigation solutions will be outlined.

### 3.3.1. Definitions of Electrical Distance

Electrical distance defines the closeness of elements in the power grid to one another, indicating a higher linkage and higher influence between power flow, voltage, or frequency. Electrical distance in a power grid can be defined using various metrics:

- **Admittance based.** Admittance, denoted as the inverse of impedance ($Y = \frac{1}{Z}$), provides a measure of how easily electrical current can flow in a system in between elements. In the context of a power system, the admittance between two nodes accounts for both the resistance and the reactance of the connecting lines. Glover *et al.* [58] suggests that lines with higher admittance are electrically closer, and therefore have more influence on power flow of each other.
- **Impedance based.** Similar to admittance-based but uses the impedance of lines to determine electrical closeness, which includes both resistance ($R$) and reactance ($X$). Impedance based electrical distance is more commonly used in protection relaying for fault detection and location.
- **Power Transfer Distribution Factors (PTDFs).** PTDFs indicate how power flows between different parts of the system will change in response to injections and withdrawals of power at various locations, providing dynamic picture of electrical distance based on load and generation [59]. PTDFs are valuable in managing congestion and adhering to system stability limits when evaluating the impact of proposed generation trades in the energy market.
- **Line Outage Distribution Factors (LODFs).** LODFs are similar to PTDFs, but indicate how power flows in lines in the system change in response to outages of other lines [60]. According to Wei [61], the values for the LODFs are higher between electrically close lines, indicating a significant impact between power flows in the lines.
- **Hop-Based Electrical Distance.** A hop represents a direct connection between two network elements. Hop-based distance can be explained as a topological estimate of electrical distance, independent of the physical properties of the lines. While it does not always accurately reflect the influence of one component on another due to neglecting line impedance or system connectivity, it does show the trend that hop-based closer elements in the grid are found to have higher electrical influence on one another [62].

### 3.3.2. Application of Electrical Distance

In practical terms, electrical distance can be used to identify operating actions that are electrically close to the affected overflowing line, as the effects are greater then. Generation dispatch can be altered based on Power Transfer Distribution Factors (PTDFs), or topology optimisation can be applied where alternate paths are identified using a metric for electrical distance, as targeting the closest branches to the overflows often results in the most significant reduction in overflow [63]. Hop-based methods to estimate electrical distance can be used to generate a quick estimate of influence, or in situations where connectivity parameters as impedance or admittance are not known.

Hop-based methods however, while not being a direct indication of the influence, but rather an estimate, are subject to many variables. It best estimates the electrical distance in systems where line parameters are similar, and grids are interconnected. In systems where line parameters vary greatly the estimate might be less precise, as hop-based distance is not dependent on physical parameters, disregarding the effect of the physical line properties. In highly meshed and interconnected grids faults or outages may have a stronger local effect, highlighting the usefulness of using hop-based estimates for electrical distance. However, in large parallel or radially connected systems the effects of outages might traverse much further due to its inherent 'closeness' of certain connections. Therefore, caution has to be taken before establishing hop-based approximations for electrical distance in certain systems.

In real-sized grids where exhaustive N-k simulations are not feasible, using electrical distance can help in quickly identifying corrective measures. Current methods for identifying contingencies do not depend on exhaustive searches, but on fast screening of most severe possible contingencies. Utilising a ranking based on performance indices contingencies are arranged based on their expected severity [64]. State-of-the-art methods are able to rank the contingencies using AC power flow methods, indicating action trajectories for power system operators [65]. Using AC methods for quickly determining the most impactful contingencies and electrical distance methods for rapidly finding topological corrective measures, practical solutions may be within real-time limits of operation for power grids.

In conclusion, the strategic manipulation of generation dispatch and topology optimisation, guided by electrical distance metrics, forms practical uses for alleviating overflows in power systems. These methods ensure that interventions are directed at the network elements most influential to the issue, providing a robust response to alleviating overflows. Hop-based approaches offer quick estimates of network influence for immediate decision-making or when connectivity details are scarce, where PTDF or LODF-guided actions allow for a more detailed understanding and system management, each method playing a vital role in maintaining the operation and stability of the power grid independently.

# 4

# Methodology

A hybrid approach combining a curriculum-trained ML agent with a specialised MCTS (MCTS) is presented. The proposed approach leverages the sample-efficient training regime of curriculum learning, while introducing an improved level of security by incorporating the simulation of promising actions through a guided tree search, improving performance during both training and testing. The search process is extended by including an electrical distance term to enhance the convergence process. For this, a quick BFS is used to calculate the lowest line count, measured in hops, consistent during any grid topology.

The proposed implementation is schematically visualised in Figure 4.1, where dotted lines indicate a transfer of data and connected lines indicate the usage of a function or module. Section 4.1 discusses the low-level expert rules used by all agents, and how to determine their operating limits. Section 4.2 outlines the curriculum-learning regime for the agent, while section 4.3 describes the improvements and training algorithm of the MCTS extension. In section 4.4 the electrical distance search is detailed.

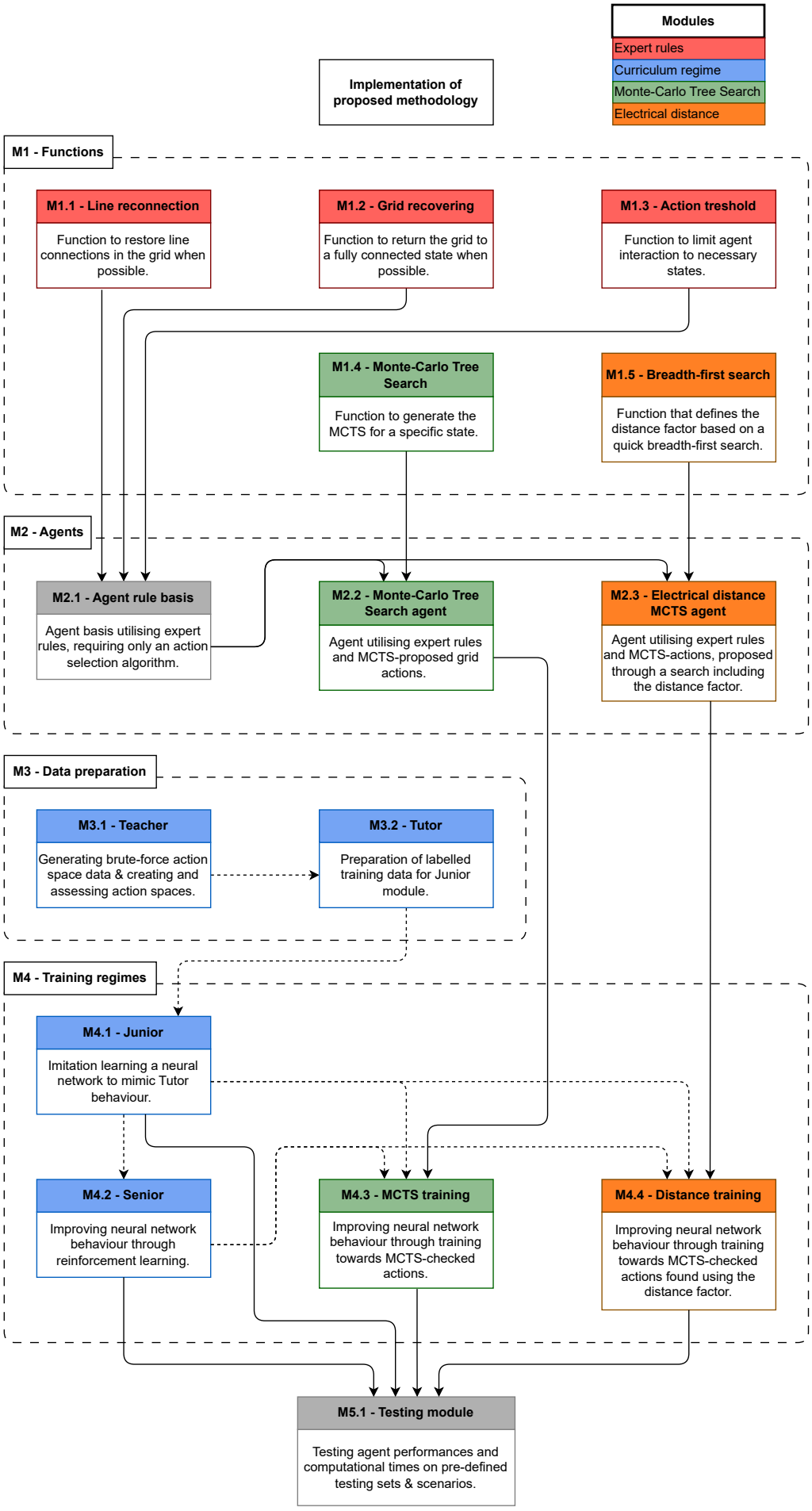



**Figure 4.1:** Full schematic overview of the workflow of the methodology.

## 4.1. Expert rules

In section 3.2 the discovered similarities between competing agents are highlighted and discussed. The following practices are found in most, if not all, of the agents, indicating their value:

- **Reduced action space**
- **Expert rules:** re-establishing line connections, recovering the grid to its most resilient, fully-connected state and having an action threshold for requesting agent actions.
- **Safety check for top-N actions**

The process of realising the reduced action space and its most optimal performance will be discussed in section 4.2, while the improved level of safety and security will be explained through the MCTS mechanism in section 4.3. This section discusses the application and implementation of the expert rules. A visual schematic overview of this part of the methodology can be seen in Figure 4.2.

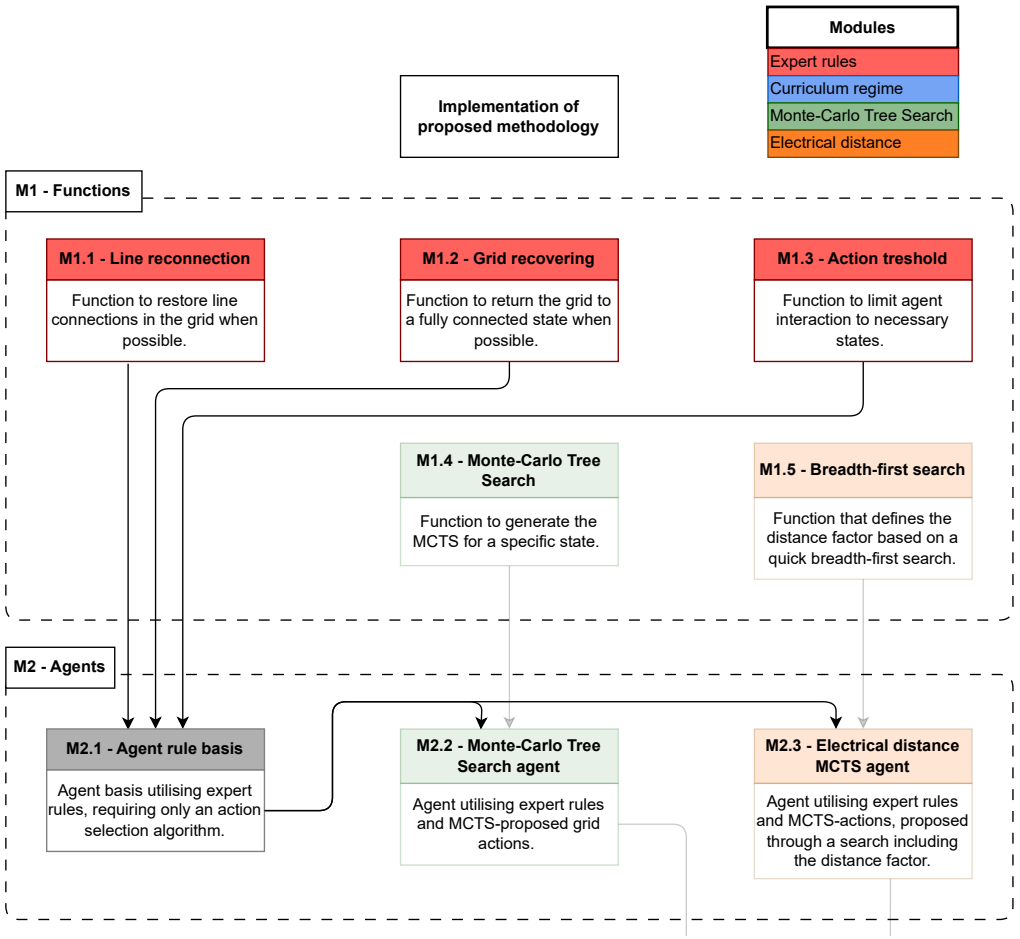



**Figure 4.2:** Modules of the methodology including the expert rules

### 4.1.1. Re-establishing line connections

Optimal grid performance is contingent on keeping all power lines operational, as it ensures full grid capacity and offers flexibility for topological changes that can prevent overflows. The Grid2Op environment provides control for this through parameters such as '*line_status*', '*connection_status*', and '*cooldown_time*'. The '*line_status*' parameter indicates if a certain line is functional or out of order, while '*connection_status*' details the buses each line is connected to. Any non-functional line that is disconnected due to an overflow or maintenance, has a non-zero value for '*cooldown_time*', which counts down the time until the line can be reconnected.

The line reconnection function first checks if there are any lines not in service, using the '*line_status*' values of the current observation. It considers '*cooldown_time*' to identify lines eligible for reconnection. To ensure reconnections do not deteriorate grid stability, the simulation capabilities are used to predict the impact on line loading. Actions that lower the maximum line flow in the grid are considered, and among those, the action that best reduces the maximum flow is selected. This methodical approach to line reconnection underscores the trade-off between maintaining grid capacity and ensuring stability. The pseudo-code for this function can be found in Algorithm 1.

**Algorithm 1** Line Reconnection

```
function reconnect_line(observation)
    if no disconnected lines then return None
    end if
    bestAction ← None
    bestReduction ← 0
    for each line in disconnectedLines do
        Skip line if still in cooldown
        Evaluate reconnection potential
        Update bestAction if improvement found
    end for
    return bestAction
end function
```

### 4.1.2. Automatic Grid Recovery

Multiple competitors in the L2RPN competition observed optimal grid resilience when all elements at the substations were were connected to the same bus. This default, fully-connected configuration maximised connectivity and flow distribution, and provided maximum flexibility for the usage of transmission control switching in critical scenarios.

Nodes in the Grid2Op environment possess the same operating parameter as lines in *cooldown_time*, determining when they can be acted upon. The state of connections at substations, whether to bus 1, bus 2, or disconnected, is captured by the vector *topo_vect*. This vector is checked to see if the grid is fully connected. If deviations are found, actions are considered if the *cooldown_time* has passed. Just like line reconnections, substations are assessed for their default state. Substation actions that lead to a fully-connected state are simulated node by node, chosen if they are predicted to optimise line load. If there are multiple beneficial actions, the one offering the most flow reduction is selected. The pseudo-code for this function can be found in Algorithm 2.

**Algorithm 2** Automatic Grid Recovery

```
function recover_topology(observation)
    if grid is fully-connected then return None
    end if
    bestAction ← None
    bestImprovement ← 0
    for each node in grid do
        if node not fully-connected then
            Skip node if still in cooldown
            Evaluate reconnection potential
            Update bestAction if improvement found
        end if
    end for
    return bestAction
end function
```

### 4.1.3. Agent action threshold

In power grid management, the topological actions of the agent taken can be critical. During safe states, where operational limits are within bounds, doing nothing is often the best action, where inappropriate grid reconfigurations can potentially destabilise the system. Agents are therefore only subject to propose topology actions when the system limits are breached, indicating a necessity for mitigating actions. To enforce this principle, an action threshold for agent actions is used.

This threshold is defined using the maximum line loading of the system for each step. If the maximum value of the line loading in the system is lower than the threshold, no agent actions are requested, and line reconnection or topology recovery actions are considered. If none of these actions are needed, the do-nothing action is employed. If the threshold is breached, mitigation efforts are needed, and the action proposed by the agent is considered. This can be an action chosen directly from a neural network, through extensive simulation or from a brute-force algorithm. The basic algorithm for all agents using expert rules is outlined in Algorithm 3.

**Algorithm 3** Expert Rule Agent Basis

```
function act(observation)
    maximum loading ← observation.max_rho()
    if maximum loading > threshold then
        bestAction ← GetActionFromAgentAlgorithm()
    else
        bestAction ← reconnect_line(observation)
        if bestAction = None then
            bestAction ← recover_topology(observation)
        end if
        if bestAction = None then
            bestAction ← doNothing
        end if
    end if
    return bestAction
end function
```

Setting the right value for the action threshold is an important process, as the correct trade-off needs to made between timely responding to critical states and preserving maximum resilience during safe operations. To assess the most optimal action threshold brute-force agents are employed, which differ in their strategy only in the values for the action threshold. To determine the optimal value, the performances are compared, based on the metric of maximum time steps survived over different scenarios. The results for the action threshold testing can be found in Chapter 5.

## 4.2. Curriculum learning regime

The curriculum learning approach for the agent is presented in this section. The used methods for creating and assessing the reduced action space is presented first, after which the preparation of the training data for the neural network is outlined. The neural network fitting and its subsequent training using RL are described in the ensuing sections. The neural network model developed during those sections are used further throughout this training as the basis for further model development or for baseline testing. Methods are based on Lehna *et al.* [66]. A visual schematic overview of this part of the methodology can be seen in Figure 4.3.

### 4.2.1. Reduced Action Space Generation ('Teacher')

The network of the 2022 L2RPN competition allows for 72,958 topological actions. Since it is computationally infeasible to evaluate all actions during real-time operations for their impact, a reduced set of actions is created, which allows the agent for optimal trade-off between computational time and flexibility. This condensed action space only comprises binary substation topological actions, which represent the bulk of potential actions. Actions involving line switching are excluded to avoid compro-

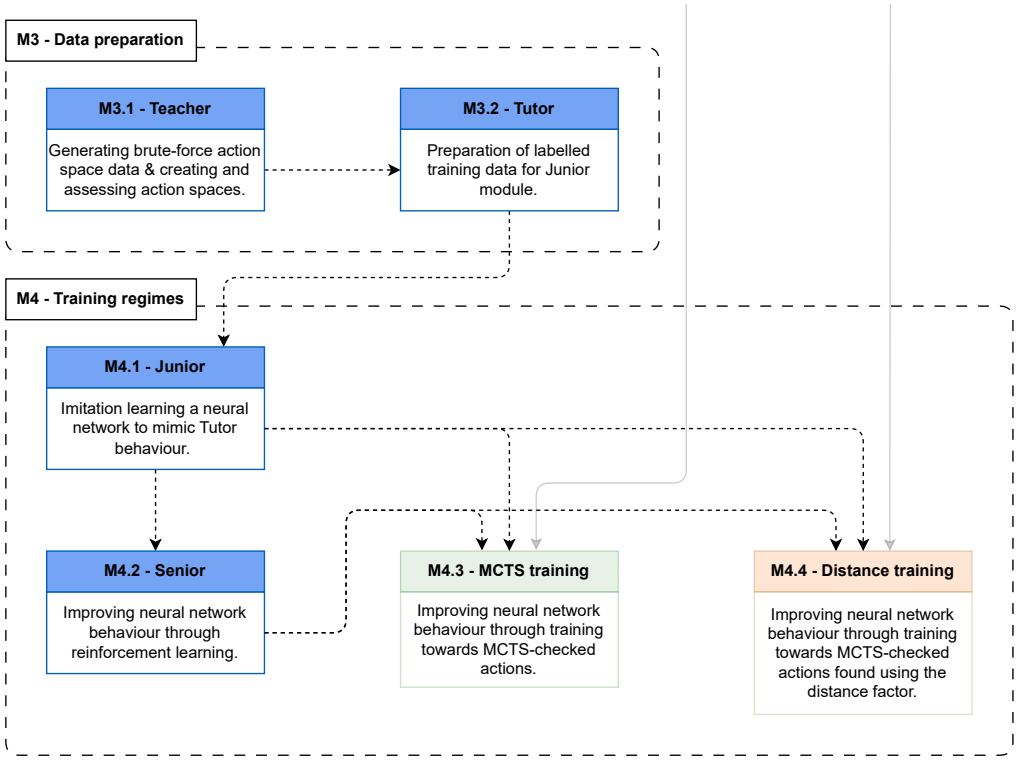



**Figure 4.3:** Modules of the methodology including the curriculum learning regime

mising system resilience, with line connections managed by a dedicated expert module. Continuous, non-topological actions like redispatching or battery interventions are also omitted allowing the most optimal use of the neural network and the proper evaluation of network control using solely economically topological actions.

### Collection of Action Data

The first step in creating the reduced action space is computationally extensive, as all actions need to be evaluated during different scenarios to assess their impact. To achieve these results, a brute-force '*Teacher*' module is used. The following steps are used by the module:

- **First step.** The agent executes through environment scenarios until an overflow occurs. When such an overflow occurs, all possible topological substation actions are simulated, and the resulting states are evaluated. Any action that is simulated to reduce the overflow is saved to an experience file.
- **Second step.** The action experience file generated in step 1 is analysed and filtered. Any action that does not relieve the overflow by $> 2\%$ is filtered out. Likewise, any 'do-nothing' actions are also filtered out.
- **Third step.** All remaining actions are sorted in order of frequency.

### Creating and Assessing Action Spaces

The action space can be generated by cutting off the remaining actions, in order to generate a set with the actions that are most frequently shown to relieve overflows. A trade-off between flexibility and computational performance is made by setting the limit for the cut-off. Introducing more actions in the action space allows for more flexibility in actions, but can hinder the convergence during training or the computational complexity during real-time operation. A brute-force algorithm is used to determine the flexibility and computational complexity of different action space sizes, after which the most optimal performing size can be chosen. The brute force agent enlists the earlier defined expert rule basis, where an action is requested once the threshold for maximum loading is surpassed. To define the 'best action' in these states, the agent simulates over all actions in the provided action space, opting

for the action that offers the best reduction. Performance, based in timesteps survived throughout the scenarios, and computational time, are saved. The brute-force algorithm can be seen in Algorithm 4. The results of the action space testing can be found in Chapter 5.

**Algorithm 4** Brute-Force Agent Algorithm

```
function act(observation)
    maximum loading ← observation.max_rho()
    if maximum loading > threshold then
        bestAction ← BruteforceSimulateAllActions()
    else
        bestAction ← reconnect_line(observation)
        if bestAction = None then
            bestAction ← recover_topology(observation)
        end if
        if bestAction = None then
            bestAction ← doNothing
        end if
    end if
    return bestAction
end function
```

### 4.2.2. Preparation of Training Data ('Tutor')

The next module in the curriculum learning process utilises the reduced action space from the '*Teacher*' module to generate a set of training data for fitting a NN, the next step in the learning pipeline. The so-called '*Tutor*' module tries to approximate optimal behaviour through a brute-force approach, which the neural network can try to mimic.

#### Creating the Training Set

A similar approach to the '*Teacher*' module is used where all actions are considered during states of overflow. However, during this stage only the best performing actions are saved, to prepare a labelled dataset that can be used for imitation learning the next step, in the form of:

$$\texttt{training dataset} = \begin{bmatrix} \texttt{state, best action} \\ \texttt{state, best action} \\ \texttt{state, best action} \\ \vdots \quad\quad\quad \vdots \\ \texttt{state, best action} \end{bmatrix}$$

The process to generate this dataset can be outlined as follows:

- **First step.** The agent executes through environment scenarios until an overflow occurs. When such an overflow occurs, all actions from the provided **reduced action space** are simulated, and the resulting states are evaluated. The best performing action in terms of overflow reduction is saved, together with the current state of the environment.
- **Second step.** All gathered data is shuffled and partitioned into a training set, a testing set and a validation set, using 80%, 10% and the remaining 10%, respectively.

### 4.2.3. Neural Network Configuration via Imitation Learning ('Junior')

The third module within the curriculum learning pipeline is called the '*Junior*', and comprises the first deep learning method. The goal of this module is to train a sequential neural network using the data generated by the '*Tutor*' module, aiming for the neural network to mimic the training data. That is, aiming for the sequential neural network to have its highest prediction output for a certain best brute-force action, when the associated state is fed forward.

The neural network has a relatively simple design, where the input layer has the shape of the state from the '*Tutor*' dataset, and the output layer has the shape of the amount of actions from the reduced action space. The states used for the input layer are filtered, as not all parameters are equally valuable in determining the best mitigation action. Including all variables would increase the computational complexity unnecessarily. Variables that were constant throughout the scenario offer no extra information, while other variables were skipped due to the agent's inability of performing those actions, such as generator ramping values or the attention budget used for giving alarms. Used and unused variables are described in Table 4.1, while further details about the used variables can be found in Table B.4.

**Table 4.1:** Comparison of used and unused state variables, as for the neural network input

| Used Variables | Unused Variables |
|---|---|
| Timestamp Features | Alarm Features |
| Generation Features | Storage Features |
| Load Features | Dispatch Features |
| Line Features | Curtailment Features |
| Bus Features | Generation Adjustment Features |
| Cooldowns | Alert and Attack Features |
| Maintenance Information | - |

#### Training Details

The sequential neural network is trained using the '*fit()*' method from TensorFlow [67]. The state entries in the data are first scaled using scikit's *MinMaxScaler()* [68]. Using this scaler all variables within the training data are scaled between 0 and 1, and the scaler parameters are saved, so that unseen validation or testing data are scaled using the same values. The training function is used on the '*Tutor*' training set, while the validation set is used to determine the loss per iteration. An early stopping mechanism, where the training is cut short if the loss has ceased to improve over a set number of iterations, is used to prevent the model from overfitting. The used sequential neural network is based on the layer design of the Binbinchen agent, where hyperparameter tuning is used to obtain the layer parameters. An overview of the layer parameters and values can be found in Table 4.2, while the additional training variables can be found in Table B.1. The training and validation accuracy and loss, and the accuracy for the top-20 predicted actions can be found in Figures A.1, A.2 and A.3, respectively.

**Table 4.2:** Junior Neural Network parameters and values

| Parameter | Value |
|---|---|
| Input Layer | 1221 Variables |
| Hidden Layer 1 | 400 Neurons |
| Dropout Layer 1 | 0.25 |
| Hidden Layer 2 | 773 Neurons |
| Dropout Layer 2 | 0.40 |
| Hidden Layer 3 | 1044 Neurons |
| Hidden Layer 4 | 344 Neurons |
| Output Layer Linear | 100 Actions |

Employing these parameters the neural network is trained to best output the correct 'best action'. However, in grids where input variables are very similar throughout training scenarios with different values for their corresponding 'best actions', achieving high values for mimicking can be difficult. Outputting the wrong grid actions might only further deteriorate grid stability during highly loaded times, calling for additional training methods to improve performance and security.

### 4.2.4. Exploration through Reinforcement Learning ('Senior')

In final stage of the curriculum learning regime the pre-trained '*Junior*' neural network is improved upon through reinforcement learning. The '*Senior*' model is identical to the trained '*Junior*' model, sharing

its layer and neuron structure. The weights of the imitation trained model are used as a head start for the RL algorithm.

#### Training Setup

The neural network is trained using the Ray package, which allows for efficient use of training resources which distributes the training between a head node, for training, and multiple worker nodes, for generating the trajectories [69]. A custom environment is created which allows the expert rules to be integrated into the decision making process for the RL algorithm, while also allowing the training model to receive the reward over steps where no agent action was required. The model is trained using the actor-critic RL algorithm PPO [52]. The gathered basic Grid2Op reward is used to determine the effectiveness of actions, which is based on the operational costs of the whole grid. Surviving for more timesteps would award the agent with a higher score, indicating a better course of action.

A population based training (PBT) algorithm is used for hyperparameter tuning. This state-of-the-art algorithm, proposed by Google's DeepMind in 2017, efficiently discovers schedule of hyperparameter settings using a fixed computational budget [70]. The algorithm uses a pre-defined value of workers, each with a set of hyperparameter settings taken from a pre-defined range. The algorithm is able to evaluate the performance during training, discarding worse-performing sets of hyperparameters, while prioritising exploration around better performing sets. This eliminates the need for more computationally expensive or time-extensive search methods. Further details about the PBT and training set-up can be found in Tables B.2 and B.3, while the results of the mean episodic reward throughout the training regime can be found in Figure A.4.

### 4.2.5. Final Agent

The final agent is realised after all curriculum learning modules are concluded. The RL trained agent from the final stage is combined with the expert rules module. An agent action is required only after the grid threshold has been breached, and the best proposed action is then applied, or the top-N actions are simulated to allow for a slight security check.

There are a few drawbacks to this approach. Firstly, actions are only chosen based on the neural network output and their direct influence, negating possible future impact. Neural network training using RL is still time-consuming. Hyperparameter tuning can be troublesome, with great impact on the training outcome, and no guaranteed convergence. When the neural network has poor initial performance, more actions have to be simulated before a safe course of action can be chosen, essentially reverting back to a greedy brute-force approach. To combat these shortcomings a guided tree search method is introduced in the following section.

## 4.3. Monte Carlo Tree Search

To overcome the limitations of brute-force based methodologies and enhance the reliability of remedial measures, it is essential to consider potential future outcomes. To tackle this problem a MCTS addition to the curriculum-learning based agent is developed. Notably, MCTS-based agents, including the AlphaZero algorithms [10] and the 2022 L2RPN competition winner [11], have demonstrated exceptional success in environments with expansive action spaces.

In the MCTS the branches of the tree are explored depending on their expected value, which is based on a prior probability and a visit count, favouring nodes with high preliminary probabilities and fewer visits. Attempting to exhaustively explore all possible states in a broad action space quickly becomes infeasible. However, this guided search ensures a great improvement in finding the right actions with minimal computational effort. The tree's structure enables the agent to evaluate the impact of immediate and sequenced actions, basing choices not only based on their simulated direct impact, but on future performances as well. Using a combination of approaches allows for the added layer of security during further training and testing. Concurrently, the neural network,responsible for predicting prior action probabilities, is pre-trained through the curriculum-learning process for enhanced sample efficiency. A visual schematic overview of this part of the methodology can be seen in Figure 4.4.

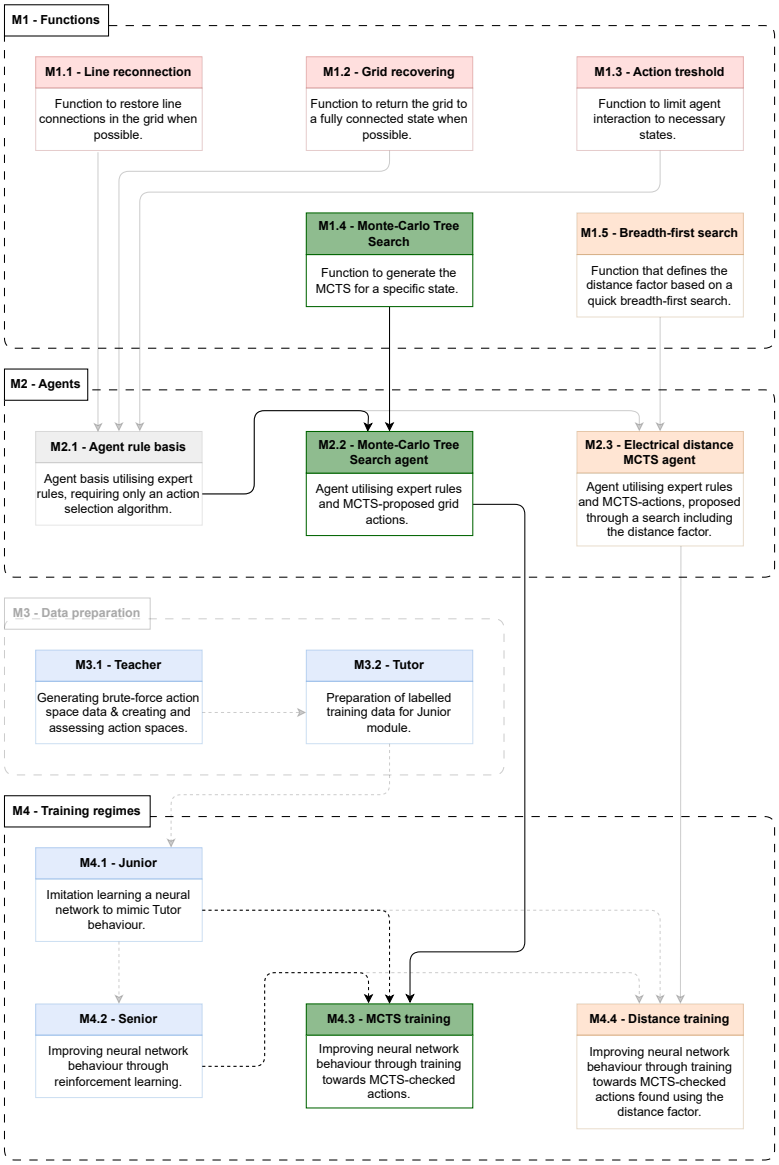



**Figure 4.4:** Modules of the methodology including the Monte-Carlo Tree Search

The needed elements for the tree traversal and selection process are outlined in section 4.3.1, after which the tree algorithm is explained in section 4.3.2. Section 4.3.3 describes how the action is chosen once the tree simulations have stopped, and section 4.3.4 explains how the secure actions chosen by the MCTS can be used to further train the prior neural network in an efficient manner.

### 4.3.1. MCTS Elements

In this section the important elements for the tree simulation, as well as the final action selection are outlined.

#### Safety Checker

The safety checker serves to evaluate the security status of a node within the tree. This status is determined by the number of timesteps a node can endure before surpassing the threshold for the load limit, necessitating agent intervention. A state is deemed safe if it can endure a pre-defined number of timesteps, referred to as '*t_skipped*', without agent intervention. The employed criterion, '*t_skipped = 10*', is supported by literature [11] and practical validation, showing its effectiveness in allowing the power grid to stabilise post-overflow events.

#### Neural Network

Within the tree, the curriculum-trained neural network predicts the initial probabilities for the actions from each node. Each newly initialised state employs a forward pass neural computation, assigning the probability of each potential action, where higher values indicate preferable actions. These probabilities shape the sequence of actions explored during tree navigation.

#### Value Network

Value networks play an important part in the action selection process of the MCTS algorithms seen in literature, but are assessed to underperform in this application. The Grid2Op reward is based on all grid operating costs, which are highly linked with the amount of timesteps survived in an algorithm, assigning little value to grids operated at lower levels of loading. To clarify the decision-making process and mitigate the inconsistencies of value networks derived from reinforcement learning, an action selection algorithm based on the 'safe' timestep count and grid loading metrics of a state is used.

#### Edges

In the tree, edges represent the transitions between nodes, indicating the actions executed. These edges possess distinct variables: the originating node, the action, and the prior probability of that action from the originating state. At each decision point within the tree, edges are used to compute their Upper Confidence Bound (UCB) score, determining the subsequent path choice.

#### Nodes

Nodes contain the state of the power grid following a specific action. Key attributes of a node include the path of actions leading to that state, the maximum timestep the node can sustain without agent interaction, as verified by the safety checker, and the specifics of the grid loading condition. The node with the most optimal grid state is chosen after simulating through the search tree, with the first entry in the action path indicating the operation to take.

### 4.3.2. Simulation Tree Algorithm

Now that the important elements of the tree are clarified, the algorithm for simulating through the tree can be described. Each time an action of the MCTS agent is required, the root node of the tree gets initialised using the current state of the environment for the specifics of the grid loading, and a forward pass through the neural network to assigning the action probabilities from that state. A decision to traverse to a new node in the tree is based on the Upper Confidence Bound (UCB):

$$U(s,a) = P(s,a)\sqrt{\frac{\sum_b N(s,b)}{1+N(s,a)}} \tag{4.1}$$

where:

$U(s,a)$: Upper Confidence Bound for action $a$ in state $s$.
$P(s,a)$: Prior probability of selecting action $a$ in state $s$.
$N(s,a)$: Number of times action $a$ has been selected from state $s$.
$\sum_b N(s,b)$: Total visit counts for all possible actions $b$ from state $s$.

The approach based on this formula prioritises the selection of states with the highest initial probabilities. Subsequently, a balance is struck at each decision point, weighing the probability of the action against its visit frequency to ensure a thorough investigation of all highly estimated states. Navigation through the decision tree involves making choices at each intersection, guided by the Upper Confidence Bound (UCB) algorithm, until a previously unexplored state is reached. This new 'leaf node' is established using the current grid conditions following the last action and employs the neural network to set the prior probabilities for potential actions. After initialising the leaf state, the process returns back to the root node to restart the action selection process. This loop persists either until a predetermined number of simulation iterations have been completed or the simulation is early-stopped.

#### Early Stopping Mechanism

To reduce the computational complexity, an early stopping mechanism is used that concludes the simulation search once a good (enough) solution is found. The criteria for stopping are defined as follows:

- Once a pre-defined number of safe states have been found, *or:*
- Once a state has been found that reaches the end of a scenario

The safe states are defined as the states which are able to skip through a set number of timesteps, '*t_skipped*', before the threshold for the load limit is crossed, necessitating agent intervention, deemed safe using the Safety Checker from Section 4.3.1. Once a pre-set number of safe states have been reached, the early stopping algorithm is terminated and the algorithm for selecting the action commences. The used value for the early stopping criterion, '*s_safe = 10*', is supported by literature [11] and practical validation, displaying a good trade-off between computational improvements and retrieval of appropriate actions. The algorithm describing the simulation tree algorithm can be found in Algorithm 5.

**Algorithm 5** Simulation Tree Algorithm

```
function run(iterations)
    iterationCount ← 0
    safeCount ← 0
    for _ in range(iterations) do
        leaf ← selectLeafUCB()
        if leaf reaches endOfEpisode then
            break
        end if
        if leaf is safe then
            safeCount ← safeCount + 1
        end if
        if safeCount ≥ safeStates then
            break
        end if
        iterationCount ← iterationCount + 1
    end for
    return simulationTree
end function
```

### 4.3.3. Action Selection Algorithm

Once the simulation tree has been fully traversed, either because the set number of iterations is reached or because the early-stopping mechanism has been employed, the action selection algorithm commences. In the original MCTS approach, from Silver *et al.* [10, 31, 32], action selection is done based on visitation count. However, to adjust for a new objective, namely operating the grid for as long as possible, the action selection progress is altered. The best node is determined depending on two different scenarios:

- **If any safe states have been discovered:** the safe state with the maximum number of reachable steps is chosen.
- **If no safe states have been discovered:** the state with the maximum number of reachable steps is chosen.

If multiple nodes have the same value for maximum number of reachable steps in either of these scenarios, the node with the lowest maximum grid loading is chosen. Once the most optimal node has been chosen, the first entry in the action path leading to that node is chosen to be applied in the environment of the root node. An example of the action selection algorithm for the case that safe states have been found can be seen in Figure 4.5, while an example of the algorithm for the case no safe states have been found can be seen in Figure 4.6.

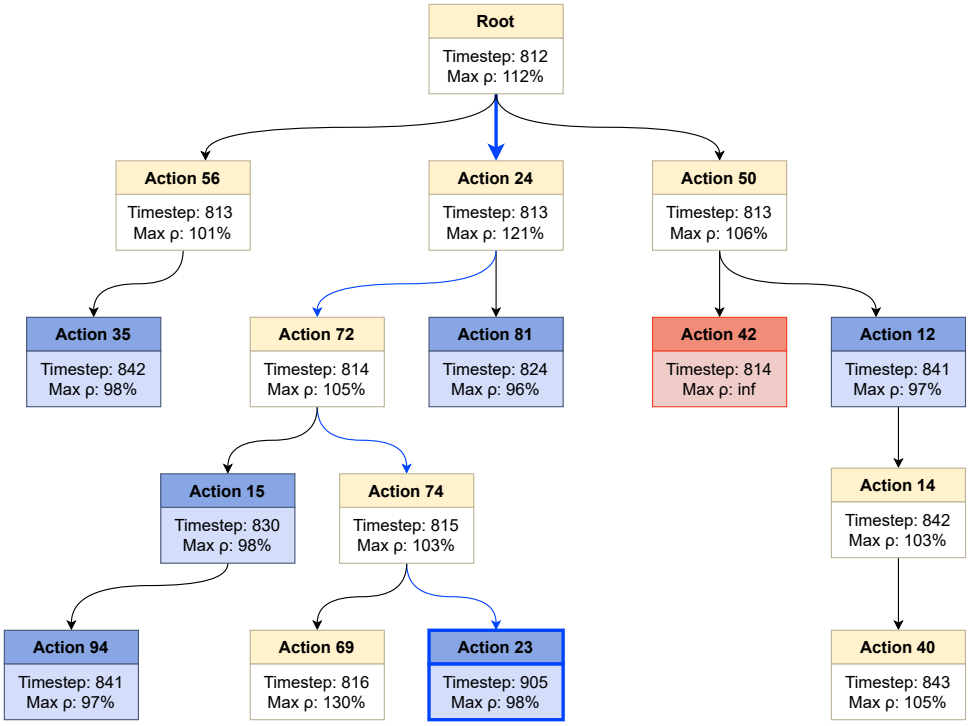



**Figure 4.5: Action selection example if safe states have been found.** Nodes represent grid states, root is the current state. Grey nodes are violating states (maximum line load > 98%), red nodes are black-out states and blue nodes are states deemed safe by the Safety Checker (able to skip '*t_skipped*' timesteps without agent intervention needed. Safe grid state able to reach the maximum number of steps is chosen, and the action leading there is chosen at the root (action 24).

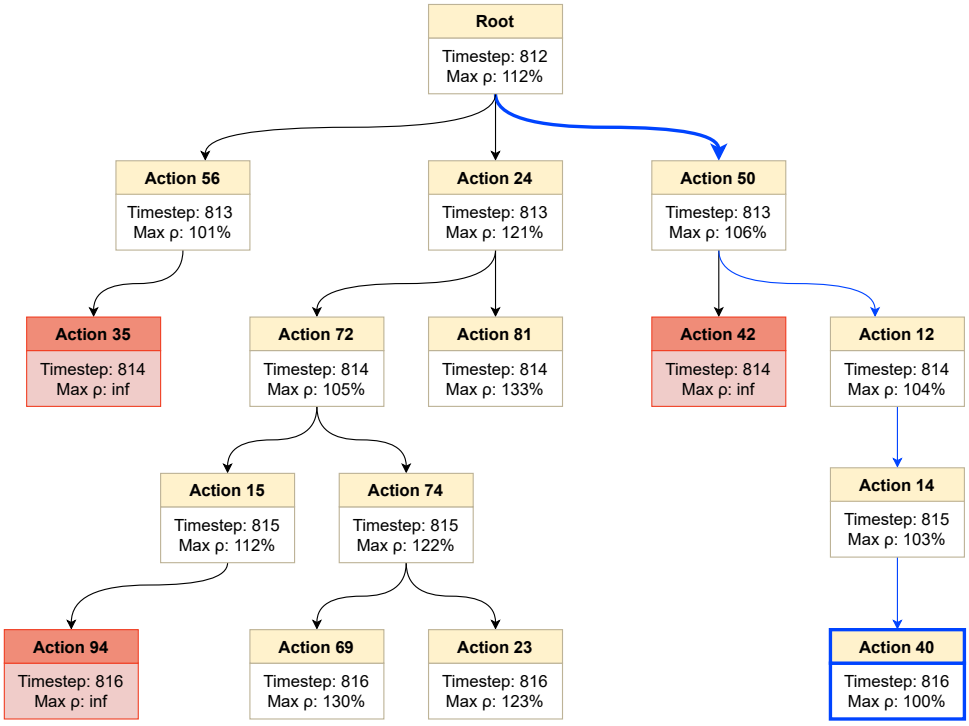



**Figure 4.6: Action selection example if no safe states have been found.** Nodes represent grid states, root is the current state. Grey nodes are violating states (maximum line load > 98%) and red nodes are black-out states. The grid state able to reach the maximum number of steps is chosen. If multiple states have the same value for reachable timesteps, the grid state with the lowest maximum loading is chosen, and the action leading there is chosen at the root (action 50).

### Action Selection Safeguard

Selecting the state with the maximum number of reachable timesteps can sometimes lead to a sub-optimal action strategy, particularly when no safe states are found. This happens when a state may be chosen solely because its path has been more frequently visited, causing it to be situated one layer more down, and thus one more timestep ahead. To enhance the security of the action selection algorithm in scenarios lacking safe states, a safeguard is implemented.

The safeguard is activated when an action has to be chosen when no safe states are found, and operates as follows:

- **Step 1.** The algorithm identifies the immediate predecessor, or 'parent', of the state selected by the primary selection algorithm.
- **Step 2.** It then assesses all other nodes in the tree with the same timestep value as this parent, based on the maximum line loading values.
- **Step 3.** If any of these nodes have a significantly lower maximum loading, determined by the '*safeguard_threshold*', the node with the lowest maximum loading is chosen instead.

The '*safeguard_threshold*' is a crucial parameter for fine-tuning the selection process. Setting it too low value can disregard the benefit of reaching further in the scenario, whereas setting it too high can minimise the opportunities to reevaluate decisions, potentially allowing unsafe actions. For the purposes of this thesis, an experimental determination has led to a '*safeguard_threshold*' set at 5%. The findings of this investigation are detailed in Chapter 5. An example of the action re-selection algorithm based on the '*safeguard_threshold*' can be seen in Figure 4.7.

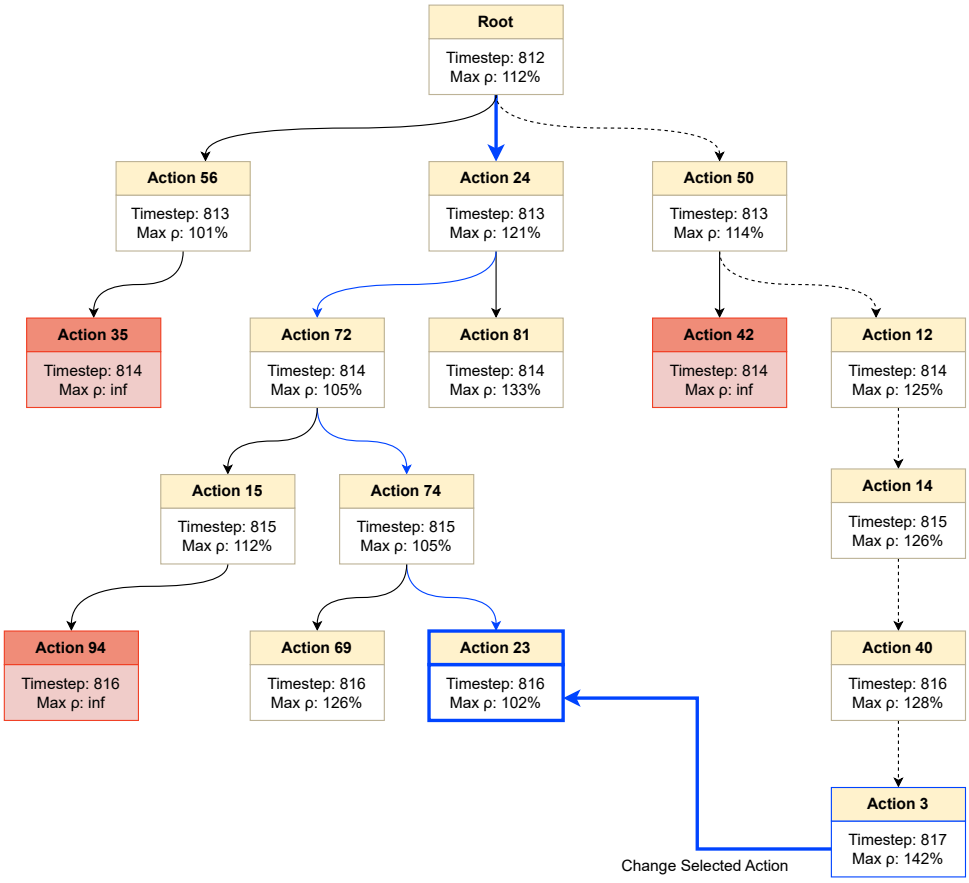



**Figure 4.7: Action re-selection example using the safeguard, if no safe states have been found.** Nodes represent grid states, root is the current state. Grey nodes are violating states (maximum line load > 98%) and red nodes are black-out states. The grid state able to reach the maximum number of steps is chosen. The safeguard compares the maximum loading of the nodes on the same timestep as the parent node of the action chosen, and re-selects the best node if a reduction in loading > '*safeguard_threshold*' is found. The action leading to the chosen node is selected at the root (action 24).

### 4.3.4. Training Regime

The MCTS addition to the curriculum-learned agent is used to improve the security of the proposed control sequences, while the strengths of the curriculum-learned approach are used initially to provide a head-start to the prior network. However, using RL approach for further training requires extensive tuning of parameters, massive exploration and therefore a great deal of training samples. Therefore, further training using the MCTS-proposed actions is recommended. The proposed approach utilises the robust simulation capabilities of the MCTS to guide the action selection process, subsequently supplying the training dataset with high-quality, simulation-tested actions. By using the foresight provided by MCTS outcomes, the algorithm can iteratively refine its policy without the necessity of exploring the full breadth of scenarios first. This shift in training approach not only accelerates the learning curve but also conservatively utilises the available samples.

Employing MCTS within the training pipeline introduces numerous advantages. Firstly, it enhances sample efficiency through simulation and concentrating on promising areas of the action space, thereby reducing the number of interactions needed to reach optimal performance. Secondly, the immediate feedback loop from the simulations reduces the variance typically associated with the trial-and-error nature of RL. This improves convergence to a well-working policy and mitigates the risk of reaching local solution optima.

#### Training Set-up

The training set-up requires a batch of training data, which are generated every time that an MCTS action is required. Due to the robust nature of the MCTS-checked actions, batching data sample to reduce variance is not needed, reducing the training complexity:

$$\texttt{training dataset} = \left[\texttt{state},\ \ \texttt{best action}\right]$$

The training of the model employs the Adam optimisation algorithm, well-known for its effective moment estimation and bias correction capabilities, which are particularly useful in navigating the challenges posed by high-dimensional spaces [71]. This optimiser refines the neural network parameters by minimising the 'sparse categorical cross-entropy' loss of the training samples, which is well-fitting for models with categorical target labels. To speed-up training and enhance hardware-efficiency the sample scenarios are split up and processed in parallel, each optimiser dedicated to the scenarios of a distinct month of the yearly dataset. After each iteration of the training data the parameters of the parallel trained models are averaged, and redistributed to the optimisers for the next training iteration. Utilising this training method ensures optimal usage of available training resources, while model averaging can even introduce a reduction of variance in the training regime [72].

### 4.3.5. Testing Regime

During the testing cases, as seen in Chapter 5, no optimisation occurs. The MCTS is fully traversed at every step necessitating agent interaction to determine the course of action, though no training data is generated, and the neural network providing the MCTS with the prior predictions for the actions does not get improved. Each full MCTS simulation concludes either due to the early-stopping algorithm, or when the set amount of maximum iterations within the tree is reached.

## 4.4. Electrical Distance Guided Search

The inclusion of an MCTS framework adds a significant layer of security to the action selection process. However, if the initial training of the neural network is not optimal, the search tree might struggle to quickly identify viable solutions, hindering a quick training process. To address this, an electrical distance guided term in the search tree is proposed, which helps streamline the action selection process within the tree and improve the convergence of the training regime. A visual schematic overview of this part of the methodology can be seen in Figure 4.8.

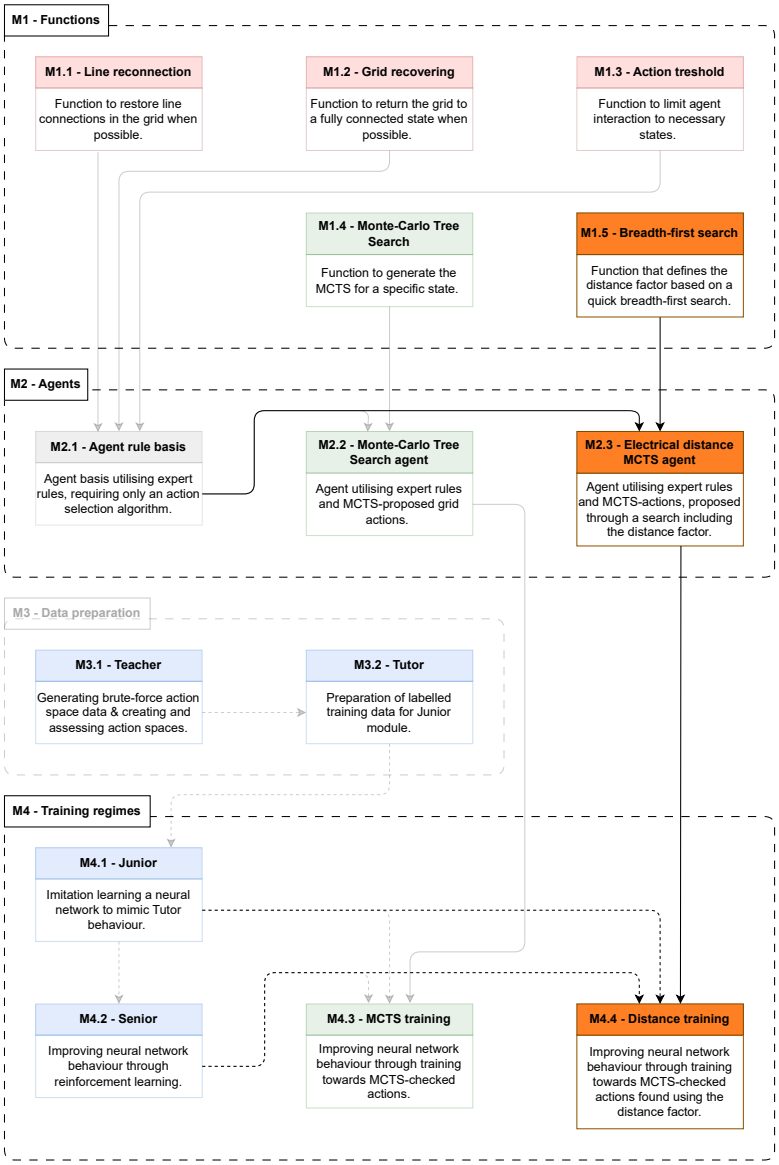



**Figure 4.8:** Modules of the methodology including the electrical distance factor

### 4.4.1. Motivation for the Usage of an Electrical Distance Factor

Literature has shown that topological interventions located closer to overflows tend to have a higher impact, increasing their effectiveness in reducing violations, as outlined in Section 3.3. Guiding the agent towards actions in the tree that are found electrically closer improves the convergence of the tree algorithm and ensures enhanced selection of effective actions for mitigation control. The metric for electrical distance used is 'hops', quantified as direct connections between an overflow and an action's location, due to its availability and usability in quickly calculating the metric. While this metric does not provide an absolute measure of influence, it can offer a valuable estimate of the impact of an action to the agent. The positive relationship between the electrical 'hop' distance and the effectiveness of mitigation actions is experimentally tested. Utilising the earlier described brute-force algorithm, outlined in Algorithm 4, a variety of scenarios are simulated, indicating the hop distance of each 'best action' encountered, which is the action that provides the highest reduction in maximum loading. The results, as shown in Figure 4.4, highlight a greater effect from actions found up to three 'hops' from the overflow, as almost 85% of best actions are to be found within a 3 hop perimeter of the line experiencing the violation.

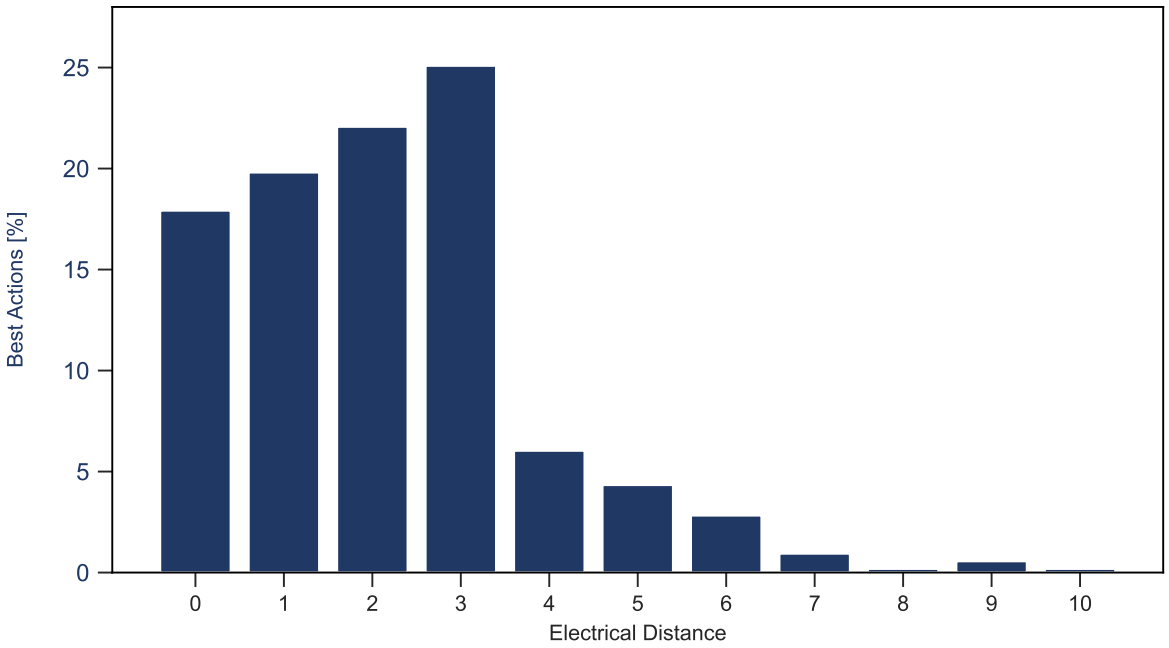



**Figure 4.9:** Percentage of actions found that most optimally reduce loading, per electrical distance as measured in 'hops'.

### 4.4.2. Determining Electrical Distance

The electrical distance metric is needed during each step of the simulation tree, necessitating a quick algorithm capable of accurately determining the closest electrical connection between an overflow and all possible actions. A BFS algorithm is proposed, offering reliable determination of the closest electrical distance, while being sufficiently computationally efficient.

The proposed BFS algorithm, embedded within the MCTS framework, efficiently calculates the shortest 'hop' distance within the power grid represented in a Compressed Sparse Row (CSR) matrix. This matrix format is optimal for sparse data such as grid systems, where each bus is connected to only a few others, resulting in a matrix dominated by zeroes.The CSR format becomes computationally advantageous with increasing grid sizes, by storing only non-zero elements and their respective indices. The BFS algorithm systematically explores adjacent connections starting from the overflowing line, avoiding revisiting previously explored nodes, until the destination node is found, guaranteeing the shortest path. This dynamic exploration algorithm succeeds in finding the shortest distance even when extensive adjustments to the grid topology have been made, possibly altering earlier direct connections. This gives the algorithm an edge over static methods of determining the distance, such as a look-up table (LUT).

### 4.4.3. Adding an Electrical Distance Term to the Tree Algorithm

To integrate the electrical distance into the simulation process, the BFS algorithm dynamically computes the hop electrical distance for use within a factor that guides the tree search algorithm. This factor is designed to prioritise actions that are electrically closer, specifically giving the highest preference to actions within a 3-hop distance, corresponding to the findings from the brute-force testing. The electrical distance factor is applied to direct the tree search toward proximate actions, balancing between leaving room for exploration, and guiding towards anticipated impactful interventions. This factor is described in a formula that adjusts the selection weight based on the hop distance, thereby optimising the search for effective solutions within the MCTS framework. The formula describing the distance factor is chosen to be used in an additive way, where the closest actions at most double their UCB advantage within the search tree, while the actions furthest away are barely altered. The cut-off for the formula is chosen to be at the electrical distance of 4, matching the findings from Figure 4.9, giving the following formula:

$$DF(d) = \frac{1}{1 + e^{(d-4)}} \tag{4.2}$$

where:

$DF(d)$: Distance factor as a function of electrical distance $d$.
$d$: Electrical distance measured in hops.

The distance factor is visualised in Figure 4.10.

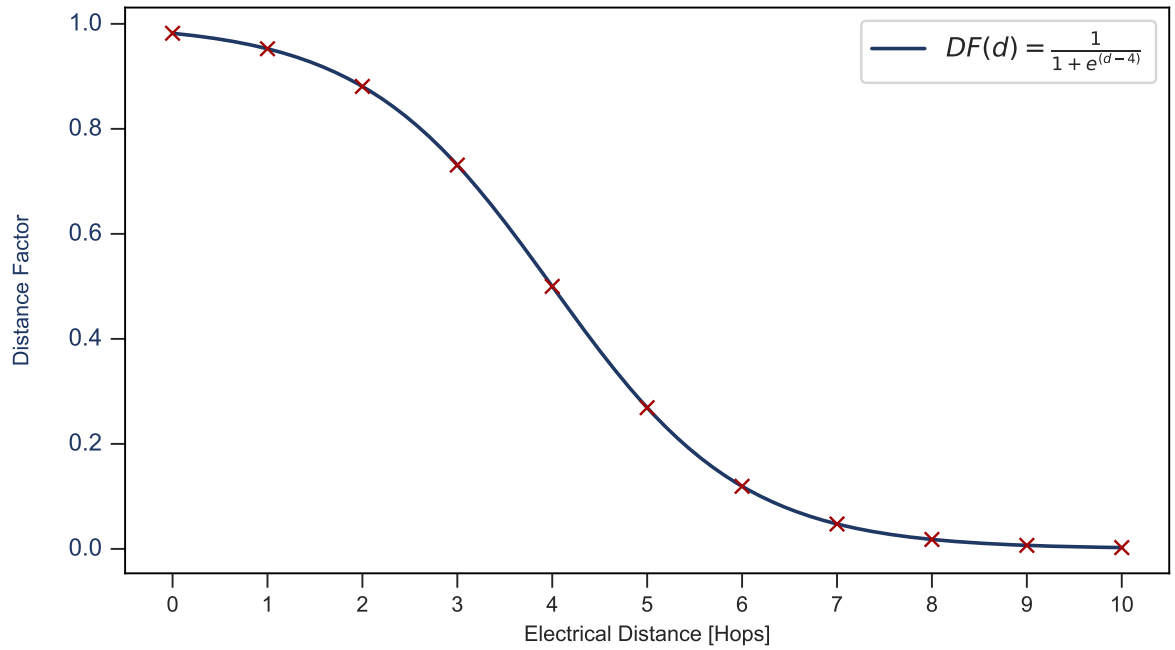



**Figure 4.10:** Graph of the Distance Factor Formula

In the context of the MCTS, the distance factor serves as an influence rather than a brute-force method. It refines the tree selection process by adjusting the Upper Confidence Bound (UCB) values of the edges originating from each node. This additive integration respects the prior action probabilities while subtly guiding the decision-making process in favour of actions that are electrically closer and, presumably, more impactful. The UCB formula, which balance exploration and exploitation, are thus modified to give a slight advantage to actions within a closer electrical distance, resulting in the following adjusted UCB formula:

$$U(s,a) = P(s,a) \cdot \sqrt{\frac{\sum_b N(s,b)}{1 + N(s,a)}} \times (1 + DF(d)) \tag{4.3}$$

where:

$U(s,a)$: Upper Confidence Bound for action $a$ in state $s$.
$P(s,a)$: Prior probability of selecting action $a$ in state $s$.
$N(s,a)$: Number of times action $a$ has been selected from state $s$.
$\sum_b N(s,b)$: Total visit counts for all possible actions $b$ from state $s$.
$DF(d)$: Distance factor as a function of electrical distance $d$.

### 4.4.4. Training Using the Distance Factor

This section describes the incorporation of an electrical distance factor into the training regime of an MCTS agent, mirroring the training methodology established in Section 4.3.4. The training process remains consistent with that of the standard MCTS agent, leveraging robust simulated actions generated by MCTS to guide the optimisation process. The introduction of the electrical distance factor is aimed at guiding the agent's policy towards optimal solutions. It serves as a guiding heuristic, initially steering the policy search but gradually decreasing in influence as the training progresses. This dynamic scaling of the distance factor is designed to ensure that as the agent's policy advances and converges, the artificial bias introduced by the distance bonuses becomes less significant, allowing the agent to rely on the strength of its learned strategy rather than the heuristic. The result is an agent that, while initially informed by the distance factor, ultimately achieves convergence through the policy itself, without the need for continued heuristic influence.

# 5

# Case Studies

## 5.1. Test Settings and Network

All testing cases are performed on the Grid2Op 118-bus '*WCCI_L2RPN_2022*' system. Figure 5.1 shows the grid, containing 118 substations, 186 powerlines, 91 loads and 62 generators.

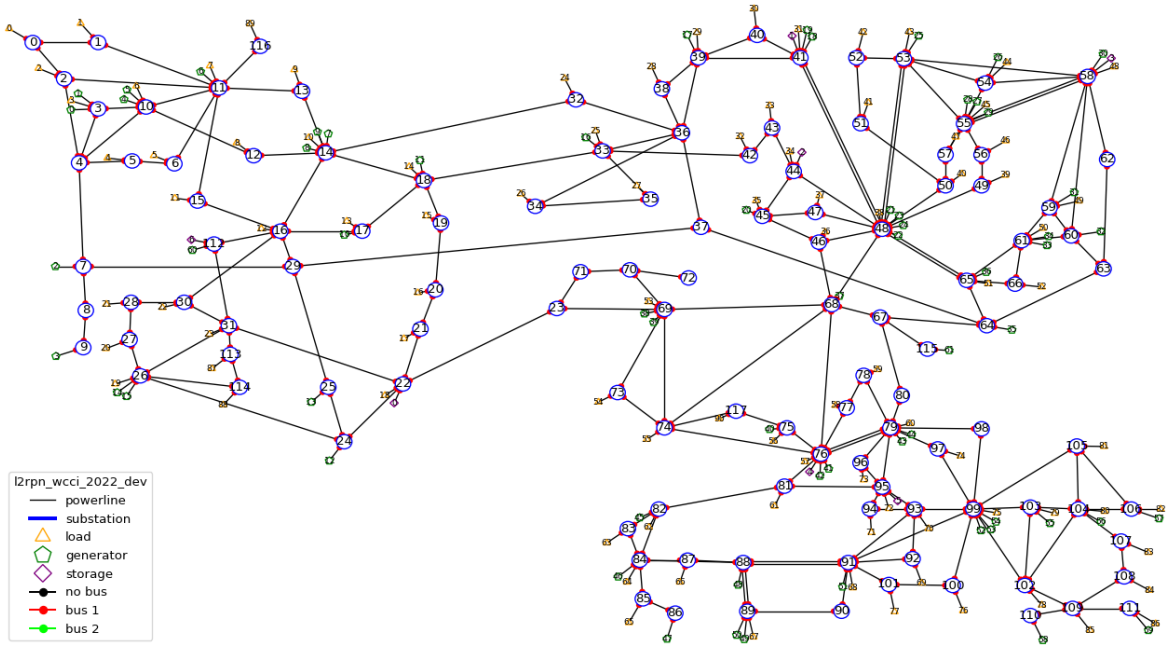



**Figure 5.1:** Schematic overview of the '*WCCI_L2RPN_2022*' grid from Grid2Op, used for all testing cases.

All test cases are performed on a test or hyperparameter set consisting of 52 weekly scenarios, composed of scenarios from each month of the year, to offer realistic variance in load consumption and difficulty. For the test set, the 52 weekly scenarios from the 2022 L2RPN competition are used [8]. The test set is distinct from the validation set used for determining the hyperparameter values, introducing no initial bias. For the validation set 52 different scenarios are used, offering a similar level of variance in load consumption and difficulty. This validation set is only used for hyperparameter determination test cases and not for other case studies. For both the test and the validation set the opponent agents are seeded randomly 4 different times, changing their attack locations and times, essentially quadrupling the test and validation set size and reducing the outcome variance. The details of the used scenarios for the test and validation set can be found in Table B.5.

The data necessary for the realisation of the action spaces is generated on a simple 16.0 GB RAM laptop with an AMD Ryzen 7 4800U 1.80 GHz core. The '*Teacher*' module collecting overflow instances and actions reducing overflows is run for 40 hours, resulting in 63,308 overflow samples. Imitation learning and RL for the *Junior* and *Senior* is also done on this 16,0 GB RAM laptop with an AMD Ryzen 7 4800U 1.80 GHz core. All testing cases are performed using DelftBlue's supercomputer Intel XEON E5-6248R 24C 3.0GHz CPU cores [73]. The 1.9.3 version of the Grid2Op package, the 0.7.3 version of the LightSim2Grid, and the 2.11.1 package of pandapower are used.

## 5.2. Hyperparameter Analyses

Before the case studies, the control and operation parameters for the agent are determined through experimental verification. The following parameters are tested in this section:

- **Reduced Action Space (RAS).** Action spaces of various sizes are compared through performance and computational time to determine the optimal size.
- **Agent Action Threshold (AAT).** The optimal value for the limit for the maximum line loading in the grid before an action of the agent is required is determined, based on performance, as seen in Section 4.1.3.
- **MCTS safeguard (MCTS-SG).** The optimal value of the safeguard, determining whether another node in the search tree should be chosen, as described in Section 4.3.3, is determined.

The brute-force algorithm, as outlined in Algorithm 4, is used for the RAS and AAT testing, where the action space size for the RAS testing and the action threshold for the AAT testing are the only variables. This brute-force algorithm utilises the expert base rules, requesting an agent action when the action threshold is surpassed. The brute-force agent will then simulate through all actions in the employed reduced action space, opting for the action resulting in the lowest maximum loading. For the MCTS-SG testing, an agent employing the MCTS algorithm for simulation and action selection, as defined in Algorithm 5 and Section 4.3.3, is used. The *Junior* neural network is used for predicting the prior probabilities of each state within the MCTS, as outlined in Section 4.2.3. The only variable during this testing is the '*safeguard_threshold*', determining when an earlier, safer node should be chosen within the search tree.

### 5.2.1. RAS Analysis

Filtering the overflow action data to distinct actions used for reducing overloads by $> 2\%$, 2,700 unique actions remain. The actions are sorted and prepared into action spaces by taking the first-N most frequent actions for an action space. The action space sizes evaluated for RAS are: 50, 100, 250, 500, 1000, 2000, and 2700. The results of the RAS testing case can be found in Figure 5.2.

The RAS testing cases clearly show the expected trend of increased computational complexity with a growing action space size, while the increase in performance stagnates after the usage of larger action space sizes than 100. The performances of the agents utilising various action space sizes show the impressive utility of only a small portion of the most influential actions, where compelling performance can be achieved using an action space of containing less than $4\%$ of the actions in the largest action space, and less than $0.15\%$ of the total action space. Interestingly, utilising the largest available action space decreases performance, underlining the importance of selecting the appropriate actions, instead of the most optimal ones for the current timestep. The computational time does not show a direct linear relation, due to the universal usage of the expert-rule bases for all agents, but increasingly growing computational complexity is still expected during neural network training and MCTS convergence, both growing exponentially with increasing actions. Therefore the reduced action space size of 100 is chosen to be used for further agent development, striking the right balance between offering good flexibility while allowing for rapid computation.

### 5.2.2. AAT Analysis

The only variable in this testing case is the value for '*action_threshold*', for which the values 80%, 90%, 95%, 98% and 100% are tested. The results of the AAT testing case can be found in Figure 5.3.

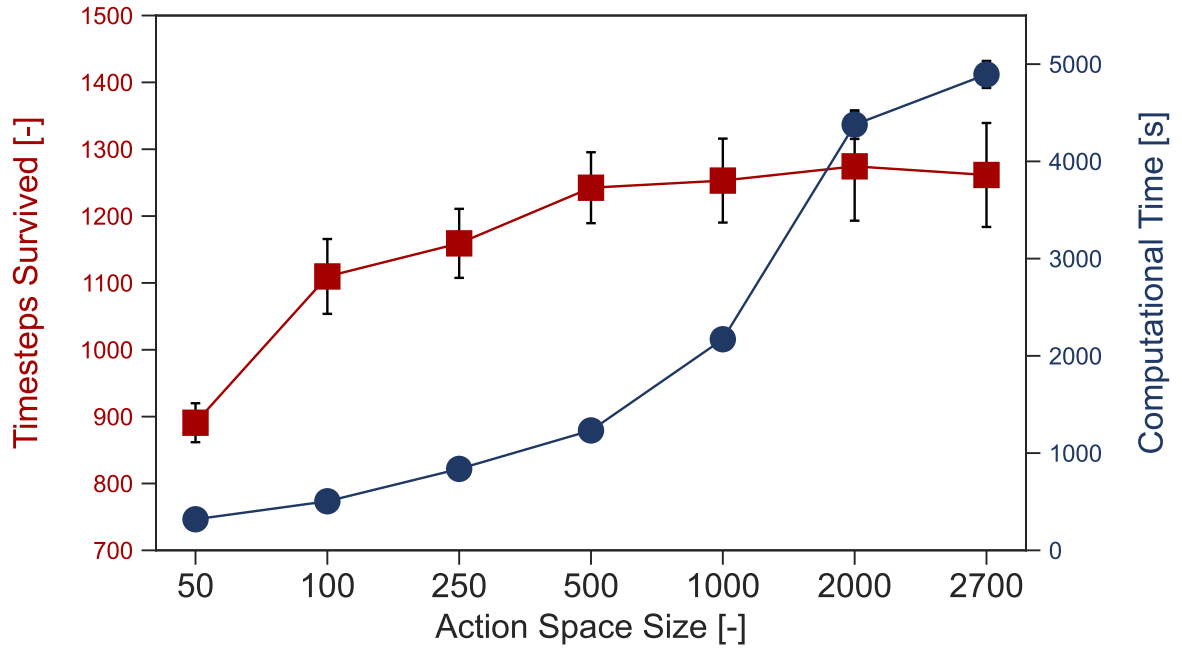



**Figure 5.2:** Effect of the action space size, used by the brute-force algorithm, on the performance and computational time per scenario. Average values and standard deviations are indicated using markers and error bars, respectively.

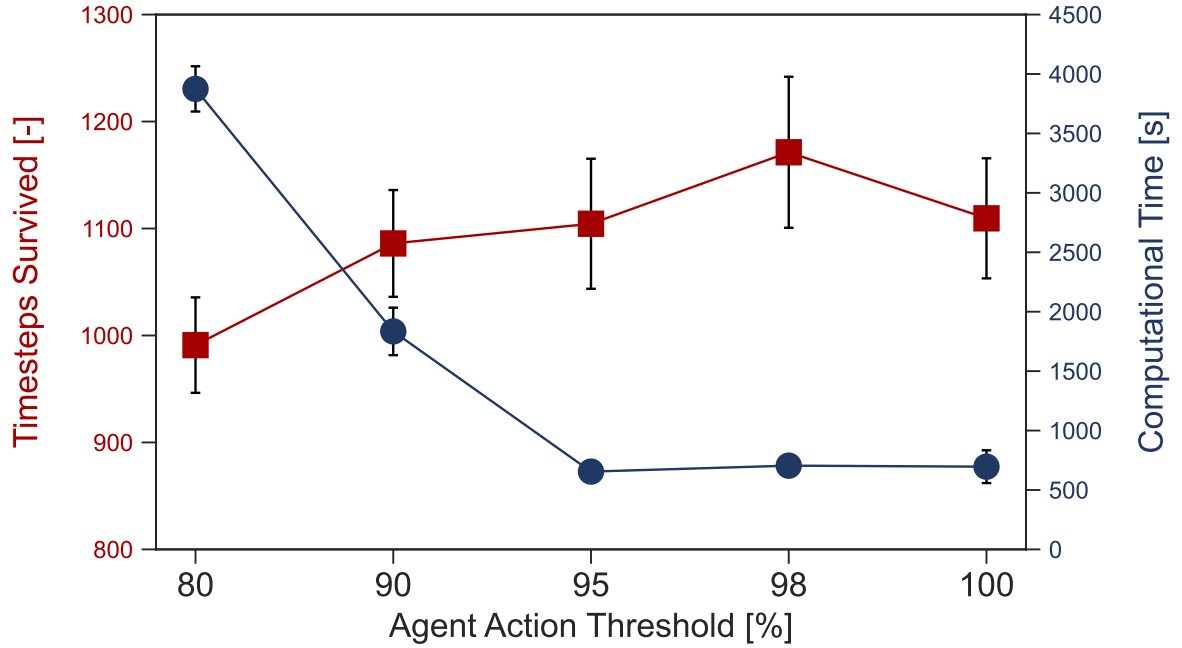



**Figure 5.3:** Effect of the Agent Action Threshold (AAT) , used by the brute-force algorithm, on the performance and computational time per scenario. Average values and standard deviations are indicated using markers and error bars, respectively.

The results of the performance show a clear preference for an agent action threshold that requires mitigation actions only when the grid has a maximum loading present of $> 98\%$. The results of the computational times show the practicality of using a higher value for the agent action threshold, as unnecessarily requesting agent actions greatly increases computational times. During further agent developments and case studies in this thesis a value for the agent action threshold of 98% is used, striking the right balance between intervening when needed without unnecessarily deteriorating the

grid during normal operations.

### 5.2.3. MCTS-SG Analysis

In this testing case, the only variable tested is the value for '*safeguard_threshold*', determining the level of reduction in maximum loading that must be found in earlier nodes before the action selection algorithms opt to change their initial selection. The values tested are 0%, 1%, 2%, 5%, 10%, 20%, and 50%. A baseline (BL) that never opts for an earlier, safer action is also included. The results of the MCTS-SG testing case can be found in Figure 5.4.

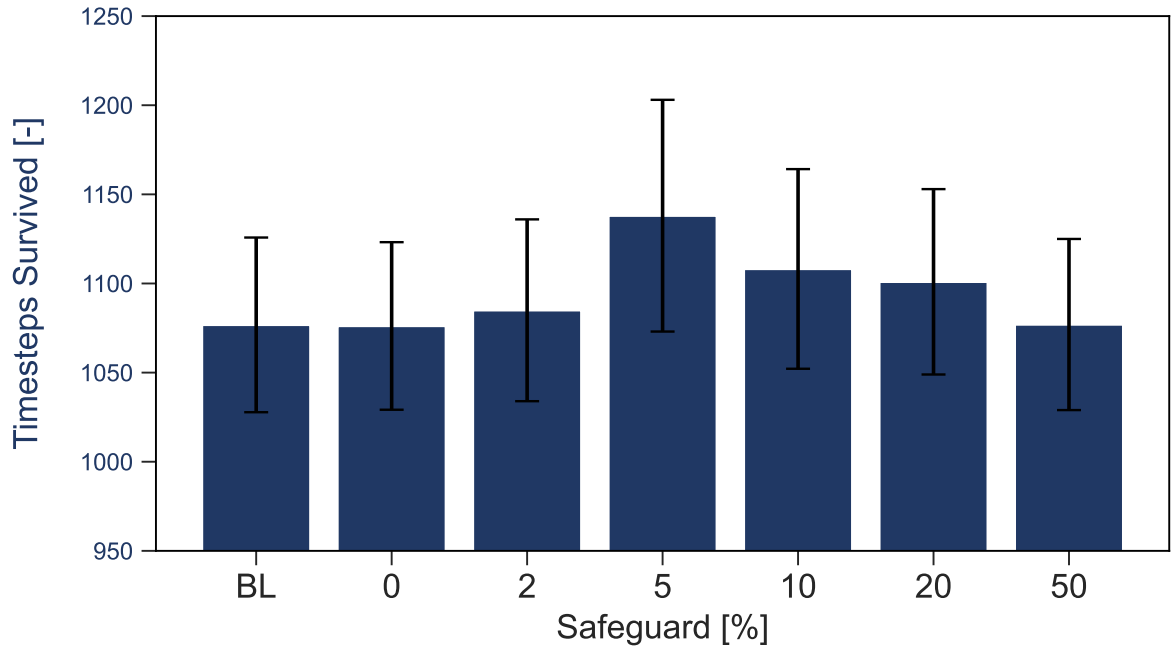



**Figure 5.4:** Effect of the safeguard value, as employed by the *Junior*-MCTS algorithm, on the average agent performance, including a baseline (BL) algorithm that never re-thinks its course of action.

The experimental findings indicate that configuring the '*safeguard_threshold*' to a value of 5% yields the most enhancements in performance metrics. Comparative analysis shows that all MCTS-SG algorithms either match or surpass the baseline performance, underlining the advantage of selecting actions that guide the system towards nodes associated with slightly reduced maximum timestep values, but resulting in superior grid performance with respect to the maximum loading observed.

## 5.3. Case Study 1: MCTS Security and Sample Efficiency

In the first set of case studies objectives 1 and 2 of the research questions are evaluated. The added level of security of the MCTS is evaluated and compared to the baselines. The performance of agents using only network topology control actions are quantified and evaluated, looking into the effects of using MCTS-sampled actions on the training stability and sample efficiency. The following metrics are used to evaluate their performance:

- **Security.** What is the level of security added by utilising a MCTS algorithm? This is quantified by the amount of timesteps survived per scenario.
- **Speed.** How fast is the proposed method compared to the baseline?
- **Sample efficiency and stability.** Does the proposed method improve the sample-efficiency and stability of the training setting?

For effective comparison of the level of security added, the prior networks of the *Junior* and *Senior* are evaluated using the expert rules and the extra safety check for their top-N actions. The NN's performance was measured in three scenarios settings: using the top prediction alone, simulating

the top-5 actions to select the one with the most significant reduction, and similarly with the top-25 actions. These were benchmarked against an MCTS agent incorporating the NN's for prior estimations, constrained to 150 iterations per step due to computational limits. Additionally, comparisons were made with two baseline agents: a do-nothing agent, and a brute-force agent. The brute-force agents simulates through the entire reduced action space once an action is requested, opting for the action with the resulting lowest value for maximum loading, as outlined in Algorithm 4. The do-nothing agent does not perform any topological mitigation actions, solely re-establishing line connections and the default topology as per the expert rules.

The MCTS training protocol, as outlined in Section 4.3.4, harnessed the entirety of 32 years' worth of grid scenarios, divided into 1,663 weekly segments. For a single training iteration all 1,663 scenarios are simulated to completion or game-over. The optimiser trains the neural network separately for each month of the training scenarios, after which the parameters of the parallel trained model are averaged, and distributed to the optimiser for the next training iteration, as outlined in Section 4.3.4. For each iteration, the opponent is seeded randomly, such to not repeat precisely similar training scenarios. The MCTS-trained model used for evaluation is trained for 3 iterations.

### 5.3.1. MCTS Security Analysis

First, the results for the MCTS security testing are presented.

Figure 5.5 presents the comparisons of the performance of the neural network prior predictions of the *Junior* and *Senior* compared to the baselines on the left side, while the right side presents the computational times per timestep of the agents.

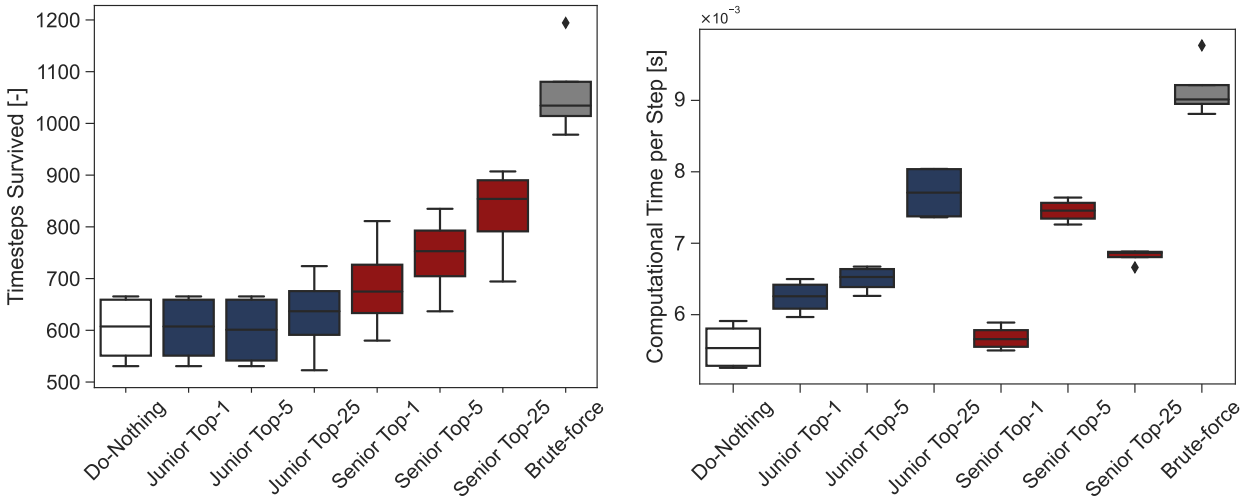



**Figure 5.5:** Timestep survival performance (left) and computational time per step (right) of the NN-agents, using their top prediction, or simulating the top-5/25 predicted actions, compared to the baselines.

The agents employing neural networks trained via imitation learning exhibited performance that is only marginally improved over the do-nothing baseline, where the agent simulating the 25 top actions is the only agent able to improve over this baseline. The agents trained through reinforcement learning (RL) show better performance, already surpassing the do-nothing baseline and all imitation learned agents only by using its top prediction, showing a clear improvement. The improvements in performance of the *Senior* agents simulating its top-5/top-25 actions shows how the security can improve by including the safety check expert rule. However, both the *Junior* and the *Senior* agents still fall short of surpassing the brute-force baseline. This underscores the criticality of optimal action selection, as most actions tend to be sub-optimal in the majority of states. The assessment of computational time reveals a positive trend to relate with performance. The *Senior* (RL) agents have reduced computational times, indicating the selection of actions that bring more stability, leaving the grid to operate within limits for longer without

renewed agent intervention. The neural network agents however do not offer any performance increase to benefit from their quicker computation.

Figure 5.6 displays the performance of the MCTS agents on the left side, which includes a comparison with both baseline agents and the agents utilising only expert rules with a simulation safety check, while the right side presents the computational times per step for each agent.

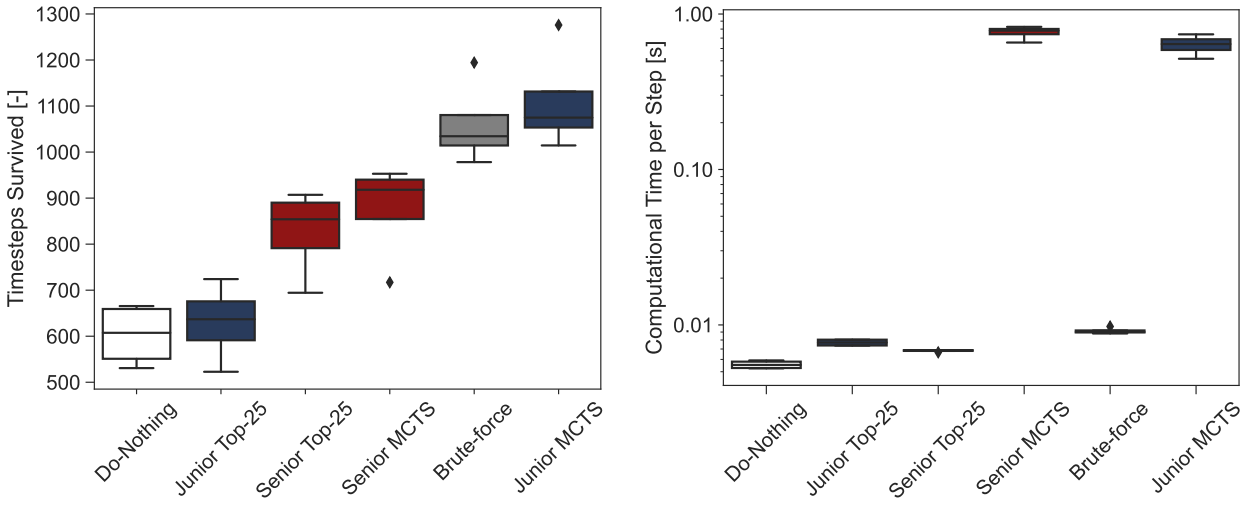



**Figure 5.6:** Timestep survival performance (left) and computational time per step (right, using logarithmic scale) of the MCTS agents, compared to the *Junior*/*Senior* neural network agents employing a top-25 simulation security check, and the baselines.

The results of the performance and the computational time per step are also detailed in Tables 5.1 and 5.2 for proper comparison.

**Table 5.1:** Performance statistics (average and standard deviation) for the MCTS and prior prediction agents compared to the baselines

| | **Time Steps Survived [-]** | **Computational Time per step [s]** |
|---|---|---|
| Do-Nothing | $603\,(\pm 59)$ | $5.6 \cdot 10^{-3}\,(\pm 2.9 \cdot 10^{-4})$ |
| *Junior* top-25 | $630\,(\pm 73)$ | $7.7 \cdot 10^{-3}\,(\pm 3.3 \cdot 10^{-4})$ |
| *Senior* top-25 | $827\,(\pm 83)$ | $6.8 \cdot 10^{-3}\,(\pm 9.2 \cdot 10^{-5})$ |
| *Senior* MCTS | $877\,(\pm 94)$ | $0.76\,(\pm 0.06)$ |
| Brute-Force | $1060\,(\pm 81)$ | $9.2 \cdot 10^{-3}\,(\pm 3.7 \cdot 10^{-4})$ |
| *Junior* MCTS | $1110\,(\pm 99)$ | $0.63\,(\pm 0.08)$ |

**Table 5.2:** Direct comparison of the MCTS and non-MCTS methods

| | *Junior* Agent | *Senior* Agent |
|---|---|---|
| Performance Increase | $+76\%$ | $+6\%$ |
| Computation Time Increase | x$81$ | x$112$ |

A significant enhancement in security of the actions chosen is observed for the neural network agents through the integration of a MCTS in the action selection phase. The agent leveraging the *Junior* neural network with MCTS surpasses the brute-force baseline in terms of performance, while the agent based on the *Senior* neural network sees improvements but does not outperform the brute-force approach. The differences in performance of both MCTS agents highlight the effect that combining different training regimes may have, where advanced performance through both methods, RL and the MCTS, do not automatically add. The lower performance of the *Senior*-MCTS agent may imply that the RL-trained

model has higher confidence in its predicted actions, outputting higher values for singular actions, versus a more diverse array actions with relatively low predicted values. When this model is not trained to an optimal extent yet, the performance increase due to the actions chosen might not weigh up versus the reduced exploration in the search tree due to the more particular predictions. The results of both the *Junior* and *Senior* agents show a positive effect of combining an imitation learned pre-trained neural network with a MCTS for added security. These findings affirm the effectiveness of incorporating MCTS in action selection, which aids the agent with the identification of action sequences that are better-suited for long-time operation rather than solely focusing on the immediately most advantageous actions. Even though this performance boost comes at the cost of significant computational time increase, computation stays under 1 second per 5 minute timestep, showcasing applicability within real-time operations.

### 5.3.2. MCTS Training Regime Analysis

In this section the results for the training regime improvements are presented.

Figure 5.7 shows the performances of the MCTS-training algorithms throughout each of the iterations of the training data, where all 1,663 scenarios, split up into monthly segments, are used to generate MCTS-robust data samples for the algorithm to train towards. The averaged parameters of the 12 monthly models are used at the start of the new iteration.

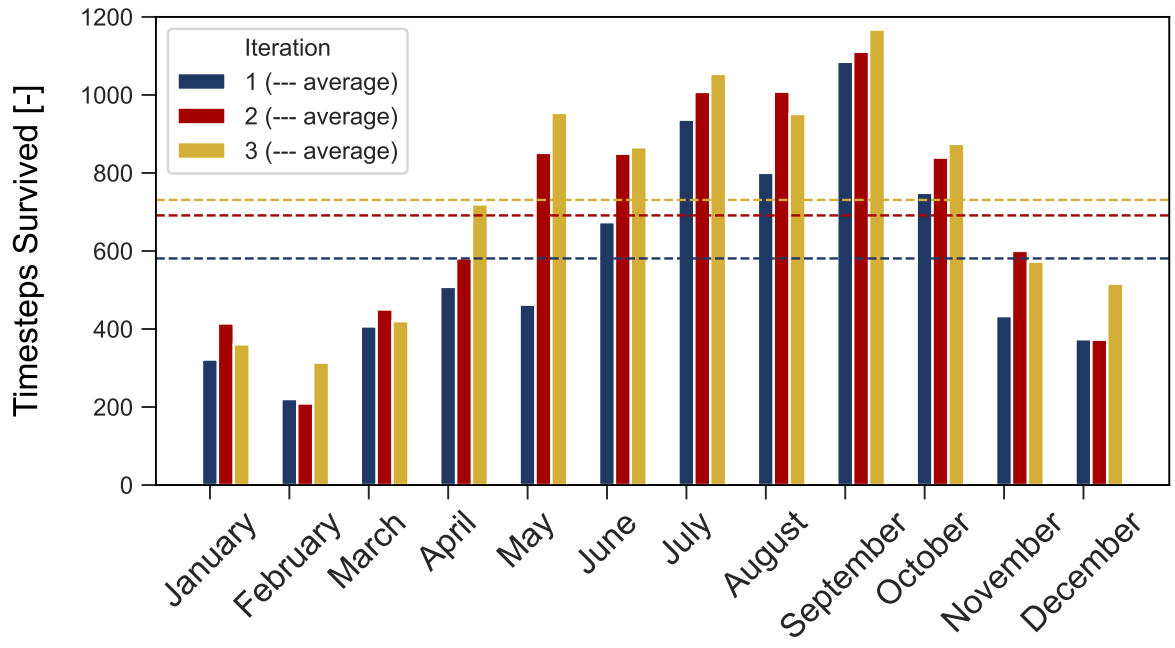



**Figure 5.7:** Performance of the MCTS-trained algorithm, averaged for each of the training months, throughout its 3 training iterations.

The results of the MCTS-training show that MCTS-approved robust training samples of best actions can be efficiently used to train the MCTS agent. Leveraging these robust actions more than a $25\%$ percent improvement in performance is seen throughout only 3 iterations of 1,663 samples, a fraction of the amount of interactions used by RL algorithms through training. The first iteration performs poorly for the month of May, under-performing its average, compared to the the second and third iteration outperforming its average in this month. Even omitting this outlier, an improvement of more than $20\%$ can be seen throughout the training iterations. This shows improved performance in sample-efficiency, using far fewer samples than a RL-training approach, while also highlighting the stability of the training regime. The results showcase only very minor monthly dips in performance throughout the steadily improving training iterations. An overarching curve can be seen throughout all training data, having its peak throughout the summer months, and experiencing the highest difficulty in finishing scenarios

throughout the winter months. The higher electricity consumption during colder times in the winter months put the network at a higher level of capacity, hindering options for topology mitigation actions and their effectiveness. However, the influence and effect of the adversarial agent is also clearly visible, where changing opponent seeds through each iteration can hinder performance, indicating a limit towards what can be achieved using solely node-switching actions during violations caused by the unexpected outages of highly loaded lines. These training results demonstrate the effectiveness of training using robust samples, but show that optimal performance independent of the scenarios might only prove possible using a combined approach of switching actions and redispatch actions.

Figure 5.8 presents the performance of the MCTS-trained neural network on the left side, compared with the *Junior* and *Senior* NN-agents and the baselines, while the right side presents the computational times per step of the agents.

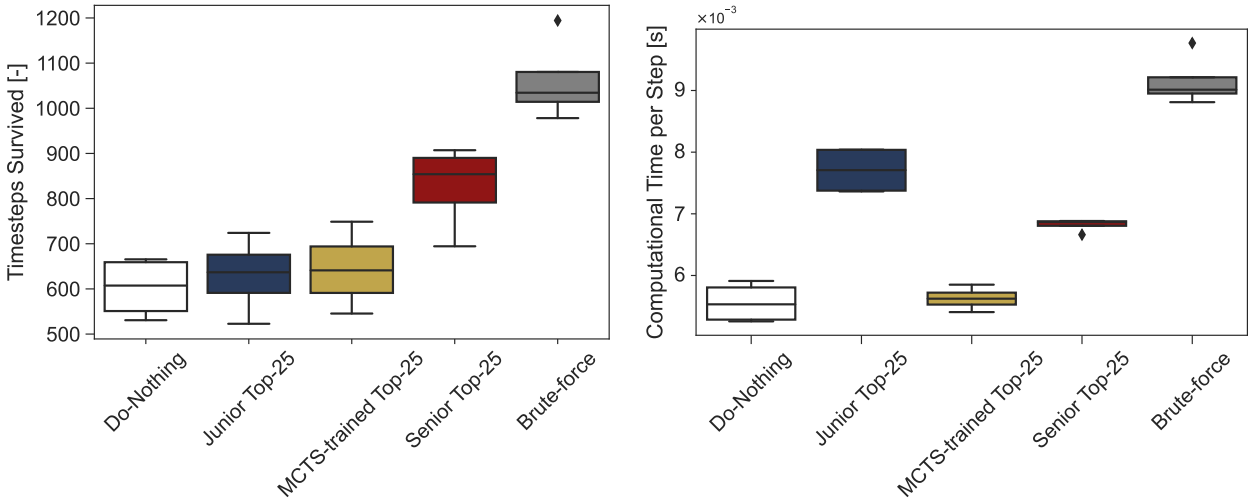



**Figure 5.8:** Timestep survival performance (left) and computational time per step (right, using logarithmic scale) of the MCTS-trained prior prediction agent compared to the *Junior*/*Senior* neural network agents, all employing a top-25 simulation security check, and the baselines.

The prior predictions of the neural network from the MCTS-trained agent exhibits performances that are only marginally better compared to the *Junior* neural network agent, despite refining its prior network through many iterations of robust training samples, showcasing that the training regime using MCTS-samples has not reached a decent level of convergence yet. If such a level would have reached, the optimal action in each state of overflow would be proposed. However, due to the usage of the MCTS during action selection, the most optimal action does not require a maximum preference from the neural network, as long as the optimal action has a high enough prediction value allowing it to be found within the set amount of iterations. The MCTS-trained neural network does not surpass the agent using a RL neural network network, not very surprisingly, as the RL agent has trained specifically for the purpose of optimising its prior predictions, where the MCTS-approach offers some leeway through its expansive simulation methods.

Figure 5.9 displays the performance and iteration count for the MCTS-trained agent, compared to the *Junior*/*Senior* MCTS agents. The results show a positive effect of utilising robust MCTS-confirmed actions for the training process, improving the performance of the *Junior* MCTS agent while using little extra in terms of computational times. The improvements throughout the training iterations of the MCTS-trained agent do not directly translate to the testing set, where only a slight improvement can be seen, compared to the $25\%$ increase throughout the training regime. This again shows constraining limits in the form of the adversarial agent, whose unexpected disconnections might not be overcome using only node-switching actions. The comparison to the RL trained MCTS agent showcases its effectiveness in sample efficient training. Where the RL has been trained using more interactions, fewer but more robust samples of the MCTS-streamlined training approach offer a higher utility of

training scenarios. The results showcase the difference in training approach between RL and MCTS-training, where exploration is preferred in RL, advancing its prior predictions, while the MCTS-training showcases applicability in advancing robust operation through the scenarios.

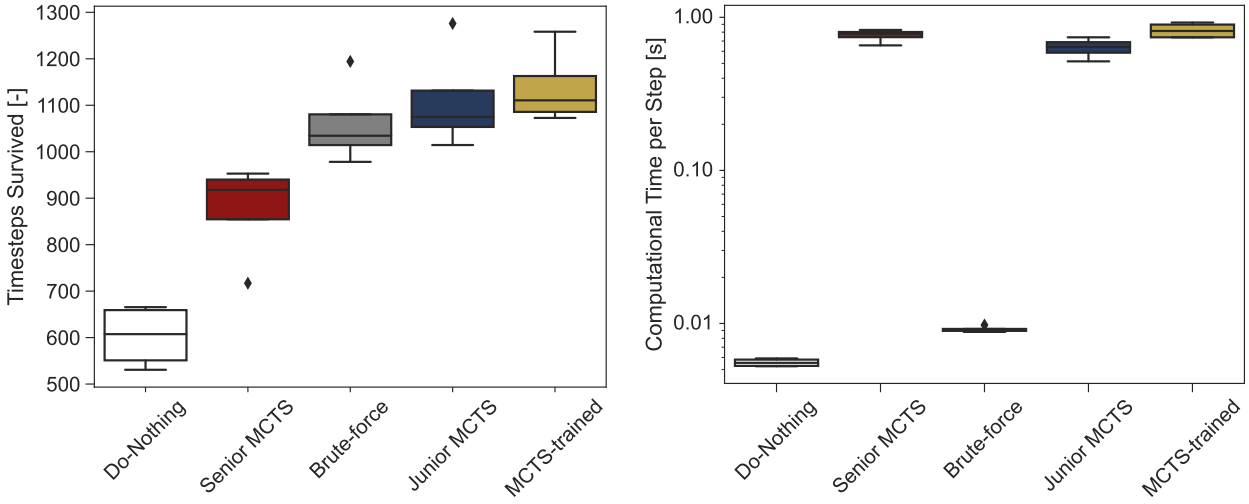



**Figure 5.9:** Timestep survival performance (left) and computational time per step (right) of the MCTS-trained MCTS agent compared to the *Junior*/*Senior* MCTS agents, and the baselines.

## 5.4. Case Study 2: Distance Factor

In the second set of case studies objective 3 of the research questions is evaluated. The effect of the inclusion of a proximity factor on MCTS-simulation and performance convergence is evaluated and compared to the previously developed models. The following metrics are used to evaluate the performance:

- **Performance.** What is the level of direct performance increase when the distance factor is used to streamline the action selection process of the MCTS during testing?
- **Simulation Converge** Does the inclusion of the distance factor reduce the amount of MCTS simulations needed to find a good solution?
- **Training convergence.** Does the inclusion of the distance factor during training advance convergence of the action policy?

The distance factor will be added to the *Junior*-MCTS agent, the best performing agent so far, for optimal evaluation. The time steps reached and simulation iterations will be compared for the non-MCTS trained base case. The surviving abilities and iteration performance of the trained models will be evaluated, compared with the reinforcement learned *Senior* model to allow for direct comparison in sample efficiency. Both the MCTS-trained and distance-trained model used for evaluation are trained for three full iterations of 1,663 scenarios in the training regime, as explained in Sections 4.3.4 and 4.4.4.

### 5.4.1. Distance Bonus Analyses

Firstly, the results for the case where the distance bonus is added to the simulation search of the MCTS-*Junior* agent are presented, compared to the base-case without the distance bonus and the MCTS-trained agent. Baselines are not included in the results for the MCTS iterations, as they do not employ a tree search. The results for the performance can be found on the left side of Figure 5.10, while the iteration results can be found on the right side. Details of the results, including computational times can be found in Table 5.3.

The results show that a noticeable increase in performance can be obtained when using a distance-guided search, performing at similar levels as the model that has been subjected to three iterations

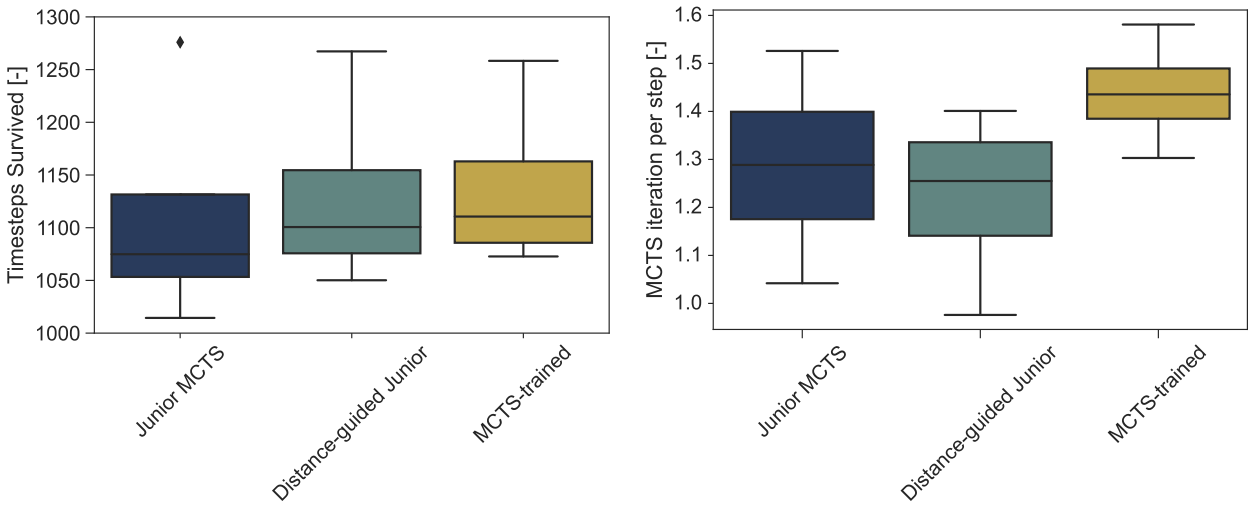



**Figure 5.10:** Timestep survival performance (left) and MCTS iterations used per step (right) for the distance-guided *Junior*-MCTS agent, compared to its non-guided base case and MCTS-trained agent.

**Table 5.3:** Direct comparison of the distance guided and non-distance guided MCTS-algorithms. The base case used for comparison is the *Junior*-MCTS algorithm. Values are given with averages and standard deviations.

| | Base Case | Distance-guided | MCTS-trained |
|---|---|---|---|
| Performance [-] | $1110\,(\pm 99)$ | $1130\,(\pm 83)$ | $1138\,(\pm 73)$ |
| Performance Increase | - | $+1.8\%$ | $+2.5\%$ |
| Iteration Count per Step [-] | $1.29\,(\pm 0.18)$ | $1.22\,(\pm 0.16)$ | $1.44\,(\pm 0.10)$ |
| Iteration Count Increase | - | $-5.4\%$ | $+11.6\%$ |
| Computational Time per Step [s] | $0.63\,(\pm 0.08)$ | $0.75\,(\pm 0.10)$ | $0.82\,(\pm 0.08)$ |
| Computational Time Increase | - | $+19\%$ | $+26\%$ |

of training data. The performance boost seems small, gaining only about a 2% increase in timesteps survived. However, such an increase corresponds to an extra 3 hours of survival through highly loaded states each scenario, allowing The results for the MCTS-iterations needed per step show that the distance-guided search is able to find appropriate course of action with fewer simulations needed within the search tree, without having to sacrifice in performance. This results offers positive insights into future application into a larger grid, where the greater distances and amplified level of connections can quickly threaten guaranteed tree convergence. In such larger grids, local influences can help guide a more vast search tree towards operation within real-time constraints. Equally insightful is the performance with respect to computational speed of the distance-guided method. The usage of a quick BFS algorithm does not hinder the speed significantly, staying well within 1s per 30 minute timestep. However, larger grids containing a greater, more difficult structure unfit for using this search method within operational time-limits might have to resort to different, such as static, methods.

The final case study tests regard the differences between training results obtained by the distance-guided training regime compared to the 'standard' MCTS-training regime, as outlined in sections 4.3.4 and 4.4.4. The results highlight the effect the distance factor has on training performance. It presents the average amount of timesteps completed per scenario, and it outlines the amount of iterations per timestep needed for the MCTS. The results are shown on the left side of Figure 5.11 for both performances and on the right side for the comparison between iteration count needed per timestep. The details of these results are described in Table 5.4.

The results stemming from training the MCTS algorithms with and without the distance-guided search shows various findings. Firstly, it shows that assessing all scenarios during the training process proves to be more difficult than the selected testing scenarios, achieving lower performances for timesteps

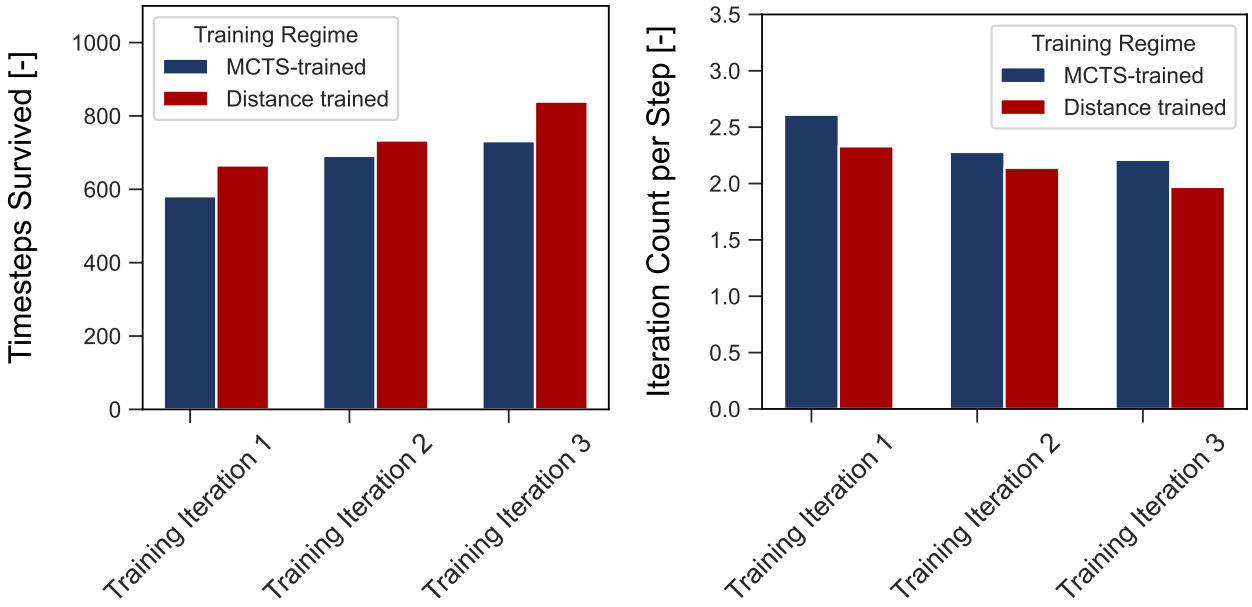



**Figure 5.11:** Timestep survival performance (left) and iteration count per timestep (right) survived for the standard MCTS-training regime compared to the distance-guided training regime, averaged per training iteration.

**Table 5.4:** Direct comparison of the distance guided and non-distance guided MCTS-algorithms during training

| Training Iteration | | MCTS-trained | Distance Trained |
|---|---|---|---|
| 1 | Performance | 581 | 665 |
| | Iteration Count per Step | 2.61 | 2.33 |
| 2 | Performance | 691 | 733 |
| | Iteration Count per Step | 2.28 | 2.14 |
| 3 | Performance | 731 | 839 |
| | Iteration Count per Step | 2.21 | 1.97 |

survived throughout the entire training set than compared to the testing set. This further displays the effect that the selection of testing scenarios has on the performance, where optimal performance for testing scenarios throughout the year is hard to achieve. Improved training convergence can be seen throughout the training iterations for the distance-guided training regime, which sees a performance boost of nearly $45\%$ for the distance trained agent trained over 3 iterations compared to its starting point. The average iteration count throughout the training regimes show higher values compared to the training set, which can correspond with worse performance. If better courses of actions are found, overflows might be relieved quicker, leaving the grid to operate within its limits for an extended period of time. Such periods of stability lower the average iteration count per timestep. In conclusion, the results of the training regimes affirm the hypothesis that using a distance-guided search during training can improve performance in both the ability of the agent to survive through the scenarios, as the ability of the MCTS to find more optimal control sequences through fewer simulation iterations.

# 6

# Conclusion and Discussion

The drive for decarbonisation combined with a rapid increase in electricity demand is shifting the operational approach of the grid to a distributed solution with higher variance in generation and flows throughout the system. Ensuring a constant, reliable and safe electricity supply is increasingly becoming a challenge for network system operators. Recently, network topology control has been shown as an under-exploited flexibility in the grid, able to redirect power flows to alleviate overflows and voltage problems. Conventional computational methods are struggling with the complexity of finding optimal control strategies for this expansive problem. System operators often default to experiential intuition or pre-defined control solutions, resulting in sub-optimal operation. This urges the need for a method able to quickly approximate optimal network topology solutions, improving system security through the flexibility inherent to the system. In this thesis, a combined ML and tree search approach is introduced to address the challenges associated with computational complexity of the transmission switching problem.

## 6.1. Proposed Approach

The proposed approach aims to utilise the sample efficient training regime from a curriculum learned neural network, combined with the added security of a specialised Monte-Carlo Tree Search (MCTS). Using various modules of increasing difficulty labelled training data is generated for a neural network, which is then trained using imitation learning and reinforcement learning to mimic the training data and to improve its performance through experimentation, respectively. These neural networks are combined with the MCTS, which incorporates future outcomes through guided simulation, aiding in selecting actions that show better long-term performance. A training algorithm based on MCTS-verified actions is used for a sample efficient and stable training regime. To include further grid dependent information an electrical distance estimate based on hops is used. This proximity factor is used to guide the simulation tree towards actions closer to overloads, streamlining its selection process.

## 6.2. Research Questions and Answers

In this chapter the research questions are revisited and discussed based on the outcomes of the case studies. The resulting discussions and drawn conclusions are outlined in the following section.

### 6.2.1. Hybrid Curriculum Learning and Monte-Carlo Tree Search Approach

The first academic contribution of the thesis concerns the hybrid approach between the curriculum learning regime and a specialised MCTS to employ both the sample efficiency and the added security. The associated objective and research questions for this contribution are as follows:

**Objective 1:** Develop a hybrid network topology control agent, utilising a curriculum-learning pre-training regime and a Monte-Carlo Tree Search during operation.

- Q1 How well can a hybrid agent leverage both the pre-training sample-efficiency from the curriculum-learning regime and the improved security of the Monte-Carlo Tree Search?
- Q2 To what extent is the agent capable of maintaining network operations within limits using only network topology control actions?

**Q1 - Sample efficient and secure hybrid agent** The first research question defines the applicability of obtaining the advantages of both regimes within a hybrid approach. The proposed approach is able to leverage the curriculum learning training regime to pre-train the neural networks for the hybrid agent. The earlier modules within the curriculum learning regime are able to define the reduction potential for the action space size, and the effects of opting for an action space size of 100. The imitation learning module is able to quickly pre-train the neural network for usage in further models, where the RL module can be used to improve the performance of the neural networks. While RL offered an increase in performance compared to solely the imitation learning module, the agents combined with the MCTS showed significant enhancements in their performance. The integration of MCTS improved the decision-making process by focusing on longer-term strategic actions, allowing the agents to surpass the brute-force baseline, particularly when employing the *Junior* neural network.

That being said, the optimisation of hyperparameters and variables used throughout the training and testing regimes are found to be very influential, possibly offering an performance improvement throughout training and testing. In the case of this thesis there are two main groups of hyperparameters. Firstly, there are the parameters related to the agent and the environment: the action space it is able to use, the loading threshold before an agent action is required, and parameters concerning the action selection in the search tree. The action space used throughout this thesis offered greatly reduced computational times while offering only slightly lower flexibility. However, significant performance improvements are able to be obtained using larger action space sizes. Ideally, an optimal reduced action space does not compromise on flexibility, while greatly reducing computational times. The results from gathering action space data confirms that such an action space can be found, where less than 4% of the total actions available are shown to be used to reduce the loading.

The second group of hyper-parameters relate to the architecture and training settings of the neural networks. Hyper-band optimisation is used to determine the optimal training settings for the imitation learning stage. Tweaking the hyperparameters greatly influenced the obtained results, however, even using the found optimal settings, a relatively low accuracy for imitating the best actions is obtained. This is in accordance to state-of-the-art methods [66], where performance of the imitation learning was found to be relatively low. The results obtained better performance, however, only a subset of the 118-bus grid was used, entailing 35 buses. These results show that neural networks employing different architectures with greater generalisation capabilities might be needed to capture the complex relationship between the grid variables and the optimal transmission switching action.

To conclude, utilising a hybrid approach of a curriculum learning regime together with a specialised MCTS can improve the performance of a topology agent, proposing secure action trajectories able to surpass the baseline that does not incorporate future outcomes. However, more researched tuning of network architecture and training parameters is needed to accurately approximate optimal transmission switching actions.

**Q2 - Network topology control limitations** The second research questions aims to determine to what extent the grid can be maintained using solely topological reconfigurations for control actions. The results from the case studies show that utilising the MCTS to selectively simulate future outcomes can greatly help improve the performance of an agent utilising only grid reconfigurations, even surpassing the brute-force baseline. However, the details of the attacks by the adversarial agents greatly influence the performance, showcasing a limit to what can be obtained utilising the current approach.

While the topological agent is able to find better action trajectories of topological actions to alleviate overloads through highly loaded scenarios, it may fail to perform after unforeseen adversarial attacks. The adversarial agent can be compared to N-1 security, where the outage of a single component should not compromise the system security. While employing several base-level expert rules, N-1 simulation capabilities are not available for the tested agent. The expert rule employed to ensure automatic grid recovery is used to increase resilience during times when no overflows are present. However, this also means that grid reconfigurations to better prepare for N-1 scenarios during times where operational limits are about to be breached are not applied. Topological actions stemming from the MCTS are also not checked for N-1 stability, possibly decreasing grid resilience in times where overflows are present. This further implies unused flexibility in the case of preventive transmission switching, instead of the proposed method of corrective transmission switching.

To conclude, all agents seem to reach a level of performance after which improvements stagnate, where large advances in the performance throughout the training set do not directly translate to the same level of improvements throughout the testing set. This implies a ceiling, which might not be able to be surpassed using topological reconfiguration actions alone. This supports the further development and research into methods combining redispatch actions and topological reconfiguration, where smart usage of the flexibility of the grid can greatly reduce operational costs of the system, and where re-dispatching actions can come during states of high stress, which transmissive switching only can not solve.

### 6.2.2. Monte-Carlo Tree Search based Training Regime

The following academic contribution of the thesis that will be discussed concerns the usage of MCTS-checked actions for training, allowing direct training and improving sample efficiency. The associated objective and research questions for this contribution are as follows:

**Objective 2:** Utilise the capabilities of the Monte-Carlo Tree Search to generate training samples for the agent to optimise its performance.

- Q1 Can the robust actions proposed by the Monte-Carlo Tree Search, tested through simulation, be utilised as viable training samples for the agent?
- Q2 How significantly do the Monte-Carlo Tree actions improve the sample efficiency and performance stability during training?

**Q1 - MCTS training regime attainability** The first research question of this objective examines the attainability of using MCTS-prepared samples to establish a stable and sample-efficient training regime. The proposed approach is able to use the available 1,663 scenarios for sequential simulation, employing the MCTS according to the specified agent variables. The actions proposed by the MCTS are simulation tested and robust, and are able to be used directly, to improve the performance throughout the training set with more than $25\%$ after only three iterations.

The training approach is set up to utilise a distributed training regime, where each month is trained separately to increase computational times. The optimiser trains the neural network after each training sample, offering direct improvement and easy training, not necessitating a large, non-local reachable buffer. However, state-of-the-art training methods utilise random sampling and batch training to reduce training biases and variance [31]. So, while the proposed training regime is able to optimise the neural network performance directly through MCTS-verified actions, more research can be done to determine more optimal convergence in approximation capabilities through random sampling and batch training.

**Q2 - Improved sample efficiency and performance stability** The second research question of this objective aims to determine the improvements in sample efficiency and performance stability to be gained throughout the MCTS-training regime, especially compared to the RL training. The training regime utilises the available 1,663 scenarios for a single iteration, significantly improving performance through each iteration using only a fraction of the amount of interactions needed for the RL algorithm. Even while having complete a simulation tree for each action, computational times are still reduced

through the improved stability. The stability is showcased through the monthly performances throughout the training iterations, where only a few slightly worse performances are recorded from the next iteration to the one before. This indicates that the newer trained model, averaged over all the monthly trained models of the last iteration, has improved approximation capabilities to improve performance throughout the yearly scenarios.

The differences in performances of the prior prediction agents and the MCTS-agents reveal that the training regime chosen has a high impact on performance in different scenarios. Even though the RL trained agent outperforms the MCTS-trained when utilising only its prior predictions, the RL trained agent did not achieve the same level of performance when combined with the MCTS. The RL trained neural network, while more optimised towards providing better actions directly, might allow for less exploration within the search tree, hindering improved performance. An optimally trained RL neural network should allow the same action trajectories to be found, however, when this convergence has not yet been reached, performance might weaken.

In conclusion, the use of robust MCTS-confirmed actions during the training phase proved beneficial. It allowed for direct improvement in robust operations, increasing sample efficiency through verifying sample actions and increasing stability by incorporating exploration in the simulation tree, and limiting exploration needed during the training regime.

### 6.2.3. Incorporating Proximity through an Electrical Distance Factor

The final academic contribution of the thesis that will be discussed involves the added proximity factor within the search tree to increase simulation and training convergence. The associated objective and research questions for this contribution are as follows:

**Objective 3:** Incorporate the proximity of actions to the overflow as additional information in the Monte-Carlo Tree Search.

- Q1 How can the proximity be considered for estimating electrical distance and enhancing Monte-Carlo Tree Search convergence?
- Q2 How effectively does the inclusion of a proximity factor improve the convergence of both the Monte-Carlo Tree Search and the agent action policy during training?

**Q1 - Hop based proximity factor** The first question regarding the final objective tries to determine how the proximity of actions can be incorporated into the search tree. This question touches on the suitability of using a hop based approximation for electrical distance and the operational applicability when applied using a breadth-first search. Test results showed the positive relationship between hop distance and effectiveness of actions, indicating the applicability of using the hop approximation to enable quick proximity calculations. Utilising the proximity factor within the search tree facilitated a performance boost similar to the increase seen after three MCTS training iterations, while reducing the iterations needed to reach convergence within the search tree. Employing the breadth-first search for defining the proximity factor showcases that, while the computational time increases, no significant hinder is found through this increase, keeping the operational times within a second for a 30 minute time frame.

However, the proposed method does not utilise line or connection parameters to calculate electrical distance to determine a proximity factor. While LODF methods are very quick in determining line flow changes once the initial LODF matrix has been computed, it is less applicable to use in the setting where altering the topology is used to alleviate overflows. Each time a new topology configuration would be used, the LODF matrix would have to be recalculated. A state-of-the-art method uses transformed LODFs to allow rapid computation of the effects of busbar splitting on the DC load flow [42]. However, more research is needed to allow the usage within larger grids to make a better estimation of the actual electrical influence a topological action can have on an overflowing line. Further research is also needed to tune the formula for defining and applying the proximity factor. In the current proposed method, the proximity factor is used to boost the initial prior predictions of the actions, prioritising those closer to the highest overflow in the system. More elegant methods might improve the performance

by integrating information about the level of overflow, or information about multiple lines close together experiencing an overflow, shifting the focus towards electrical distance when needed most.

In conclusion, while a hop based proximity factor can be used to improve convergence within the MCTS, more research can be done to explore better approximations of electrical distance, and how to elegantly apply the factor when needed most.

**Q2 - Effectiveness in improving convergence** The final research question aims to define the improvements to be gained within the convergence of the simulation tree and of the entire MCTS-training regime, when utilising the proximity factor within the tree search. The results of the case studies show that direct application of the proximity factor within the *Junior*-MCTS agent improves performance similar to three training iterations, while reducing the amount iterations needed per timestep by more than $5\%$. This showcases a significant reduction, taking into account that in many timesteps the simulation tree will have to go to its limit in both test cases, as no actions are available to provide long-term reduction and early-stop the simulation algorithm. Similar to higher reductions in iteration counts were also found throughout the MCTS-training regime adapting the proximity factor, highlighting quicker convergence within the search tree. The performance regarding timesteps survived is able to improve quicker throughout the proximity factor MCTS-training regime, indicating that quicker convergence of the training algorithm can be obtained.

However, the current approach applies the proximity factor experimentally for each iteration, reducing the influence throughout the training iterations, guiding the network towards a converged state where no more 'bonuses' should be needed. Further research can be done towards determining the optimal function, or other method, for applying the distance factor gradually throughout training. This can go hand-in-hand with further research into better approximation methods for the electrical distance or how to best define the proximity factor formula.

In conclusion, the quick computational methods and effectiveness in reducing convergence within the search tree paves the way for application within larger, real-time grids, efficiently identifying optimal control sequences. However, further research needs to be done towards determining better approximations or calculations for actual electrical distance, and how to best apply this within a training regime for an AI model.

In final conclusion, while fully autonomous AI-based optimal grid operation still seems far away, the findings of the case studies investigating the developed agents show that the combination of smart heuristics, ML and model-based methods opens the door to robust and possibly scalable solution for assistive grid topology control.

## 6.3. Future Work

Extending on this thesis, further developments of the proposed approach first and foremost involves the inclusion of an N-1 check. Within grid operation, N-1 security is not an addition, but a necessity. LODFs can be utilised to quickly determine the most drastic effects of possible unplanned outages, where topological actions can be used as a preventive measure to increase resilience. Further developed research into Bus Split Distribution Factors (BSDF) may allow the rapid assessment of N-1 security within the search tree, where the top-k proposed actions might be simulated to assess the resilience of the resulting state.

The optimal tuning of the agent parameters, such as the reduced action space and the agent action limit should be further researched. While a reduced action space of 100 was chosen for this approach due to prior estimated computational complexity, larger action spaces might offer better improvements in performance while still allowing real-time operation. A more extensive hyperparameter search should be applied to determine their influence for different combinations.

Further extensions can be made regarding the application of combined actions. Currently, only topological actions regarding a single substation are used at any one timestep. While such an action may

comprise multiple switching actions within the same substation, combining actions of multiple substations might prove useful to relieve overloads quickly in instances where single actions cannot achieve this. While allowing sub-steps of actions within a single timestep would quickly increase computational complexity, smart methods might be used to determine several valuable combinations, similar to defining the best single actions using the curriculum learning modules. These combinations might be used as either an emergency module, or to simulate with the top-k proposed actions to check for suitability. Similarly, line reconnections and grid recovery might be combined to be allowed within times of overflows as well, as long as resilience is not simulated to decrease.

Added research can be done in the utility of the pre-trained neural networks. Varying neural network architectures should be considered and tested, including extensive hyperparameter testing using hyperband optimisation. The introduced limitations of using a pre-trained neural network for a MCTS approach should be analysed in the case the training performance is not as desired.

Inclusion of an agent that proposes actions on the continuous operations within the grid can help extending the conclusions about performance of utilising both topological and redispatching actions. Currently, an L2RPN baseline employs an agent based on CVXPY and the DC approximation of the power flow, defining optimal actions for the continuous operations within the grid [74]. Solely using this can determine a baseline in performance and economic effects of using redispatching, while combining it with the proposed method can help determine their combined effectiveness. While this method is relatively slow, it could be employed only at times where the MCTS cannot find any fitting topological actions, reducing the computational times to appropriate levels.

# References


1. *Tennet steekt 13 miljard in Versnelde vernieuwing Stroomnetwerk* July 2022. https://nos.nl/artikel/2435398-tennet-steekt-13-miljard-in-versnelde-vernieuwing-stroomnetwerk.
2. Korad, A. S. & Hedman, K. W. Robust corrective topology control for system reliability. *IEEE Transactions on Power Systems* **28,** 4042–4051 (2013).
3. Ruiz, P. A., Foster, J. M., Rudkevich, A. & Caramanis, M. C. Tractable transmission topology control using sensitivity analysis. *IEEE Transactions on Power Systems* **27,** 1550–1559 (2012).
4. Koglin, H. & Müller, H. Corrective switching: a new dimension in optimal load flow. *International Journal of Electrical Power & Energy Systems* **4,** 142–149 (1982).
5. Kelly, A., O'Sullivan, A., de Mars, P. & Marot, A. Reinforcement learning for electricity network operation. *arXiv preprint arXiv:2003.07339* (2020).
6. Marot, A. *et al.* L2rpn: Learning to run a power network in a sustainable world neurips2020 challenge design. *Réseau de Transport d'Électricité, Paris, France, White Paper* (2020).
7. Marot, A. *et al.* Learning to run a power network with trust. *Electric Power Systems Research* **212,** 108487 (2022).
8. Serré, G. *et al.* Reinforcement learning for Energies of the future and carbon neutrality: a Challenge Design. *arXiv preprint arXiv:2207.10330* (2022).
9. Matavalam, A. R., Guddanti, K. P., Weng, Y. & Ajjarapu, V. Curriculum Based Reinforcement Learning of Grid Topology Controllers to Prevent Thermal Cascading. *IEEE Transactions on Power Systems* (2022).
10. Silver, D. *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362,** 1140–1144 (2018).
11. Dorfer, M., Fuxjäger, A. R., Kozak, K., Blies, P. M. & Wasserer, M. Power Grid Congestion Management via Topology Optimization with AlphaZero. *arXiv preprint arXiv:2211.05612* (2022).
12. Petinrin, J. & Shaabanb, M. Impact of renewable generation on voltage control in distribution systems. *Renewable and Sustainable Energy Reviews* **65,** 770–783 (2016).
13. Behnert, M. & Bruckner, T. *Causes and effects of historical transmission grid collapses and implications for the German power system* tech. rep. (Beiträge des Instituts für Infrastruktur und Ressourcenmanagement, 2018).
14. Gourtani, A., Xu, H., Pozo, D. & Nguyen, T.-D. Robust unit commitment with n-1 n-1 security criteria. *Mathematical Methods of Operations Research* **83,** 373–408 (2016).
15. Fernández-Guillamón, A., Gómez-Lázaro, E., Muljadi, E. & Molina-García, Á. Power systems with high renewable energy sources: A review of inertia and frequency control strategies over time. *Renewable and Sustainable Energy Reviews* **115,** 109369 (2019).
16. Hines, P., Balasubramaniam, K. & Sanchez, E. C. Cascading failures in power grids. *Ieee Potentials* **28,** 24–30 (2009).
17. Lowe, R. & Drummond, P. Solar, wind and logistic substitution in global energy supply to 2050–Barriers and implications. *Renewable and Sustainable Energy Reviews* **153,** 111720 (2022).
18. Gils, H. C. Balancing of intermittent renewable power generation by demand response and thermal energy storage (2015).
19. Bird, L. *et al.* Wind and solar energy curtailment: A review of international experience. *Renewable and Sustainable Energy Reviews* **65,** 577–586 (2016).
20. Ruiz, P. A. & Caspary, J. *Transmission Topology Optimization: a software solution for improving congestion management* Energy Systems Integration Group. https://www.esig.energy/resources/transmission-topology-optimization-a-software-solution-for-improving-congestion-management-webinar-recording/.
21. Zhou, Z.-H. *Machine learning* (Springer Nature, 2021).
22. Arulkumaran, K., Deisenroth, M. P., Brundage, M. & Bharath, A. A. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

23. Rumelhart, D. E., Widrow, B. & Lehr, M. A. The basic ideas in neural networks. *Communications of the ACM* **37,** 87–93 (1994).
24. Silver, D. *Deep Reinforcement Learning Tutorial* Lecture slides available at David Silver's personal website. 2020. https://www.davidsilver.uk/wp-content/uploads/2020/03/deep_rl_tutorial_small_compressed.pdf.
25. Learning, M. Alexander Jung.
26. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. *Neural networks* **2,** 359–366 (1989).
27. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural networks* **4,** 251–257 (1991).
28. Reed, R. & MarksII, R. J. *Neural smithing: supervised learning in feedforward artificial neural networks* (Mit Press, 1999).
29. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).
30. Silver, D. *Introduction to Reinforcement Learning with David Silver, Lecture 2: Markov Decision Processes* https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf.
31. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *nature* **529,** 484–489 (2016).
32. Silver, D. *et al.* Mastering the game of go without human knowledge. *nature* **550,** 354–359 (2017).
33. Campbell, M., Hoane Jr, A. J. & Hsu, F. Deep blue. *Artificial intelligence* **134,** 57–83 (2002).
34. Tesauro, G. & Galperin, G. On-line policy improvement using Monte-Carlo search. *Advances in Neural Information Processing Systems* **9** (1996).
35. Schrittwieser, J. *et al.* Mastering atari, go, chess and shogi by planning with a learned model. *Nature* **588,** 604–609 (2020).
36. Marot, A. *et al. Learning to run a power network challenge: a retrospective analysis* in *NeurIPS 2020 Competition and Demonstration Track* (2021), 112–132.
37. RTÉ France. *Grid2Op Documentation* Revision a819a777. https://grid2op.readthedocs.io/en/latest/.
38. Fisher, E. B., O'Neill, R. P. & Ferris, M. C. Optimal transmission switching. *IEEE Transactions on Power Systems* **23,** 1346–1355 (2008).
39. Numan, M. *et al.* The Role of Optimal Transmission Switching in Enhancing Grid Flexibility: A Review. *IEEE Access* (2023).
40. Heidarifar, M. & Ghasemi, H. A network topology optimization model based on substation and node-breaker modeling. *IEEE Transactions on Power Systems* **31,** 247–255 (2015).
41. Subramanian, M., Viebahn, J., Tindemans, S. H., Donnot, B. & Marot, A. *Exploring grid topology reconfiguration using a simple deep reinforcement learning approach* in *2021 IEEE Madrid PowerTech* (2021), 1–6.
42. Van Dijk, J., Viebahn, J., Cijsouw, B. & van Casteren, J. Bus Split Distribution Factors (2023).
43. Mnih, V. *et al.* Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
44. Lan, T. *et al. AI-based autonomous line flow control via topology adjustment for maximizing time-series ATCs* in *2020 IEEE Power & Energy Society General Meeting (PESGM)* (2020), 1–5.
45. GEIRINA, RTE & INRIA. *Learning to Run a Power Network* Competition: https://competitions.codalab.org/competitio Requires Python >= 3.6. License: GNU Lesser General Public License v3.0. 2019. https://github.com/shidi1985/L2RPN.
46. Wang, Z. *et al. Dueling Network Architectures for Deep Reinforcement Learning* in *Proceedings of The 33rd International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger, K. Q.) **48** (PMLR, New York, New York, USA, June 2016), 1995–2003. https://proceedings.mlr.press/v48/wangf16.html.
47. Matavalam, A. R. *Learning to Run the Power Network using a Reinforcement Learning Actor Agent* RTÉ. 2020. https://drive.google.com/file/d/1e6Q0taq7UaN-h-OPLMkVmId4xg_XrBL4/view?usp=drive_link.
48. Narvekar, S. *et al.* Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research* **21,** 7382–7431 (2020).
49. Zhou, B. *et al. Action set based policy optimization for safe power grid management* in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Confer-*

*ence, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part V 21* (2021), 168–181.
50. Chauhan, A., Baranwal, M. & Basumatary, A. *Powrl: A reinforcement learning framework for robust management of power networks* in *Proceedings of the AAAI Conference on Artificial Intelligence* **37** (2023), 14757–14764.
51. Yoon, D., Hong, S., Lee, B.-J. & Kim, K.-E. *Winning the l2rpn challenge: Power grid management via semi-markov afterstate actor-critic* in *International Conference on Learning Representations* (2020).
52. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
53. Wang, C., Xie, W., Zhang, X., Qin, R. & Yu, Y. *L2RPN-ICAPS 2021 presentation* EPRI. 2021. https://www.youtube.com/watch?v=WOt8xgpC370&list=PLWVnM03ByAqJ9KmaUQ-LTu7lVnhvvlhLZ&index=4.
54. Martinez, H. *Learning to run a power network competition - ICAPS 2021 presentation* EPRI. 2021. https://www.youtube.com/watch?v=WOt8xgpC370&list=PLWVnM03ByAqJ9KmaUQ-LTu7lVnhvvlhLZ&index=4.
55. enliteAI. *Maze-RL L2RPN - ICAPS 2021 Submission* https://github.com/enlite-ai/maze-l2rpn-2021-submission. 2021.
56. Bengio, Y., Louradour, J., Collobert, R. & Weston, J. *Curriculum learning* in *Proceedings of the 26th annual international conference on machine learning* (2009), 41–48.
57. AsprinChina. *NeurIPS Competition 2020: Learning to Run a Power Network (L2RPN) - Robustness Track* https://github.com/AsprinChina/L2RPN_NIPS_2020_a_PPO_Solution. The solution based on this repository ranks 2nd in the NeurIPS 2020, Track 1 L2RPN competition. 2020.
58. Glover, J. D., Sarma, M. S. & Overbye, T. J. *Power systems analysis and design* 2012.
59. Dobson, I. *et al.* Electric power transfer capability: concepts, applications, sensitivity and uncertainty. *PSerc Publication* (2001).
60. Bergen, A. R. *Power systems analysis* (Pearson Education India, 2009).
61. Wei, N. *A Line Outage Study for Prediction of Static Power Flow Redistribution* PhD thesis (Virginia Tech, 2016).
62. Von Meier, A. *Electric power systems: a conceptual introduction* (John Wiley & Sons, 2006).
63. Wood, A. J., Wollenberg, B. F. & Sheblé, G. B. *Power generation, operation, and control* (John Wiley & Sons, 2013).
64. Fu, C. & Bose, A. Contingency ranking based on severity indices in dynamic security analysis. *IEEE Transactions on power systems* **14,** 980–985 (1999).
65. Sekhar, P. & Mohanty, S. *Power system contingency ranking using Newton Raphson load flow method* in *2013 Annual IEEE India Conference (INDICON)* (2013), 1–4.
66. Lehna, M., Viebahn, J., Marot, A., Tomforde, S. & Scholz, C. Managing power grids through topology actions: A comparative study between advanced rule-based and reinforcement learning agents. *Energy and AI* **14,** 100276 (2023).
67. Martín Abadi *et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015. https://www.tensorflow.org/.
68. Pedregosa, F. *et al.* Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **12,** 2825–2830 (2011).
69. Moritz, P. *et al. Ray: A distributed framework for emerging {AI} applications* in *13th USENIX symposium on operating systems design and implementation (OSDI 18)* (2018), 561–577.
70. Jaderberg, M. *et al.* Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).
71. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
72. Naftaly, U., Intrator, N. & Horn, D. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems* **8,** 283 (1997).
73. (DHPC), D. H. P. C. C. *DelftBlue Supercomputer (Phase 1)* https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1 (2022).
74. France, R. *L2RPN_Baselines: Repository hosting reference baselines for the L2RPN challenge* https://github.com/rte-france/l2rpn-baselines. 2023.

# A

# Figures

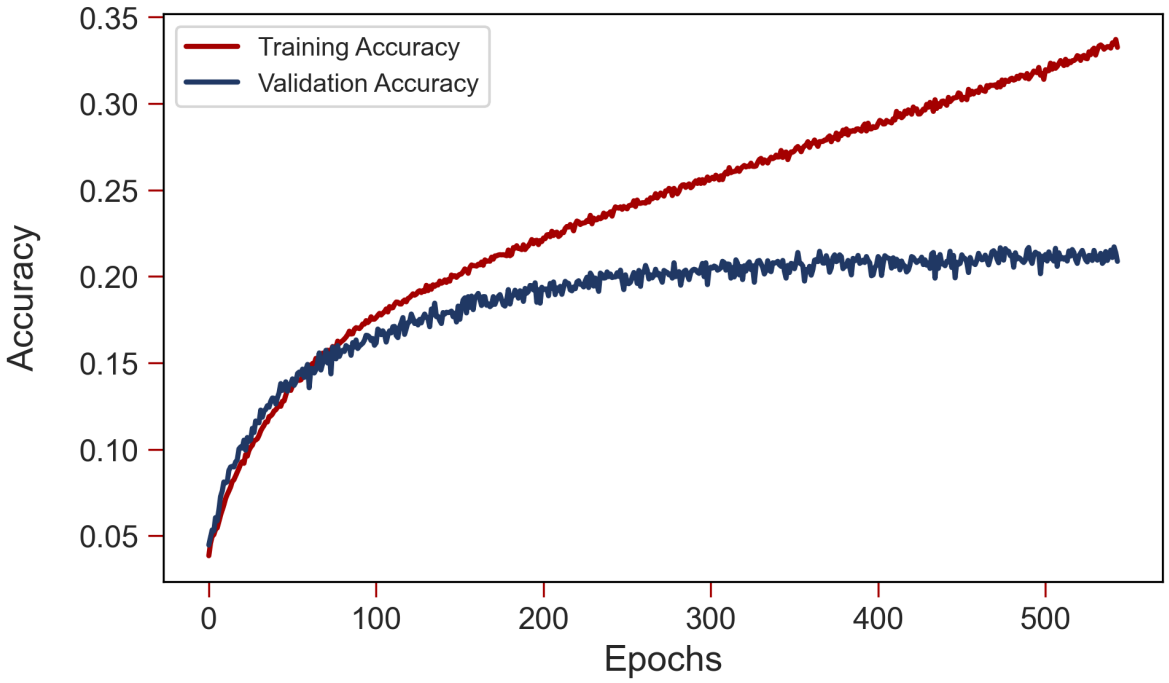



**Figure A.1:** Training and validation accuracy of the neural network during the Junior training regime

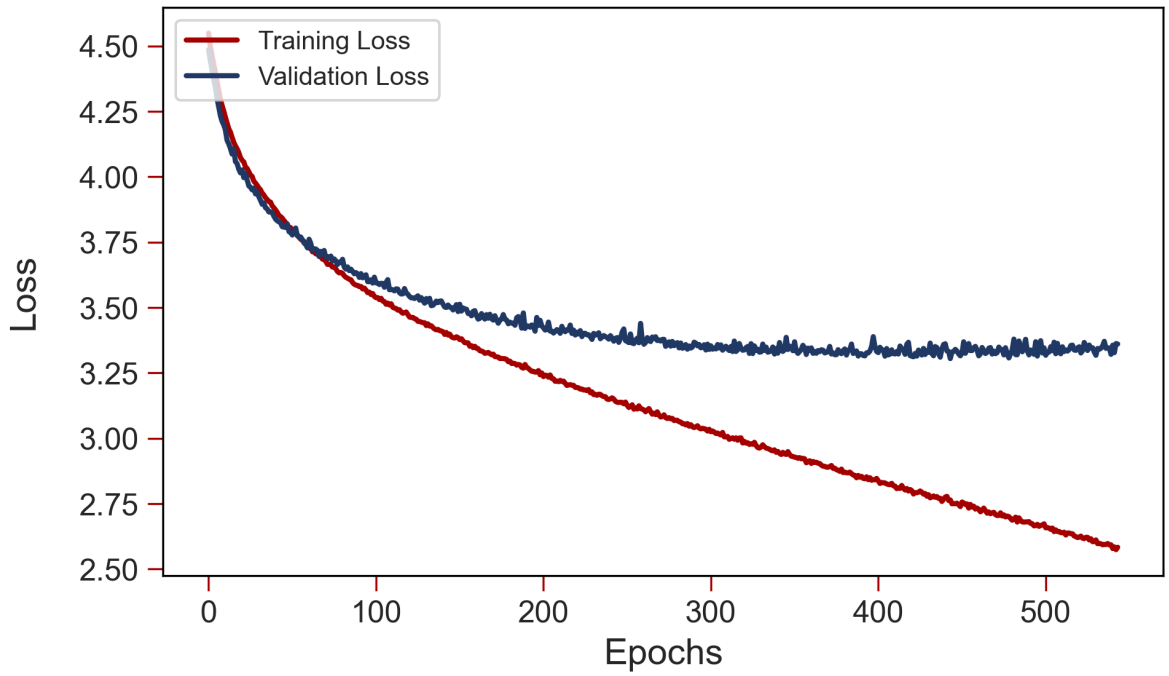



**Figure A.2:** Training and validation loss of the neural network during the Junior training regime

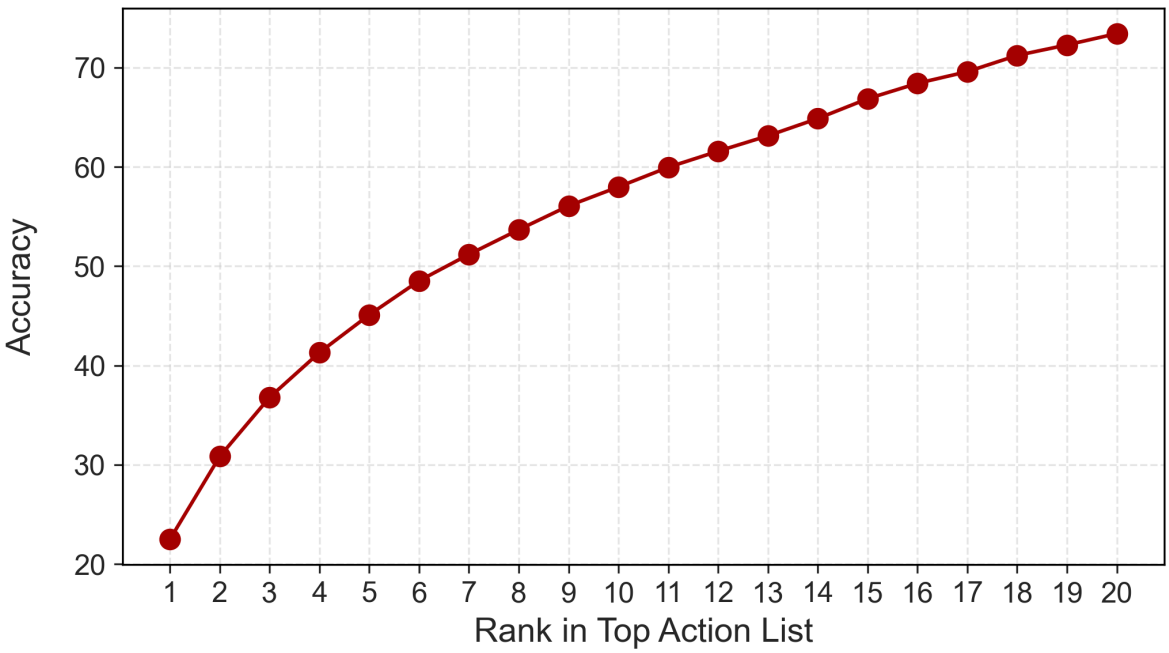



**Figure A.3:** Validation accuracy for the top-20 actions for the Junior trained neural network

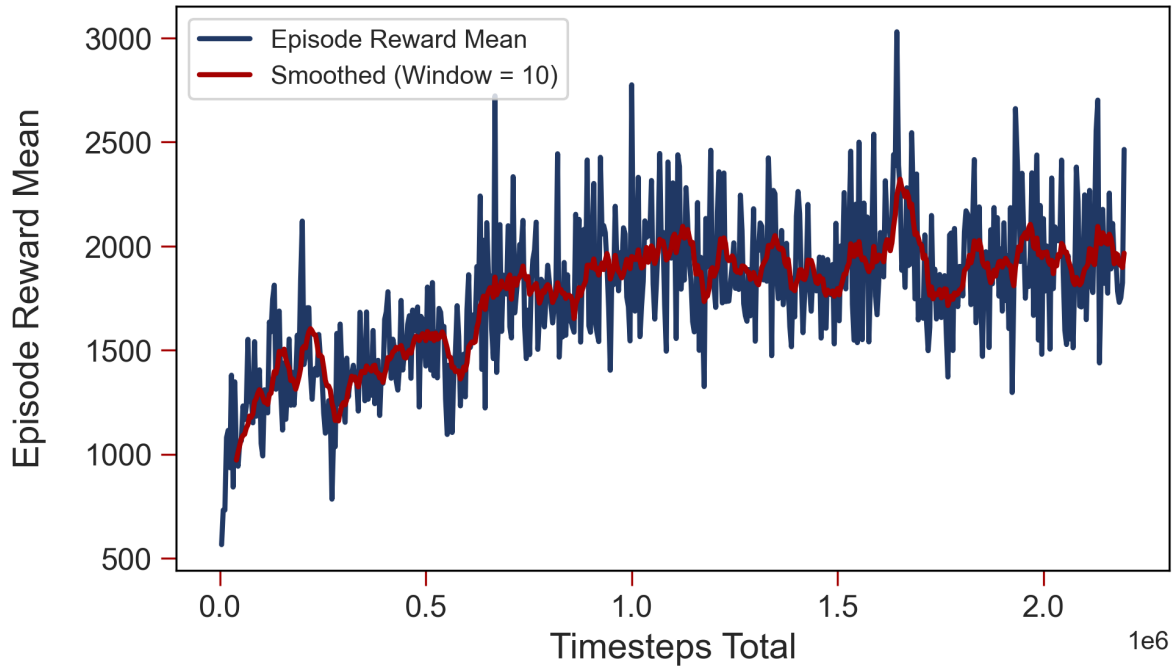



**Figure A.4:** Mean episodic reward per timestep during the Senior training regime, smoothed with a window of value 10

# B

# Tables

**Table B.1:** Junior Neural Network parameters and values

| Parameter | Value |
|---|---|
| Input Layer | 1221 Variables |
| Hidden Layer 1 | 400 Neurons |
| Dropout Layer 1 | 0.25 |
| Hidden Layer 2 | 773 Neurons |
| Dropout Layer 2 | 0.40 |
| Hidden Layer 3 | 1044 Neurons |
| Hidden Layer 4 | 344 Neurons |
| Output Layer Linear | 100 Actions |
| Activation | Relu Activation |
| Batchsize | 768 |
| Initialiser | Orthogonal |
| Learning Rate | 5e-5 |
| Epochs | 1000 |
| Early Stopping | 100 steps |

**Table B.2:** Population Based Training Configuration for the Senior module, using Ray's RLlib

| Parameter | Value |
|---|---|
| Time Attribute | *"training_iteration"* |
| Metric | *"episode_reward_mean"* |
| Mode | *"max"* |
| Perturbation Interval | 50 |
| Resample Probability | 0.5 |
| Learning Rate Mutations | [1e-3, 5e-4, 1e-4, 5e-5, 1e-5] |
| Number of SGD Iterations Mutations | Randint(3, 10) |
| VF Loss Coefficient Mutations | Randuniform(0.5, 1) |
| Clip Parameter Mutations | Randuniform(0.01, 0.5) |
| Gamma Mutations | Randuniform(0.975, 1) |
| Entropy Coefficient Mutations | Randuniform(0.00001, 0.01) |

**Table B.3:** Senior Neural Network parameters and values

| Parameter | Value |
|---|---|
| Learning rate | 1e-5 |
| Clip Parameter | 0.07689201 |
| Entropy Coefficient | 0.00093928 |
| Gamma | 0.98829384 |
| VF Loss coefficient | 0.79847594 |
| Number of SGD Iterations | 4 |
| Input Layer | 1221 Variables |
| Hidden Layer 1 | 400 Neurons |
| Dropout Layer 1 | 0.25 |
| Hidden Layer 2 | 773 Neurons |
| Dropout Layer 2 | 0.40 |
| Hidden Layer 3 | 1044 Neurons |
| Hidden Layer 4 | 344 Neurons |
| Output Layer Linear | 100 Actions |

**Table B.4:** Further description of the used variables during the training process of the Junior and Senior neural networks

| Category | Variable | Description |
|---|---|---|
| Time Values | *month* | Month of the Scenario |
| | *day* | Day in time step $t$ |
| | *hour_of_day* | Hour of time step $t$ |
| | *minute_of_hour* | Minute of time step $t$ |
| | *day_of_week* | Weekday of time step $t$ |
| Generation and Load | *gen_p*, *gen_q* | Active and reactive production value of each generator |
| | *gen_v* | Voltage magnitude of each generator at connected bus |
| | *load_p*, *load_q* | Active and reactive load value of each consumption |
| | *load_v* | Voltage magnitude of each consumption at connected bus |
| Lines | *p_or*, *q_or* | Active and reactive power flow at the origin end of each power line |
| | *v_or* | Voltage magnitude at the origin end of each power line |
| | *a_or* | Current flow at the origin end of each power line |
| | *p_ex*, *q_ex* | Active and reactive power flow at the extremity end of each power line |
| | *v_ex* | Voltage magnitude at the extremity end of each power line |
| | *a_ex* | Current flow at the extremity end of each power line |
| | *rho* | Capacity of each power line |
| | *line_status* | Whether the line is connected or disconnected |
| | *time_step_overload_low* | Number of time steps since overflow |
| Bus Information | *topo_vector* | Vector of the topology configurations |
| Cooldowns | *time_before_cooldown_line* | Time before an action can be done on the line |
| | *time_before_cooldown_sub* | Time before an action can be done on the substation |
| Maintenance | *time_next_maintenance* | Time when the next maintenance of line is planned |
| | *duration_next_maintenance* | Duration of planned maintenance |

**Table B.5:** Scenarios for both the Test and Hyperparameter set, used during hyperparameter validation and case studies

| Scenarios Test Set | Scenarios Hyperparameter Set |
|---|---|
| *2050-01-03_17* | *2050-01-03_7* |
| *2050-01-10_17* | *2050-01-10_5* |
| *2050-01-24_17* | *2050-01-17_2* |
| *2050-02-07_18* | *2050-01-24_6* |
| *2050-02-14_3* | *2050-01-31_2* |
| *2050-02-21_18* | *2050-02-07_4* |
| *2050-03-07_17* | *2050-02-14_6* |
| *2050-03-14_17* | *2050-02-21_3* |
| *2050-03-21_17* | *2050-02-28_2* |
| *2050-03-28_17* | *2050-03-07_6* |
| *2050-04-04_17* | *2050-03-14_14* |
| *2050-04-11_17* | *2050-03-21_6* |
| *2050-04-11_3* | *2050-03-28_17* |
| *2050-04-25_17* | *2050-04-04_6* |
| *2050-05-02_17* | *2050-04-11_6* |
| *2050-05-09_17* | *2050-04-18_19* |
| *2050-05-16_17* | *2050-04-25_14* |
| *2050-05-23_17* | *2050-04-25_6* |
| *2050-05-30_14* | *2050-05-09_6* |
| *2050-05-30_17* | *2050-05-16_4* |
| *2050-06-06_17* | *2050-05-23_6* |
| *2050-06-13_17* | *2050-05-30_6* |
| *2050-06-20_17* | *2050-06-06_6* |
| *2050-06-27_17* | *2050-06-13_12* |
| *2050-07-04_1* | *2050-06-20_6* |
| *2050-07-04_17* | *2050-06-27_10* |
| *2050-07-11_17* | *2050-07-04_6* |
| *2050-07-18_17* | *2050-07-11_6* |
| *2050-07-25_17* | *2050-07-18_11* |
| *2050-08-01_17* | *2050-07-25_6* |
| *2050-08-08_17* | *2050-08-01_6* |
| *2050-08-15_17* | *2050-08-08_6* |
| *2050-08-15_3* | *2050-08-15_6* |
| *2050-08-22_17* | *2050-08-22_6* |
| *2050-08-29_17* | *2050-08-29_6* |
| *2050-09-05_17* | *2050-09-05_6* |
| *2050-09-19_17* | *2050-09-12_6* |
| *2050-09-26_17* | *2050-09-19_16* |
| *2050-10-03_17* | *2050-09-26_6* |
| *2050-10-03_18* | *2050-10-03_6* |
| *2050-10-10_17* | *2050-10-10_6* |
| *2050-10-17_3* | *2050-10-17_6* |
| *2050-10-24_17* | *2050-10-24_6* |
| *2050-10-31_17* | *2050-10-31_6* |
| *2050-11-07_17* | *2050-11-07_6* |
| *2050-11-28_17* | *2050-11-14_6* |
| *2050-12-05_17* | *2050-11-21_1* |
| *2050-12-12_1* | *2050-11-28_6* |
| *2050-12-12_17* | *2050-12-05_6* |
| *2050-12-19_17* | *2050-12-12_6* |
| *2050-12-26_17* | *2050-12-19_9* |
| *2050-12-26_7* | *2050-12-26_11* |