FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Graph Reinforcement Learning for Improving Smart Grid Services

António Bernardo Linhares Oliveira

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. António Costa & Prof. Rosaldo Rossetti

August 13, 2024

Graph Reinforcement Learning for Improving Smart Grid Services

António Bernardo Linhares Oliveira

Mestrado em Engenharia Informática e Computação

August 13, 2024

Resumo

Os grafos são representações que descrevem problemas e os seus objectos em domínios orientados para as redes, como as redes eléctricas, os transportes ou as redes sociais. Estas representações podem captar não só os conceitos e as suas respectivas propriedades, mas também as relações entre esses conceitos, resultando em estruturas de dados frequentemente complexas e esparsas que são especialmente úteis para representar problemas em que a topologia de uma rede desempenha um papel importante. No contexto da utilização destas estruturas em algoritmos de aprendizagem computacional, o seu desempenho torna-se dependente não só da sua conceção e seleção dos atributos relevantes e parâmetros, mas também das representações subjacentes utilizadas para captar a essência das estruturas em grafo.

A Aprendizagem por Reforço em Grafos ou *Graph Reinforcement Learning* (GRL) é um tópico que tem merecido grande atenção por parte dos académicos nos últimos anos. Ao permitir que as técnicas de Aprendizagem por Reforço aprendam e optimizem processos de decisão sequenciais em ambientes baseados em grafos, os sistemas podem ser melhorados de forma a tirar partido das características da topologia dos grafos em domínios de aplicação associados a redes. Com os avanços no final da década de 2010 em Redes Neuronais de Grafos, ou *Graph Neural Networks*, na aprendizagem e extração de representações eficientes de grafos, foram propostos métodos mais sofisticados de GRL e o tópico começou a atrair mais curiosidade dos académicos. Embora, nos últimos anos, muito trabalho tenha sido feito nesta área, a pesquisa à volta do GRL ainda é considerada estar em fase inicial.

Além disso, considerando os actuais desafios globais associados à sustentabilidade e aos sistemas energéticos, há uma necessidade crescente de avanços em sistemas inteligentes focados em modernizar as redes de distribuição e transmissão de energia. Atualmente, as fontes de energia renováveis desempenham um papel importante na redução da dependência dos combustíveis fósseis, o que altera a topologia dos sistemas de distribuição de energia à medida que os consumidores adquirem a capacidade de gerar energia renovável. Com as melhorias na Inteligência Artificial e Aprendizagem Computacional, os sistemas podem ser adaptados à descentralização da produção e gerir eficientemente a monitorização, distribuição e transmissão da energia. Neste trabalho, a ênfase principal reside na melhoria dos algoritmos de GRL que serão aplicados no contexto do problema da distribuição dinâmica e económica de energia como seu principal domínio de aplicação, considerando fontes de energia renováveis e sistemas de armazenamento de energia.

Desta forma, esta dissertação tem como objetivo fazer avançar a investigação existente sobre técnicas de Aprendizagem por Reforço em Grafos através de: (1) realizar uma revisão exaustiva da literatura recente relativa às várias abordagens de GRL propostas e de sistemas de distribuição dinâmica e económica de energia, de modo a obter uma perspetiva global das técnicas recentes mais avançadas e das suas limitações; (2) realizar um estudo empírico comparativo e sistemático das diferentes técnicas de GRL no problema de distribuição dinâmica e económica, considerando cenários de estudo de caso de diferentes dimensões modelados por uma simulação de uma rede de distribuição de energia eléctrica; (3) propor um modelo que melhor integre as capacidades das

técnicas de *Deep Reinforcement Learning* e *Graph Neural Networks*, com base nos resultados do estudo empírico e com melhorias no desempenho e escalabilidade face aos modelos propostos pela literatura.

Abstract

Graphs are structures that depict problems and their objects in network-oriented domains such as power grids, transport or social networks. These representations can capture no only the concepts and their respective properties but the intricate relationships between those concepts, resulting in often complex and sparse data structures that are especially useful for representing problems where network topology plays a major role. In the context of using these structures in machine learning algorithms, their performance becomes not only dependent on its design and the selection of relevant features and parameters, but also on the underlying representations used to capture the essence of graph structures.

Graph Reinforcement Learning (GRL) is a topic that has earned significant attention from academics in the last few years. By enabling Reinforcement Learning techniques to learn and optimize sequential decision-making processes in graph-based environments, systems can be improved and gain the ability to leverage graph topology features in network-oriented application domains. With the advancements in the late 2010s on Graph Neural Networks on learning how to extract efficient graph representations from a given scenario, more sophisticated methods of GRL were proposed and the topic started to attract the curiosity of scholars. Although, in recent years, a lot of work has been done in this area, research on techniques is still considered to be in an early stage.

Furthermore, considering the current global challenges associated with sustainability and energy systems, there is an increasing need for advancements in energy-focused intelligent systems to modernize the current power grids. In the present, renewable energy sources play a major role in reducing the reliance on fossil fuels, which changes the topology of energy distribution systems as consumers gain the capability to generate renewable power. With the improvements in Artificial Intelligence and Machine Learning, systems can be adapted to the decentralization of energy production and efficiently manage the monitoring, distribution and transmission of energy systems. In this work, the primary emphasis lies on improving GRL algorithms which will be applied to solve the dynamic economic power dispatch problem as its main application domain, considering renewable energy sources and energy storage systems.

In this manner, this dissertation aims to advance the existing research on Graph Reinforcement Learning techniques by: (1) conducting a thorough review of the recent literature regarding various proposed GRL approaches and Dynamic Economic Dispatch Systems to gain an overall perspective of the recent state-of-the-art techniques and their limitations; (2) performing a comparative and systematic empirical study on the different GRL techniques on the Dynamic Economic Power Dispatch problem, considering case study scenarios of different sizes modelled by a power distribution grid simulation (3) propose a model that better integrates the capabilities of Deep Reinforcement Learning Agents with Graph Neural Networks, based on the results of the empirical study, with improvements in performance and scalability.

Keywords: Graph Reinforcement Learning, Graph Neural Networks, Deep Reinforcement Learning, Smart Grid, Dynamic Economic Dispatch

ACM Classification: Computing Methodologies $\rightarrow$ Machine Learning $\rightarrow$ Learning Paradigms $\rightarrow$ Reinforcement Learning

Acknowledgements

To my supervisors for all the guidance and teachings during the development of this work
To my parents who always supported me and expected me to become the best version of myself

António Oliveira

“Man is not worried by real problems
so much as by his imagined anxieties about real problems”

Epictetus

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

IT Information Technology

ANN Artificial Neural Network

MLP Multilayer Perceptron

CNN Convolutional Neural Network

RL Reinforcement Learning

MDP Markov Decision Process

DRL Deep Reinforcement Learning

GNN Graph Neural Network

GRL Graph Reinforcement Learning

SAC Soft Actor-Critic

DDPG Deep Deterministic Policy Gradient

DQN Deep Q-Network

PPO Proximal Policy Optimization

GCN Graph Convolutional Network

GAT Graph Attention Network

GIN Graph Isomorphism Network

ADN Active Distribution Network

DED Dynamic Economic Dispatch

ESS Energy Storage System

RES Renewable Energy Source

ML Machine Learning

Chapter 1

Introduction

In this introductory chapter, the context and motivation regarding this dissertation, as well as its key objectives are presented in sections 1.1, 1.2 and 1.3, respectively. Additionally, the structure of this report is exposed and its logical divisions are described is section 1.4.

1.1 Context

Several real-world problems and their objects can be instinctively represented by graph structures. These representations not only capture the main properties of a given domain but also the intricate topology of relationships in a network-oriented problem. Graph representations are often sparse and complex and to appropriately leverage their various topology features machine learning algorithms require underlying methods to efficiently generalize and produce adequate representations from these structures, considering the trade-off data completeness and computational efficiency.

In the case of sequential decision-making problems, the same is verified. Learning how to map good sequences of decisions in network-oriented domains can depend, in some cases, on accounting for the environment topological features [9, 59, 60, **?**]. As the main paradigm of machine learning that addresses sequential decision-making problems, RL algorithms need to be adapted to reflect these considerations, which establishes the foundations of Graph Reinforcement Learning (GRL). This compromises the main focus of this work, which will be applied in the application domain of *Smart Grid* Services.

In other regards, this report is written in the context of the course of Dissertation Preparation (PDIS) inserted in the Master's Degree in Informatics and Computing Engineering (MEIC) of the Faculty of Engineering of the University of Porto (FEUP). In addition, this project is accommodated in the Artificial Intelligence and Computer Science Laboratory (LIACC).

1.2 Motivation

Reflecting on the current global issues associated with the energetic crisis and climate change, there is an increasing need for sustainable and economic energetic systems to modernize the current power grids. This modernization is translated into the transition to the *smart grid*, a power grid equipped with intelligent control and monitoring systems to efficiently manage power distribution [9, 30], voltage regulation, system restoration [69], grid reliability [40] or other associated processes. Currently, renewable energy sources play a major role in reducing the reliance on fossil fuels [2], which changes the topology of energy distribution systems as consumers gain the capability to generate renewable power. Furthermore, investment in energy storage is becoming a priority in the energy sector [46], resulting in improvements on storage capacity and approximating the current solutions to the average consumer [32].

The exposed issues serve as the prime motivation for performing this work on the application domain of *smart grid* services, with the expectation that by proposing concrete improvements in GRL techniques a well-performing solution to the dynamic economic dispatch problem can be presented. Beyond this, we hope the proposed solution and its architecture is also adaptable and applicable to other smart grid problems, resulting in a significant contribution to these services.

Furthermore, the main reasons for addressing GRL algorithms lies on their novelty and complexity, the lack of well-documented literature regarding these approaches, and the need for systematic and comparative studies confronting the different proposed techniques and architectures.

1.3 Objectives

Considering this work's context and motivations, we define its main objectives:

1. Perform a review of literature regarding GRL approaches and Dynamic Economic Dispatch (DED) systems
2. Conduct a comparative and systematic empirical study of different GRL solutions of the DED problem
3. Propose a GRL model and concrete improvements facing the literature proposed models

The first goal addresses the analysis of existent techniques, as well as its limitations, by reviewing the relevant research on GRL approaches and DED systems. Secondly, we will focus on implementing the different observed approaches to solve the DED problem and perform a comparative study to analyse and confront the gathered results. Lastly, we hope the accomplishment of the first to goals to enable the proposition of a state-of-the-art GRL model and specific improvements to these techniques.

1.4 Report Structure

This report is organized as follows: (1) an introductory chapter; (2) a chapter explaining the relevant background concepts to perform this study; (3) the review of the literature regarding GRL techniques and DED systems; (**??**) the statement of the main problem and the presentation of the proposed solution and finally, (6) the main conclusions, reflections and expected contributions of this work.

Chapter 2

Background Knowledge

In this chapter, we will present the underlying concepts related to this dissertation. This knowledge consists in important context for the rest of this work by explaining the its background concepts.

This chapter is divided into section 2.1 addressing Artificial Neural Networks, section 2.2 regarding Reinforcement Learning problem and algorithms, 2.3 explaining Graph Representation Learning, 2.4 exposing the current GNN approaches and 2.5 addressing Smart Grid Services..

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are a class of machine learning algorithms based on the neural process of biological learning. The simplest form of an ANN is a Multilayer Perceptron (MLP), also called a feedforward network, whose main objective is to approximate to function f that models relationships between input data x and output data y of considerate complexity [7, 20]. It defines a mapping $y = f(x;\theta)$ and learns the best composition of parameters θ to approximate it to the unknown model. The MLP serves as a fundamental part of developing the other more complex types of neural networks [7].

2.1.1 Feedforward Neural Networks

The main building block of a MLP is the *Perceptron*, pictured in figure 2.1, a simple computational model initially designed as a binary classificator that mimics biological neurons' behaviour [7]. A neuron might have many inputs x and has a single output y. It contains a vector of *weights* $w = (w_1 ... w_m)$, each associated with a single input, and a special weight b called the *bias*. In this context, a perceptron defines a computational operation formulated as equation 2.1 portrays [7].

$$f(x) = \begin{cases} 1 & \text{if } b + \mathbf{w} \cdot \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Functions that compute $b + \mathbf{w} \cdot \mathbf{x} > 0$ are called *linear units* and are identified with Σ [7, 20]. An activation function g was introduced to enable the output of non-linear data. The default

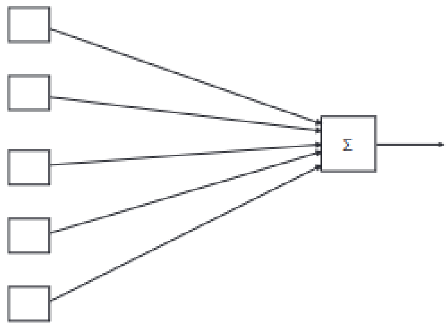

Figure 2.1: The Perceptron [7]

recommendation is the *Rectified Linear Unit* (*ReLU*), with the Logistic curve (sigmoid) also being very common [20].

Feedforward networks are composed of an input layer formed by the vector of input values, an arbitrary number of hidden layers and an output layer, which is the last layer of neurons [7]. The greater the amount of layers the higher the *depth* of the network [7, 20].

On its own, the model amounts only to a complex function. Still, with real-world correspondence between input values and associated outputs, a feedforward network can be trained to approximate the unknown function of the environment. In more concrete terms, this involves updating all of the different weight and bias values of each neuron to achieve an output as close as possible to the real or desired value or minimize the total loss, which indicates how distant the network model is to the real function to approximate [7, 20]. *Loss functions* are used to calculate this value.

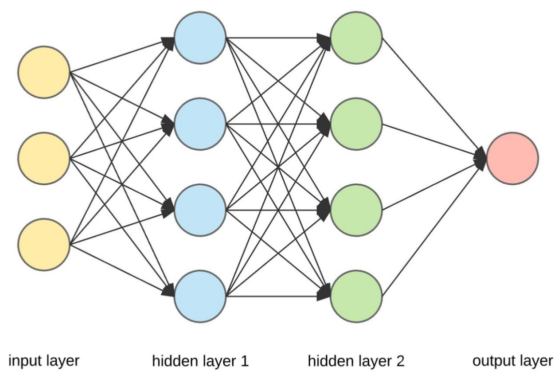

Figure 2.2: Architecture of a Feedforward Neural Network [1]

2.2 Reinforcement Learning

RL consists of a field and a class of machine learning algorithms that study how to learn to take good sequences of actions to achieve a goal associated with a maximizing received numerical reward [6]. The main objective is to maximize the received cumulative reward by trying between the available actions and discovering which ones yield the most reward [47]. This sequential decision-making process becomes more complex when a delayed reward is considered, given that an action with immediate reward may not always reflect the delayed consequences of that decision [47]. It's also the learner's job to consider this during the learning process. These concepts of *delayed reward* and *trial-and-error search* make up the most important characteristics of Reinforcement Learning [47]. The classic formalisation of this problem is the MDP through defining the agent-environment interaction process, explained in the following subsection 2.2.1.

A major challenge in this machine learning paradigm is the trade-off between *exploring* new unknown actions and *exploiting* the already known "good" actions [47]. To choose the sequence of actions that return the highest reward, the agent must choose actions it found effective in similar past situations or *exploit* what it learned from experience. Furthermore, given that the agent may not know the action-reward mappings initially, it has to *explore* possible actions that were not selected previously or may initially seem to yield a low reward to compute accurate reward estimates. The main problem is that neither exploitation nor exploration can be favoured exclusively without failing at the task [47]. Additionally, an agent's environment is uncertain, and changes in the environment's dynamics may also involve re-estimating action rewards.

In conclusion, RL techniques enable the implementation of sequential decision-making agents that seek to maximize a reward signal analogous to an explicit (complex) goal. The agents need to balance between actions that yield a reward on posterior time steps and actions that produce immediate rewards. In addition, these agents are also faced with the task of balancing the exploitation of information from past experiences and the exploration of new decision paths that could potentially return a higher reward down the road [47].

2.2.1 Markov Decision Process

Markov Decision Processs (MDPs) are a classical formalization of a sequential decision-making process, constituting the mathematical definition of the RL problem [47, 37]. Beyond estimating potential rewards for the available actions, the problem defined by MDPs involves learning which actions are optimal in specific situations, i.e. learning a mapping between states of the environment and actions [47].

The central component of MDPs is the agent, which acts as a decision-maker and learns from interactions with the environment it's inserted. In a continuous process, the agent takes actions that affect the environment's state, which in turn presents new situations [47]. The environment also responds with the reward signals which the agent aims to maximize over time through its decision process.

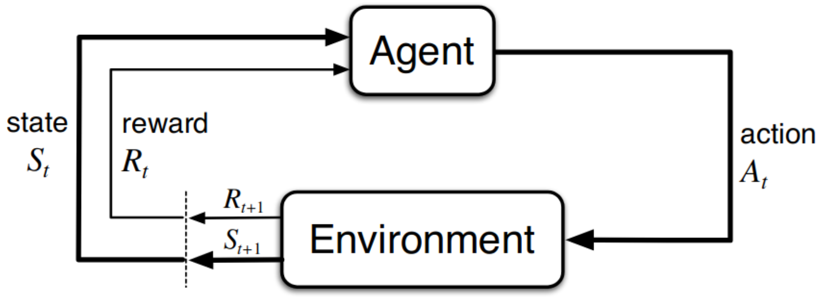

Figure 2.3: Agent-environment interaction in a MDP [47]

Formally, the agent-environment interactions, as figure 2.3 entails, occur in a sequence of discrete time steps t, where at each step, the agent receives a representation of the state of the environment $S_t \in \mathcal{S}$ which is used to select an appropriate action $A_t \in \mathcal{A}(s)$, where $\mathcal{S}$ is the set of possible states called the *state space* and $\mathcal{A}(s)$ is the set of available actions for state s [47, 37]. In the next step, the agent receives, as a consequence of its decision, a numerical reward signal $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and is faced with a new state S_{t+1} [47]. Ultimately, the MDP agent follows a logical sequence that occurs as equation 2.2 states. The collection of a state S_t, action taken A_{t+1}, reward R_{t+1} received and next state S_{t+1} constitutes an *experience tuple* [37].

$$S_0, A_0, R_1, S_1, A_2, R_2, S_2, A_2, R_3, \ldots \tag{2.2}$$

In addition, when the set of possible actions, states and rewards ($\mathcal{A}$, $\mathcal{S}$ and $\mathcal{R}$) are finite, the MDP is said to be *finite* [47]. This results in S_t and R_t having well-defined discrete probability distributions in function of the preceding state and chosen action [47]. Therefore, the probability of receiving a particular reward and state given the previous state and selected action, which characterizes a finite MPD's dynamics, may be characterized by function p defined in equation 2.3

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\} \tag{2.3}$$

For all $s, s' \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$, where $\doteq$ denotes a mathematical formal defintion. This encompasses the assumption that the probability of each possible state, S_t, and reward, R_t, pair is only dependent on the preceding state, S_{t-1}, and action taken, A_{t-1} [47]. Instead of observing this as a restriction on the decision process, it's more convenient to view it as a constraint on the state variable, considering that it must contain all the necessary information from experience to make a valuable decision in the immediate step. If this condition is satisfied, the state is declared to have the *markov property* [47].

From function p in equation 2.3, the state-transtion probabilities, also called the *transition function*, can be computed as described by equation 2.4 [47, 37].

$$p(s'|s, a) \doteq Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{2.4}$$

In addition, the expected rewards can be calculated for state-action pairs (equation 2.5) or state-action-next-action triples (equation 2.6) [47, 37].

$$r(s,a) \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a) \tag{2.5}$$

$$r(s,a,s') \doteq \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r | s, a)}{p(s' | s, a)} \tag{2.6}$$

2.2.2 Rewards and Returns

As stated in the previous subsections, the main goal of a RL agent defined by the numeric reward signal, $R_t \in \mathbb{R}$, it receives from the environment [47]. In this context, the agent's objective is to maximize the total reward it receives, considering not only immediate but also the cumulative reward over time. In the ubiquitous work of [47], the *reward hypothesis* is stated as follows:

> That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward). [47]

This also entails that the process of reward maximization from the agent has to be closely tied to it achieving its defined goals in a practical sense. Otherwise, the agent will fail at fulfilling the desired objectives [47].

Formally, the goal of an RL agent can be defined by the maximization of the cumulative reward received of time called the *expected return*, Return_t [47].

$$\text{Return}_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \tag{2.7}$$

T describes the final time step. This definition can be applied in domains with a natural notion of a terminal state or final time step. In these cases, the agent-environment interaction process can be broken into logically independent subsequences called *episodes* [47]. Each episode ends in a special state, called the terminal state, restarting a new sequence of states and actions completely independent from the previous episode [47]. In this context, episodes can be considered to end in the same terminal state, with different accumulated rewards for the different outcomes [47].

In contrast, there are situations where the decision-making process doesn't divide itself into logically identifiable episodes but goes on indefinitely. In this case, $T = \infty$ and according to equation 2.7, the expected return the agent aims to maximize would be infinite [47]. In this manner, another concept is added in the expected return definition called the *discount rate*, γ where $0 \leq \gamma \leq 1$, representing how strongly the agent should account for future rewards in the expected return calculations, as equation 2.8 [47].

$$\text{Return}_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.8}$$

From this equation, we can compute the expected discounted return on a given time step t in the function of the immediate reward signal received and the expected return for the next time step $t+1$, which eases the job of calculating expected returns for reward sequences [47]. This is entailed by equation 2.9.

$$\text{Return}_t = R_{t+1} + \gamma G_{t+1} \tag{2.9}$$

In this manner, a MDP can be defined by a tuple with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function p, a reward function r and a discount factor γ, as equation 2.10 portrays [6].

$$M = (\mathcal{S}, \mathcal{A}, p, r, \gamma) \tag{2.10}$$

2.2.3 Policies and Value Functions

RL techniques typically involve the estimation of what is understood as *value functions*, functions that estimate the expected return based on the current state value or state-action pair. This characterizes how good is for an agent to be in a specific state or to take an action in a specific state, respectively, using the expected return to characterize the overall *goodness* of these scenarios [47, 37]. These functions are tied to a specific way of determining the action in a given state. Formally, this is defined as a *policy* π, that defines the probability $\pi(a|s)$ of taking action a in state s [47]. In this context, the *state-value function*, $v_\pi(s)$ and *action-value functions* $q_\pi(s,a)$ for policy π can be defined by equations 2.11 and 2.14, respectively [47].

$$v_\pi(s) \doteq \mathbb{E}_\pi[\text{Return}_t | S_t = s], \forall s \in \mathcal{S} \tag{2.11}$$

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[\text{Return}_t | S_t = s, A_t = a] \tag{2.12}$$

The utility of such functions rely on the possibility of estimating them with regard to past experience of the agent [47]. A fundamental property of value functions is that it can, as was the case with the expected return (equation 2.9), satisfy recursive relationships with the next immediate value as equation 2.13 entails [47]. This equation is called the *Bellman equation for* v_π, which characterizes the relationship between the value of current and subsequent states.

$$v_\pi \doteq \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{2.13}$$

$$q_\pi(s,a) \doteq \sum \mathbb{E}_\pi[\text{Return}_t | S_t = s, A_t = a] \tag{2.14}$$

2.2.4 Types of RL

Regarding RL algorithms, they can be divided into model-free and model-based techniques [39]. These categories are distinguished by whether an agent uses a provided or learned *model* of the set of transition and reward functions, another optional element of RL techniques [37, 39]. In the positive case, the method is said to be model-based, otherwise, it's model-free. Having an accurate model of the environment allows the RL agent to focus on planning ahead by calculating

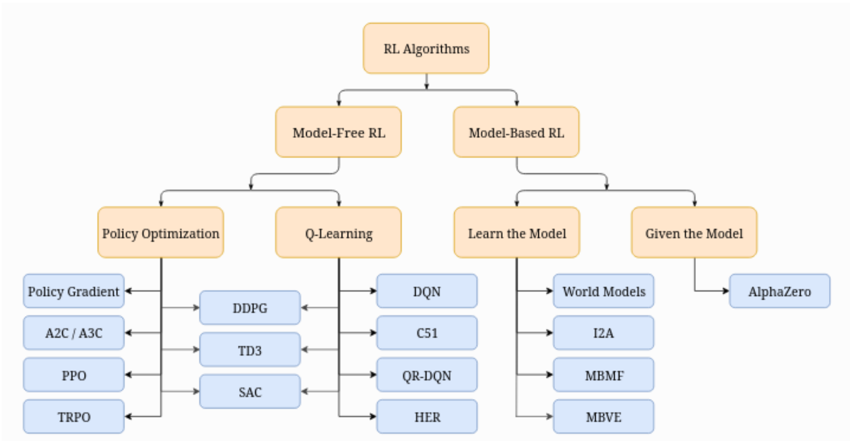

Figure 2.4: Taxonomy of algorithms in modern RL [39]

future scenarios and creating policies based on the results of the planning process. An example of a famous system of this kind is AlphaZero [45]. However, in most cases, agents can't access a ground-truth model of the environment, leaving only the scenario where an agent learns a model purely from experience. This creates several challenges, the most prominent of which relies on the fact that the model, in most times, doesn't fully capture the environment's transition dynamics, equipping it with bias in relation to the actual dynamics. With this, learning how to generalise the model to real-world environments so that the bias is not over-exploited becomes a very complex task [39]. Model-free algorithms can be also further divided into Q-learning and Policy Aproximmation techniques.

Furthermore, algorithms can also be subdivided into on-policy and off-policy methods. [37] On-policy algorithms evaluate and improve a single policy used to determine the agent's behaviour [37]. The methods under the policy optimization category such as A2C and A3C [35] or the Proximal Policy Optimization (PPO) [44] almost always fall into this label. In contrast, off-policy algorithms learn how to improve a different target policy based on the results that arise from the policy used to determine the system behaviour initially [37]. Such approaches include Q-Learning algorithms such as Deep Q-Networks (DQNs) [36, 39]. Deep Deterministic Policy Gradient (DDPG) [33] combines policy optimization with q-learning, consisting of an off-policy method that learns both a q-function and a policy. DDPG constitutes the adaption of q-learning methods to continuous action spaces [39]. Another example of an off-policy algorithm is the Soft Actor-Critic (SAC) [21] method, which bridges stochastic policy optimization with the DDPG approach and has entropy regularization as one of its central features, which translates into training a policy that maximizes the expected return and entropy, a measure of randomness in the policy [39].

Lastly, with the advent of deep learning becoming one of the most ubiquitous techniques in machine learning, RL algorithms have evolved beyond the traditional tabular methods [37]. Traditional RL has evolved to Deep Reinforcement Learning (DRL), which studies how to use deep neural networks in RL problems to leverage their generalization abilities for solving more complex problems.

2.3 Graph Representation Learning

Several objects and problems can be naturally expressed in the real world using graphs, such as social networks, power grids, transportation networks, recommendation systems or drug discovery. The usefulness of such representations is tied to how they instinctively represent the complex relationships between objects. However, graph data is often very sparse and complex, and their sophisticated structure is difficult to deal with [34, 68].

Furthermore, the performance of machine learning models strongly relies not only on their design but also on good representations of the underlying information [34]. Ineffective representations, on the one hand, can lack important graph features and, on the other, can carry vast amounts of redundant information, affecting the algorithms' performance in leveraging the data for different analytical tasks [34, 55].

In this context, **Graph Representation Learning** studies how to learn the underlying features of graphs to extract a minimal but sufficient representation of the graph attributes and structure [22, 68, 13]. Currently, the improvements in deep learning allow representation learning techniques consisting of the composition of multiple non-linear transformations that yield more abstract and, ultimately, more useful representations of graph data [13].

2.4 Graph Neural Networks

In the present, deep learning and ANN have become one of the most prominent approaches in Artificial Intelligence research [13]. Approaches such as recurrent neural networks and convolutional networks have achieved remarkable results on Euclidean data, such as images or sequence data, such as text and signals [56]. Furthermore, techniques regarding deep learning applied to graphs have also experienced rising popularity among the research community, more specifically **GNNs** that became the most successful learning models for graph-related tasks across many application domains [13, 56].

The main objective of GNNs is to update node representations with representations from their neighbourhood iteratively [48]. Starting at the first representation $H^0 = X$, each layer encompasses two important functions:

- **Aggregate**, in each node, the information from their neighbours
- **Combine** the aggregated information with the current node representations

The general framework of GNNs, outlined in [48], can be defined mathematically as follows:

Initialization: $H^0 = X$
For $k = 1, 2, \ldots, K$

$$a_v^k = \text{AGGREGATE}^k\{H_u^{k-1} : u \in N(v)\}$$
$$H_v^k = \text{COMBINE}^k\{H_u^{k-1}, a_v^k\}$$

Where $N(v)$ is the set of neighbours for the v-th node. The node representations H^K in the last layer can be treated as the final representations, which sequentially can be used for other downstream tasks [48].

2.4.1 Graph Convolutional Network

A **GCN** [26] is a popular architecture of GNNs praised by its simplicity and effectiveness in a variety of tasks [34, 48]. In this model, the node representations in each layer are updated according to the following convolutional operation:

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^kW^k) \tag{2.15}$$

$= A + I$ - Adjacency Matrix with self-connections
$I \in \mathbb{R}^{N\times N}$ - Identity Matrix
$\tilde{D}$ - Diagonal Matrix, with $\tilde{D}_{ii} = \sum_j {}_{ij}$
σ - Activation Function
$W^k \in \mathbb{R}^{F\times F'}$ - Laywise linear transformation matrix (F and F' are the dimensions of node representations in the k-th and $(k+1)$ layer, respectively)

$W^k \in \mathbb{R}^{F\times F'}$ is a layerwise linear transformation matrix that is trained during optimization [48]. The previous equation 2.15 can be dissected further to understand the *AGGREGATE* and *COMBINE* function definitions in a GCN [48]. For a node i, the representation updating equation can be reformulated as:

$$H_i^k = \sigma\Big(\sum_{j\in\{N(i)\cup i\}} \frac{\tilde{A}_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}} H_j^{k-1}W^k\Big) \tag{2.16}$$

$$H_i^k = \sigma\Big(\sum_{j\in N(i)} \frac{A_{ij}}{\sqrt{\tilde{D}_{ii}\tilde{D}_{jj}}} H_j^{k-1}W^k) + \frac{1}{\tilde{D}_i}H^{k-1}W^k\Big) \tag{2.17}$$

In the second equation, the *AGGREGATE* function can be observed as the weighted average of the neighbour node representations [48]. The weight of neighbour j is defined by the weight of the edge (i, j), more concretely, A_{ij} normalized by the degrees of the two nodes [48]. The *COMBINE* function consists of the summation of the aggregated information and the node representation itself, where the representation is normalized by its own degree [48].

Spectral Graph Convolutions

Regarding the connection between GCNs an spectral filters defined on graphs, spectral convolutions can be defined as the multiplication of a node-wise signal $x \in \mathbb{R}^N$ with a convolutional filter $g_\theta = diag(\theta)$ in the *Fourier domain* [34, 48], formally:

$$g_\theta \star \mathrm{x} = U_{g_\theta} U^T \mathrm{x} \tag{2.18}$$

$\theta \in \mathbb{R}^N$ - Filter parameter
U - Matrix of eigenvectors of the normalized graph Laplacian Matrix $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

The eigendecomposition of the Laplacian matrix can also be defined by $L = U \Lambda U^T$ with Λ serving as the diagonal matrix of eigenvalues and $U^T \mathrm{x}$ is the graph Fourier transform of the input signal x [48]. In a practical context, g_θ is the function of eigenvalues of the normalized graph Laplacian matrix L, that is $g^\theta(\Lambda)$ [34, 48]. Computing this is a problem of quadratic complexity to the number of nodes N, something that can be circumvented by approximating $g_\theta(\Lambda)$ with a truncated expansion of Chebyshev polynomials $T_k(x)$ up to K-th order [34, 48]:

$$g_{\theta'}(\Lambda) = \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) \tag{2.19}$$

$\tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - \mathrm{I}$
$\lambda_{\max}$ - Largest eigenvalue of L
$\theta' \in \mathbb{R}^N$ - Vector of Chebyshev coefficients
$T_k(x)$ - Chebyshev polynomials
$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$

By combining this with the previous equation, the first can be reformulated as:

$$g_\theta \star \mathrm{x} = \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) \mathrm{x} \tag{2.20}$$

$\tilde{L} = \frac{2}{\lambda_{\max}} L - I$

From this equation, it can be observed that each node depends only on the information inside the K-th order neighbourhood and with this reformulation, the computation of the equation is reduced to $O(|\xi|)$, linear to the number of edges ξ in the original graph G.

To build a neural network with graph convolutions, it's sufficient to stack multiple layers defined according to the previous equation, each followed by a nonlinear transformation. However, the authors of GCN [26] proposed limiting the convolution number to $K = 1$ at each layer instead of limiting it to the explicit parametrization by the Chebyshev polynomials. This way, each level only defines a linear function over the Laplacian Matrix L, maintaining the possibility of handling complex convolution filter functions on graphs by stacking multiple layers [26, 48]. This means the model can alleviate the overfitting of local neighbourhood structures for graphs whose node degree distribution has a high variance [26, 48].

At each layer, it can further considered $\lambda_{\max} \approx 2$, which the neural network parameters could accommodate during training [48]. With these simplifications, the equation is transformed into:

$$g_{\theta'} \star \mathrm{x} \approx \theta'_0 \mathrm{x} + \theta'_1 \mathrm{x}(L - I_N)\mathrm{x} = \theta'_0 \mathrm{x} - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2.21}$$

θ_0' and θ_1' - Free parameters that can be shared over the entire graph

The number of parameters can, in practice, be further reduced, minimising overfitting and minimising the number of operations per layer as well [48] as equation 2.22 entails.

$$g_\theta \star \mathrm{x} \approx \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})\mathrm{x} \tag{2.22}$$

$\theta = \theta_0' = -\theta_1'$

One potential problem is the $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ matrix whose eigenvalues fall in the $[0,2]$ interval. In a deep GCN, the repeated utilization of the above function often leads to an exploding or vanishing gradient, translating into numerical instabilities [34, 48]. In this context, the matrix can be further renormalized by converting $I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ into $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ [34, 48]. In this case, only the scenario where there is one feature channel and one filter is considered which can be then generalized to an input signal with C channels $X \in \mathbb{R}^{N \times C}$ and F filters (or hidden units) [34, 48]:

$$H = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X W \tag{2.23}$$

$W \in \mathbb{R}^{C \times F}$ - Matrix of filter parameters

H - Convolved Signal Matrix

2.4.2 Graph Attention Network

GAT [51] is another type of GNNs that focuses on leveraging an attention mechanism to learn the importance of a node's neighbours. In contrast, the GCN uses edge weight as importance, which may not always represent the true strength between two nodes [48, 51].

The Graph Attention Layer defines the process of transferring the hidden node representations at layer $k-1$ to the next node presentations at k. To ensure that sufficient expressive power is attained to allow the transformation of the lower-level node representations to higher-level ones, a linear transformation $W \in \mathbb{R}^{F \times F'}$ is applied to every node, followed by the self-attention mechanism, which measures the attention coefficients for any pair of nodes through a shared attentional mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ [48, 51]. In this context, relationship strength e_{ij} between two nodes i and j can be calculated by:

$$e_{ij} = a(WH_i^{k-1}, WH_j^{k-1}) \tag{2.24}$$

$H_i^{k-1} \in \mathbb{R}^{N \times F'}$ - Column-wise vector representation of node i at layer $k-1$ (N is the number of nodes and F the number of features per node)

$W \in \mathbb{R}^{F \times F'}$ - Shared linear transformation

$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ - Attentional Mechanism

e_{ij} - Relationship Strength between nodes i and j

Theoretically, each node can attend to every other node on the graph, although it would ignore the graph's topological information in the process. A more reasonable solution is to only attend nodes in the neighbourhood [51, 48]. In practice, only first-order node neighbours are used, including the node itself, and to make the attention coefficients comparable across the various nodes, they are normalized with a *softmax* function:

$$\alpha_{ij} = \text{softmax}_j(\{e_{ij}\}) = \frac{exp(e_{ij})}{\sum_{l \in N(i)} exp(e_{il})}$$

Fundamentally, α_{ij} defines a multinomial distribution over the neighbours of node i, which can also be interpreted as a transition probability from node i to each node in its neighbourhood [48]. In the original work [51], the attention mechanism is defined as a single-layer Feedforward Neural Network that includes a linear transformation with weigh vector $W_2 \in \mathbb{R}^{1\times 2F'}$ and a LeakyReLU nonlinear activation function with a negative input slope $\alpha = 0.2$ [48, 51]. More formally, the attention coefficients are calculated as follows:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(W_2[WH_i^{k-1}||WH_j^{k-1}]))}{\sum_{l \in N(i)} \exp(\text{LeakyReLU}(W_2[WH_i^{k-1}||WH_l^{k-1}]))} \tag{2.25}$$

$||$ - Vector concatenation operation

The novel node representation is a linear composition of the neighbouring representations with weights determined by the attention coefficients [51, 48], formally:

$$H_i^k = \sigma(\sum_{j \in N(i)} \alpha_{ij} W H_j^{k-1}) \tag{2.26}$$

Multi-head Attention

Multi-head attention can be used instead of self-attention, determining a different similarity function over the nodes. An independent node representation can be obtained for each attention head according to the equation bellow [51, 48]. The final representation is a concatenation of the node representations learned by different heads, formally:

$$H_i^k = \Big\|_{t=1}^{T} \sigma(\sum_{j \in N(i)} \alpha_{ij}^t W^t H_j^{k-1})$$

T - Number of attention heads

α_{ij}^t - attention coefficient computed from the t-th attention head

W^t - Linear transformation matrix of the t-th attention head

Lastly, the author also mentions that other pooling techniques can be used in the final layer for combining the node representations from different heads, for example, the average node representations from different attention heads [51, 48].

$$H_i^k = \sigma(\frac{1}{T}\sum_{t=1}^{T}\sum_{j\in N(i)}\alpha_{ij}^t W^t H_j^{k-1}) \tag{2.27}$$

2.5 Smart Grid Services

Given the global ecological emergency and the increasing energetic crisis, there is a necessity for advancements in energy distribution and transmission systems now more than ever. To fulfil the need for energy sustainability, traditional centralized distribution grids must be adapted to, on the one hand, accommodate the rise of distributed renewable energy sources in corporate and domestic consumers and, on the other, to make more efficient and reliable distribution of energy resources [15, 53].

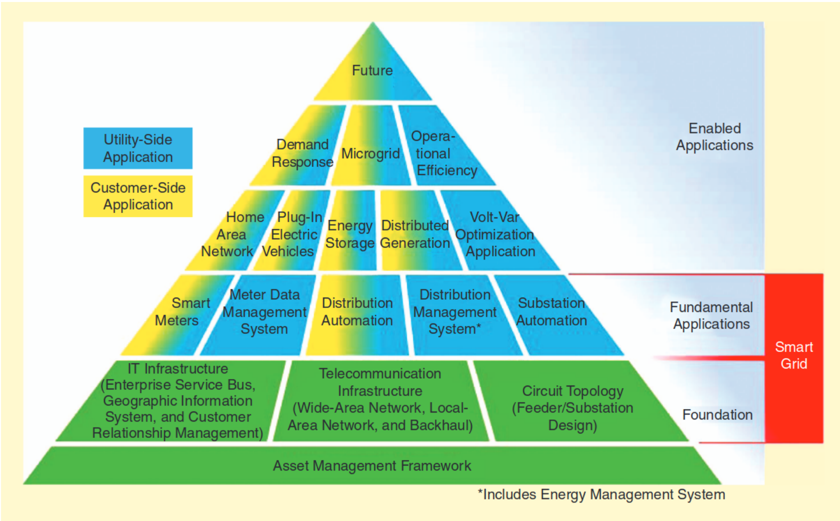

Figure 2.5: Smart Grid Capabilities Pyramid [15]

The *Smart Grid* or the *Smart Power Grid* conceptualizes this modernization of the electricity network by leveraging the technological advancements in information technology and communication science to create intelligent systems that manage and monitor the distributed generation of energy [5, 15]. Figure 2.5 describes the smart grid pyramid, which has asset management at its base. On this foundation, the foundation of the smart grid is laid out by the circuit topology, IT systems and telecommunications infrastructure, the basic ingredients for the emergence of fundamental applications such as smart meters and distribution automation [15]. In turn, these serve as building blocks for creating more intelligent systems that leverage upper-layer applications, enabling the true smart grid capabilities [15].

Chapter 3

Literature Review

In this chapter, the literature review regarding GRL approaches and DED systems is documented. This unit is divided into three sections with the first exposing the research methodology and the others reviewing GRL approaches and the DED Systems, the second is further subdivided into plain GCN, attention-based and other approaches.

3.1 Research Methodology

This literature review focuses on analysing the existent literature around the main objective of this dissertation, which is improving GRL techniques in the context of this work's application domain, smart grid services. In this manner, the main research questions are presented as:

- RQ1 - *How can RL algorithms be adapted to effectively solve sequential decision-making problems on graph-based environments?*
  - RQ1.1 - *What are the existent GRL approaches?*
  - RQ1.2 - *What are their limitations?*
- RQ2 - *How can GRL algorithms improve smart grid services?*
  - RQ2.1 - *How can GRL algorithms improve Dynamic Economic Dispatch (DED) in power distribution grids?*

These interrogations aggregate the relevant topics we aim to address in this review. The main requirement on the analyzed literature was to only consider research from the past five years, with the exception of seminal works. We also exclusively considered literature published in scientific conferences and top-tier journals.

In this manner, the initial exploratory research was conducted, primarily using *Scopus* and *Web of Science* for searching the published literature and alternatively using *Google Scholar* for finding cross-references when necessary. The research questions were translated into fundamental search queries where the research process was based.

- `"Graph Reinforcement Learning" OR "Reinforcement Learning on Graphs"`
- `"Reinforcement Learning" AND "Power" AND "Dispatch"`
- `"Graph Reinforcement Learning" OR "Reinforcement Learning on Graphs" AND "Power" AND "Dispatch"`

The gathered literature was screened and the relevant works were thoroughly reviewed, the following sections present the main findings.

3.2 Graph Reinforcement Learning Approaches

GRL or Reinforcement Learning on Graphs is a relatively new area in the broader field of machine learning. GRL techniques have shown significant progress in solving problems with underlying graph-based representations such as power grid management [30, 10], smart transport [59, 3] or task offloading [19, 29]. In this work, the main focus lies on studying the development of GRL techniques and subsequent application to smart grid services such as dynamic economic energy dispatch systems [9, 57], residential electricity behaviour identification and energy management [10], or Volt-VAR regulation [25].

Research on this topic has significantly increased in the last few years with the improvements of DRL techniques and the developments in GNNs in the mid-2010s [26, 51, 31, 18]. GNNs became the state-of-the-art for solving numerous data mining tasks involving graph-structured data, excelling at classification, link prediction and representation learning [61, 38]. This advancement brought more sophisticated RL applications on graphs and the surge of a new field studying how to combine the improvements of graph mining and reinforcement learning techniques [52, 38].

3.2.1 Plain GCN-Based GRL

A common approach in Graph Reinforcement Learning model implementation is the use of graph convolutions with the GCNs architecture for leveraging graph-based structures to extract and aggregate the essential features of data in hand and improve the performance of RL agents in those environments. The techniques listed in this subsection constitute approaches that integrate a GCN with RL algorithms. The gathered literature can be observed in table 3.1.

[30] implements a GRL system to improve the decision quality of economic dispatch under high penetration of distributed energy generations. To accomplish this, a SAC system is employed with the main objective of finding the optimal action policy for minimizing generation cost with the appropriate reliability concerns. This problem is represented by an undirected graph with nodes describing the power grid elements with their respective attributes and edges describing the underlying energy connections between those units. To extract the structural features of the graph, this work implements a full connected layer to perform feature transformation with a two-layer GCN followed by three full connected layers for the non-linear mapping of state-to-action policy in both actor and critic modules. [9] develops a similar approach, with both concluding that

it significantly reduces learning time for achieving better training results in comparison to plain SAC and showing significant improvement on economy and flexibility of the system on more complex and sparse state graphs. The use of GCNs enables the system to adapt to changes in the state space by leveraging the network's generalization ability.

In [60] a three-layer GCN is used to extract node feature and graph topology information and is integrated into a Rainbow-based [24] DRL algorithm for electric vehicle charging guidance. In this article, the testing results show promising performance in reducing operation costs for electric vehicle users, portraying a model with good generalization ability in untrained scenarios. This work and [9] further tested their proposed model's performance when inducing topology changes, yield promising results in the adaptability of the two GRL algorithms in scenarios where the environment suffers dynamic changes.

Another interesting implementation of this approach is [8], which studies and compares different solutions for optimizing autonomous exploration under uncertain environments. It analyses combinations of a single agent Deep Q-Network (DQN) and Advantageous Actor-Critic (A2C) with Graph Convolutional Networks, Gated Graph Recurrent Networks and Graph U-Nets. The algorithms are executed in test cases of various sizes and the paper reports that the GCN-DQN was the model that achieved the highest reward during policy training, followed by the GGNN-A2C model, although in the end, it concludes that the second showed improved scalability in relation to the first model. Other similar approaches include [10] for residential electricity behaviour identification and energy management with a behaviour correlation graph and [62] for line flow control in a power grid simulation.

The reviewed literature fails to consider methods for analysing the scalability of proposed models in relatively larger scenarios, except [8], with some proposing this as relevant future work [9, 60]. Beyond that, only [60], [10] [9] defined methods for evaluating the performance of the algorithms under topology variations. This shows a gap for more complete studies in plain GCN-based approaches that also focus on studying the scalability of GRL to large scenarios and the adaptability of the models to dynamic topology variations. In most approaches, namely in [60, 9, 30, 62], the presented results portray GRL techniques as significantly more effective than plain DRL models without a GCN already suggesting that these techniques are a potential solution for solving sequential decision-making problems with graph-based environments.

Table 3.1: GCN-Based GRL Literature

Reference	DRL Algorithm	GNN Algorithm	Application Domain
[66]	MDP	GCN	Interpret GNNs at model-level
[54]	DDPG	GCN	Automatic transistor sizing
[64]	A3C	GCN	Automatic Virtual Network Embeddings
[49]	Deep Q-Learning	GCN	Task Offloading in Edge Computing
[60]	Rainbow	GCN	Electrical Vehicle Charging Guidance
[30], [9]	SAC	GCN	Dynamic economic energy dispatch
[28]	SAC	GCN	Multi-access Edge Computing
[62]	DQN	GCN	Line flow control
[8]	DQN	GCN	Autonomous Exploration under uncertainty
[10]	DQN	GCN	Residential electricity behavior identification and energy management

3.2.2 Attention-based GRL

Another effective approach in extracting relevant topology and graph features relies on using attention mechanisms to weigh different nodes' contributions dynamically. While this encompasses techniques that use the GAT architecture, which is a GNN design with the attention mechanism at its core, various scholars propose GCN approaches integrated with attention mechanisms such as [69] and [14]. In the top part of table 3.2 the single-agent reviewed attention-based approaches can be observed, while at the bottom some relevant multi-agent approaches are listed.

[57] proposes a DDPG-based algorithm improved with a GAT block with three graph Attention Layers for extracting and learning the topology information for achieve real-time optimal scheduling for Active Distribution Networks (ADNs). This paper compares the obtained test results against a GCN-DDPG model and shows increased performance over the GCN method in reducing cost and power loss. Beyond this, the work demonstrates that the GAT's attention mechanism enables the algorithm to focus on more important nodes and improve the signal-to-noise ratio compared to its GCN counterpart. [11] and propose a multi-agent approach to the same domain but more focused on voltage regulation with a multi-agent SAC instead of a single-agent DDPG algorithm.

In [59], another model for the electric vehicle charging guidance is proposed, consisting of a bi-level approach of a Rainbow-based algorithm with a GAT block. The upper level focuses on the decision-making process regarding charging, while the lower level handles routing. The proposed model proved to be more effective than a shortest distance path-based [58] and a DRL-based [43] approach. [63] develops a similar approach with a Double-prioritized DQN for the same application domain. In [69] and [14], the sequential distribution system restoration problem is addressed with a multi-agent RL algorithm equipped with a GCN with an attention mechanism. In the first case, multi-head attention is used as the convolution kernel for the GCN with a DQN algorithm. In the second, self-attention is used for improving the centralized training of the used multi-agent actor-critic algorithm, more concretely, by embedding it in the critic networks. At the same time, the GCN is integrated into the actor networks for extracting the graph features. Both solutions proved more efficient than traditional RL techniques, with the first highlighting its solution generalization ability and the second showing increased scalability facing the non-GRL techniques. In general, the literature regarding attention-based approaches is a lot sparser and scarcer than GCN-based approaches. Only three relevant single-agent works were found [57, 59, 63] which might either suggest that scholars have a slight preference for multi-agent systems when implementing GRL with attention mechanisms or that these approaches in single-agent systems still require further research. Nevertheless, some of the works showed models with good adaptability to topology variations [57, 4, 11] and scalability to large scenarios [69, 57, 4, 11]. Notably, some works suggest directly embedding the GNN in th RL framework to boost the methodology computation performance and robustness [57, 57].

Table 3.2: Attention-Based GRL Literature

Reference	**DRL Algorithm**	**GNN Algorithm**	**Application Domain**
[57]	DDPG	GAT	Optimal Scheduling for ADNs
[59]	Rainbow	GAT	Electric Vehicle Charging Guidance
[63]	DQN	GAT	Electric Vehicle Charging Guidance
[69]	DQN	GCN	Multi-agent Sequential Distribution System Restoration
[4]	DQN	GCN	Active Power Rolling Dispatch
[14]	AC	GCN	Multi-agent Service Restoration
[11]	SAC	GAT	Multi-agent Voltage Regulation

3.2.3 Other Approaches

This subsection includes other relevant and promising GRL approaches that combine of other GNNs architectures with RL algorithms . In [40], a GraphSAGE network [**?**] is used with a

Deep Dueling Double Q-Network (D3QN) for emergency control of Undervoltage load shedding for power systems with various topologies. The author presents promising results for the GraphSAGE-D3QN model compared to a GCN-D3QN, achieving higher cumulative reward and faster voltage recovery speed, although it required longer decision times. The proposed model performed excellently in the application domain and successfully generalized the learned knowledge to new topology variation scenarios. Another approach that showed good performance with the GraphSAGE architecture was [?] in the context of the dynamic economic dispatch problem.

[67] focused on solving the Job shop scheduling problem through a priority dispatching rule with a Graph Isomorphism Network (GIN) [61] and an actor-critic PPO algorithm where the GIN is shared between actor and critic networks. The method showed superior performance against other traditional manually designed priority dispatching rule baselines, outperforming them by a large margin.

Table 3.3: Other GRL Approaches

Reference	**DRL Algorithm**	**GNN Algorithm**	**Application Domain**
[40]	DQN	GraphSAGE	Undervoltage Load Shedding
[?]	PPO	GraphSAGE	Dynamic Economic Dispatch
[67]	PPO	GIN	Dispatch for Job Shop Scheduling

3.3 Dynamic Economic Dispatch Systems

RL algorithms are already regarded as a well established potential solution for solving economic dispath problems [41]. In table 3.4 the relevant RL approaches to the problem, including GRL techniques, can be observed. In a general level, the reviewed literature regarding DED solutions take into account a wide range of considerations and constraints. The recent works show an almost ubiquitous study of Energy Storage Systems (ESSs) management and Renewable Energy Source (RES) curtailment, given the current global energetic issues and the relevance of these technologies for solving them.

Some systems [23, 9, ?] take further steps in ensuring grid stability by also considering voltage deviations while a notable paper considers battery degradation when calculating the cost of dispatch [?]. In general, GRL algorithms show superior performance in relation with plain DRL approaches [9, 30, ?] as also concluded in section 3.2.1, with studies successfully ensuring adaptability of the GRL models to variations in the scenario's topology [9, ?]. However, only [4] conducts tests in test cases of different sizes, which shows a gap for studies addressing the scalability limitations of the developed models.

Table 3.4: Dynamic Economic Dispatch RL Systems

Reference	Approach	Application
[65]	DDPG with Prioritized Experience Replay Mechanism and L2 Regularization	Integrated Energy Systems, Utility (Selling, Purchasing and Gas) and ESSs
[27]	DDPG	Thermal Power, Renewable Energy Sources and ESSs
[23]	SAC with Imitation learning	Thermal Power, Renewable Energy Sources and Voltage deviation
[9]	GCN-SAC with Replay Buffer	Thermal Power, Renewable Energy Sources, ESSs and Voltage Deviation
[30]	GCN-SAC	Utility (Time-of-Use), Thermal Power, Renewable Energy Sources and ESSs
[?]	GraphSAGE-PPO	Thermal Power, Renewable Energy Sources, ESSs and Voltage Deviation
[?]	Multi-Agent RL with Function Approximation and Diffusion Mechanism	Utility (Time-of-Use), Thermal Power (Diesel) and ESSs (Considering degradation)
[4]	Multi-Agent GAT-DQN	Thermal Power, Renewable Energy Sources and ESSs

3.4 Conclusions

In this chapter, we reviewed relevant literature for this dissertation's main research topic, GRL algorithms. GRL is very promising field, where several different applications and techniques were already studied. GNNs architectures such as GCN have been extensively applied with DRL algorithms for enabling feature extracting from graph-based state representations [9, 8]. Architectures such as the GraphSAGE and other attention-based have also been successfully applied with very promising results [40, 57] in comparison with GCNs. However, less research regarding their integration with DRL algorithms was discovered. This suggests that a possible improvement and research direction in the development of GRL techniques might be connected with exploring the use of different GNNs architectures and using the rising attention-based techniques.

- RQ1 - **How can RL algorithms be adapted to effectively solve sequential decision-making problems on graph-based environments?**
  - RQ1.1 - **What are the existent GRL approaches?** Existent approaches ubiquitously use DRLs with GNNs for efficiently extract graph features. They can be divided into plain GCN approaches, Attention-based approaches and others. GCNs compromise

a popular method while attention-based approaches showed to be less researched but more promising [57, 69]. Tables 3.1, 3.2 and 3.3 depict the reviewed GCN, attention-based and other implementations, respectively.

 - RQ1.2 - **What are their limitations?** Research already depicts GRL techniques as better performing in relation to plain RL algorithms in graph-based environments, which portrays GRL is a potential solution to this problem [9, 30, **?**, 59, 60, 62]. The reviewed literature proved to be quite scarce and sparse, probably due to the novelty of the field. After a thorough review, we failed to find works that listed specific limitations of GRL algorithms. This suggests the existence of a research gap for works with a thorough and well-documented scientific process as well as comparative and systematic studies between the different approaches, highlighting models performance in large scenarios and under topology variation. Furthermore, some papers suggested the directly embedding of the GNN into the RL framework [57, 59].

- RQ2 - **How can GRL algorithms improve smart grid services?**
  - RQ2.1 - **How can GRL algorithms improve Dynamic Economic Dispatch (DED) in power distribution grids?** RL algorithms are already regarded as a well established potential solution for solving economic dispath problems [41]. We were able to find evidence of GRL being successfully implemented in DED systems such as [9] and [30] in a power grid simulation context, showing efficient performance and scalability in comparison with plain DRL models in the same issue. However, as it was the case with GRL algorithms, the sparseness of different considerations, constraints and formalizations of the DED problem, highlighted in section 3.3 bring added complexity when comparing different approaches, since models are build with different DED requirements.

Chapter 4

Methodological Approach

In this chapter, we will formally state the problem this work aims to address, as well as its the proposed solution. In section 4.1 the problem statement is exposed, section **??** lists the solutions main functional, non-functional and structural requirements, section 4.3 addresses the architecture of the solution, section **??** exposes the established methodology and section **??** contains the proposed work plan.

4.1 Problem Statement

> *The reviewed literature addressing solutions to sequential decision-making problems in graph-based environments is sparse and scarce, leading to a research gap for comparative and systematic GRL approaches analysing scalability under different sized scenarios and adaptability to topology variation.*

As addressed in the previous chapter, graphs are ubiquitous representations that can serve to instinctively represent several problems and their objects. In some network-oriented domains, these representations reveal underlying features that can't be naturally represented by plain Euclidean data. This problem becomes even more difficult considering that graph data is complex and sparse, something that brings the need for methods that efficiently extract representations.

By conducting a thorough literature review of the relevant studies in this context, we observed that current RL algorithms are not as efficient as GRL techniques in handling such complex environments, because of not considering and generalizing environment topology features in the decision-making process. This deeply affects the performance of decision systems inserted in network-oriented domains where the intricate relationships between the objects may be relevant for mapping the observable environment states to optimal action policies.

More and more GRL attracts the curiosity of academics, only increasing the relevance of this problem. With the recent advancements of GNNs, the popularity around GRL has risen because of their excellent efficiency in creating optimal graph representations and other graph machine learning problems. However, with GRL being a field whose research is still in initial phase, the gathered literature is very sparse, with a lack of works addressing the benefits, disadvantages and performance of the various proposed models. Moreover, the literature also highlights the importance of studying GRL models in scenarios under topology variations and of different sizes for analysing their scalability.

4.1.1 Scope

This dissertation will focus on studying this problem in the context of single-agent RL algorithms, given that multi-agent systems are significantly more complex to implement. Furthermore, in the context of the dissertation's application domain, which is smart grid services, the possible improvements in GRL techniques will be implemented to the Dynamic Economic Dispatch (DED) problem that studies solutions that optimize power generation cost while maintaining reliable grid stability. Additionally, the GRL proposed models may be also implemented to solve other smart grid systems such as Undervoltage Load Shedding and Volt-VAR Regulation.

4.2 Problem Formalization

In this section, the main problem of study of this dissertation is formally introduced. Firstly, the problem is approached from an application domain perspective concerning the DED problem and its considered features and characteristics. In the second subsection **??**, the DED problem is formalized as a dynamic sequential decision-making problem, and its characteristics are presented under the form of its corresponding MDP.

These two formalizations are crucial for giving the appropriate view on how the GRL algorithms and their environment will be approached and studied.

4.2.1 Dynamic Economic Dispatch

$F(t)$	Total Operational Cost
$F_{\text{NRES}}(t)$	Total Non-Renewable Generators Operational Cost
$F_{\text{NRES}}(t)$	Total Renewable Generators Operational Cost
$F_{\text{ESS}}(t)$	Total ESS Operational Cost
T	Terminal Timestep
t	Timestep
c_i^{NRES}	Cost of conventinal generator i in €/MWh
$P_i^{\text{NRES}}(t)$	Current generated power of non-renewable generator i in MW
β_{RES}	RES wasted energy penalty term
$P_i^{\text{RES}}(t)$	Current generated power of renewable generator i in MW
$\overline{P_i^{\text{RES}}}(t)$	Current power of renewable generator i before curtailmentin MW
c_i^{ESS}	Operational Cost of ESS i in €/MW
$P_i^{\text{ESS}}(t)$	Discharging/Charging Power of ESS i
$P_i^{\text{LOAD}}(t)$	Active Power demand of load i
$Q_i^{\text{LOAD}}(t)$	Reactive Power demand of load i
F_i or $F_{i,j}$	Powerline i status
rho_i or $\text{rho}_{i,j}$	Relative flows of powerline i or with origin in substation i and destination in substation j. Ratio of the flow divided by its thermal limit.
$P_i^{\text{G}}(t)$	Current generated active power of generator i in MW
$Q_i^{\text{G}}(t)$	Current generated reactive power of generator i in MW
N	Number of non-renewable generators
M	Number of renewable generators
K	Number of loads
L	Number of powerlines

4.2.1.1 Objective Function

The DED problem addresses the issue of balancing the necessity for ensuring stability, security and reliability of a power grid while also minimizing its operating cost. This problem is hardened by the paradigm observed in current power systems where ESS and RES are available and also need to be taken in consideration when studying solutions for this problem.

In the idealized power system, the main components taken into account are renewable and non-renewable generators, loads, powerlines, substations and storage systems. Equation 4.1 highlights the main objective function for the DED problem and is composed of three main components: $F_{\text{NRES}}(t)$ represents the cost of energy produced by conventional generators, $F_{\text{RES}}(t)$ is the term that depicts the cost of wasted energy from RES.

$$\min \sum_{t=1}^{T} (F_{\text{NRES}}(t) + F_{\text{RES}}(t)) \tag{4.1}$$

Conventional generation cost calculation is presented in equation 4.2. For the sake of simplicity, this component is reduced to a linear equation and represented by a static c_i term for generator i in €/MWh . Regarding the cost of wasted RES energy, a penalty term β_{RES} is introduced that defines a cost for the abandoned renewable energy.

$$F_{\text{NRES}}(t) = \sum_{i=0}^{N} c_i^{\text{NRES}} P_i^{\text{NRES}}(t)\Delta t \tag{4.2}$$

$$F_{\text{RES}}(t) = \beta_{\text{RES}} \sum_{i=0}^{M} (\overline{P_i^{\text{RES}}}(t) - P_i^{\text{RES}}(t))\Delta t \tag{4.3}$$

4.2.1.2 Constraints

In order to maintain stability and reliable power distribution, the system must follow several operational constraints. These restrictions are related to generators, voltage, powerlines and its overall stability.

The main rule in a power system environment consists in the compliance with power balance. This implies that, at all times, the sum of production output of all generators must surpass the total system load demand, as portrayed by equation 4.4.

$$\sum_{i}^{N} P_i^{\text{NRES}}(t) + \sum_{i}^{M} P_i^{\text{RES}}(t) = \sum_{i}^{L} P_i^{\text{LOAD}}(t) + \delta, \delta > 0 \tag{4.4}$$

Considering non-renewable generations, its output value it is bounded by absolute maximum and minimum values respective to each generator, $\underline{P_i^{\text{NRES}}}$ and $\overline{P_i^{\text{NRES}}}$, as depicted in equation 4.5. Furthermore, as illustrated in equation 4.6, the varying power between two different timesteps is also restricted by maximum ramp up and down limits, $\underline{\eta_i}$ and $\overline{\eta_i}$.

$$\forall t, i : \underline{P_i^{\text{NRES}}} \leq P_i^{\text{NRES}}(t) \leq \overline{P_i^{\text{NRES}}} \tag{4.5}$$

$$\forall t, i : \underline{\eta_i}\Delta t \leq P_i^{\text{NRES}}(t+1) - P_i^{\text{NRES}}(t) \leq \overline{\eta_i}\Delta t \tag{4.6}$$

In contrast with these static production limits, renewable generator production limits are dynamic. Depending on the type of RES, the maximum available generation output is subject to the environmental characteristics dictated by the environment. In such manner, the ouput value of renewable generations for any timestep t is bounded by a maximum production value, $\overline{P_i^{\text{RES}}}(t)$ and 0. Equation 4.7

$$\forall t, i : 0 \leq P_i^{\text{RES}}(t) \leq \overline{P_i^{\text{RES}}}(t) \tag{4.7}$$

Regarding powerlines, there are two thresholds that restrict their operation in cases of line overflow. They are applied to safeguard the condition of powerlines and emulate the behaviour of a real power system rho, in two difference scenarios:

- **Hard Overflow** - In case the relative flow of any powerline surpasses the hard overflow threshold, θ_{hard}, it's instantly disconnected. This is known as time overcurrent;
- **Soft Overflow** - A line is allowed to surpass the soft overflow threshold, θ_{soft}, for T_{soft} timesteps before being disconnected for safety reasons. This consists in an instantenous overcurrent;

In any case, when a powerline is disconnected for safety reasons, it's affected by a reconnection cooldown, $T_{\text{reconnect}}$.

4.2.2 Markov Decision Process (MDP)

In this section, the DED problem is formalized in the as a sequential decision-making problem and its MDP is uncovered.

Action The considered action space includes the change in conventional generator redispatching ΔP_i^{NRES}, renewable energy curtailment P_i^{RES} and ESS absorption/production power P_i^{ESS}. The first concerns only to non-renewable dispatchable generators and is done in increments and decrements, while the second is the ratio of curtailment applied to the total amount of generation ouput of a renewable generator.

This second type of actions refers to the energy discarded from RES for grid stability and security purposes, but they're represented as the upper bound on a renewable generator's output level. In this manner, while positive redispatch actions are directly tied with a higher energy output and, consequently, operational cost of the system, curtailment actions have the opposite effect, given that they are represented as the upper bound on a renewable generator output levels. Therefore, higher curtailment values result in a reduction of the wasted energy provided from RES, which makes these actions crucial for an optimal grid operation.

$$A = \{P_1^{\text{NRES}}, P_2^{\text{NRES}}, \ldots, P_N^{\text{NRES}}, P_1^{\text{RES}}, P_2^{\text{RES}}, \ldots, P_M^{\text{RES}}\} \tag{4.8}$$

Observation The observation space is composed of four components concerning Generators, RES, Loads and Powerlines, highlighted by the following equations:

$$o_1(t) = \begin{bmatrix} P_1^{\text{NRES}} & P_2^{\text{NRES}} & \ldots & P_N^{\text{NRES}} \end{bmatrix} \tag{4.9}$$

$$o_2(t) = \begin{bmatrix} \frac{P_1^{\text{RES}}}{P_1^{\text{RES}}} & \frac{P_2^{\text{RES}}}{P_2^{\text{RES}}} & \ldots & \frac{P_M^{\text{RES}}}{P_M^{\text{RES}}} \end{bmatrix} \tag{4.10}$$

$$o_3(t) = \begin{bmatrix} P_1^{\text{LOAD}} & P_2^{\text{LOAD}} & \dots & P_K^{\text{LOAD}} \\ Q_1^{\text{LOAD}} & Q_2^{\text{LOAD}} & \dots & Q_K^{\text{LOAD}} \end{bmatrix} \tag{4.11}$$

$$o_4(t) = \begin{bmatrix} F_1 & F_2 & \dots & F_L \\ \text{rho}_1 & \text{rho}_2 & \dots & \text{rho}_L \end{bmatrix} \tag{4.12}$$

$$o(t) = \{o_1(t), o_2(t), o_3(t), o_4(t)\} \tag{4.13}$$

This tuple encompasses the main elements of the powergrid an their characteristics, without capturing their topology. This is the space considered for pure-DRL baselines and it was taken from similar literature that addresses DED with equivalent considerations.

Concerning the topology of the network, for GRL algorithms this tuple was replaced by a graph structure representing the power grid configuration. This graph is defined by its adjacency matrix, weight matrix and feature matrix. In equation 4.14, the tuple of features for node i at timestep t can be observed.

$$o_i(t) = \{P_i^{\text{LOAD}}, Q_i^{\text{LOAD}}, P_i^{\text{NRES}}, P_i^{\text{RES}}, \overline{P_i^{\text{RES}}}, t\} \tag{4.14}$$

The adjacency matrix is derived from *line status* and the weight matrix corresponds to th *rho* value of each line. Finally, a graph representation is obtained from the initial equation 4.13, enabling their exploit and posterior study by GRL algorithms.

$$o(t) = G(\text{Adj}, W, X) \tag{4.15}$$

where,

$$\text{Adj}_{i,j} = F_{i,j} \tag{4.16}$$

$$W_{i,j} = \text{rho}_{i,j} \tag{4.17}$$

$$X_i = o_i \tag{4.18}$$

Reward Considering reward functions, three formalizations were considered. The initial implementation can be described by equation 4.19 and corresponds to the total amount of cost saved at time step t, with the maximum cost, $\overline{F_{\text{NRES}}}$, representing the total cost of running the power system for the current step with all generators running at maximum output.

$$\begin{aligned} r_1(t) &= \overline{F_{\text{NRES}}} - F_{\text{NRES}}(t) \\ &= \overline{F_{\text{NRES}}} - \sum_{i=0}^{N} c_i^{\text{NRES}} P_i^{\text{NRES}}(t)\Delta t \end{aligned} \tag{4.19}$$

This definition fully encompasses the main objective of DED directly tying reward maximization and the reduction of operation cost. However, it disregards a secondary objective concerning the maximization of RES generated energy.

In this context, another component is added to consider the cost of wasted RES energy, depicted in equation 4.20. This cost is calculated using the average cost per MW of all non-renewable generators of the system, $\overline{c^{\text{NRES}}}$, and a penalty factor, β_{RES}, to control the impact of wasted energy in reward value.

$$\begin{aligned} r_2(t) &= -F_{\text{RES}}(t) \\ &= \overline{c^{\text{NRES}}} \beta_{\text{RES}} \sum_{i=0}^{M} (\overline{P_i^{\text{RES}}}(t) - P_i^{\text{RES}}(t))\Delta t \end{aligned} \tag{4.20}$$

In addition, this component was also idealized as a bonus instead of a penalty. In contrast with the last definition, the maximization of renewable energy can also be considered through a positive reward for its utilization instead of penalizing its waste. This second version of the RES component can be observed in equation 4.21

$$r_2(t) = \overline{c^{\text{NRES}}} \beta_{\text{RES}} \sum_{i=0}^{M} P_i^{\text{RES}}(t)\Delta t \tag{4.21}$$

$$r(t) = r_1(t) + r_2(t) \tag{4.22}$$

4.3 Solution Architecture

Considering the conclusions taken from the reviewed literature, in chapter 3, our proposed solution combines efficient Deep Reinforcement Learning (DRL) approaches with Graph Neural Networks (GNNs), the state-of-the-art approach for graph representation learning. Generally, the system receives graph-based representations of the environment and encodes them using the GNN algorithm. By also leveraging deep learning techniques, DRL maps the encoded embeddings to optimal action sequences with the goal of meeting the real-time load demand and reducing the operating cost of the power distribution grid. The general architecture of the solution can be observed in figure 4.1.

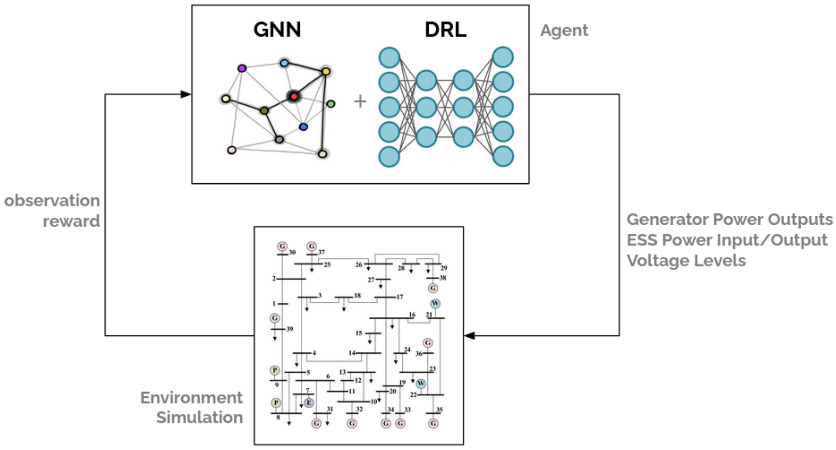

Figure 4.1: Solution Architecture

To fulfil this, the agent adjusts the generator power output and voltage levels and manages the ESS operation in real-time. The proposed solution is trained and tested in a power grid simulation that models the appropriate elements, properties, constraints and operation of the power distribution grid.

In order to achieve this purpose, the GRL models were implemented using *stable-baselines3* DRL implementations and *torch_geometric* GNN models. Beyond having several different implementations available for both their respective types of algorithms, utilizing these two technologies simplifies the integration of models from each, allowing for more straightforward testing of various combinations.

Although stable-baselines3 allows for the implementation of custom feature extractors, this method was not computationally optimal. DRL models of this library that use mini-batching have an increased complexity due to the extractor being called on each batch observation at every step. This issue is derived from another problem which is obviously the storage of raw observations in the replay buffer, where the data is batched from, instead of the extracted features.To solve this, the GNN feature extraction layer was included directly in the observation space which ensured the execution of module only once per step.

4.4 Evaluation Metrics

The solution described in the previous subsections 4.3 and **??** clearly involves intricate operational mechanisms, something that calls for a sophisticated evaluative process. Furthermore, given the comparative nature of this dissertation the evaluation and analysis methods will be key factors in studying and confronting the different combinations of GNNs and DRL, as well as possible improvements in the integration of these techniques. In this manner, we define the four dimensions for evaluating and analysing the different GRL models:

Learning efficiency This dimension assesses how effectively the models learn and improve their decision-making process over time. It involves evaluating how quickly they converge to optimal or near-optimal dispatch strategies through the convergence rate.

Computational Efficiency It's crucial that the solution is able to perform well on real-time execution. In this context, not only it's important to assess the solution's decision computation performance but also to measure the time necessary for the model offline training, as well as the observed CPU/GPU resource utilization.

Dispatch Efficiency The performance of the GRL model in managing power distribution from the various generators will be mainly measured by the average operating cost (in *Euros*) derived from the agent's sequence. Furthermore, other system behaviours will be analysed such as Renewable Energy Source Penetration, through the average ratio of maximum power and real power output, and ESS utilization, through the average energy stored.

$$\text{Daily Operating Cost} = \tag{4.23}$$

$$P^{\text{RES}}_{\text{Wasted}} = \tag{4.24}$$

Scalability and Adaptation The solution will be applied to scenarios of several sizes for analysing its scalability. Furthermore, we will induce variations in the simulation scenarios to test the model's ability to handle topology changes.

To ensure a proper and unbiased evaluation of the proposed models, the data provided by the used scenarios is splitted into train and test samples with a ratio of 90% and 10%, respectively. Cross-validation was considered, but discarded because of the slow performance of the algorithms and apparent reduced learning efficiency.

4.5 Technologies

Beyond being a well-documented, fast, robust and modular language, *Python* is an ubiquitous technology for implementing machine learning algorithms. Furthermore, the main libraries used to implement the simulation environment and algorithm are native to this language.

In table x, the main requirements for the project can be observed, along with their respective version number and a short description of their purpose and functionality.

Grid2Op is a critical requirement used to simulate the power grid operation in different manageable scenarios. This is further discussed in the next subsection 4.6. Stable-Baselines3 was chosen for its performance, customization and descriptive documentation, while PyTorch Geometric was favoured because of the wide variety of GNNs Implementations included. Additionally, these libraries mix well because stable-baselines3 already uses PyTorch in its DRL implementations.

Name	Description	Version Number
Python	Functional Programming Language	3.11.9
Numpy	Numerical Computing Library	1.24.3
Ray	Used for hyperparameter tuning	2.21.0
PyTorch	ML Library used by stable-baselines3	2.3.0
PyTorch Geometric	GNN Implementations	2.5.3
Grid2Op	Power Grid Simulation Platform	1.9.8
Pandas	Data Manipulation and Analysis	1.5.3
LightSim2Grid	Fast Grid2Op Backend	0.8.2
Gymnasium	RL Single-Agent API	0.29.1
Stable Baselines 3	DRL Implementations	2.3.2

Table 4.1: Project Requirements

Lastly, *Gymnasium* is used as the standard RL single-agent API in the entirety of the project, adding robustness, scalability and flexibility to agent-environment interaction process in the implementation.

4.6 Simulation Environment

The *Grid2Op* framework [17] will be used for modelling the sequential decision-making process on simulated power distribution grids. *Grid2Op* is designed by RTE (*Réseau de Transport d'Électricité*), the electricity transmission system operator of France, and is equipped with a variety of pre-defined scenarios used in coding competitions and based on real-world data [17].

This *python* library implements the creation of power grid environments that emulate a subset of the elements and physical constraints of real power systems. On one hand, the main objects and their interactions are captured realistically, while on the other, the abstraction level used to model the fundamental elements of a power grid facilitates the implementation of intelligent systems in safe and controllable scenarios.

Furthermore, *Grid2Op* models the dynamic topology of power systems as graph-structured data, which also eases the study of how this information can influence decision-making systems performing on these environments. In figure 4.2, a graphical plot of the environment state at an arbitrary timestep can be observed.

This framework is compatible with the *Gymnasium* framework [16], a widely used toolkit for developing RL algorithms, which will also will be used in this work together with the Grid2Op simulation environment.

- redispatch - change the production output of non-renewable generators in incrementing or decrementing intervals
- curtail - change the production setpoint of renewable generators bellow the current maximum available power [17]

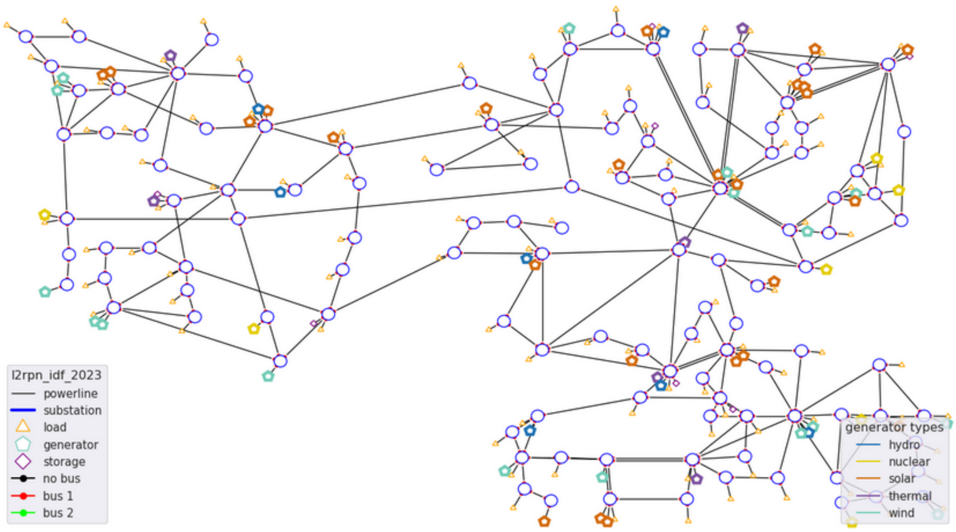

Figure 4.2: Grid2Op *l2rpn_idf_2023* 118-bus test case [17]

4.6.1 Elements

The main elements that are modelled by the simulation environment are presented in this subsection. The power grid is reduced to its fundamental components and their properties:

- **Buses** are the fundamental objects of the power grid, representing nodes where power sources, loads and other elements are connected[17].
- **Powerlines** represent edges in the power grid and connect the different buses together. They represent the physical transmission and distribution lines and allow power to flow from one part of the grid to another [17].
  - status
  - rho
- **Generators** are critical grid elements connected to buses whose main role is to produce power and maintain grid stability by balancing the energy supply and demand. They can be Conventional Thermal Generators, Wind Turbines or Photovoltaic Cells [17].
  - $P_i^{\mathrm{NRES}}(t)$ - Power generation at non-renewable generator i
  - $P_i^{\mathrm{RES}}(t)$ - Power generation at renewable generator i
  - $\overline{P_i^{\mathrm{RES}}}(t)$ - Maximum power generation at renewable generator i
  - $\overline{P_i^{\mathrm{NRES}}}, \underline{P_i^{\mathrm{NRES}}}$ - Maximum/Minimum power generation at renewable generator i
- **Loads** consume power from the grid, simulating electricity use. They're also associated to an individual bus [17].
  - P_i^{LOAD} - Active power at load i

  - Q_i^{LOAD} - Reactive power at load i

- **Storage Units** can act as both consumers and producers. They're able to retain energy from the power grid when production surpasses demand for later injecting back power when convenient. Storage units are bound by a maximum energy storage capacity [17].

Beyond *grid2op* to build and run the test cases and *gymnasium* as the RL framework, this project will use *PyTorch* [42] with *PyTorch Geometric* [50] library for developing the GNN because of its extensive list of available implemented models. The different combinations of algorithms will be applied to a set of modified scenarios that fulfil the settled requirements. For Deep Reinforcement Learning algorithms, solutions with plain SAC, DDPG and PPO approaches and combined with the GCN, GAT and GIN architectures.

Concerning result analysis, it's also important to point out the use of quantitative methods for evaluating the different implemented models, a topic that is further explored in the following section 4.4.

4.7 Parameters

Grid2Op also defines a number of customizable environment parameters that control some very important grid properties and settle some bounds on the agent-envrionment interaction. As there is a significant number of parameters that consider details that are out of the scope of this project, only the relevant parameters are exposed in table 4.2.

Table 4.2: Grid2Op Relevant Parameters

Parameter	**Description**	**Default**
`NO_OVERFLOW_DISCONNECTION`	If `true`, the powerlines over their thermal limit will never be disconnected	`false`
`NB_TIMESTEP_OVERFLOW_ALLOWED`	Number of timesteps a powerline is allowed to be in a soft overflow before it gets disconnected by time overcurrent	2
`NB_TIMESTEP_RECONNECTION`	Number of timesteps that a line disconnected for security reasons will stay disconnected	10
`HARD_OVERFLOW_THRESHOLD`	If the power flow of a line is above the settled hard overflow threshold it's instantaneously and automatically disconnected	2.0
`SOFT_OVERFLOW_THRESHOLD`	If the power flow of a line is above the defined soft overflow threshold for over `NB_TIMESTEP_OVERFLOW_ALLOWED` steps, it's automatically disconnected	1.0
`ENV_DC`	If `true`, the environment will use direct current approximations instead of alternative current	`false`
`LIMIT_INFEASIBLE_CURTAILMENT_STORAGE_ACTION`	If set to `true`, the environment will automatically limit curtailment (and storage) actions that otherwise would lead to an infeasible system state. Might help the training and learning process of the model and significantly increases the survivability of the agent	`false`

Most of the parameters, with the exception of one, were left at their default values depicted in table 4.2. This includes the overflow thresholds, all of the timestep limits mentioned and direct current flag, whose values were found to be sensible enough to maintain a realistic yet feasible model of the environment. The only parameter that was tampered was `LIMIT_INFEASIBLE_CURTAILMENT_STORAGE` with the goal of improving the learning efficency of the algorithms during training.

4.8 Scenarios

Table 4.3: Test Case Sizes

	l2rpn_case14_sanbox	l2rpn_icaps_2021_large	l2rpn_idf_2023
Buses	14	36	118
Powerlines	20	59	186
Generators	6	22	62
Loads	11	37	99
Episodes	1004	2952	832

This work will use the pre-defined *grid2op l2rpn_case14_sandbox*, *l2rpn_icaps_2021_large* and *l2rpn_idf_2023* test environments. The scenarios define the properties and characteristics of loads and generators at each time step, as well as the grid layout [17] for a weekly episodes. Its sizes of test cases can be further observed in table 4.3.

The first case was used for the main implementation and development of the algorithms given its small grid size. Motivated by its considerable grid size and amount of training data, the second test case was mainly used for the calibration of the model's main parameters. It's a subset of the original 118-bus system [12] with 50 years worth of data divided into independent [1] weekly scenarios. Finally, the third scenario, is based on the same original test case as the previous one and it includes modifications that aim to accommodate the *possible energy mix* of France in 2035, containing 16 years of data [17]. It was mainly used to study the algorithm's scalability and performance in larger scenarios in comparison with the baseline plain-DRL model.

[1] non-consecutive

Chapter 5

Result Discussion

In this chapter, the obtained results are thoroughly analysed and discussed. The performed experiments can be subdivided into 6 categories: Curtail Lower Limit (section 5.2), Limit Infeasible Curtailment Actions 5.1, Reward Tests5.3, GNNs Hyperparameter Tuning, GNN Implementation Comparison 5.5 and Scability Tests 5.5.

5.1 Limit Infeasible Curtailment Actions

5.2 Curtailment Lower Limit

The initial exploratory experiments and posterior analysis of training performance showed significant instability and lack of convergence of DRL models. This is aggravated by the inclusion of curtailment actions in the action space, which in turn introduced additional complexity to the decision process and severely affected the model's survivability.

Considering the secondary goal of maximization of generated renewable energy, the idealization of a lower bound to restrict the curtailment action space became a potential solution to aid in the model's convergence. Additionally, two methods of

Results showed in graph GRAPH proved that the introduced curtailment lower bounds considerably increased the models training performance and convergence, with the square root decay method achieving the overall highest average accumulated reward during validation. Introducing a decay in the lower bound limit implies an enhanced overall performance and survivability of the algorithm.

5.3 Rewards Tests

Several implementations posed as relevant formulas to describe the reward function of the environment, namely three: Economic Reward, RESs Penalty Factor Reward and RESs Bonus Reward. The first considered only the saved cost in non-renewable generators operation, while the second and third accounted for RES maximization in a penalty and bonus factor, respectively.

Learning Rate	1e-4
Gamma	0.85
Entropy Coefficient	'auto'
Gradient Steps	1
Buffer Size	1e6
Batch Size	256
Tau	0.001
Target Update Interval	1
Training Frequency	1
Target Entropy	'auto'
Optimizer	Adam
Activation Function	ReLU
Number of Units per Hidden Layer	128
Number of Hidden Layers	6

Table 5.1: Soft Actor-Critic Parameters

Input Channels	6
Hidden Channels	18
Number of Layers	2
Output Channels	6
Dropout Rate	0.1
Activation Function	ReLu
Aggregation Function	'sum'
act_first	True
Flow	Source to Target
Node Dimension	-2
Decomposed Layers	-1
Improved	False
Cached	False
Normalize	True
Bias	True

Table 5.2: GCN Parameters

Considering the new parameter introduced by both the Penalty and Bonus Factor rewards, β, the experiments took into account three distinct values: $\{0.2, 0.4, 0.6\}$.

It's critical to take into account that when evaluating and comparing Reward Functions, the average accumulative reward values have different meanings. Furthermore, other crucial metrics such as the survival rate and the average daily operating cost are favoured.

Results revealed that the proposed implementations were superior to the already defined Economic Reward both in survivability and overall performance of the models. The with the best results were observed in the Penalty Factor Reward with $\beta = 0.2$.

5.4 GNNs Hyperparameter Tuning

As a key area of this research work, the GNN component of the proposed algorithm was given a particular focus in what accounts for hyperparameter tuning. Concrete experiments were devised to assess the performance of different parameter combinations, mainly using the GCN-SAC algorithm.

The GNN aggregation function plays a fundamental role in the architecture of the solution so the tuning process first focused on this parameter. There are five available types of aggregation schemes:

- **Sum** -
- **Max** -
- **Min** -
- **Mean** -
- **Mul** -

In a first experiment, five models each with the different types of aggregation functions were trained for 5000 episodes. The main parameters and results can be observed in appendix A, section A.6. The function that obtained the best average accumulative reward during validation was `max`, achieving around 4% more than the second placed `sum` function. Furthermore, this model managed to do better than average in most metrics while simultaneously being outperformed in all of them, which shows the balance between the importance of cost saving while considering the maximization of renewable energy.

Regarding the number of GNN layers, section **??** of appendix A represents the results from the conducted experiment with this scope. This test considered the `max` aggregation function and a number of GCN layers between $[1, 6]$. In contrast with the last experiment, the best model which used a 6-layer GNN outperformed the second place with a significant difference of 22.4%. This results led to the use of these two parameters in the GCN model.

5.5 GNN Implementation Comparison

5.6 Scalability Tests

Chapter 6

Conclusions

In conclusion, this report establishes a solid foundation for implementing and proposing concrete improvements to GRL techniques and efficiently solve the problem of DED in graph-based environments.

We concluded in the literature review, that GRL is already a promising solution for DED, yielding better performance than other approaches. In general, GNNs are an ubiquitous method, throughout the gathered literature, for extracting efficient environment topological representations. The identified limitations of existent implementations lie on their lack of: (1) scalability to larger environments, resulting in a significant decrease in computation performance; (2) a seamless integration between GNN and DRL algorithms and (3) adaptability to real-time topology changes.

Moreover, we aim to study how to tackle these limitations by implementing different GRL approaches for solving the DED problem and conducting a comparative study between the different techniques. The implementations will be compromised of various combinations of GNNs, for efficiently extracting the relevant environment features, and DRL algorithms for mapping the extracted representations into optimal action sequences. Case studies will be carried out within different size modified IEEE bus systems for testing the various models and confront their results. Lastly, the models will be analysed considering their training efficiency, computational performance, dispatch efficiency, scalability to large scenarios and adaptability to topological changes. Based on the conclusions drawn from the comparative study, we will propose a GRL that implements concrete enhancements.

6.1 Expected Contributions

In this section, we list the expected contributions derived from this work on a Scientific, Technological and Application levels:

Scientific A systematic and comparative study of different GRL approaches. This fills the research gap for systematic studies comparing different proposed techniques. Furthermore, this work will bring a clearer insight regarding the best practices on implementing GRL models

Technological A model resulting from this study with concrete improvements over the current GRL techniques proposed so far and tackling their limitations. Beyond this,

Application An efficient solution for the DED problem with GRL algorithms, compromising significant contribution to the research on DED systems in complex scenarios

References

[1] Feedforward Neural Network. https://orgs.mines.edu/daa/wp-content/uploads/sites/38/2019/08/1_Gh5PS4R_A5drl5ebd_gNrg@2x.jpg.

[2] Renewables supplied 88% of Portugal's electricity consumption in January. *Reuters*, February 2023.

[3] Paul Almasan, José Suárez-Varela, Krzysztof Rusek, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. *Computer Communications*, 196:184–194, December 2022.

[4] Yuyang Bai, Siyuan Chen, Jun Zhang, Jian Xu, Tianlu Gao, Xiaohui Wang, and David Wenzhong Gao. An adaptive active power rolling dispatch strategy for high proportion of renewable energy based on distributed deep reinforcement learning. *Applied Energy*, 330:120294, January 2023.

[5] R. Bayindir, I. Colak, G. Fulli, and K. Demirtas. Smart grid technologies and applications. *Renewable and Sustainable Energy Reviews*, 66:499–516, December 2016.

[6] Emma Brunskill. CS234: Reinforcement Learning Winter 2023.

[7] Eugene Charniak. *Introduction to Deep Learning*. The MIT Press, Cambridge, Massachusetts London, England, 2018.

[8] Fanfei Chen, John D. Martin, Yewei Huang, Jinkun Wang, and Brendan Englot. Autonomous Exploration Under Uncertainty via Deep Reinforcement Learning on Graphs. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6140–6147, October 2020.

[9] Junbin Chen, Tao Yu, Zhenning Pan, Mengyue Zhang, and Bairong Deng. A scalable graph reinforcement learning algorithm based stochastic dynamic dispatch of power system under high penetration of renewable energy. *International Journal of Electrical Power & Energy Systems*, 152:109212, October 2023.

[10] Xinpei Chen, Tao Yu, Zhenning Pan, Zihao Wang, and Shengchun Yang. Graph representation learning-based residential electricity behavior identification and energy management. *Protection and Control of Modern Power Systems*, 8(1):1–13, December 2023.

[11] Yongdong Chen, Youbo Liu, Junbo Zhao, Gao Qiu, Hang Yin, and Zhengbo Li. Physical-assisted multi-agent graph reinforcement learning enabled fast voltage regulation for PV-rich active distribution network. *Applied Energy*, 351:121743, December 2023.

[12] Rich Christie. Power Systems Test Case Archive.

[13] Peng Cui, Lingfei Wu, Jian Pei, Liang Zhao, and Xiao Wang. Graph Representation Learning. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 17–26. Springer Nature, Singapore, 2022.

[14] Bangji Fan, Xinghua Liu, Gaoxi Xiao, Yu Kang, Dianhui Wang, and Peng Wang. Attention-Based Multi-Agent Graph Reinforcement Learning for Service Restoration. *IEEE Transactions on Artificial Intelligence*, pages 1–15, 2023.

[15] Hassan Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1):18–28, January 2010.

[16] Farama Foundation. Gymnasium Documentation. https://gymnasium.farama.org/index.html.

[17] RTE France. Grid2Op's Documentation. https://grid2op.readthedocs.io/en/latest/index.html.

[18] Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2083–2092. PMLR, May 2019.

[19] Zhen Gao, Lei Yang, and Yu Dai. Fast Adaptive Task Offloading and Resource Allocation in Large-Scale MEC Systems via Multi-Agent Graph Reinforcement Learning. *IEEE Internet of Things Journal*, pages 1–1, 2023.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 2016.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870. PMLR, July 2018.

[22] William L Hamilton. Graph Representation Learning.

[23] Xiaoyun Han, Chaoxu Mu, Jun Yan, and Zeyuan Niu. An autonomous control technology based on deep reinforcement learning for optimal active power dispatch. *International Journal of Electrical Power & Energy Systems*, 145:108686, February 2023.

[24] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018.

[25] Daner Hu, Zichen Li, Zhenhui Ye, Yonggang Peng, Wei Xi, and Tiantian Cai. Multi-agent graph reinforcement learning for decentralized Volt-VAR control in power distribution systems. *International Journal of Electrical Power & Energy Systems*, 155:109531, January 2024.

[26] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks, February 2017.

[27] Lei Lei, Yue Tan, Glenn Dahlenburg, Wei Xiang, and Kan Zheng. Dynamic Energy Dispatch Based on Deep Reinforcement Learning in IoT-Driven Smart Isolated Microgrids. *IEEE Internet of Things Journal*, 8(10):7938–7953, May 2021.

[28] Lixiong Leng, Jingchen Li, Haobin Shi, and Yi'an Zhu. Graph convolutional network-based reinforcement learning for tasks offloading in multi-access edge computing. *Multimedia Tools and Applications*, 80(19):29163–29175, August 2021.

[29] Nan Li, Alexandros Iosifidis, and Qi Zhang. Graph Reinforcement Learning-based CNN Inference Offloading in Dynamic Edge Computing. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, pages 982–987, December 2022.

[30] Peng Li, Wenqi Huang, Zhen Dai, Jiaxuan Hou, Shang Cao, Jiayu Zhang, and Junbin Chen. A Novel Graph Reinforcement Learning Approach for Stochastic Dynamic Economic Dispatch under High Penetration of Renewable Energy. In *2022 4th Asia Energy and Electrical Engineering Symposium (AEEES)*, pages 498–503, March 2022.

[31] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow. Gated graph sequence neural networks. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[32] Paul Lienert. GM takes on Tesla in home and commercial energy storage, management. *Reuters*, October 2022.

[33] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019.

[34] Zhiyuan Liu and Jie Zhou. *Introduction to Graph Neural Networks*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham, 2020.

[35] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1928–1937. PMLR, June 2016.

[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[37] Miguel Morales. *Grokking Deep Reinforcement Learning*. Manning, Shelter Island [New York], 2020.

[38] Mingshuo Nie, Dongming Chen, and Dongqi Wang. Reinforcement Learning on Graphs: A Survey. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4):1065–1082, August 2023.

[39] OpenAI. Spinning Up Documentation. https://spinningup.openai.com/en/latest/index.html.

[40] Yangzhou Pei, Jun Yang, Jundong Wang, Peidong Xu, Ting Zhou, and Fuzhang Wu. An emergency control strategy for undervoltage load shedding of power system: A graph deep reinforcement learning method. *IET Generation, Transmission & Distribution*, 17(9):2130–2141, May 2023.

[41] A. T. D. Perera and Parameswaran Kamalaruban. Applications of reinforcement learning in energy systems. *Renewable and Sustainable Energy Reviews*, 137:110618, March 2021.

[42] PyTorch. PyTorch. https://pytorch.org/.

[43] Tao Qian, Chengcheng Shao, Xiuli Wang, and Mohammad Shahidehpour. Deep Reinforcement Learning for EV Charging Navigation by Coordinating Smart Grid and Intelligent Transportation System. *IEEE Transactions on Smart Grid*, 11(2):1714–1723, March 2020.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017.

[45] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm, December 2017.

[46] Liam Stoker and Liam Stoker. Energy storage outranks solar in company investment plans. *Reuters*, December 2023.

[47] Richard S. Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, nachdruck edition, 2014.

[48] Jian Tang and Renjie Liao. Graph Neural Networks for Node Classification. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 41–61. Springer Nature, Singapore, 2022.

[49] Zhiqing Tang, Jiong Lou, Fuming Zhang, and Weijia Jia. Dependent Task Offloading for Multiple Jobs in Edge Computing. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–9, August 2020.

[50] PyG Team. PyG - pytorch_geometric. https://pyg.org/.

[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, February 2018.

[52] N. Vesselinova, R. Steinert, D.F. Perez-Ramirez, and M. Boman. Learning Combinatorial Optimization on Graphs: A Survey with Applications to Networking. *IEEE Access*, 8:120388–120416, 2020.

[53] Tamilmaran Vijayapriya and Dwarkadas Pralhadas Kothari. Smart Grid: An Overview. *Smart Grid and Renewable Energy*, 02(04):305–311, 2011.

[54] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, July 2020.

[55] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors. *Graph Neural Networks: Foundations, Frontiers, and Applications*. Springer Nature Singapore, Singapore, 2022.

[56] Lingfei Wu, Peng Cui, Jian Pei, Liang Zhao, and Le Song. Graph Neural Networks. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 27–37. Springer Nature, Singapore, 2022.

[57] Qiang Xing, Zhong Chen, Tian Zhang, Xu Li, and KeHui Sun. Real-time optimal scheduling for active distribution networks: A graph reinforcement learning method. *International Journal of Electrical Power & Energy Systems*, 145:108637, February 2023.

[58] Qiang Xing, Zhong Chen, Ziqi Zhang, Ruisheng Wang, and Tian Zhang. Modelling driving and charging behaviours of electric vehicles using a data-driven approach combined with behavioural economics theory. *Journal of Cleaner Production*, 324:129243, November 2021.

[59] Qiang Xing, Yan Xu, and Zhong Chen. A Bilevel Graph Reinforcement Learning Method for Electric Vehicle Fleet Charging Guidance. *IEEE Transactions on Smart Grid*, 14(4):3309–3312, July 2023.

[60] Qiang Xing, Yan Xu, Zhong Chen, Ziqi Zhang, and Zhao Shi. A Graph Reinforcement Learning-Based Decision-Making Platform for Real-Time Charging Navigation of Urban Electric Vehicles. *IEEE Transactions on Industrial Informatics*, 19(3):3284–3295, March 2023.

[61] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks?, February 2019.

[62] PeiDong Xu, YangZhou Pei, Xinhu Zheng, and Jun Zhang. A Simulation-Constraint Graph Reinforcement Learning Method for Line Flow Control. In *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, pages 319–324, October 2020.

[63] Peidong Xu, Jun Zhang, Tianlu Gao, Siyuan Chen, Xiaohui Wang, Huaiguang Jiang, and Wenzhong Gao. Real-time fast charging station recommendation for electric vehicles in coupled power-transportation networks: A graph reinforcement learning method. *International Journal of Electrical Power & Energy Systems*, 141:108030, October 2022.

[64] Zhongxia Yan, Jingguo Ge, Yulei Wu, Liangxiong Li, and Tong Li. Automatic Virtual Network Embedding: A Deep Reinforcement Learning Approach With Graph Convolutional Networks. *IEEE Journal on Selected Areas in Communications*, 38(6):1040–1057, June 2020.

[65] Ting Yang, Liyuan Zhao, Wei Li, and Albert Y. Zomaya. Dynamic energy dispatch strategy for integrated energy system based on improved deep reinforcement learning. *Energy*, 235:121377, November 2021.

[66] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. XGNN: Towards Model-Level Explanations of Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 430–438, New York, NY, USA, August 2020. Association for Computing Machinery.

[67] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1621–1632. Curran Associates, Inc., 2020.

[68] Liang Zhao, Lingfei Wu, Peng Cui, and Jian Pei. Representation Learning. In Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao, editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 3–15. Springer Nature, Singapore, 2022.

[69] Tianqiao Zhao and Jianhui Wang. Learning Sequential Distribution System Restoration via Graph-Reinforcement Learning. *IEEE Transactions on Power Systems*, 37(2):1601–1611, March 2022.

Appendix A

Appendix

A.1 Limit Infeasible Curtail Action Experiment Results

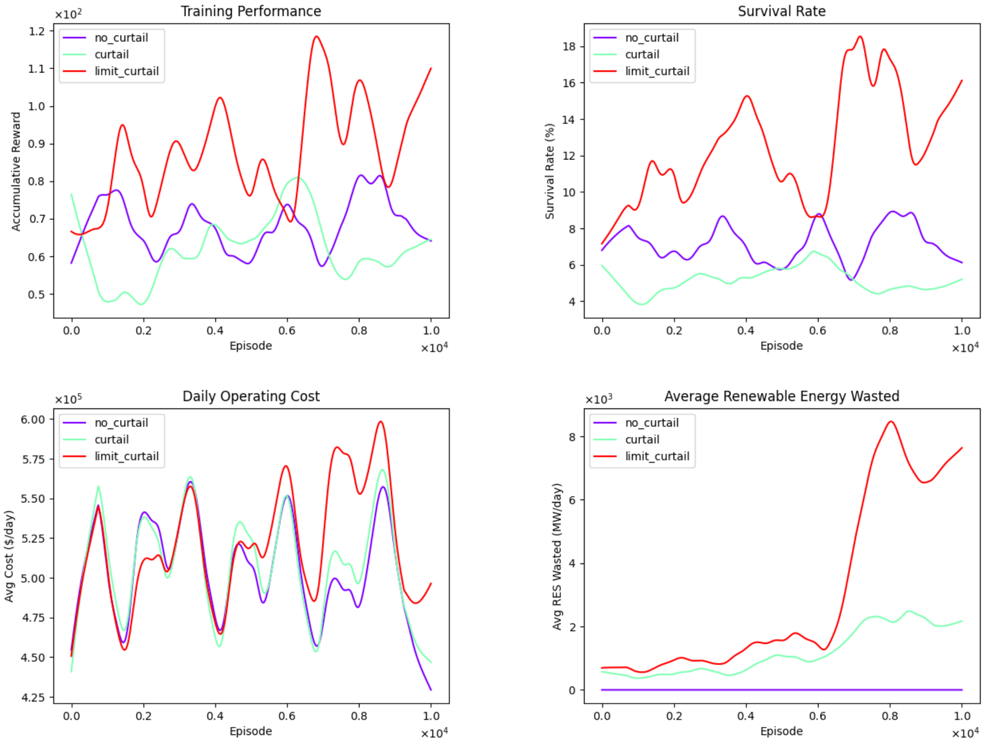

Figure A.1: Training Results of the Experiments concerning Limit Infeasible Curtail Actions.

Model	**Avg. Accumulative Reward**	**Avg. Length (Steps)**	**Avg Daily Operating Cost (€)**	**Avg. Renewables Wasted (MW/day)**	**Total Time (Seconds)**
no_curtail	68.87	225.46	533238.32	0.0	812.81
curtail	60.81	108.27	551028.29	4565.01	510.97
limit_curtail	78.62	759.82	575798.57	7640.76	2329.76

Table A.1: Validation Results of the Experiments concerning Limit Infeasible Curtail Actions.

A.2 Curtailment Lower Limit Smoothing

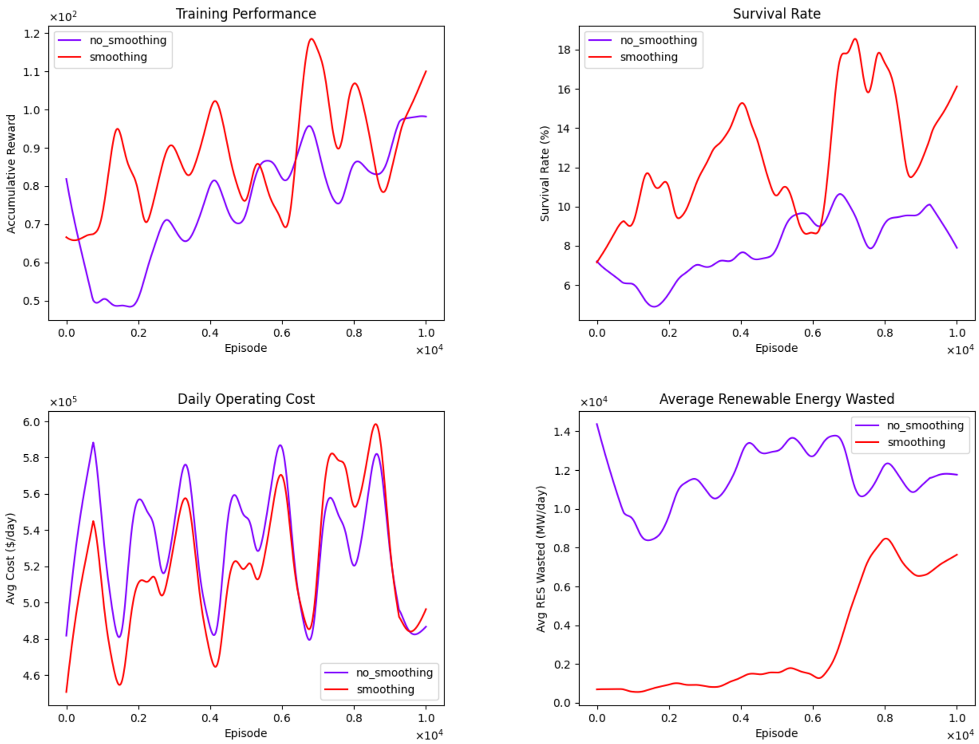

Figure A.2: Training Results of the Experiments concerning Curtailment Lower Limit Smoothing.

Model	**Avg. Accumulative Reward**	**Avg. Length (Steps)**	**Avg Daily Operating Cost (€)**	**Avg. Renewables Wasted (MW/day)**	**Total Time (Seconds)**
no_smoothing	94.91	476.48	565094.39	10106.84	3011.93
smoothing	78.62	759.82	575798.57	7640.76	2329.76

Table A.2: Validation Results of the Experiments concerning Limit Infeasible Curtail Actions.

A.3 Reward Experiments

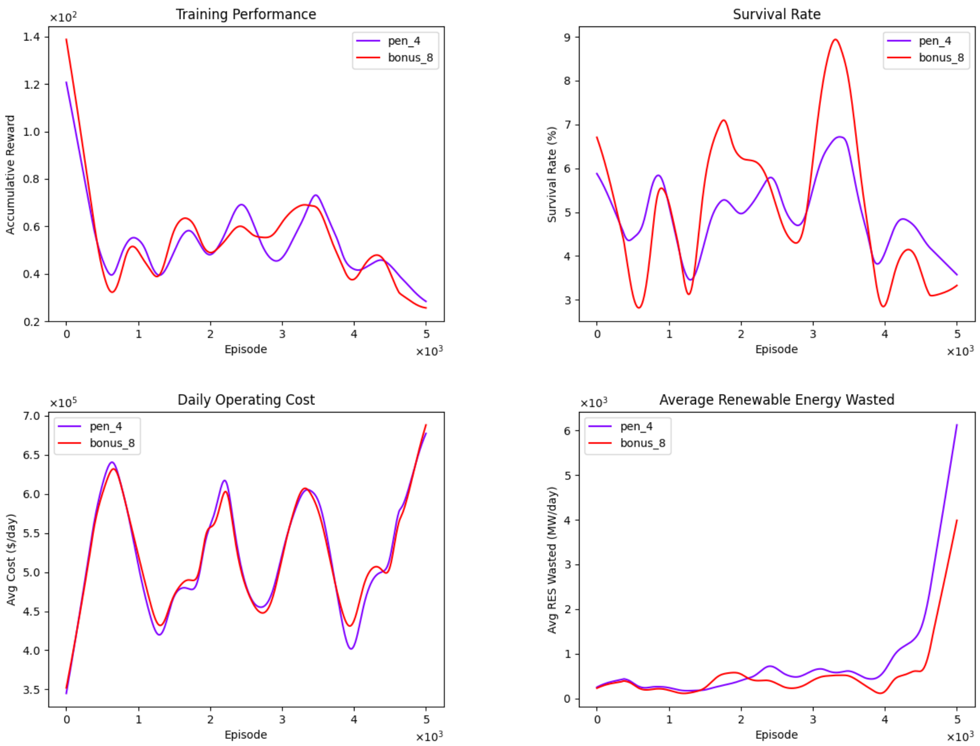

Figure A.3: Training Results of the best Penalty and Bonus Factor Rewards.

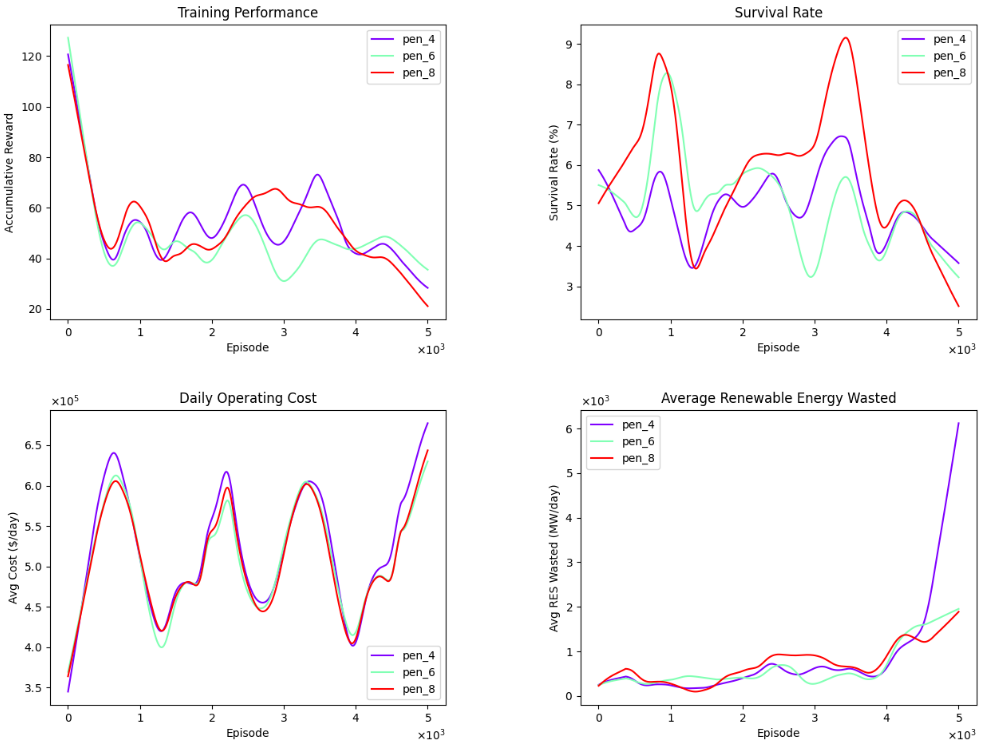

Figure A.4: Training Results of the Penalty Factor Rewards.

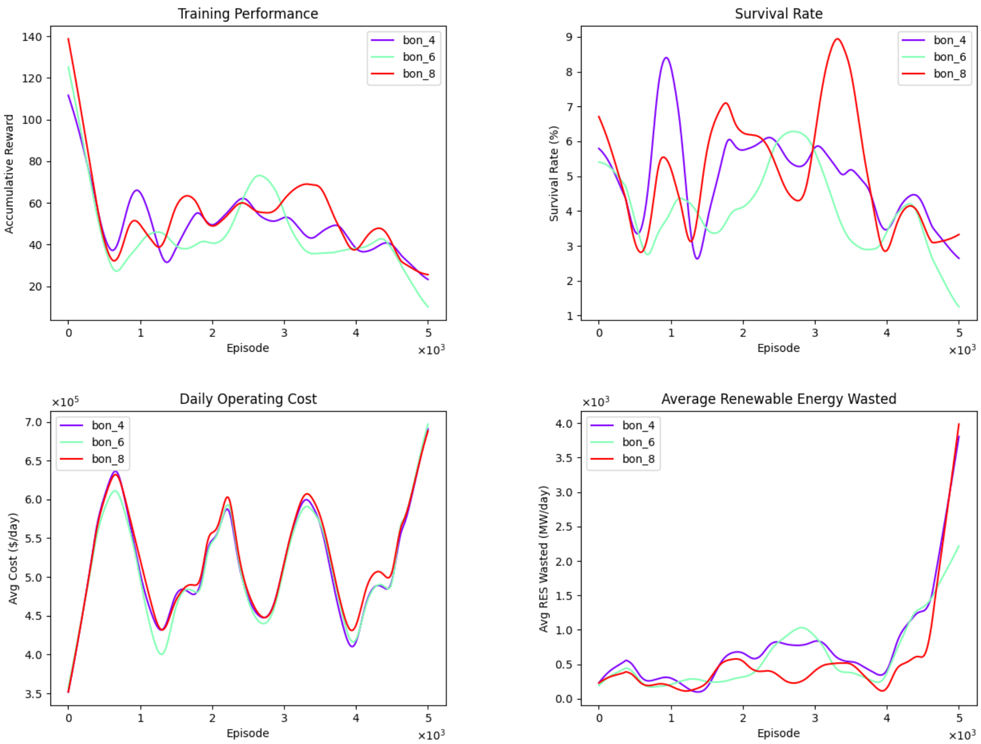

Figure A.5: Training Results of the Bonus Factor Rewards.

Model	Avg. Accumulative Reward	Avg. Length (Steps)	Avg Daily Operating Cost (€)	Avg. Renewables Wasted (MW/day)	Total Time (Seconds)
pen_4	40.57	80.75	565893.63	4100.20	230.99
pen_6	37.54	81.40	558362.82	2570.71	231.09
pen_8	38.22	76.35	555582.14	2612.34	222.58
bon_4	40.13	76.23	565757.16	3723.72	223.90
bon_6	30.78	55.06	566786.07	2750.34	189.01
bon_8	40.88	96.43	560927.60	3115.18	256.59

Table A.3: Validation Results of the Experiments concerning Limit Infeasible Curtail Actions.

Parameter	Values
Aggregation Function	{'sum','mean', 'min', 'max', 'mul'}
Number of Layers	{1,2,3,4,5}
Hidden Channels	{6, 12, 18, 24, 36}
Ouput Channels	{3, 6, 12, 18, 24, 36}
Dropout Rate	[0.1, 0.4]
Activation First	True, False
Heads	1,2,3,6
GATv2	True, False

Table A.4: General GNN Parameters Tuned

Implementation	Parameter	Values
GCN	Improved	True, False
GAT	Heads	1,2,3,6
GAT	GATv2	True, False

Table A.5: Model-Specific Parameters Tuned

A.4 Experiments on the GNN Tuning

A.5 Experiments on the GNN Parameters *act_first* and *improved*

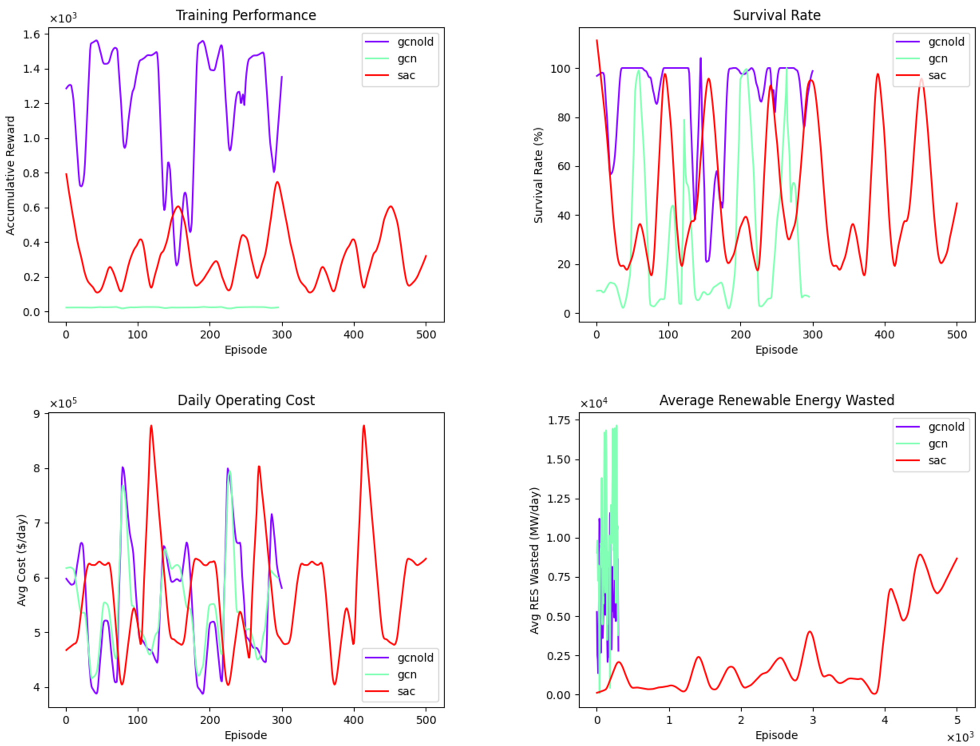

Figure A.6: Training Results of the act_first and improved Parameter Tests.

Parameter	**Values**
Aggregation Function	{'sum','mean', 'min', 'max', 'mul'}
Number of Layers	1
Hidden Channels	18
Ouput Channels	6
Dropout Rate	0.1
Activation First	True

Table A.6: Parameters of Aggregation Function Experiment

A.6 Experiments on the GCN Aggregation Function

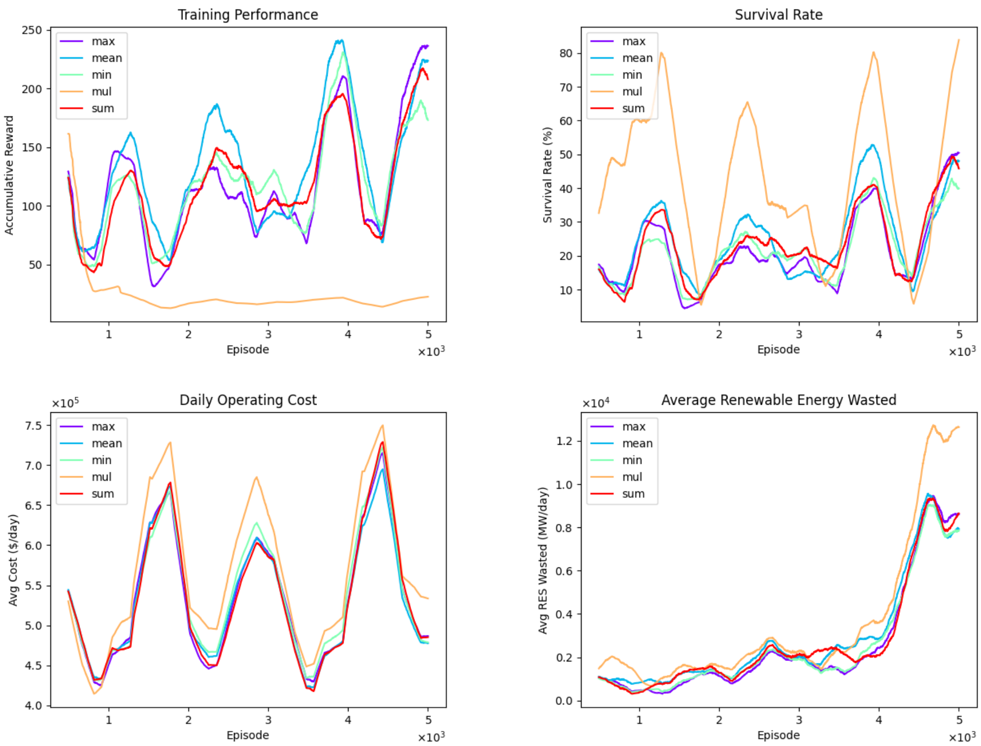

Figure A.7: Training Results of different types of GNN aggregation functions

Model	**Avg. Accumulative Reward**	**Avg. Length (Steps)**	**Avg Daily Operating Cost (€)**	**Avg. Renewables Wasted (MW/day)**	**Total Time (Seconds)**
max	119.16	677.75	560219.45	6564.49	552.51
sum	114.58	768.58	569952.37	7268.28	606.28
mean	92.90	551.24	557637.84	6268.06	454.24
min	80.89	495.70	559033.63	6687.94	416.63
mul	15.76	1104.06	608319.66	10305.96	846.43

Table A.7: Validation Results of different types of GNN aggregation functions.

Parameter	Values
Aggregation Function	'max'
Number of Layers	[1, 6]
Hidden Channels	18
Ouput Channels	6
Dropout Rate	0.1
Activation First	True

Table A.8: General GNN Parameters Tuned

A.7 Experiments on the number of GCN layers

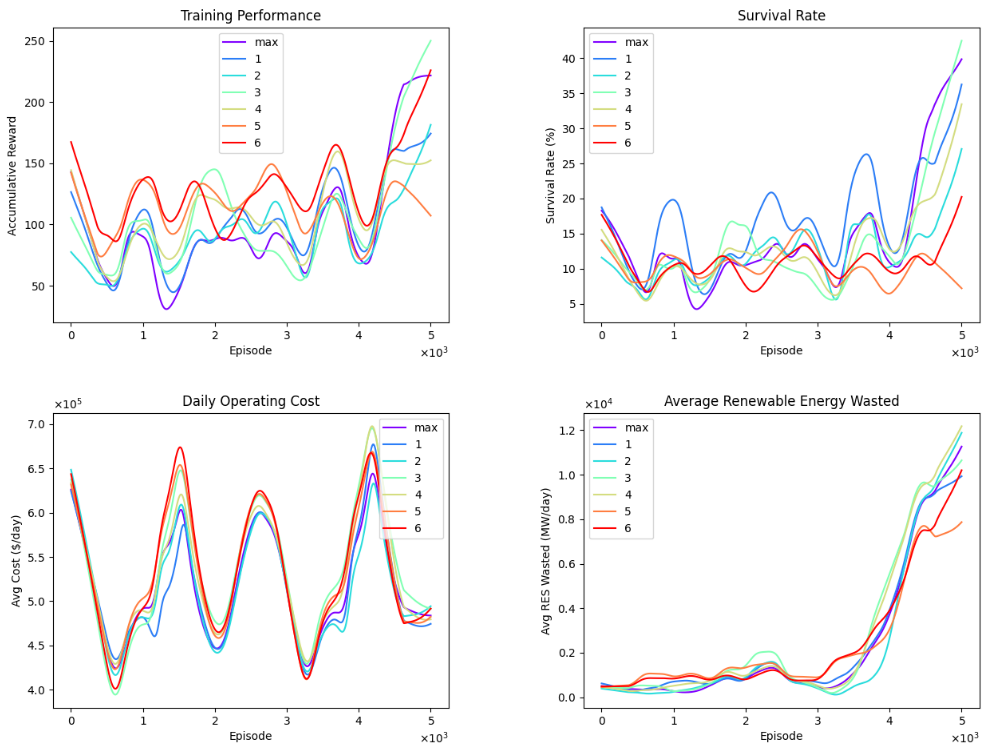

Figure A.8: Training Results of models with 2, 3, 4 and 5 GNN Layers

Model	Avg. Accumulative Reward	Avg. Length (Steps)	Avg Daily Operating Cost (€)	Avg. Renewables Wasted (MW/day)	Total Time (Seconds)
6	135.75	487.09	560147.77	5862.49	655.44
3	110.73	736.32	567510.58	7116.24	744.50
1	101.24	709.76	565865.82	6855.58	573.30
4	98.67	451.51	577464.90	8641.05	529.76
5	92.97	286.65	550205.21	6397.41	386.19
2	91.58	694.67	575093.64	5989.95	638.11

Table A.9: Validation Results of the Experiments concerning Limit Infeasible Curtail Actions.

A.8 Experiments on the number of GAT Heads

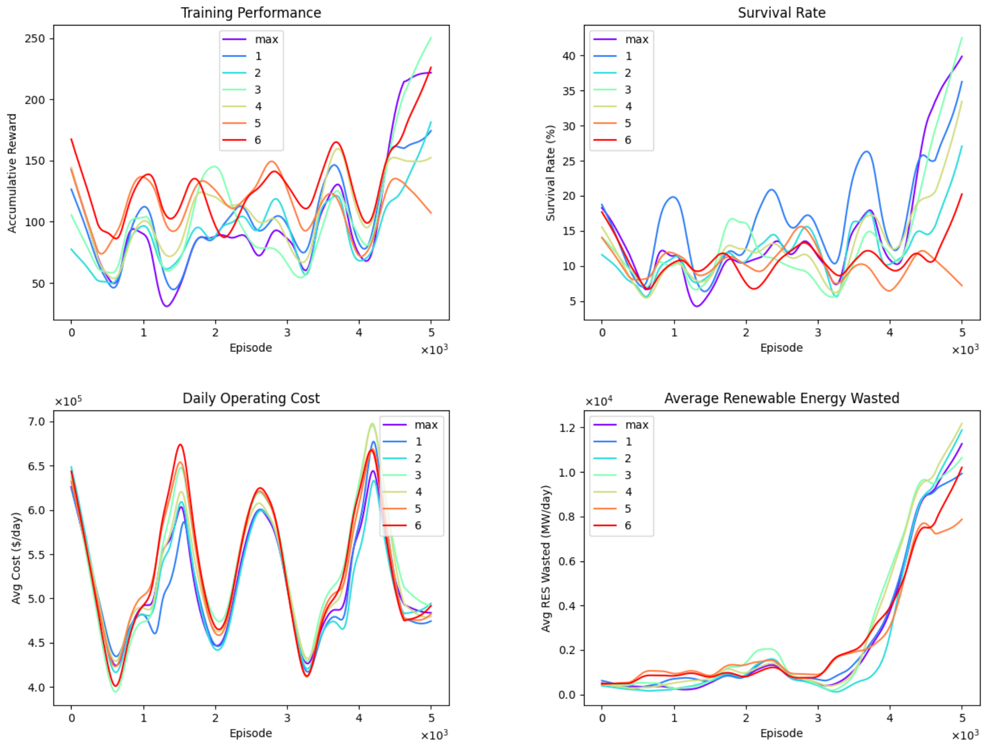

Figure A.9: Training Results of models with 1, 2 and 3 GAT Heads

A.9 GNNs vs. SAC Experiments

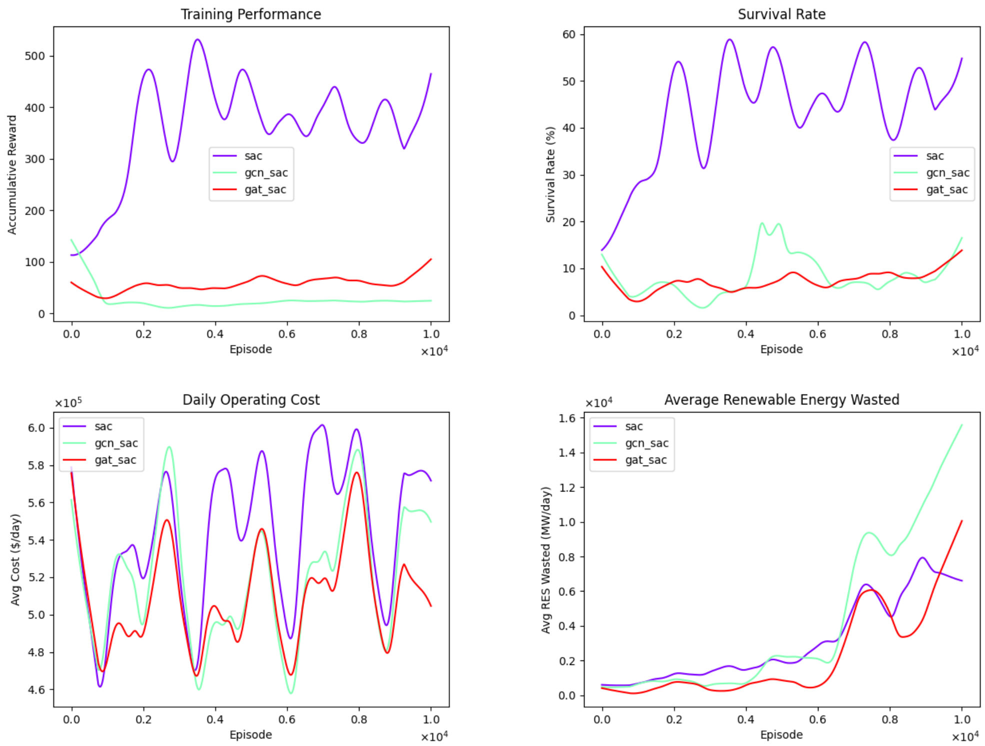

Figure A.10: Training Results of models with 1, 2 and 3 GAT Heads

Model	**Avg. Accumulative Reward**	**Avg. Length (Steps)**	**Avg Daily Operating Cost (€)**	**Avg. Renewables Wasted (MW/day)**	**Total Time (Seconds)**
sac	342.84	1118.24	601896.44	5443.64	741.23
gcn_sac	23.38	573.73	549592.13	9885.84	467.29
gat_sac	55.80	481.72	553388.36	6064.43	603.91

Table A.10: Validation Results of the Experiments concerning Limit Infeasible Curtail Actions.