Reinforcement Learning and Tree Search Methods for the Unit Commitment Problem

Patrick de Mars

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
of
University College London.

Bartlett School of Environment, Energy and Resources
University College London

20th October 2022

I, Patrick de Mars, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The unit commitment (UC) problem, which determines operating schedules of generation units to meet demand, is a fundamental task in power systems operation. Existing UC methods using mixed-integer linear programming are not well-suited to highly stochastic systems. Approaches which more rigorously account for uncertainty could yield large reductions in operating costs by reducing spinning reserve requirements; operating power stations at higher efficiencies; and integrating greater volumes of variable renewables. A promising approach to solving the UC problem is reinforcement learning (RL), a methodology for optimal decision-making which has been used to conquer long-standing grand challenges in artificial intelligence. This thesis explores the application of RL to the UC problem and addresses challenges including robustness under uncertainty; generalisability across multiple problem instances; and scaling to larger power systems than previously studied. To tackle these issues, we develop *guided tree search*, a novel methodology combining model-free RL and model-based planning. The UC problem is formalised as a Markov decision process and we develop an open-source environment based on real data from Great Britain's power system to train RL agents. In problems of up to 100 generators, guided tree search is shown to be competitive with deterministic UC methods, reducing operating costs by up to 1.4%. An advantage of RL is that the framework can be easily extended to incorporate considerations important to power systems operators such as robustness to generator failure, wind curtailment or carbon prices. When generator outages are considered, guided tree search saves over 2% in operating costs as compared with methods using conventional $N - x$ reserve criteria. The strategies adopted by guided tree search improve our understanding of the problems studied and demonstrate that RL is a rich methodology for solving the UC problem, offering practical value to system operators through more intelligent operation of complex and uncertain power systems.

Impact Statement

There is active interest from electricity network operators in applications of artificial intelligence methods including reinforcement learning (RL) to improve the stability and efficiency of power systems. These interests are in part motivated by the increasing complexity in power systems deriving from increasing renewables penetration, electrification of end-use sectors and increasing decentralisation of generation, among other trends. This thesis shows that RL methods can be applied to solve the unit commitment (UC) problem, a fundamental task in power systems operation which has received comparatively little attention from RL researchers. The guided tree search methods presented in this thesis show substantial improvements in solution quality as compared with current mathematical optimisation methods, which could benefit system operators through significant operating cost reductions if applied at scale. There are further benefits of our method in terms of system security, shown through experiments studying high levels of uncertainty from wind generation, demand and generator outages. These studies reflect the challenges faced by system operators in current and future power systems, which demand novel solution methods to ensure reliable and cost-effective supply of electricity.

Applied RL research requires problem-specific environments, the development of which is typically time-intensive and requires expert knowledge of the problem domain. The open-source UC environment developed for this research enables researchers from both artificial intelligence and power systems backgrounds to approach the UC problem, accelerating research in this area. This research has significance for RL research more broadly, as it bridges the gap between state-of-the-art methods and practical applications. It is widely recognised that significant challenges remain in developing RL methods that are applicable in the real world; this thesis addresses these problems in the context of power systems. In particular, UC is a challenging problem for existing RL methods due to its very large discrete action space. Guided tree search addresses this issue using a novel method to reduce the solution space, and could be applied in other fields sharing this characteristic where existing RL methods do not succeed such as vehicle routing, employee scheduling or portfolio optimisation.

With future advances in RL methods and due to the profound importance of UC for the effective operation of power systems, there will undoubtedly be further

research conducted in this area. This thesis provides the foundation for future work, showing the value of combining model-free and model-based methods; developing techniques for incorporating domain knowledge; and developing the software to facilitate rapid development of new solution methods.

Acknowledgements

Thank you to Aidan O'Sullivan, who has supported me since we first met during my bachelor's degree in 2016. This thesis would certainly not have been written if not for your backing. I would also like to thank my supervisors Ilkka Keppo and Paul Dodds, whose advice and perspective were valuable in determining the direction of this research. I would also like to thank Andreas Schäfer, who has been a great inspiration and mentor.

I am grateful to all those who have supported me since I started in 2017. In particular, I would like to thank Jonno Bourne, Connor Galbraith and Ayrton Bourn, who were brilliant colleagues at UCL and from whom I have learned so much. Thank you to my great friend Josh; you helped me see the bigger picture and patiently helped me through my more intricate problems over coffee at Fork. Thank you to all my family and friends who kept my feet on the ground. And finally, special thanks to Ellen, without whom this would have been so much harder.

Above all, this thesis is dedicated to my dad, who was consistently encouraging of my PhD but sadly did not get to see me finish.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Electricity is vital to modern societies, driving essential sectors of the global economy including communication, healthcare and finance. As countries transition towards net zero energy systems, electricity will play a key role as it can be decarbonised more easily than other energy carriers [6], leading to the electrification of further end-use sectors such as heat and transport [7]. The provision of secure, cost-effective and low-carbon electricity is therefore increasingly important for the prosperity of societies and the mitigation of climate change [8].

Power systems are the physical infrastructure connecting electricity generation with demand. A fundamental task for operators of the system, who must ensure the equal balance of supply and demand, is unit commitment (UC), determining the on/off schedules of a fleet of generators for a future period [2]. Improving UC solutions can have a significant impact on system operating costs by operating thermal generators at higher efficiencies, reducing requirements of spinning reserve and integrating larger volumes of renewable energy generation. The introduction of the current state-of-the-art in mixed-integer linear programming (MILP) methods is estimated to have saved over $1 billion annually in total operating costs in North America, through more efficient scheduling of generation [9]. In its simplest form and assuming perfect information, a UC problem with N generators and T discrete commitment periods can be described as the following cost minimisation problem:

$$\min \sum_{t=1}^{T} \sum_{i=1}^{N} C_i(t) \tag{1.1}$$

subject to

$$\sum_{i=1}^{N} p_i(t) = D(t) \qquad \forall t \in \{1..T\} \tag{1.2}$$

$$\boldsymbol{p}_i \in \Pi_i \qquad \forall i \in \{1..N\} \tag{1.3}$$

where $C_i(t)$ is the operating cost of generator i at time t; $p_i(t)$ is the power output of unit i at time t; $D(t)$ is the demand at time t; and Π_i is the region of operating levels that obey generator operating constraints. The optimisation therefore requires that supply of electricity is balanced with demand in Equation 1.2, while respecting constraints on generator operation in Equation 1.3.

The complexity of the UC problem stems from non-linear and non-convex generator cost functions $C_i(t)$ and inter-temporal generator constraints such as minimum up/down times, which make the UC problem NP-hard [10]. Furthermore, while the optimisation problem in Equations 1.1–1.3 assumes perfect foresight, in practice UC decisions must be made based on forecasts of demand, renewables generation and other power system variables. Due to the long time horizons - typically several hours or days - over which the UC problem is typically solved [11], forecasts used to inform UC decisions carry high levels of uncertainty. While the UC problem is primarily concerned with integer decision variables determining the on/off status of generators, most approaches to the UC problem simultaneously solve for the real-valued generator power outputs. This latter problem of determining the optimal dispatch of generators given a commitment decision is known as the economic dispatch (ED) problem, and is closely related to the UC problem. The ED problem is an essential task for short-term operational decision making such as real-time balancing or re-dispatch of generators in response to deviations in demand from forecasts. While the commitment of generators in Great Britain (GB) and other regions with similar market structures is not determined by a central operator, the cost-minimising UC problem given in Equations 1.1–1.3 remains the most widely-studied, archetypal formulation, and is studied throughout this thesis.

Methods for formulating and solving the UC problem have been studied for decades [12] and the current industry practice is to use deterministic MILP formulations [13]. The UC problem must typically be solved within minutes in order to exploit the most up-to-date forecasts and allow sufficient time for the system operator to conduct security analyses [11, 14]. As power systems transition towards net zero carbon emissions, growing volumes of variable renewable generation introduce additional uncertainty to the UC problem [15, 16]. This has motivated research into new solution methods which more rigorously account for uncertainty while remaining computationally tractable in short computing times [13, 17–20]. Deterministic formulations are not a natural framework for handling uncertainty, relying on reserve capacity that is determined by heuristic methods such as the widely-used

$N-1$ criterion protecting against the loss of the single largest infeed on the network. Research has shown that deterministic approaches give variable levels of system security [21] and probabilistic treatments of the UC problem using stochastic optimisation techniques are better-suited to systems with high levels of uncertainty, achieving lower operating costs and more consistent system reliability [13, 18, 22]. However, much larger computational requirements have prevented practical applications of stochastic UC [13, 17].

The results of studies using stochastic optimisation to solve the UC problem have shown that deterministic UC methods are sub-optimal, motivating research into alternative solution methods. This thesis is focused on the application of artificial intelligence (AI) methods to the UC problem. Driven by a growing abundance of computational resources, advanced optimisation algorithms and widespread digitalisation, AI algorithms are the state of the art in challenging domains such as image recognition [23], machine translation [24] and protein folding [25]. The sub-field of reinforcement learning (RL) is a framework for goal-directed decision-making under uncertainty which has surpassed the performance of existing methods for a number of complex control tasks including games-playing and robotics [26–28]. In RL, a decision-making agent learns by trial-and-error to maximise a numerical performance signal called the reward. Through repeated interactions with an environment representing the problem domain, the agent learns to improve its operational strategy, known as a policy [29].

Although RL has been recognised as a promising framework for solving the UC problem due to its ability to learn optimal policies in complex, uncertain environments [30–32], RL methods make up only a small fraction of the existing UC literature. An RL solution to the UC problem offers the possibility of achieving the solution quality of stochastic optimisation methods with a fraction of the computational cost at decision time by off-loading computation to a training period. Furthermore, combining deep learning with RL (known as deep RL) offers a powerful and flexible framework to improve decision-making by automatically extracting relevant features from large amounts of data such as weather forecasts, market trends and smart meter readings. Much of the existing research in RL for UC was conducted prior to significant breakthroughs in RL including deep RL [26], state-of-the-art model-based algorithms [27] and several policy optimisation techniques which have been shown to significantly improve training efficiency [33–36]. It is worthwhile revisiting the UC problem in light of recent advances in RL.

Currently, the dominant RL methods are model-free [28], where the optimal policy is estimated by interactions with the environment alone. Such methods have achieved state-of-the-art performance in several challenging sequential decision-making problems [26, 37, 38], and are capable of rapidly producing high quality solutions. However, the combinatorial nature of UC decisions means that the action

space contains 2^N unique actions for a system of N generators, posing a formidable challenge for existing model-free methods [39]. Other characteristics of the UC problem such as a high-dimensional state space, extreme penalties for lost load (blackouts) and long time dependencies make UC an extremely difficult problem for model-free RL. For these reasons, existing model-free RL research has been limited to UC problem instances in small power systems [40–43].

Model-based RL, which combines planning methods such as tree search with experience-driven learning, is a promising approach to solving the UC problem that has not received attention in the existing literature. Such methods combine the generalisability properties of model-free RL, training by trial-and-error in a simulated environment, with precise lookahead capabilities of tree search and are the state-of-the-art in challenging games-playing domains [27, 28, 44]. Notably, the model-based RL algorithm AlphaGo was used to beat the number one ranked player in the board game Go, a long-standing grand challenge of AI [44]. In the context of the UC problem, the ability to anticipate contingencies using model-based lookahead strategies is highly advantageous due to the crucial requirement of maintaining high levels of system security. Model-based methods can offer greater robustness in real-world contexts, and can also offer greater levels of explainability [45]. However, existing model-based RL methods are not well-suited to the UC problem, and new solution methods are required.

In this thesis we develop an RL framework for solving the UC problem that combines model-free and model-based methods. This research addresses several practical challenges that have limited previous applications of RL to the UC problem and demonstrates the superior solution quality of RL methods over conventional deterministic optimisation approaches using MILP. In Chapter 4, we tackle the scaling issues that have prevented the application of RL to larger UC problem instances by incorporating an RL-trained policy into traditional planning methods in *guided tree search*. This significantly improves the run time complexity with negligible impact on solution quality and enables the application of guided tree search to problem instances of 100 generators in Chapter 5. The variable run time of planning methods across problem instances is tackled by employing anytime methods, enabling solutions to be iteratively improved within a time budget. We address issues of trust in RL in Chapter 6 by examining more complex UC problem instances involving power generation outages. We show that RL provides greater levels of system security than deterministic mathematical programming methods and can provide system operators with a tool to develop novel and robust strategies for dispatching generators. This research makes significant contributions in the field of RL for power systems and provides a foundation for further research in this area. In the next section, we will summarise the key contributions of this thesis.

1.2 Contributions

The contributions of this thesis are summarised under five themes:

1. Power System Environment for the UC Problem
2. Guided Tree Search
3. Informed and Anytime Tree Search Methods
4. Large-Scale Applications
5. Adaptability of Guided Tree Search

Contribution 1: Power System Environment for the UC Problem

As we discuss in Section 2.2.1 and Section 2.3.1, research into the UC problem has been limited by a lack of benchmark problem instances, which limits the extent to which solution methods can be robustly compared with one another. In addition, a suitable open-source simulation environment for RL research into the UC problem is not available in the existing literature. A significant contribution of this thesis is the development of an open-source power systems simulation environment and deterministic UC benchmark solutions. The simulator enables RL methods to be applied to the UC problem and enables the UC problem to be studied by the wider RL research community. The development of accessible environments has led to significant research outputs in RL applied in other energy domains, including building control [46] using the CityLearn environment [47]; and electricity network operation [48] using the Grid2Op environment [49]. These environments are valuable not only in enabling the application of RL in energy domains, but also for studying the effectiveness of RL methods more generally and their ability to tackle complex real-world problems, in addition to widely-studied AI problems such as games-playing.

The RL environment developed in this thesis can be used to unify the evaluation of methods from across the optimisation literature reviewed in Chapter 2 as well as RL methods presented in this thesis. We show in Section 4.2.3 how UC solutions can be evaluated in terms of expected costs by Monte Carlo simulations with the RL environment, enabling fair comparison of mathematical programming with RL methods. The environment is based on historical wind generation and demand data from the GB power system to create realistic and varied UC problems. This is essential to determining the generalisability of solution methods across problem instances with different characteristics, and our experiments find that some UC problems are more challenging to solve with a given solution method. In Chapter 6, we further develop the environment to add a carbon price, wind curtailment and generator outages. These cover prominent challenges faced by system operators in current and future power systems [50–52].

Contribution 2: Guided Tree Search

To solve the UC problem, we develop guided tree search in Chapter 4, combining model-free RL with model-based planning. While most recent RL research has focused on model-free methods [28], the UC problem poses significant challenges for model-free methods due to the large, constrained, combinatorial action space; high-dimensional state space; long time dependencies; and the extreme nature of lost load events. The existing research into RL for the UC problem, reviewed in Section 2.5, has shown promising results, outperforming baseline solutions for small-scale problems [41] but RL has not previously been successfully applied to larger scale problems. We find in Section 4.5.3 that a model-free RL approach using a policy trained with PPO is not an effective approach to solving the UC problem; the agent is not able to maintain satisfactory levels of system security, resulting in high costs due to lost load. Despite vast improvements in model-free RL methods in recent years [34, 35, 53], applying these methods to the UC problem remains a significant challenge. Our research finds that employing model-based methods significantly improved quality of solution, and is essential to maintaining grid security with uncertain forecasts. In order to introduce model-based planning, tree search methods have been used successfully in previous research [27, 44, 54]. Traditional tree search methods without RL have been applied to the UC problem in [55] and shown to outperform benchmark solutions, but also face scaling issues due to the same problems of curses of dimensionality. Our experiments applying uniform-cost search (UCS) to UC problem instances in Section 4.6.2 corroborate the scaling difficulties of traditional tree search methods, with exponential time complexity in the number of generators.

While neither model-free RL nor model-based tree search methods alone are successful in solving the UC problem beyond small problem instances, combining both approaches in guided tree search is shown to be a powerful methodology that is competitive with state-of-the-art deterministic approaches. In Section 4.6.3 we show that Guided UCS is capable of outperforming MILP benchmarks for problems of up to 30 generators, significantly larger than studied elsewhere in the literature. Operating costs are found to be between 0.3–0.9% lower using Guided UCS than deterministic UC formulated using MILP. The improved scaling of Guided UCS and other guided tree search algorithms relative to conventional planning methods applied in [55] is achieved by exploiting an RL-trained policy as a guide to reduce the branching factor of the search tree. We show that run time is held roughly constant with increasing numbers of generators, while operating costs are similar to those produced by exhaustive tree search methods without RL. Guided tree search contributes to the growing literature combining model-free RL with tree search, most notably with AlphaGo [27, 44], and shows this approach is applicable to a real-world problem in power systems operation.

Contribution 3: Informed and Anytime Tree Search Methods

In Chapter 5, we apply more advanced tree search methods, using informed and anytime algorithms. The two novel algorithms developed in this chapter, Guided A* and Guided Iterative-Deepening A* (IDA*), are significantly more effective in solving the UC problem than the general-purpose algorithm, Guided UCS, used in Chapter 4. Using Guided A* search and a novel heuristic function based on priority list solution methods [1], run times are reduced by up to 94% as compared with Guided UCS, with negligible ($< 0.1\%$) impact on operating costs. These results demonstrate the value of domain expertise in designing solution methods for UC and other real-world problems. While a large proportion of RL literature has been applied in games-playing domains, applications in real-world contexts has progressed more slowly [39]. Our results show that combining domain expertise with state-of-the-art RL can improve solution methods for specific applications, accelerating the adoption of RL methods for practical benefit.

The UC problem is typically highly time-constrained, and must usually be solved within minutes [11]. The variable and unpredictable run times of fixed-depth tree search methods such as UCS and A* therefore pose practical problems for UC. As a result, we develop an anytime algorithm Guided IDA* to mitigate run time variability, constraining the run time to a fixed computational budget. We show that Guided IDA* achieves up to 1% reduction in operating costs for similar computational budgets as compared with Guided UCS, comparable to the cost savings shown by stochastic optimisation over deterministic methods [13, 18, 22]. Anytime methods are shown to be particularly well-suited to the UC problem, enabling more reliable generation of high-quality solutions as compared with fixed-depth tree search.

In Chapter 4 we show that the exponential time complexity of UCS can be overcome by using Guided UCS, employing RL to reduce the branching factor of the search tree. Chapter 5 shows that Guided UCS can be further improved for UC by modifying the algorithm to exploit properties of the problem in Guided A* and Guided IDA*. While many tree search methods are impractical for UC applications, this thesis shows that problem-specific modifications through RL and advanced search methods can enable tree search methods to be successfully applied, producing high quality solutions.

Contribution 4: Large-Scale Applications

While many real-world UC problem instances involve small numbers of generators, such as the problems solved by generating companies in self-dispatching power markets, scaling characteristics of UC solution methods are nevertheless important to assess applications to larger problems such as those solved by system operators. Guided IDA* is applied to a problem of 100 generators in Section 5.5 and found to produce operating costs that are competitive with deterministic MILP approaches

(0.1% lower using Guided IDA*). The only RL-based study of a similar scale studied a 99-generator problem, but made significant simplifications to the problem formulation by preventing intra-day commitment changes, making it an unrealistic point of comparison and limiting solution quality [56]. In addition, no comparison was made with state-of-the-art methods. The 100-generator experiment in Chapter 5 is therefore the largest in the existing literature by number of generators, and the first to show that RL is a viable methodology for solving the UC problem at scale.

Contribution 5: Adaptability of Guided Tree Search

A significant advantage of RL over mathematical optimisation methods is the ability to learn fundamentals of the given problem *tabula rasa*. This has been shown by general-purpose games-playing algorithms such as MuZero, which achieved superhuman levels of performance in Go, Chess and Shogi, with no modification of the solution method. To demonstrate the generalisability of guided tree search to different problem instances and variants of the UC problem, in Chapter 6 we develop two variations of the power system environment. The first includes curtailment decisions and carbon pricing; the second includes generator outages. We show that RL can be simply applied to solve variants of the UC problem by changing the underlying dynamics of the environment and retraining the agent. By contrast, mathematical programming methods may require significant expertise and development time to develop efficient representations of the problem variant. In some instances, this may include substantive adjustments to the problem itself, such as convexifying or linearising elements of the problem.

In the first case study of Chapter 6, we introduce a carbon price to the reward function in addition to a wind curtailment action. Carbon pricing is already an important policy mechanism to promote low-carbon electricity [57], and curtailment presents both a challenge and opportunity for system operators to effectively manage large penetrations of variable renewable energy [51]. We show that the RL framework enables different operating behaviours to be incentivised by modification of the carbon price. Operating patterns are responsive to changes in the carbon price with gas displacing coal as base-load generation as the carbon price is increased. This shows that using Guided IDA* system operators can dynamically adjust the objective function in response to current system demands, with more general support than is possible using MILP which requires a linear objective function. Guided IDA* also learns to curtail wind generation to manage large swings in demand and wind generation, improving system robustness. These strategies enable the identification of challenging or uncertain decision periods, providing insight into the nature of the UC problem instances studied. By comparison with MILP methods, introducing the additional curtailment action is straightforward, and does not require reformulation of the algorithm.

Outages of generation and other transmission assets can have catastrophic

impacts on power systems and large economic consequences; the 2003 North American blackout led to the loss of power for 50 million people with an estimated total cost of $6 billion [58]. The traditional $N-1$ criterion protects system security against the largest loss of infeed, but does not necessarily account for coincident outages, which was the cause of the recent blackout in England impacting over 1 million customers [59]. Achieving adequate levels of system security while maintaining lower system operating costs and avoiding high levels of carbon intensive spinning reserve requirements requires UC methods which more robustly account for the joint distribution of outages. The generator outages case study shows that heuristic reserve requirements based on $N-x$ reserve criteria are sub-optimal for the problem instances examined. Guided IDA* learns economic and robust reserve margins during training, taking into account a much larger number of uncertain parameters than in previous experiments without outages. The capacity of Guided IDA* to learn effective reserve margins in this case is indicative of its value in operating increasingly complex power systems with uncertainties stemming from multiple sources. In the context of increasing penetration of geographically-distributed variable renewable energy, large numbers of uncertain parameters are required in order to fully capture effects on the transmission network and the correlated forecast errors. We show in this thesis that RL can easily incorporate deep learning to learn such inter-dependencies and develop suitable strategies for managing this uncertainty. Our results find that irrespective of the problem variant studied, guided tree search is an effective solution method for UC.

1.3 Thesis Structure

This thesis is structured in seven chapters. Chapters 4, 5 and 6 are results chapters containing the original contributions of this thesis. Chapters 2, 3 and 7 are Literature Review, Methodology and Conclusion chapters, respectively.

In **Chapter 2**, we conduct a literature review of the state-of-the-art in UC research. We review in detail deterministic methods, which are the most widely-used approaches to solving the UC problem and are used to benchmark the guided tree search methods in later chapters. In addition, we review scenario-based stochastic optimisation and robust optimisation methods for the UC problem, which more rigorously account of uncertainties and have been shown to outperform deterministic methods, albeit at much higher computational cost. Finally, we review existing research using RL and tree search to solve the UC problem.

Chapter 3 provides a background to the RL and tree search methods which form the basis of our methodology. We focus particularly on policy gradient RL methods, which are best suited to the UC problem for their ability to handle high-

dimensional action spaces and to learn stochastic policies. We describe three tree search algorithms: uniform-cost search, A* search and iterative-deepening, which we apply in the guided tree search framework in Chapter 4 and Chapter 5. In addition, we cover mathematical optimisation methods for power systems, which are used to develop the power system simulation environment in Chapter 4 and provide benchmark solutions. In particular, we provide a background to MILP and methods for solving the related economic dispatch problem, which is an integral component of the simulation environment.

Chapter 4 is the first of three results chapters. In this chapter, we present the power systems environment developed for this research, which is used to conduct experiments solving UC problems with MILP and guided tree search methods. We use data from the GB power system to create several years' worth of training problems and a generative model for simulating forecast errors. We formulate the UC problem with stochastic demand and wind generation as a Markov decision process, suitable for RL methods. We then apply the traditional tree search algorithm uniform-cost search (UCS) [60] to solve small UC problem instances, showing competitive solution quality as compared with MILP approaches but practical limitations due to exponential run time complexity in the number of generators. To enable tree search methods to be applied at scale, we present *guided expansion*; the key innovation of guided tree search. Using guided expansion, an RL-trained policy can be used to reduce the branching factor of a search tree so that it can be solved with traditional methods. We apply this approach in **Guided UCS**, the first of three guided tree search algorithms, and show that run time remains roughly constant with increasing numbers of generators, while achieving similar operating costs to traditional UCS. Compared with deterministic MILP benchmarks for systems of 10, 20 and 30 generators, Guided UCS consistently achieves lower operating costs. The results of this chapter show that guided tree search is an economic and scalable approach to solving the UC problem.

The field of tree search encompasses multiple classes of algorithms with different characteristics which may be better suited to particular problem domains. **Chapter 5** extends the guided tree search framework to informed search and anytime search algorithms [61], and compares these methods to Guided UCS. Three UC-specific heuristics are developed which are used in the informed search algorithm A* search [62] to achieve greater search efficiency. Using the heuristics, **Guided A*** search is shown to be up to an order of magnitude faster than Guided UCS with no significant impact on solution quality. We then present **Guided IDA***, an anytime algorithm which can be terminated when a time budget is spent. The anytime property assures that computational resources are fully exploited, resulting in deeper search depths and improved solution quality in practice. These improvements, culminating in Guided IDA*, enable us to solve UC problems for a larger power system of 100 generators, achieving similar operating costs to deterministic MILP benchmarks. This is the

largest simulation study applying RL and/or tree search to solve the UC problem in the existing literature.

In **Chapter 6**, the power system environment is modified in two case studies that demonstrate the flexibility of the guided tree search framework across heterogeneous power system contexts. In the first case study, we introduce a **curtailment action** and **carbon price**, studying the adaptability of Guided IDA* to more heterogeneous actions and the impact of reward shaping. Guided IDA* responds flexibly to changes in the problem environment, adjusting curtailment rates and utilisation of different fuel types to manage carbon costs. In the second case study, we study a system with **generator outages**, and compare Guided IDA* with MILP using typical $N-x$ security criteria, which are widely-used to handle such contingencies. We show that Guided IDA* adaptively allocates reserves and achieves lower operating costs and better security of supply overall. In both case studies, Guided IDA* exhibits novel operational strategies that uncover properties of the problem itself and demonstrate the value of guided tree search as a decision support tool for system operators.

Chapter 7 discusses the results, contributions and limitations of the thesis as a whole and proposes further research topics.

Chapter 2

Literature Review

2.1 Introduction

The UC problem is a large-scale, stochastic, mixed-integer optimisation problem which, even when reformulated as a deterministic problem, is NP-hard [10, 63]. Due to the large and growing size of power systems globally, there are significant economic and environmental incentives for improving UC solution methods, which has motivated a large body of research. The UC problem has been formulated as a deterministic [64], stochastic [65] and robust [13] optimisation problem; formulations that use different approaches to managing uncertainty. Furthermore, a large number of solution methods have been applied, incorporating domain-specific heuristic approaches [66], mathematical optimisation [67, 68] and metaheuristics [5, 69]. Facilitated by improvements in commercial solvers such as CPLEX and Gurobi, industry-standard approaches use the deterministic formulation, solved with mixed-integer linear programming (MILP) techniques [64]. This approach was first adopted by the PJM interconnection in 2005 [70] and is estimated to have been responsible for operating cost savings of more than $1 billion per year in North American markets compared with previous methods based on Lagrangian relaxation [9]. However, the deterministic formulation has been shown to yield economically sub-optimal solutions as it is outperformed by stochastic optimisation methods which more rigorously account for uncertainties [18, 20, 22]. Stochastic formulations are substantially more expensive to solve than deterministic ones [17, 71] and cannot be used in most practical contexts as UC problems must typically be solved within the order of minutes to satisfy market and operational constraints [11]. As a result, there are significant incentives for further research into novel UC solution methods and formulations that can outperform current deterministic approaches within practical computational budgets.

The purpose of this chapter is to present existing research into the UC problem, covering the principle formulations and solution methods and the current state of the art. The literature review will be organised around the most common UC formulations. Section 2.2 describes deterministic formulations of the problem, which are the most

widely used in practical applications [13]. Section 2.3 reviews the stochastic UC literature, which more rigorously accounts for uncertainty by minimising expected operating costs over scenarios. Section 2.4 reviews the robust UC literature, which aims to minimise the worst-case operating costs. Finally, Section 2.5 reviews the small body of literature that has used reinforcement learning (RL) to solve the UC problem, which is the topic of this thesis.

2.2 Deterministic Unit Commitment

The most widely-used approach to UC formulates the problem deterministically, minimising operating costs under point forecasts of demand and generation [13]. Uncertainties are managed by enforcing a reserve constraint which is determined separately, either by statistical methods or by heuristics. Based on [64], for a power system with N generators and T decision periods the deterministic UC problem can be formulated as:

$$\min \sum_{i=1}^{N} \sum_{t=1}^{T} C_i(t) \tag{2.1}$$

subject to

$$\sum_{i=1}^{N} p_i(t) = D(t) \qquad \forall t \in \{1..T\} \tag{2.2}$$

$$\sum_{i=1}^{N} \bar{p}_i(t) \geq D(t) + R(t) \qquad \forall t \in \{1..T\} \tag{2.3}$$

$$\boldsymbol{p}_i \in \Pi_i \qquad \forall i \in \{1..N\} \tag{2.4}$$

Equation 2.1 is the objective function, which is to minimise the sum of generator operating costs $C_i(t)$ (including fuel and startup costs) over all periods t. Equation 2.2 gives the load balance constraint, requiring that the sum of generator power outputs $p_i(t)$ equals the forecast demand $D(t)$. Equation 2.3 is the reserve constraint, ensuring that the sum of available capacities $\bar{p}_i$ exceeds the sum of forecast demand $D(t)$ and the reserve constraint $R(t)$. Equation 2.4 requires that generators operate within Π_i, the region of feasible operating levels for generator i. The operating constraints include minimum and maximum operating limits as well as inter-temporal constraints such as minimum up/down times, limiting the frequency of startups and shutdowns.

The deterministic UC problem is NP-hard [10], and even finding sub-optimal solutions is a challenging and expensive optimisation problem. The fuel costs included in $C_i(t)$ are usually modelled with quadratic cost curves [2], which require

linear approximations for many solution methods such as MILP [72]. In addition, startup costs are often modelled as a function of time using a step [5] or exponential function [64]. The presence of binary variables denoting commitment decisions makes the problem non-convex. Brute force solution methods are not scalable to large power systems [12], and numerous methods have been proposed including heuristic-based approaches, conventional mathematical optimisation methods and metaheuristics which are discussed in Section 2.2.2. Performance of different solution methods are evaluated on benchmark problems, defining generator cost functions and operating constraints, as well as demand and reserve profiles. The following subsection describes existing benchmark problems for the deterministic UC problem.

2.2.1 Benchmark Problems

UC research uses test problems to evaluate the performance of solution methods. These problems vary substantially from small-scale systems of < 10 generators [40], to large problems representing regional-scale transmission grids such as the North American mid-continent network [14]. Additional features include pumped hydropower [66, 73, 74] and consideration of additional problem features such as transmission constraints [75] or ramping constraints [76, 77]. Early UC research was generally conducted on proprietary data [66, 68, 78], making it difficult to directly compare solution methods. Benchmark problems have since been developed which have been used more widely across the UC literature [5, 11, 79].

The benchmark problem proposed by Kazarlis et al. [5], specifying cost functions and constraints for 10 generators and a 24-hour demand profile, has been extensively used in the literature [1, 64, 80–88]. The reserves are fixed to be 10% of demand. By duplicating generators, the Kazarlis problem has been scaled to up to 1000 generators [88] with proportional scaling of the demand and reserve profiles. It has also been amended in [79] which duplicates generators in varying combinations to generate a more diverse set of problem instances. Selected solutions to the 100-generator Kazarlis benchmark are shown in Table 2.1, demonstrating the wide range of methods which have achieved better solution quality (i.e. lower operating costs) and lower run times as compared with the original genetic algorithm (GA) solution [5]. A limitation of the Kazarlis benchmark is that it includes a single 24-hour profile for demand and reserve, making it difficult to evaluate the generalisability of solution methods across problems. Algorithms can be tuned aggressively to achieve very low operating costs on a single problem instance, and it is difficult to compare solution methods by comparing solution quality on the Kazarlis benchmark alone.

Knueven et al. [11] propose three benchmark systems specifically for comparing deterministic mixed-integer linear programming (MILP) formulations which aim to address this research limitation by creating a large number of diverse problem instances. The systems are based on data from the California ISO (CAISO), Fed-

Method	Year	Cost ratio (%)	Time ratio (%)	Reference
GA	1996	100.00	100.00	[5]
EA	1999	99.94	38.90	[93]
LR	2000	99.75	25.71	[80]
GA	2002	99.98	22.54	[94]
PL	2003	99.66	0.41	[1]
LR	2004	99.93	4.64	[95]
EA	2004	99.65	7.63	[96]
LR	2004	99.61	2.19	[82]
SA	2006	99.83	4.42	[97]
PL	2006	99.79	2.38	[98]
SA	2006	99.71	1.31	[99]
MILP	2006	99.60	0.78	[64]
GA	2007	99.77	Not reported	[100]
EA	2008	99.78	12.84	[76]
DP	2009	99.69	Not reported	[101]
EA	2009	99.68	0.51	[102]
PSO	2011	99.66	1.88	[103]
EA	2011	99.57	1.86	[104]
PSO	2012	99.53	1.33	[105]
MILP	2013	99.74	99.31	[77]
EA	2015	99.60	2.52	[106]
PL	2015	99.51	0.05	[88]
PSO	2016	99.54	4.20	[107]
GA	2018	99.54	0.14	[108]
EA	2018	99.53	Not reported	[109]

Table 2.1: Selected solutions to 100-generator Kazarlis benchmark problem [5] for deterministic UC. Included methods are genetic algorithms (GA), evolutionary algorithms (EA), Lagrangian relaxation (LR), priority list (PL), particle swarm optimisation (PSO), dynamic programming (DP), simulated annealing (SA), mixed-integer linear programming (MILP). Ratios compare operating costs and run time with the original GA solution in [5].

eral Energy Regulatory Commission (FERC) [89] and the IEEE RTS-GMLC [90] test system, each with multiple demand and renewables generation profiles. This benchmark system has been used to systematically compare MILP-based approaches to the UC problem in [11, 91, 92]. While the benchmark problems proposed in [11] are well-suited to comparing deterministic formulations of the UC problem, the literature lacks more general benchmarks for comparing non-deterministic solution methods on a variety of problem instances. In Section 4.2 we address this research gap, introducing a simulation environment that can be used to evaluate solution methods on a diverse set of UC problems.

Type	Method	Advantages	Disadvantages
Heuristic	Priority list	Low computational cost and scalable to large power systems	Strong reliance on problem-specific heuristics; typically poor solution quality; no optimality gap
Mathematical optimisation	Dynamic programming	Guaranteed to converge to an optimal and feasible solution	Exponential time complexity in the number of generators means heuristics are required to achieve tractability
	Lagrangian relaxation	Good solution quality; scalable to large power systems; measurable duality gap	Not guaranteed to produce a feasible solution without heuristics
	Branch and bound	Excellent solution quality; generally quick to execute using commercial solvers; measurable optimality gap	Can be slow to find a feasible solution under certain conditions or parameters
Metaheuristic	Genetic algorithms Simulated annealing Particle swarm optimisation	Potential for excellent solution quality; can be quick to execute	Generally requires extensive parameter tuning; no convergence guarantee or optimality gap

Table 2.2: Summary of reviewed solution methods for the deterministic UC problem.

2.2.2 Solution Methods

The computational challenges and NP-hardness [10] of the UC problem has motivated a large body of research proposing solution methods that achieve good solution quality in short computing times. In this section, we will cover the most widely-studied heuristic, mathematical optimisation and metaheuristic methods for solving the UC problem, which are summarised in Table 2.2. Direct comparison between solution methods is challenging, in part due to the limitations of existing benchmarks that were discussed in Section 2.2.1. This section will discuss the properties of each method and make side-by-side comparisons based on benchmark problems where appropriate.

Priority List

Priority list (PL) methods are among the most conceptually simple and computationally inexpensive approaches to solving the UC problem and have been used extensively [1,66,88,98,110–115]. A summary of the main advantages and drawbacks of PL methods is given in Table 2.2. PL methods were among the first computational methods for UC, superseding manual approaches [66,110–112]. PL methods order generators in terms of capacity, startup costs, or fuel costs [1,88,116], and for each period commit generators in decreasing order of preference until the reserve constraint in Equation 2.3 is met. Figure 2.1 shows an example UC schedule produced by a priority list method. By committing units independently for each period, this

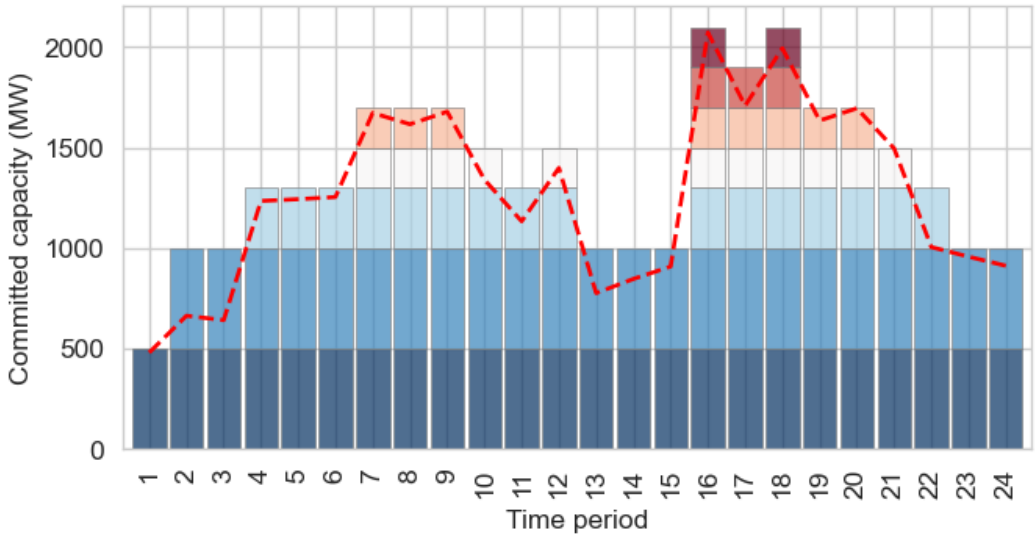


Figure 2.1: Example priority list unit commitment schedule. Generators are ordered in a priority list of decreasing preference (e.g. by fuel cost) and committed in this order until demand plus a reserve constraint (red line) is met. Inter-temporal constraints such as minimum up/down times are not guaranteed to be met with this algorithm and must typically be fixed using heuristic methods [1].

simple algorithm neglects inter-temporal constraints such as minimum up/down time constraints, and hence solutions are not guaranteed to be feasible, i.e. satisfying Equation 2.4. Therefore, the initial solutions are typically 'fixed' by further heuristics in order to produce a feasible schedule [66]. As a result, a large number of PL variants have been proposed combining different heuristics and optimisation techniques to improve the run time and solution quality [1, 88, 98, 113]. The most recent PL-based approach in the surveyed literature combines MILP to generate an initial solution and neighbourhood search to fix the inter-temporal constraints [117]. This is the best performing solution on the Kazarlis benchmark problem, surveyed in Table 2.1, achieving 0.49% lower operating costs as compared with the original GA approach with a 2000-fold decrease in run time. The PL method proposed in [1] also achieves lower operating costs than the original GA solution with a 240-fold speed-up.

Historically, PL methods replaced manual commitment methods and were reported to have achieved roughly 1% annual operating cost savings in an industrial-scale application to the Connecticut power system [66]. Due to their low computational cost, PL methods are highly scalable to larger power systems, as evidenced by applications to systems of ≥ 100 generators from 1971 [66] and 1988 [115] which are significantly larger than most contemporary research. PL methods rely heavily on problem-specific heuristics to improve solution quality. As a result, while the PL solutions reported in Table 2.1 are among the best performing, extensive algorithmic tuning can be used to inflate performance of PL methods for specific problem instances and the results do not indicate generalisability to other power systems or

demand profiles. In general, PL methods are considered to have low solution quality relative to more general mathematical optimisation approaches [77, 118]. In addition, they lack a measurable optimality gap for measuring solution quality (e.g. relative to an estimated lower bound of operating costs) that is offered by mathematical optimisation methods such as branch and bound and Lagrangian relaxation. These reasons have limited practical applications of PL methods since their use in early power systems [66, 110–112].

Dynamic Programming

Dynamic programming (DP) was among the first mathematical optimisation methods employed to solve the UC problem [69, 78, 87, 101, 119–122]. Unlike heuristic-based methods such as PL, traditional DP benefits from guaranteeing an optimal solution. However, the time complexity of DP is generally exponential in the number of generators and DP implementations also typically use heuristics in practice to reduce the solution space and improve scalability. For instance, studying a system of 14 generators, [78] ignores inter-temporal constraints and uses DP to calculate the lowest cost commitment for demand levels between minimum and maximum capacity of the generation mix. This is calculated offline and used to look-up commitment decisions for levels of forecast demand. Another heuristic approach adopted in [69] is to group generators based on their properties, and commit groups of generators using DP rather than individual units. Using this method, a problem of 33 generators is reduced to a tractable problem considering 7 subsets. In addition, a common approach only considers states which commit generators in PL order - known as DP sequential combinations (DP-SC) [120]. In this method, a system of N generators will only have $N+1$ states: all generators off, and the N combinations which consider the first $\{1, 2, \ldots, N\}$ generators committed in PL order. Further problem-specific decision rules are used in [121] to achieve tractability for a system of 26 generators.

Studies which directly compare PL and DP show conflicting results. To the best of our knowledge, DP has not been used to solve the Kazarlis benchmark. On a proprietary test case of 33 generators, [69] finds a 0.36% improvement in operating costs of DP over PL. [113] and [115] find 2% and 4% improvements of PL over DP, respectively. Three DP algorithms are compared with a simple PL algorithm in [120] for six test problems of up to 17 generators. Savings of between 0.20–0.79% are found using the DP-SC approach described above as compared with PL, but with 20–50 times higher run times. Due to the strong reliance of both PL and DP on heuristics, the precise implementation of both methods evidently has a significant impact on solution quality. Both PL [66] and DP [69] were employed in early industrial contexts.

Lagrangian Relaxation

Motivated by the lack of guarantees on solution quality of PL and the potentially large run times of DP, Lagrangain relaxation (LR) methods were developed for

the UC problem [68, 80, 82, 95, 123–128] and were widely used by utilities during the 1990s and 2000s [12]. Compared with DP and PL, LR has better solution quality, as measured by operating costs, than PL methods and is more scalable to larger power systems than DP [113]. LR is a mathematical optimisation method for finding approximate solutions to constrained optimisation problems. LR 'relaxes' constraints by including them as terms in the objective function, penalised with Lagrange multipliers. This relaxed ('dual') problem is easier to solve than the original ('primal') problem. Iterative methods are used to update the Lagrange multipliers and converge to a solution which is close to the optimum of the primal problem [129]. Applying LR to the UC problem, the most common approaches decompose the dual problem into individual generator sub-problems which are solved with DP [68, 123–126]. However, this method is not guaranteed to produce a feasible solution to the primal problem, and heuristics are generally used to satisfy the violated constraints [125, 126]. Several papers have used further metaheuristic approaches such as particle swarm optimisation (PSO) [95] or genetic algorithms (GA) [80] to update the Lagrange multipliers.

Improved solution quality of LR over PL methods are shown in early research (1988) [130], which finds large cost reductions of 10–33% using LR. In applications to the 100-generator Kazarlis benchmark summarised in Table 2.1 LR is applied in [80, 82, 95] and achieves lower operating costs in run times that are between 2–24% of the original GA run time. In addition to higher solution quality, LR exhibits two practical advantages over DP and PL methods. First, it is more scalable in the number of generators without strong reliance on heuristics. Due to the decomposition into generator sub-problems, LR methods have roughly linear time complexity in the number of generators [126]. A proprietary, 172-unit test system for Electricite de France (EDF) was studied in [68] and 100-unit systems in [125, 130] which are among the largest studies of their era. The second advantage of LR is that unlike PL and DP methods, LR benefits from having a known 'duality gap' giving the difference between the feasible solution cost and a lower bound for the optimal solution cost. The duality gap is measured as the difference between the dual solution cost and the primal solution cost [68]. This can be used as a stopping criterion, and provides assurances for system operators regarding the solution quality, despite the reliance of LR on heuristics. For these reasons, LR was the dominant method for practical UC applications in the 1990s and 2000s [12, 64].

Metaheuristics

Several metaheuristic methods have been proposed to solve the UC problem, including genetic algorithms (GA) [5, 94, 100, 131, 132]; other evolutionary algorithms [76, 93, 96, 102, 104, 106, 109]; particle swarm optimisation (PSO) [83, 86, 103, 105, 107, 133, 134]; simulated annealing (SA) [97, 99, 135–137]. These methods use probabilistic techniques and rules to iteratively improve a solution or population of solutions. Metaheuristics

do not generally guarantee an optimal solution, but may achieve good sub-optimal solutions in short run times.

The survey of the 100-generator Kazarlis benchmark problem in Table 2.1 shows several metaheuristic approaches are among the best performing techniques both in terms of operating cost and run time [99, 102, 108]. Many of these metaheuristic approaches include problem-specific rules to improve the convergence time and solution quality [83, 93, 96, 99, 108]. As a result, good performance on the Kazarlis benchmark does not necessarily imply generalisability across other problem instances, as metaheuristic approaches can be aggressively tuned to improve solution quality and reduce solution time. This reliance on expert rules and often extensive parameter tuning have prevented practical applications of metaheuristics [127].

Mixed-Integer Linear Programming

Currently, the most prevalent solution methods use a mixed-integer linear programming (MILP) formulation which is solved by branch-and-bound techniques (described in detail in Section 3.7.2) [11, 14, 64, 67, 73, 77, 138–140]. Advanced optimisation techniques such as pre-solve and cutting planes are implemented in commercial solvers such as CPLEX and Gurobi, enabling large-scale deterministic UC problems to be solved even with limited hardware [64]. Like LR, branch-and-bound produces an optimality gap measuring the proximity of the current solution costs to a lower bound.

Only two of the surveyed solutions to the 100-generator Kazarlis benchmark in Table 2.1 used MILP [64, 77]. Although both solutions achieve similar operating costs, [64] has roughly 130-fold lower run time than [77]. The variation in run time of MILP approaches to the UC problem is influenced by the tightness (size of the feasible solution region) and compactness (the problem size, such as number of constraints and decision variables) of the formulation [140], as well as the efficiency of the solver. Designing tight and compact MILP formulations for the UC problem has been studied extensively [11, 64, 79, 140, 141]. 41 MILP formulations are compared in [11], using 68 problem instances from the 3 large-scale benchmarks described in Section 2.2.1 to systematically evaluate the efficiency of MILP formulations. This study finds large variability in solution times, with relative solution times of different formulations also varying across problem instances. Machine learning has recently been used to improve the efficiency of branch-and-bound for MILP [91, 142]. This approach is applied to solve the DUC problem in [91], and is found to reduce the optimality gap by a factor of 2 for similar computing times as compared with a conventional solver.

As the dominant solution method, MILP has been widely used to solve variations on the deterministic UC problem and in planning studies. Problem variants have considered combined heat and power (CHP) [143], storage assets [73] and UC in a microgrid setting [144]. These studies have focused on deriving efficient MILP

formulations to include additional decision variables and constraints. Formulating UC problem variants for MILP is often challenging due to the non-convexities and non-linearities arising from integer decision variables and inter-temporal constraints such as minimum up/down times and ramping constraints [143]. Due to its accuracy and scalability to large power systems, MILP is also appropriate for simulation studies on large transmission networks and has been used to estimate startup costs [145] and curtailment rates [146] in future power systems, as well as for power system planning models [147].

The efficiency improvements of MILP solvers and solution techniques as well as tighter and more compact deterministic MILP formulations have led to this becoming the most widely used approach for solving the UC problem in practical applications [13]. The experiments in [11] which consider up to 939 generators in the FERC benchmark [89] and studies on the MISO grid of approximately 1400 generators [14] show that deterministic MILP is scalable to large power systems. In addition, the optimality gap assures high quality solutions are achieved. However, the deterministic UC problem formulation is limited by its consideration of the point forecasts alone and use of heuristic reserve constraints to manage uncertainties. In the following section, we review stochastic formulations of the UC problem. These formulations more rigorously account for uncertainties and achieve lower expected operating costs in practice, at the expense of higher computational requirements.

2.3 Scenario-Based Stochastic Unit Commitment

In contrast with the deterministic UC formulation which manages uncertainties using reserve constraints, scenario-based stochastic formulations have been adopted where the objective function minimises expected cost over a finite number of scenarios [65]. Using the same notation as for the deterministic UC problem in Equations 2.1–2.4, the stochastic UC problem is formulated as:

$$\min \sum_{s \in \mathcal{S}} P(s) \sum_{i=1}^{N} \sum_{t=1}^{T} C_i(t, s) \tag{2.5}$$

subject to

$$\sum_{i=1}^{N} p_i(t,s) = D(t,s) \qquad \forall t \in \{1..T\}, \forall s \in \mathcal{S} \tag{2.6}$$

$$\boldsymbol{p}_i(s) \in \Pi_i \qquad \forall i \in \{1..N\}, \forall s \in \mathcal{S} \tag{2.7}$$

$$\tag{2.8}$$

Compared with the deterministic UC formulation defined in Equations 2.4–2.4, the stochastic formulation introduces $\mathcal{S}$, a finite set of scenarios representing realisations of the uncertain processes such as demand and renewables generation. Scenarios are typically represented in a tree, where each node represents the realisation of uncertainties at a given decision period, branching into one or more child nodes at the next decision period. The objective function aims to minimise the weighted sum of operating costs under each scenario, with scenarios weighted by probability $P(s)$. Commitment decisions must remain fixed across scenarios, while the generator setpoints p_i can differ between scenarios to satisfy the load balance constraint in Equation 2.6. This formulation is therefore a two-stage stochastic program, where the first-stage commitment decisions are scenario independent and the second-stage (also known as real-time) dispatch decisions are made on the basis of uncertain outcomes [148].

The stochastic formulation of the UC problem omits the reserve constraint in Equation 2.3. Instead, reserve is implicitly allocated as solutions must satisfy the load balance constraints under all scenarios. If the set of scenarios $\mathcal{S}$ contains sufficiently extreme scenarios, solving the stochastic UC problem yields robust solutions. Like the deterministic UC problem, several extended formulations have been proposed, including those discussed in Section 2.3.4 which add explicit reserve constraints to improve robustness where including large numbers of scenarios in the problem formulation is not tractable. Further formulations have incorporated hydropower scheduling [149–151] and the consideration of AC network constraints [152]; however, in this section we will focus on research studying the commitment of thermal power stations without network constraints.

2.3.1 Benchmark Problems

Unlike the deterministic UC benchmarks described in Section 2.2.1, the stochastic UC problem lacks established problems that allow for solution methods to be systematically compared. Among the popular test systems is the IEEE RTS benchmark systems generators which has been studied with varying numbers of generators [22, 152–154]. In addition, variations on the IEEE 118 network have been used in [20, 155–157], the CAISO system in [17] and Irish system in [18, 158, 159]. However, due to the random nature of the stochastic UC problem, even the results of simulation studies conducted

on the same system cannot be easily compared. It is noted in [22] that stochastic UC simulation studies usually report costs under the 'in-sample' scenarios $\mathcal{S}$ which were included in the formulation. Differences in random seeds as well as methods for scenario generation and reduction [154] mean that the set $\mathcal{S}$ is typically unique for each stochastic formulation. Since the costs reported consider the 'in-sample' scenarios $\mathcal{S}$, results are generally not comparable.

Instead of reporting costs under the in-sample scenarios $\mathcal{S}$, Monte Carlo methods for evaluating solution quality have been used. The second-stage costs are calculated under a large number of out-of-sample scenarios, and expected costs are estimated with the empirical mean [17,22,154,160]. With a sufficiently large number of scenarios, results from different studies can be compared when using this approach provided the distributions of random processes are the same. Benchmarks have not so far been developed for systematically comparing solution methods using Monte Carlo approaches.

2.3.2 Rolling Horizon Optimisation

In this literature review and throughout this thesis, we focus on solutions to the day-ahead UC problem. However, due to large computational costs associated with stochastic optimisation, a significant proportion of the stochastic UC literature has used a rolling horizon approach, where UC decisions are made for a limited time horizon, such as a 3, 6 or 12 hours, then rescheduled periodically [18,158,159,161,162]. This greatly improves the computational tractability of stochastic UC, as the scenario tree of each stochastic program generally grows exponentially with the planning period.

Several studies have used the WILMAR planning tool [19] to run rolling horizon stochastic UC models for the Irish power system [18,158,159,163]. These studies have mainly focused on simulating future power systems, for purposes such as assessing the sizing of reserve requirements [159] and the impact of increased penetration of renewables [158,163]. In [18], the authors use a rolling horizon model to assess the benefits of stochastic optimisation with a rolling horizon in the Irish power system, in terms of operating costs and security supply, with comparison made against deterministic UC as well as optimisation with perfect foresight. However, rolling horizon methods are not applicable in day-ahead electricity markets, where decisions must be fixed for a 24-hour period. As most exchange-traded power demand is settled in day-ahead markets [164], rolling horizon approaches are currently more limited than day-ahead methods in terms of practical applications. Next, we will describe solution methods that have been employed to solve the stochastic, day-ahead problem.

2.3.3 Solution Methods

Unlike deterministic UC reviewed in Section 2.2, there is no dominant solution method for the stochastic UC problem. The scenario-based stochastic UC formulation is NP-hard, and considerably more expensive to solve than the deterministic version [17]. MILP approaches such as branch-and-bound, which are the state of the art in solving deterministic UC problems, have been used to solve the stochastic UC problem [20] but face challenges in large power systems due to run time and memory constraints [17]. Lagrangian relaxation approaches has been widely used to decompose the stochastic UC problem into generator sub-problems or scenario sub-problems [17, 65, 165, 166]. Further decomposition methods such as Benders' decomposition [152, 167] and Dantzig-Wolfe decomposition [168] have also been used to solve large-scale stochastic UC problems. The decomposed problems can be solved in parallel across multiple machines, which offers promising opportunities for achieving tractability using high performance computing resources [17]. Nevertheless, the large computational cost of stochastic optimisation remains a barrier to the adoption of stochastic UC for practical applications.

Due to the large computational burden of scenario-based stochastic UC, determining the size and content of the scenario set $\mathcal{S}$ is an important decision in stochastic UC solution methods [154, 166]. The set should capture important statistical information about the underlying stochastic processes within a small number of scenarios to achieve computational tractability. Common approaches generate a set of scenarios using a Monte Carlo method and select a subset based on the Kantorovich distance, minimising the difference between the original and reduced scenario sets [19, 169]. K-means clustering and importance sampling approaches are among those presented in a comparison of approaches [154], which finds that the reduction method impacts both operating cost and run time. Nevertheless, due to the high computational requirements of solving stochastic formulations, the number of scenarios is generally small and may not capture extreme scenarios in cases with large numbers of uncertain parameters [22]. For this reason, reserve requirements have been introduced to some stochastic UC formulations to improve solution security, as described in Section 2.3.4.

2.3.4 Formulations with Reserve Constraints

A common extension to the stochastic UC formulation in Equations 2.5–2.7 is to add an explicit reserve constraint to improve solution robustness [22, 170]. Since the number of scenarios $|\mathcal{S}|$ must usually be relatively small in order for the problem to be computationally tractable, it often cannot include extreme situations [22]. As a result, including a reserve constraint as in the deterministic formulation (Equation 2.3) decreases the probability of lost load under large deviations of demand or generation from their forecasts. Furthermore, the performance of stochastic UC models without

reserve constraints is highly dependent on the statistical approach to modelling uncertainties and generating the scenario set $\mathcal{S}$. By including reserve constraints, solutions are made more robust against random generation of scenarios [159].

Using the Monte Carlo evaluation method described in Section 2.3.1, [22] compares stochastic UC with and without reserve constraints in simulations on the IEEE RTS system with 32 generators. Solutions with reserve constraints are found to be more robust to changes in the underlying probability distributions, reflecting inaccurate models of the real world. In addition, solutions have lower expected costs when reserves are included, and the reserve requirements are found to be smaller than for deterministic UC approaches. The results of [22] show that including reserve requirements can be used to reduce the computational burden of stochastic UC by requiring fewer scenarios to achieve robust solutions. However, this requires heuristic methods to determine reserve constraints, and comes at the expense of lower solution quality as the number of scenarios increases. In the following section, we compare the solution quality of stochastic UC approaches, with and without reserve requirements, to deterministic methods.

2.3.5 Comparison with Deterministic UC

In comparison with deterministic methods reviewed in Section 2.2, solutions to the stochastic UC problem have been shown to achieve lower operating costs due to its more rigorous consideration of uncertainties [17, 18, 20, 22, 65, 159, 161, 171–173]. Simulating one year's rolling operation of the Irish 2020 power system, [18] shows that total operating costs of a stochastic model are 0.25–0.9% lower than a deterministic approach. In [22], in which reserve requirements are used, both demand uncertainty and generator outages are modelled, and costs of the stochastic approach are found to be 1.3% lower than deterministic UC in experiments on the IEEE RTS system with 32 generators. Studying a representation of the GB power system, [161] uses a rolling planning framework to evaluate the relative costs of stochastic and deterministic UC over two years, considering transmission network constraints and pumped storage. Stochastic UC outperforms deterministic UC by 0.3% in terms of operating costs.

Operating cost reductions compared to deterministic UC are shown to increase with uncertainty, for example resulting from increasing wind penetration [17,20,22,161, 172]. Deterministic UC solutions to the problem in [20] are unable to accommodate wind penetrations above 8% without suffering lost load, while stochastic UC solutions are secure for penetrations up to 20%. Cost reductions in [18] are attributed to less cycling of flexible generation, with more consistent use of base-load with fewer startups. In [22], which includes reserve constraints in the stochastic formulation, the required reserve levels are found to be lower as compared with deterministic UC, causing generators to operate at greater efficiencies.

Stochastic UC has also been compared with UC methods with perfect foresight of

demand and renewables generation in a rolling horizon optimisation context [18, 158]. The cost of uncertain forecasts is quantified in [158] by comparing the costs of commitment schedules with perfect forecast and stochastic UC: perfect foresight is found to result in between 0.05% and 1.2% lower operating costs in case studies for the Irish power system. A similar comparison is made in [18], finding that scenario-based stochastic UC achieves lower 1.5% higher operating than the optimisation with perfect foresight, while a deterministic UC with reserve constraints results in 1.75% higher costs. Stochastic UC is therefore shown to mitigate the impact of uncertainty as compared with deterministic methods, but there are still significant cost impacts of uncertain forecasts.

The primary drawback of scenario-based stochastic optimisation approaches are the computational requirements, which are much larger than those of deterministic methods [17]. Run times for stochastic UC with 12 scenarios and reserve requirements are found to be between 1 and 3 orders of magnitude larger than a deterministic approach in [22]. Distributed implementations for solution methods which use decomposition techniques can be used to improve computational tractability, but still require significant high performance computing resources [17]. An additional drawback is that the underlying probability distributions of stochastic demand and renewables generation may be hard to obtain, making scenario generation difficult and solutions potentially sensitive to deviations from the modelled distributions [13]. However, the stochastic UC literature has demonstrated that the dominant deterministic UC approaches described in Section 2.2 are sub-optimal, and significant operating cost savings can be achieved by methods which aim to optimise the expected cost over scenarios, rather than considering only the point forecast. These improvements are of a similar magnitude to those achieved by significant advances in deterministic UC methods, such as the transition to MILP and branch-and-bound reviewed in Section 2.2.2 which reportedly achieved annual operating cost reductions of $1 billion in North America [9]. The potential for such large improvements in solution quality that has been demonstrated by the stochastic UC literature is one of the principle motivations for further investigating UC solution methods. Robust optimisation, reviewed in Section 2.4, mitigates some of the issues of stochastic UC, having lower computational costs while accounting for uncertainties without heuristic reserve constraints.

2.4 Robust Unit Commitment

Stochastic UC approaches have been shown to be effective in reducing operating costs, but practically difficult to implement due to high computational costs [17] and requiring knowledge of the distributions underlying stochastic processes [22, 174]. Robust optimisation has been applied to the UC problem, which is typically

less expensive to compute and makes fewer assumptions regarding the uncertain distributions [13]. In general, robust optimisation tackles problems where solutions must minimise worst-case costs over a deterministic uncertainty set. The uncertainty set is the region of possible realisations of uncertain parameters, which is usually determined based on a desired level of robustness. Feasible solutions to robust optimisation problems must satisfy constraints for all realisations of the data in the uncertainty set while optimising for the worst-case outcome [175].

Using the same notation as in previous formulations, robust UC problem formulation is:

$$\min \max_{d \in \mathcal{D}} \sum_{i=1}^{N} \sum_{t=1}^{T} C_i(t, d) \tag{2.9}$$

subject to:

$$\sum_{i=1}^{N} p_i(t, d) = D(t, d) \qquad \forall t \in \{1 \,..\, T\}, \forall d \in \mathcal{D} \tag{2.10}$$

$$\boldsymbol{p}_i(d) \in \Pi_i \qquad \forall i \in \{1 \,..\, N\}, \forall d \in \mathcal{D} \tag{2.11}$$

$$\tag{2.12}$$

The set $\mathcal{D}$ is a probabilistic uncertainty set, defining the region of realisations of uncertainty; $C_i(t, d)$ is the cost of generator i at time t under the realisation of uncertainties $d \in \mathcal{D}$. The objective of the robust UC formulation is to find a commitment schedule which minimises operating costs in the worst (most expensive) case realisation of uncertainties. Note that unlike the stochastic objective in Equation 2.5, the scenario probability $P(s)$ is not included in the objective function Equation 2.9. Instead, the set is a deterministic uncertainty set. As a result, a model for the underlying distributions of uncertain parameters is not required for robust UC. Like the stochastic formulation, the robust formulation does not include a reserve constraint, with reserves being implicitly allocated to satisfy extreme demand deviations or other contingencies in the uncertainty set.

Robust UC problem formulations have been used widely in the UC literature to manage uncertainties [13, 71, 153, 176–189]. Compared with stochastic UC, robust UC has the following benefits: (1) it is often more computationally tractable; (2) it does not require knowledge of underlying probability distribution governing stochastic processes like renewables generation; (3) it is not sensitive to scenario generation and reduction methods [13, 176, 177]. However, solutions tend to be more conservative as they optimise for the worst-case outcome in the uncertainty set, regardless of its probability, and may produce higher expected operating costs as a result [190]. In

Section 2.4.4 we review research which has compared robust UC with deterministic and stochastic UC. In the following subsection, we discuss alternative formulations which unify stochastic UC and robust UC, mitigating the over-conservatism of robust optimisation.

2.4.1 Hybrid Stochastic and Robust Formulations

UC formulations have been proposed which aim to combine the advantages of stochastic UC (minimising expected costs over scenarios) and robust UC (minimising costs under worst-case scenarios) [71,153,191,192]. These hybrid approaches produce solutions which are less conservative than traditional robust optimisation approaches. In [153], it is assumed that the probability distribution over scenarios is known, and scenarios are sampled and then aggregated into a scenario tree, similar to stochastic UC approaches. A robust optimisation approach is then formed, aiming to minimise the worst-case costs under the sampled scenarios. Minimisation of a weighted sum of expected costs and worst-case costs are used in [191], creating a hybrid stochastic/robust objective function with a parameter controlling the level of robustness. This hybrid approach is shown to produce solutions which are less conservative and have lower expected operating costs than robust UC. However, a drawback of this approach is that, like stochastic UC, it assumes that the underlying probability distributions of uncertain variables are known.

2.4.2 Benchmark Problems

Robust UC faces similar challenges regarding benchmark problems as those faced in stochastic UC research, described in Section 2.3.1. Namely, given the stochastic nature of the problem formulation, UC solutions must be evaluated against an equal set of possible realisations of uncertainties. A broad range of test systems has been studied including New England ISO network with 312 generators [13], variations of the IEEE 118 system [71,178,180,184], IEEE RTS with 96 generators [153,182] and the Polish transmission network [184]. Furthermore, studies have variously considered transmission constraints [13,180,184], dispatchable wind [71] and pumped hydro resources [177]. There has been no established benchmark in the literature that has allowed for robust UC methods to be rigorously evaluated against one another.

Several research papers have implemented multiple solution methods and compared solution quality using the Monte Carlo approach described in Section 2.3.1 [13,71,178,182]. Using this method the robustness of schedules is evaluated against out-of-sample scenarios [71]. In Section 2.4.4, we present the results of experiments comparing robust UC with deterministic and stochastic UC approaches.

2.4.3 Solution Methods

Like the stochastic UC problem, branch-and-bound approaches using commercial MILP solvers cannot be effectively applied in most cases to solve the robust UC problem [189] and most research has focused on decomposition techniques, notably Benders' decomposition [13, 177, 186, 189].

The computational tractability of robust UC depends to a large extent on the form of the uncertainty set $\mathcal{D}$. Demand and wind variation may be represented using continuous uncertainty regions (e.g. with polyhedral constraints) [13] or with discrete sets representing scenarios [177, 188]. To account for generator outages, knapsack constraints restricting coincident outages to a maximum of x are used to implement security criteria protecting against the loss of the x largest infeeds [186, 189]. While robust UC formulations are generally more computationally tractable than stochastic ones, robust UC has longer execution time than deterministic UC [13]. In addition, large-scale problems with many uncertain parameters and complex uncertainty sets can also be intractably expensive to solve for practical use cases [185].

The potential for over-conservative solutions and consequently high operating costs has been managed in several cases by introducing an uncertainty budget parameter defining the extremity of scenarios included in the uncertainty set $\mathcal{S}$ [13, 178]. Tuning the uncertainty budget is shown in [13] to have a significant impact on performance and prevent over-conservative schedules. In the following section, we will compare the performance of robust UC approaches to the deterministic and stochastic UC formulations.

2.4.4 Comparison with Deterministic and Stochastic UC

Operating costs for solutions to the robust UC problem have been compared with deterministic UC in [13, 180, 183, 184]. Comparisons of average operating costs for robust UC and deterministic UC approaches have found that while robust UC operating costs are often higher for nominal demand [180] or demand profiles with low uncertainties [13], operating costs have lower variance under different scenarios. Comparing robust optimisation with deterministic UC and evaluating solutions with a Monte Carlo approach with 1000 scenarios, [13] reports expected cost reductions of between 0.34–5.48% on a 312-unit test system reflecting the New England ISO. The standard deviation of operating costs is also found to be between 8–14 times larger when using the deterministic approach. Operating costs for the robust UC approach under the nominal demand scenario are found to be 0.8% higher than for a deterministic solution in [180], demonstrating the relative conservatism of robust optimisation. However, for the worst-case scenario, the deterministic solution is found to be infeasible (i.e. resulting in lost load), whereas the robust approach is able to maintain system security. The impact of the robust UC uncertainty budget

described in Section 2.4.3 which provides control over the conservatism of the solution is analysed in [13]. A statistical method is proposed to tune the budget parameter to improve economic efficiency of solutions.

Comparing robust UC with scenario-based stochastic methods, [178] finds that stochastic UC achieves lower expected operating costs but larger regret (deviation from optimal solution cost) on average under high levels of uncertainty. A stochastic approach is compared with robust optimisation in [71], and again found to achieve lower average operating costs but with more frequent lost load events and higher worst-case operating costs. Increasing the number of scenarios for the stochastic UC approach reduces expected operating costs, but at 30 scenarios, stochastic UC is found to be 71 times slower to solve than robust UC. Studying a small 4-bus system with 14 generators, [185] finds that stochastic UC does not significantly outperform robust UC in terms of expected operating costs, even with low uncertainty budgets. In addition, the stochastic UC is not applied to a larger IEEE 118-bus system, due to intractable compute times.

In summary, robust UC is generally a more computationally tractable alternative to stochastic UC, with potential advantages over deterministic UC methods due to more rigorous consideration of uncertainties [13]. Robust optimisation has not been widely applied on such large-scale power systems as deterministic methods. Most research has been conducted on small to medium size systems; larger simulation studies have considered the New England ISO [13] and Polish transmission network [184] but these remain smaller than large-scale deterministic UC simulation studies, such as MISO [14] and FERC [11]. The principle disadvantage of robust UC compared with stochastic UC is that schedules may be overly conservative and hence economically sub-optimal [153, 178]. In addition, the computational cost may be relatively high as compared with deterministic methods when large numbers of uncertain parameters are considered [185].

2.5 Reinforcement Learning

A small body of research has been dedicated to solving the UC problem with reinforcement learning (RL) [40–43, 55, 56], which is the focus of this thesis. These papers are summarised in Table 2.3. In contrast with the other methods discussed in this chapter, RL methods rely on the formulation of the UC problem as a Markov Decision Process (MDP), which is used to solve sequential decision-making problems. An 'agent' learns by trial-and-error, typically using a simulation of the real power system, to make decisions which minimise operating costs. We discuss MDPs and provide a background to RL more generally in Section 3.2.

Q-learning, a popular class of RL methods, has been applied to the UC problem in [40–43]. These papers have applied Q-learning in a tabular format [40] and

Authors	Method	Function Approximation	Max. Gens	Stochastic	Multiple Training Days	Unseen Test Days	Notes
Jasmin et al., 2009 [40]	Q-learning	No	4	No	No	No	
Jasmin et al., 2016 [41]	Q-learning	Yes	10	Yes	No	No	
Li et al., 2019 [43]	Q-learning	Yes	10	Yes	No	No	
Navin & Sharma [42]	Multi-Agent Fuzzy Q-Learning	Yes	10	No	No	No	
Dalal et al., 2016 [56]	Cross-Entropy	Yes	99	Yes	Yes	No	Day-ahead UC simplified to a single decision per day, choosing between 20 actions
Dalal & Mannor, 2015 [55]	SARSA	Yes	8	No	No	No	
Dalal & Mannor, 2015 [55]	Tree Search	No	12	No	NA	NA	

Table 2.3: Summary of research applying RL to the UC problem. We show the method used; the maximum problem size by number of generators; whether function approximation was used; whether stochastic demand or renewables generation are included in the problem setup; whether multiple days were used in training and whether testing was conducted on unseen test days.

with function approximation [41–43] with applications to problems of up to 10 generators. As shown in Table 2.3, all studies consider optimisation for a single profile; training and testing on a single day. As a result, they do not demonstrate the ability of the trained policy to generalise to unseen problems. Two of the Q-learning studies include uncertainty in the problem setup through stochastic renewables generation [41, 43]. Fuzzy Q-learning is used in [42] to solve the widely-studied Kazarlis et al. benchmark problem with 10 generators, and is shown to outperform several existing deterministic UC solution methods. The Q-learning methods studied suffer from curses of dimensionality in the state and action spaces for the UC problem, which has limited their application to systems of up to 10 generators.

A larger, 99-generator system is studied in [56], which solves a combined problem of day-ahead UC and real-time ED, represented using two interleaved MDPs. The UC component is solved with the cross-entropy RL method. This is by far the largest study applying RL to solve the UC problem. In addition, stochastic demand and wind generation are considered, and the agent is trained over multiple profiles. However, the UC component of this problem is simplified significantly to selecting a single commitment decision for each 24-hour period, with no intra-day commitment changes. In addition, the set of commitment decisions $\mathcal{X}$ is reduced to just 20 decisions (of a possible 2^{99}) using a heuristic approach. Compared with simple heuristic solutions to the UC problem (e.g. committing a random subset of generators or the cheapest subset), the cross-entropy method employed is shown to result in lower total operating costs. However, by reducing the UC solution space to just 20 actions and not allowing intra-day commitment changes, this approach is not guaranteed to converge towards cost-optimal solutions and solution quality is not validated by comparison with mathematical optimisation approaches such as MILP.

Three algorithms are proposed in [55]: one using the SARSA method [193] and two using tree search approaches. SARSA, which is a model-free RL algorithm, was found to be the least effective algorithm and could not be scaled beyond a problem of 8 generators due to slow convergence. To the best of our knowledge, [55] is the only research which applies tree search methods to solve the UC problem and the only research to formulate the UC problem as a search tree. Of the two tree search methods, a simple lookahead strategy with a limited time horizon was found to be the most effective approach, outperforming a metaheuristic solution by 27% in terms of operating costs for a deterministic problem of 12 generators. Unlike the other methods reviewed, the tree search method proposed does not store a policy which is iteratively improved upon and hence it is not strictly an RL algorithm. However, by comparison with the RL algorithm SARSA, exploiting lookahead capability is shown to be a more effective strategy for the UC problem.

In summary, the existing research on RL for UC has shown some promising results, but curses of dimensionality in both state and action spaces has limited these

methods to small-scale problems. The only large-scale application [56] simplifies the UC problem substantially by making a single commitment decision per day without intra-day commitment changes, significantly limiting the solution quality potential. Furthermore, no comparison has been made in the existing literature with the state-of-the-art in deterministic UC methods using MILP. RL approaches to the UC problem differ fundamentally from the other formulations discussed in this chapter. The goal is to learn a function mapping from problems to solutions by trial-and-error. The learned function can in principle generalise to unseen problems, without retraining. This property of RL, which offers the opportunity for most of the computational burden to be shifted to offline training, has not yet been explored by the existing literature which consider training and testing on the same profile. In addition, a number of recent breakthroughs in RL, including hybrid approaches using tree search [27] and efficient policy optimisation algorithms such as proximal policy optimisation [35] have not been applied to the UC problem at the time of writing. Compared with the other approaches discussed in this chapter, there is significant scope for further research investigating novel solution methods applying RL to the UC problem.

2.6 Conclusion

In this section we have reviewed the state of the art in UC methods. Deterministic UC approaches described in Section 2.2 remain the most widely-used in industry due to their low computational cost in comparison with stochastic optimisation approaches. However, they have no capacity to integrate uncertainty in day-ahead forecasts into the problem formulation, instead relying on heuristic reserve constraints. Scenario-based stochastic UC formulations, which were developed to handle uncertainty in forecasts using a set of scenarios, have been shown to outperform deterministic UC in terms of operating costs [18, 20, 22], but are generally too computationally expensive for practical applications [13]. Robust formulations discussed in Section 2.4 have been shown to exhibit advantages over deterministic methods, such as lower variance of operating costs [13] and lower worst-case operating costs [180, 184]. Robust optimisation techniques generally produce conservative solutions with high expected operating costs on average [190]. In addition, large-scale problems with many uncertain parameters can be intractably expensive to solve [185]. There is motivation to develop new UC solution methods that achieve better solution quality than deterministic approaches while remaining computationally tractable.

This literature review has shown that comparison between solution methods is difficult due to a lack of established benchmark problems with multiple problem instances. The problems proposed by Knueven et al. [11] have recently been proposed for the deterministic problem to address this issue, but have not yet seen widespread

use. In addition, these benchmark problems do not provide stochastic models for Monte Carlo evaluation of stochastic or robust UC methods. Table 2.1 surveyed solutions to the 100-generator Kazarlis benchmark [5], but this is limited to a single problem instance. For the stochastic, robust and reinforcement learning approaches, there are no widely-used benchmarks. In machine learning domains, the establishment of robust test problems such as ImageNet [194] for computer vision and SQUAD [195] has accelerated research and enabled algorithms to be systematically compared. The establishment of a universal benchmark for the UC problem which can be used to evaluate methods from across deterministic, stochastic, robust and RL-based UC literature would enable further progress in this domain.

Only a small body of research has investigated applying RL to solve the UC problem. These papers have generally investigated applications of model-free RL to small problem instances. In addition, no comparison has been made with the state-of-the-art in deterministic UC methods. Only one paper reviewed has applied tree search [55], which was successfully applied in a deterministic UC environment for problem instances of up to 12 generators. The limitations of these studies has consistently been scalability to large power systems, which so far has not been shown. Nevertheless, significant methodological breakthroughs in the field of RL have yet to be applied in the UC problem, and offer potentially large improvements relative to existing literature as has been seen in other domains [27, 28]. In the next chapter, we will provide a background to RL and tree search, which will form the basis of the novel RL-aided tree search method presented in Chapter 4.

Chapter 3

Methodology

3.1 Introduction

In this chapter, we will describe in detail the methods used in this thesis, providing the theoretical foundation for the novel UC solution methods developed and applied in Chapters 4–6. In Section 2.5 we reviewed the small body of research that has used RL to solve the UC problem. The existing literature has shown promising results for small power systems, but has been unsuccessful in overcoming the curses of dimensionality in state and action spaces required to scale to larger problems. Furthermore, recent RL methods which have been used to achieve state-of-the-art in challenging domains in AI research [27,28,35] have not yet been exploited to tackle the UC problem. Among these developments are RL approaches which use traditional planning methods such as tree search to enable lookahead strategies [27, 28, 54]. Combining RL with tree search to tackle the UC problem is the subject of this thesis, and forms the basis of the methodology developed in Chapter 4 and Chapter 5.

Sections 3.2–3.4 provides a review of RL material relevant to this thesis, with a particular focus on policy gradient methods which form the basis of the RL methods used in this thesis. In Chapter 4, we will introduce guided tree search, which combines policy gradient RL with tree search algorithms. Sections 3.5–3.6 provide a background to tree search and describe in detail the tree search algorithms uniform-cost search, A* search, and iterative deepening methods. These algorithms are used in the guided tree search framework in Chapters 4–6. Finally, Section 3.7 provides a review of traditional mathematical optimisation techniques for power systems. We describe mixed-integer linear programming for the UC problem and the lambda-iteration method for economic dispatch, which are used to develop the power system environment and benchmark solutions in Section 4.2.3.

3.2 Background to Reinforcement Learning

This section provides a background to RL, reviewing the theory necessary for the original guided tree search methods developed in this thesis. For a comprehensive introduction to RL, we refer the reader to [29].

Along with supervised and unsupervised learning, RL is one of the three main paradigms of machine learning. RL is the task of learning, by trial-and-error, how to act in order to maximise a numerical reward signal [29]. This type of goal-oriented learning differs from supervised learning, which relies on labelled data for regression and classification tasks, and unsupervised learning, which aims to discover patterns in unlabelled data. In this section we will describe the general principles of RL, beginning with the concept of an *agent* interacting with an *environment*.

3.2.1 Agent-Environment Interaction

The problem of RL is often informally described in terms of a decision-making *agent* interacting with its surroundings, the *environment*. The agent interacts with the environment through *actions*, which it chooses based on contextual information about the environment's *state*. The agent's actions can have an impact on the state, akin to decisions in the real world. Each time the agent acts, it receives feedback known as a *reward*, and a observation of the new state. These steps form a routine that is repeated in a discrete-time process: the agent observes a state, chooses an action, receives a reward, observes a new state, and so on. Each iteration of the (state, action, reward) routine is known as a *timestep*. The agent's goal is to learn a behavioural strategy called a *policy* that will maximise the sum of future rewards. The process is more formally described as a Markov Decision Process (MDP) which we cover in Section 3.2.2.

Almost all real-world decision-making processes can be formulated in terms of an agent-environment interaction. Some examples of agent-environment interactions which have been studied in the RL literature include:

- Chess [27]: the agent moves pieces (action) based on the board position (state). At the end of the agent registers whether the game was won or lost (reward).

- News recommender system [196]: the agent recommends an article (action) based on information about the user (state). The agent is notified whether the user read the article (reward).

- Microgrid management [197]: the agent controls a battery to meet household electricity demand. A solar panel can be used to charge the battery at no cost, or the battery can draw from the grid. The agent receives a weather forecast (state) and charges and discharges the battery (action), aiming to

maximally exploit the solar panel. The agents receives feedback in the form of an electricity cost (reward).

The examples illustrate several features of RL problems. In some problems, such as chess, the agent receives feedback infrequently (i.e. at the end of the game). In these contexts, the agent must learn which moves contributed to the win (or loss), which may include strategic moves early in the game. This is known as the *credit assignment problem*. In the recommender system context, the agent must trade-off recommending articles from topics the agent already knows the user is interested in, with new topics that the agent has rarely recommended before. This is an example of the *exploration-exploitation* trade-off. While there are benefits to exploiting actions that are known to lead to high rewards, the agent will only improve its performance and achieve even higher rewards by exploring new actions. A characteristic of the microgrid management problem is uncertain feedback, making this a case of *decision making under uncertainty*. Since the solar output depends on the weather, which is unpredictable, the agent may be fortunate when the day is sunnier than the forecast predicted, receiving a higher reward than on less sunny days. The agent must accumulate a wide range of experience in order to learn a strategy that maximises its *expected* reward.

3.2.2 Markov Decision Processes

As described in Section 3.2.1, the agent-environment interaction involves the agent taking actions and observing a new state of the environment at each timestep. This can be formalised in a Markov decision process (MDP), which describes sequential decision-making problems of this kind. An MDP is defined by a set of states $\mathcal{S}$, set of actions $\mathcal{A}$, reward function $R(s, a)$, and a transition function $F(s', s, a)$. The transition function describes the dynamics of the environment, giving the probability of transitioning to state s' after having taken action a in state s: $F(s', s, a) = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$. In some cases, a distinction is made between what the agent perceives, the *observation*, and the underlying definition of the environment, the state. MDPs where the agent does not observe the full state are known as partially-observable MDPs (POMDPs).

An agent operating in the MDP decides which action a to take given a state (or observation) s. The function that maps from states to actions is called a policy $\pi(a|s)$:

$$\pi(a|s) = \Pr(A_t = a | S_t = s) \tag{3.1}$$

As the agent acts following the policy $\pi(a|s)$, the following sequence develops [29]:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots \tag{3.2}$$

This sequence continues, until a special state called a *terminal state* is reached, if one exists. Problems which have terminal states (such as checkmate in a game of chess) are known as *episodic*, and each sequence is called an *episode*. In this thesis, we will only consider episodic problems. We refer the reader to [29] for a discussion of continuing problems.

3.2.3 The Objective of RL

The task of RL is to learn a policy which maximises the sum of the agent's rewards in the long run. More precisely, the objective is to find a policy $\pi(a|s)$, giving the probability of taking action a in state s, which will maximise the expected *return* G_t, which is the discounted sum of rewards:

$$G_t = \sum_{k=1}^{\infty} \gamma^{k-1} R_{t+k} \tag{3.3}$$

where $0 \leq \gamma \leq 1$ is a discount factor. The discount factor determines the present value of future rewards [29]. If $\gamma = 1$, the return is undiscounted and the agent assigns as much credit to distant rewards as to immediate ones; if $\gamma = 0$, the agent only aims to maximise the immediate reward. Discounting is used to tackle the problem of credit assignment described in the context of chess in Section 3.2.1. The field of RL is concerned with algorithms that gradually improve $\pi(a|s)$ from experience in the environment; that is, sampling states, actions and rewards.

3.2.4 Simulation Environments

In many contexts, it is not practical to generate enough experience to learn effective policies by taking actions in the real world. As a result, most RL training is conducted in simulation, with the trained policy then deployed in the real environment. Simulation environments are essential tools for training RL agents that can be deployed in the real world. Some environments with simple, deterministic dynamics can be easily simulated (such as board games), while others are much more challenging and may only roughly approximate the real world. In recent years, a wide range of simulation environments have been developed for the purpose of RL research in domains such as biomechanics [198], electricity distribution networks [199] and building management [47]. In Section 4.2.3 we develop a simulation environment for the UC problem, enabling the application of RL methods.

3.3 Taxonomy of RL Algorithms

Having introduced the problem framework of RL and some key concepts, this section will provide a general taxonomy of RL algorithms. A large number of RL algorithms

exist which are suited to different problem domains, and several methods have already been applied to solve small-scale UC problem instances as described in Section 2.5. The choice of RL method is a key decision for our RL-based approach to the UC problem described in Chapter 4. In this section, we will first discuss value-based and policy gradient methods, and the relevant properties of each. Second, we will discuss methods which exploit or learn a model of the environment (model-based), and those which do not (model-free).

3.3.1 Value-Based and Policy Gradient Methods

Most RL methods can be divided into either value-based or policy gradient methods. Here we will first describe each type of method and then compare their properties, benefits and drawbacks.

Value-Based Methods

Many methods for solving the RL problem described in Section 3.2.3 involve estimating a *value function*, describing the expected return from a given state or following a state-action pair. Learning value functions is a fundamental component of almost all RL algorithms, including both value-based and policy gradient methods.

Two types of value functions are used in RL: state-value $V^\pi(s)$ and action value $Q^\pi(s,a)$. Given a policy $\pi(a|s)$, the state-value function for state s is:

$$V^\pi(s) = \mathbb{E}[G_t|S_t = s] \tag{3.4}$$

The state-value function gives the expected return G_t (defined in Equation 3.3) when starting in state s and acting according to policy $\pi(a|s)$ thereafter. Similarly, the action-value function for a state-action pair (s,a) is:

$$Q^\pi(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a] \tag{3.5}$$

This is the expected return of taking action a in state s and acting according to $\pi(a|s)$ thereafter. The optimal state-value and action-value functions are often denoted $V^*(s)$ and $Q^*(s,a)$, which are the value functions when π is optimal:

$$V^*(s) = \max_\pi V^\pi(s) \tag{3.6}$$

$$Q^*(s,a) = \max_\pi Q^\pi(s,a) \tag{3.7}$$

The basis of value-based RL methods is to iteratively improve the value function estimating $Q^\pi(s,a)$ or $V^\pi(s)$ for a given policy π, and then adjust the policy to be greedy with respect to the new value function (selecting the action with highest estimated action-value). This is known as *generalised policy iteration* (GPI), and is

the basis of Q-learning methods [200]. Using Q-learning, the action-value function approaches the optimal $Q^*(s,a)$. Once the optimal action-value function $Q^*(s,a)$ is known, then finding an optimal policy $\pi^*(a|s)$ becomes trivial:

$$\pi^*(a|s) = \arg\max_a Q^*(s,a) \tag{3.8}$$

Experience (observed states, actions and rewards) is accumulated by the agent through interaction with the environment (as described in Section 3.2.4), which is used to update the estimated action-value function $Q^\pi(s,a)$ to be consistent with π, in a step known as policy evaluation. In simple problems, $Q^\pi(s,a)$ can be represented in tabular form with an entry for each (s,a) combination. However, this approach is infeasible for most real-world tasks including those with continuous state or action spaces. Deep Q-learning, a widely used RL algorithm, represents the action-value function as a deep neural network (deep Q-network, or DQN) [26]. Variants of this approach such as double Q-learning [33], dueling DQN [201] and Rainbow DQN [202] have become widely-used RL algorithms.

Policy Gradient Methods

Rather than deriving a policy from learned state-values or action-values, policy gradient methods aim to directly map from a numerical representation of the state onto a distribution over actions. The policy is parameterised as a function $\pi_\theta(a|s)$ with parameters θ. Commonly, this function is represented by a neural network with weights θ. Based on experience gained through interaction with the environment, policy gradient methods repeatedly estimate the gradient of the following performance measure with respect to the policy parameter θ:

$$J(\theta) = \sum_{s\in\mathcal{S}} \mu^{\pi_\theta}(s) V^{\pi_\theta}(s) \tag{3.9}$$

where $\mu^{\pi_\theta}(s)$ is the state probability distribution of the MDP when acting under policy π_θ, $\mathcal{S}$ is the set of states and $V^{\pi_\theta}(s)$ is the state-value function acting under policy π_θ from state s. Gradient ascent is then used to update the policy to a stronger one. From the state-value function in Equation 3.4, the performance measure gives the state probability-weighted expected return of acting under policy π_θ. As policy gradient methods form the basis of the RL methods applied to the UC problem in Chapters 4–6, we discuss policy gradient methods in more detail in Section 3.4.

Comparison of Value-Based and Policy Gradient Methods

The relative merits of value-based and policy gradient methods are problem-specific, and both have been used to achieve state-of-the-art performance in different domains. Policy gradient methods benefit from better convergence properties than many

value-based methods, where the policy can change dramatically following an update due to the arg max evaluation in Equation 3.8. In addition, policy gradients can be used to learn stochastic policies, which value-based methods do not naturally accommodate [29]. Lastly, policy gradients are generally better suited to domains with large or continuous action spaces, since they do not require the evaluation of Q-values over all actions [203]. However, policy gradient methods tend to require more samples, whereas value-based approaches can reuse experience more efficiently [34, 204]. In contexts where experience is expensive to gather either in simulation or in the real-world, value-based methods may be more appropriate. Furthermore, there are practical challenges in reliably estimating the gradient of the performance measure $J(\theta)$, and policy gradient methods can converge prematurely to local optima.

In practice, there is not always a clear distinction between value-based and policy gradient methods. The vast majority of policy gradient algorithms involve simultaneously learning a value function to stabilise training, the so-called actor-critic approach which we describe in Section 3.4.2. Other methods such as deep deterministic policy gradients (DDPG) [205] and twin-delayed DDPG (TD3) [206] combine Q-learning with policy gradients, allowing value-based methods to be applied in continuous action spaces.

3.3.2 Model-Based and Model-Free Methods

An important dichotomy exists between model-based RL algorithms which exploit a learned or pre-determined model of the environment's dynamics versus model-free algorithms which do not. In model-based methods, transitions can be sampled using the model in order to estimate the outcome of actions before they are taken. This brings two advantages. First, it improves *sample efficiency* - the number of interactions required in the environment in order to reach a given performance level - which can be particularly beneficial when access to the real environment is limited. Second, planning methods such as tree search can in some contexts be applied when a model is available, enabling greater action precision and benefits of foresight in decision-making [28]. By employing planning, model-based methods can offer greater levels of interpretability and robustness [45]. However, since model-free methods do not require an environment model, they can be applied more generally than model-based ones and have been the subject of the majority of recent RL research. Nevertheless, model-based methods are the state of the art in several domains where they have been applied, outperforming model-free methods [27, 28, 207, 208].

In model-based methods, the environment dynamics may be learned or pre-determined offline based on knowledge of the problem. Due to the complexity of many problems, learning an environment model is often prohibitively difficult or computationally expensive, such as for MDPs with high dimensional state spaces. However, in simple instances, model-based algorithms such as Dyna [209] can be

used which simultaneously learn reward and transition functions and exploit these to simulate additional experience without interacting with the environment. In addition, recent research has found that key features of complex environments such as the widely-studied Atari benchmarks problems can be learned using deep learning methods in order to apply model-based methods [28, 210]. Model-based approaches such as MuZero [28] have been shown to significantly outperform state-of-the-art model-free methods and are substantially more sample efficient [210].

Aside from learning the environment dynamics, it is also possible in some instances to pre-determine the environment dynamics offline. This approach has been applied in simple games-playing domains such as Go, where the transition function is cheap to evaluate and is known a priori. The AlphaGo [44] and AlphaZero [27] algorithms used this approach to achieve state-of-the-art performance in the game of Go. A similar approach was also applied in [54], which was used in the game Hex.

A disadvantage of model-based compared with model-free methods is the additional computation required to evaluate the environment model [211]. In cases where the models are learned, this may be relatively inexpensive. However, in some cases where complex physical models are used (such as electricity network management) evaluating the environment model may add significant computational expense that partially or completely offsets the improved sample efficiency of model-based over model-free methods.

While RL research has focused largely on model-free methods due to their more general application, significant milestones in AI have been reached by exploiting knowledge of the model dynamics through tree search, most notably in games-playing domains [27, 28, 44]. In Section 4.5 we show that exploiting model dynamics in guided tree search also improves on a purely model-free approach for the UC problem. Our methodology utilises model-free policy gradient methods for training, and model-based tree search methods in evaluation. In the follow section, we provide a background to policy gradient RL.

3.4 Policy Gradient Methods

Section 3.3.1 introduced value-based and policy gradient methods as two classes of RL algorithms. In Chapter 4 we will develop a novel approach to solving UC problems based on the widely-used policy gradient method, proximal policy optimisation (PPO). This section will provide a background to policy gradient methods in general and a detailed description of PPO.

3.4.1 Estimating the Policy Gradient

Equation 3.9 defined the performance measure $J(\theta)$ which policy gradient methods aim to improve. $J(\theta)$ is the sum of state-values over all states, weighted by their state-probability when acting under a policy with parameters θ. The improvements are made by maximising the performance measure $J(\theta)$ using stochastic gradient ascent:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{3.10}$$

where $\nabla J(\theta_t)$ is the gradient of the performance measure with respect to the parameters θ; i.e. the policy gradient.

We will begin this section by describing the policy gradient theorem, which defines how $\nabla J(\theta)$ can be approximated using samples of states, actions and rewards observed through experience in the environment. The policy gradient theorem analytically expresses the gradient of the performance measure $J(\theta)$ with respect to θ, and is fundamental to policy gradient RL methods. It states:

$$\nabla J(\theta) \propto \sum_s \mu^{\pi_\theta}(s_t) \sum_a Q^{\pi_\theta}(s_t, a_t) \nabla \pi_\theta(a|s) \tag{3.11}$$

$$= \mathbb{E}_{\pi_\theta}\left[G_t \nabla \log \pi_\theta(a_t|s_t) \right] \tag{3.12}$$

as proved in [212]. $\mu^{\pi_\theta}(s)$ is the state probability distribution when acting under policy π_θ and G_t is the return, as defined in Equation 3.3. Equation 3.12 provides a means of estimating the policy gradient based on samples of states, actions and rewards observed when following policy π_θ. The expectation is thus taken over states and actions when following π_θ.

Combining Equation 3.10 and Equation 3.12 produces the REINFORCE [213] update, which is the simplest policy gradient method:

$$\theta_{t+1} = \theta_t + \alpha G_t \nabla \log \pi_\theta(a_t|s_t) \tag{3.13}$$

To implement REINFORCE, the agent simulates an episode following policy $\pi_\theta(a|s)$, generating a sequence of states actions and rewards:

$$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T \tag{3.14}$$

Then for each timestep t, the return G_t is calculated using the discounted sum of future rewards defined in Equation 3.3. When π_θ is a neural network, the gradient of $\log \pi_\theta(a_t|s_t)$ with respect to θ can be computed efficiently using backpropagation. The update increases the probability of actions which resulted in large returns G_t in a way that is inversely proportional to its original probability. Low probability

actions with high returns are therefore promoted most strongly.

While REINFORCE is the simplest policy gradient method, by using Monte Carlo estimates of the return G_t (i.e. without bootstrapping) it generally has very high variance and poor sample efficiency [29]. Implementations of REINFORCE and other policy gradient methods often sample a 'batch' of multiple episodes and calculate the mean policy gradient over the batch to reduce the variance of updates. In addition, a simple approach to reducing the variance of updates is to learn an approximation of $V^\pi(s_t)$ (known as a baseline), and replace G_t in the REINFORCE update (Equation 3.12) with $G_t - V^\pi(s_t)$. Intuitively, $G_t - V^\pi(s_t)$ is an estimate of the quality of the action trajectory relative to what was expected. REINFORCE with baseline is still far from the state-of-the-art and has relatively high variance compared to more advanced actor-critic methods that use bootstrapping [29].

3.4.2 Actor-Critic Methods

Several policy gradient expressions that are analogous to Equation 3.12 have been proposed which achieve better training performance than REINFORCE or REINFORCE with baseline. These policy gradient expressions are of the form [204]:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\Psi_t \nabla \log \pi_\theta(a_t|s_t) \right] \tag{3.15}$$

where $\Psi_t = G_t$ for REINFORCE and $\Psi_t = G_t - V^\pi(s_t)$ for REINFORCE with baseline. Those which involve learning a value function and use bootstrapping (where value estimates are based on the value estimates of future states) are known as *actor-critic* methods, where the learned value function (critic) is used to calculate Ψ_t. The actor-critic framework forms the basis of a wide range of state-of-the-art policy gradient methods including soft-actor critic (SAC) [36], A3C [53], trust region policy optimisation (TRPO) [34], and proximal policy optimisation (PPO) [35] which is used in our research and discussed in detail in Section 3.4.3.

One common policy gradient expression in the form of Equation 3.15 uses an *advantage function* $\Psi_t = A^\pi(s_t, a_t)$, which (similar to REINFORCE with baseline) quantifies the value of taking action a_t relative to other actions from state s_t. The advantage function can be formulated in terms of action-value and state-value functions:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t) \tag{3.16}$$

Using the advantage function, the policy gradient encourages better-than-average actions (those having $A^\pi(s_t, a_t) > 0$). However, action-value and state-value functions are not easily known and must be approximated, often by a neural network. Since $Q(s_t, a_t) = R_{t+1} + \gamma V(s_{t+1})$, the advantage is often expressed in terms of the one-step return (i.e. the immediate reward) and the state-value function:

$$A^{\pi}(s_t, a_t) = R_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \tag{3.17}$$

The learned state-value function is known as the *critic*, which is used to estimate the advantage function. Note that $A^{\pi}(s_t, a_t)$ involves bootstrapping: the advantage is calculated by summing the one-step return and the value function for the following state, rather than the full return G_t used in REINFORCE with baseline. By learning a value function which is used to bootstrap, using the advantage function in Equation 3.17 gives an actor-critic method, whereas REINFORCE with baseline (which does not use a value function (critic) for bootstrapping) is not.

Multi-step Advantage Estimation

Equation 3.17 gives the one-step advantage of choosing a_t over other actions from state s_t. That is, we use the immediate reward R_{t+1} and the state-value function of the following state $V^{\pi}(s_{t+1})$. A more general advantage uses the n-step return, $G_{t:t+n}$:

$$A^{\pi}(s_t, a_t) = G_{t:t+n} - V^{\pi}(s_t) \tag{3.18}$$

$$G_{t:t+n} = \sum_{l=1}^{n} \gamma^{l-1} R_{t+l} + \gamma^{n} V^{\pi}(s_{t+n}) \tag{3.19}$$

With n-step return we take the discounted sum of the next n observed rewards, and bootstrap the remaining steps of the episode using the state-value function $V^{\pi}(s_{t+n})$. Using the n-step return to calculate the advantage function as in Equation 3.18 and varying n can be used as a way to trade-off bias and variance. Setting $n = \infty$ yields the REINFORCE with baseline update, which has low bias and high variance. With $n = 1$, updates have high bias and low variance.

Generalised Advantage Estimation

Instead of using the n-step return to calculate the advantage as in Equation 3.18, it is common to use generalised advantage estimation (GAE) [204]. GAE uses the λ-return G_t^{λ} [29], which is an exponentially weighted average of n-step returns:

$$G_t^{\lambda} = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n} \tag{3.20}$$

The GAE is calculated by comparing the λ-return to the learned value function:

$$A(s_t, a_t) = G_t^{\lambda} - V(s_t) \tag{3.21}$$

GAE has been shown to result in more stable training than calculating the

advantage using the n-step return [204] and is currently the standard technique for advantage estimation [214].

3.4.3 Proximal Policy Optimisation

We will now introduce proximal policy optimisation (PPO) [35], one of the most widely-used actor-critic algorithms. We use PPO in order to train the guided tree search algorithms developed in Chapters 4 and 5. PPO prevents large policy updates from drastically impacting policy performance, a problem which has been observed in other policy gradient methods such as A3C [53]. For power system applications, high variance, unpredictable convergence during training can have substantial real-world impacts on system reliability.

As described in Section 3.3.1, policy gradient methods are generally less sample efficient than value-based methods. Many policy gradient methods improve sample efficiency by performing multiple gradient updates using Equation 3.13 on a single batch of data. However, the performance $J(\theta)$ can be sensitive to small changes in the policy parameters θ, and multiple updates can therefore destabilise training and lead to catastrophic performance losses in practice [35]. Trust region policy optimisation (TRPO) and proximal policy optimisation (PPO) are two state-of-the-art actor-critic algorithms which tackle this problem by enforcing a constraint on updates to ensure that the updated policy remains similar to the original. As a result, the performance is more resilient to multiple policy gradient steps.

TRPO enforces a constraint based on the Kullback-Leibler (KL) divergence between the original policy $\pi_{\theta_{\text{old}}}$ and the updated policy π_θ. TRPO aims to maximise the surrogate objective:

$$J^{\text{TRPO}}(\theta) = \mathbb{E}_{\pi_\theta}[r_t(\theta)A_t] \tag{3.22}$$

subject to (*trust region* constraint):

$$\mathbb{E}[\text{KL}(\pi_{\theta_{\text{old}}}, \pi_\theta)] \leq \delta \tag{3.23}$$

where δ is a constant tolerance parameter defining the size of the trust region. For brevity, the advantage function is written as $A_t = A(s_t, a_t)$. r_t is the probability ratio between the two policies:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \tag{3.24}$$

As a result, the TRPO objective in Equation 3.22 measures how the new policy π_θ performs relative to the old policy θ_{old}. As in Equation 3.12, the expectation is taken over states and actions when following π_θ.

Based on a similar principle, proximal policy optimisation (PPO) [35] was

proposed with a similar goal of preventing catastrophic performance decreases while allowing for multiple updates to be performed on a single batch of transitions. The objective function for PPO is:

$$J^{\text{PPO}}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{3.25}$$

The PPO objective takes the minimum of unclipped and clipped objective functions. The unclipped objective is identical to the TRPO objective (Equation 3.22). The clipped objective truncates $r_t(\theta)$ to be between $[1-\epsilon, 1+\epsilon]$. By taking the minimum of these two terms, PPO ensures that the update is 'pessimistic'. That is, when $A_t > 0$, the probability ratio is clipped at $1+\epsilon$, while when $A_t < 0$, r_t is clipped at $1-\epsilon$, thus preventing large, greedy updates.

The PPO implementation is simpler and more general than TRPO, allowing for parameters to be shared between actor and critic networks. Empirically, it has been shown to perform as least as well as TRPO on a wide range of tasks [35] and has become a very widely used, state-of-the-art RL algorithm [214–218].

3.4.4 Entropy Regularisation

Policy optimisation often suffers from premature convergence to local optima with additional techniques to promote exploration. As a result, it is common to include an additional term to actor-critic objective functions to promote higher entropy policies. For a policy π, the entropy is defined as:

$$H(\pi) = \mathbb{E}_{a\sim\pi(\cdot|s)}[-\log \pi(a|s)] \tag{3.26}$$

An additive term that is proportional to $H(\pi)$ may be included in any actor-critic objective function of the form described in Equation 3.15. The entropy regularised objective function for PPO is:

$$J^{\text{PPO+}H} = \mathbb{E}[J^{\text{PPO}}(\theta) + \beta H(\pi_\theta)] \tag{3.27}$$

where β is a constant parameter controlling the amount of entropy regularisation. The entropy of a random variable X is defined as:

$$H(X) = -\sum_i^n (P(x_i)) \log P(x_i) \tag{3.28}$$

where $P(x_i) = \Pr X = x_i$. The entropy function of a Bernoulli random variable (i.e. a coin flip) X is shown in Figure 3.1. An unbiased coin (that is, a Bernoulli random variable with $\Pr(X=0) = \Pr(X=1) = 0.5$) has maximum entropy, while a biased coin has lower entropy. Entropy regularisation in Equation 3.27 promotes policies which more evenly distributed policy mass across actions and is generally

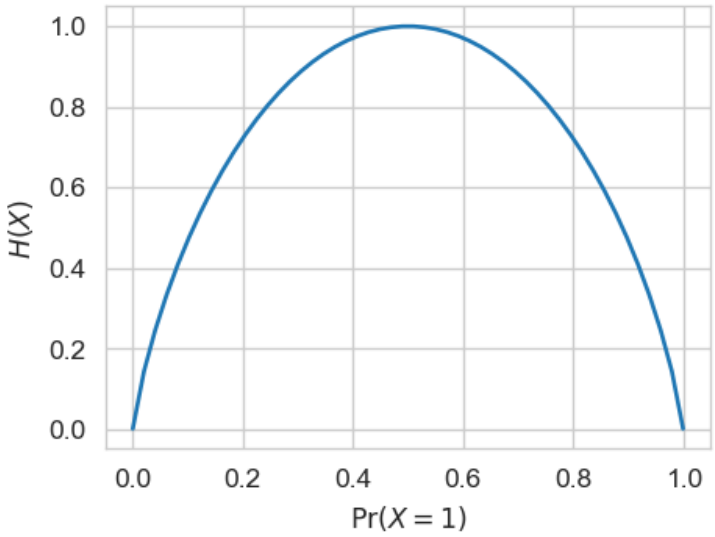


Figure 3.1: Entropy $H(X)$ of a Bernoulli random variable (i.e. a coin flip). Entropy is largest when X is fair $\Pr(X = 1) = 0.5$, and decreases as X becomes more biased.

used to encourage more stochastic behaviour in training, preventing the agent from converging prematurely to a local optimum [53].

In Section 4.5.3 we use PPO with entropy regularisation to train an RL agent to solve the UC problem. We then combine the trained agent with tree search methods, introduced in the next section, to produce 'guided tree search'.

3.5 Background to Tree Search

Sections 3.2–3.4 focused on RL, which is used to solve sequential decision-making problems formulated as MDPs. In particular, we focused on model-free policy gradient methods in Section 3.4 which are used to train the guided tree search algorithms developed in later chapters. In this section, we will focus on tree search, a class of model-based planning algorithms for decision-making and a key component of guided tree search. Rather than using trial-and-error as in RL, tree search uses one-shot decision-making and does not learn from prior experience. However, the fields of tree search and RL are closely related and have been combined in powerful algorithms such as AlphaGo [27, 44] and MuZero [28]. In the context of the UC problem, exploiting a model via tree search methods enable the agent to evaluate the outcome of actions or action sequences with respect to a model during decision-making, allowing for more robust planning. We will discuss the theory relevant to this thesis and focus on the tree search algorithms employed in Chapters 4–6. For a more comprehensive introduction to the broader field of tree search, we refer the reader to Chapters 3 and 4 of [61].

3.5.1 Definitions

Tree search problems are concerned with finding the lowest cost path from an initial node to a goal node [219]. The path is a sequence of edges taken to traverse the tree, equivalent to the actions defined in the MDP setting. Tree search algorithms rely on formulating the problem as a *search tree*, where nodes represent states and edges represent actions. The initial state is called the root node, and is the root of the search tree. A transition function (defined identically to MDPs), determines the state that follows a state-action pair. Crucially, a model of the transition function is always available to the decision-maker in a tree search problem, unlike in model-free RL where the transition function can only be sampled by interacting with the environment. In addition to states, actions and transitions, a tree search problem additionally requires the definition of a goal test function, which can be consulted to determine whether a node is a goal node. Examples of problems which can be formulated as tree search problems include:

- Route-finding [220]: the objective is to find the fastest route (least cost path) from an initial location (root node) to a destination (goal node).
- Cargo loading (knapsack problem) [221]: beginning with an inventory of items with set value and weight (root node), load items until the total weight of the loaded cargo is as large as possible (goal node). The objective is to maximise the value of the cargo loaded (least cost path).
- Rubik's Cube [222]: beginning with a scrambled initial configuration of the Rubik's Cube (root node), twist the faces until all of the faces have a uniform colour (goal node). The optimal solution is the one which uses the least number of twists (least cost path).

An example of a generic search tree is shown in Figure 3.2. The green nodes represent goal nodes, and the numerical values indicate step costs. Leaf nodes are those which have no children, and the node at the top of tree (that which has no parents) is the root node. In Figure 3.2, dotted lines indicate that the search tree is incomplete and further actions are available. The lowest cost path in the example search tree takes the following branches: [left (3), middle (1), right (2)], which has a total path cost of 6. Note that by convention, tree search generally considers costs of transitioning between states via actions, while MDPs uses rewards which are equivalent to negative costs.

A tree is a special type of directed acyclic graph (DAG), one where each node has each exactly one parent. As a result, the path from the root to any other node is unique and can be found by following the path from child to parent, beginning at the destination node. A possible brute force solution to a tree search problem is therefore to enumerate all nodes, consult the goal test function to determine the

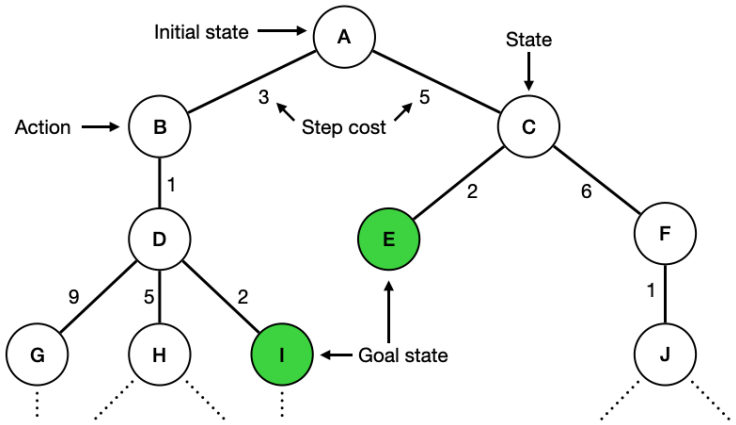


Figure 3.2: Example search tree. Nodes represent states, edges represent actions, numeric values represent costs. Dotted lines represent further branches that have not yet been added to the tree. The lowest cost path from the root node to a goal node takes the following branches, with corresponding step costs: [left (3), middle (1), right (2)].

goal nodes, and compute the lowest cost path that leads to a goal nodes. However, many problems contain an extremely large number of states, making this approach intractable. Tree search algorithms, like those described in Section 3.6, build a search tree beginning at the root in a principled way to efficiently find the optimal solution path, visiting as few nodes as possible.

Although the field of tree search is independent from RL, some MDPs used to represent problems for RL can be reformulated as a search tree and solved with tree search methods. Next, we discuss how MDPs can be formulated as search trees, allowing RL, tree search and hybrid methods to be applied.

3.5.2 MDPs as Search Trees

This thesis focuses on hybrid RL and tree search methods to solve the UC problem. As introduced in Section 3.2.2, RL requires an MDP formulation of the problem, while tree search methods require a search tree. The translation of MDPs to search trees has been employed in the games-playing literature, notably in [27, 28, 44, 54], which use Monte Carlo tree search to solve games such as Go. Similarly, we formulate the UC problem as an MDP in Section 4.3 and as a search tree in Section 4.4.1.

As in previous work [27, 28, 44, 54], we focus on MDPs with deterministic transition functions, and discrete states which can be represented by search trees of the form visualised in Figure 4.7. We discuss the deterministic representation of a search tree for the UC problem in Section 4.4.1. States and actions can be mapped directly on to nodes and states in the search tree context under these circumstances, with terminal states represented by goal nodes. MDPs with stochastic transition functions can be represented by And/Or trees, which are solved with a different

class of methods. We refer the reader to [219] for a description of And/Or trees and solution methods.

A further requirement of the tree search methods used in this thesis, which we describe in Section 3.6 is for non-negative step costs, or non-positive rewards. This constraints is naturally satisfied in the UC problem context, which is typically represented as a cost minimisation problem such as in the deterministic, stochastic and robust formulations given in Chapter 2. Methods for finding least cost paths in search trees with negative step costs include the Bellman-Ford algorithm [223], which is typically slower than methods which assume non-negative step costs.

The search tree representation of the UC problem is relied upon in later chapters to solve the UC problem with tree search methods. In the following section, we describe the taxonomy of methods used to solve tree search problems.

3.5.3 Taxonomy of Tree Search Methods

Numerous tree search methods exist which trade-off generalisability across problem domains, optimality guarantees, search efficiency and other factors [60, 62, 224–226]. As a result, algorithms have benefits and drawbacks that depend on characteristics of the task. Tree search methods can be classified along several dimensions, such as their exploitation of domain-specific knowledge; whether they can be used for online decision making; and if they can be interrupted. Here we introduce three classifications which are used to inform the development of guided tree search methods in Chapter 4 and Chapter 5.

Informed Search

Informed search methods use domain-specific knowledge about the problem to more efficiently reach a solution. A heuristic function $h(n)$ is used, which approximates the cost of the optimal path from node n to a goal node (sometimes called the *cost-to-go*). Examples of informed tree search methods are greedy best-first search [61] and A* search [62], which we describe in detail in Section 3.6.2. Some informed search methods have the same optimality guarantees as uninformed methods but may be much more efficient in practice. However, by relying on domain-specific heuristics, informed search algorithms cannot be applied out-of-the-box across different problem domains.

Real-Time Search

In some cases, it is not possible for an agent to solve the entire tree search problem offline. This is a problem if the search tree is very large, or if the agent is presented with new information after each action, such that the search tree must be rebuilt from scratch. Real-time search algorithms repeatedly solve limited sub-problems (e.g. depth-limited or time-limited) at each timestep. After an action is taken, the

search is run again to a greater depth [61]. By limiting the search horizon, real-time search does not carry optimality guarantees of other methods but may be much more practical to implement and can enable the application of tree search to problems with large state spaces.

Anytime Search

Anytime (or interruptible) algorithms can be terminated at any point and return a solution, with solution quality improving with run time [227]. Non-anytime algorithms such as uniform-cost search and A* search terminate once a goal node is found. As a result, a solution cannot be retrieved until the algorithm has run to completion, which can be problematic for time-constrained problems. Many tree search methods for games-playing such as Monte Carlo tree search [226] and iterative deepening (discussed in Section 3.6.3) are anytime. While anytime approaches may be necessary for real-world time-constrained applications, they do not guarantee optimal solutions.

3.6 Tree Search Algorithms

Having introduced the key concepts and taxonomy of tree search algorithms in Section 3.5, in this section we will provide a detailed description of the tree search algorithms employed in the novel guided tree search algorithms developed in Chapters 4 and 5. We will cover uniform-cost search [60], an uninformed algorithm; A* search [62], an informed algorithm; and iterative deepening search algorithms [224], a class of algorithms that can be considered anytime search algorithms.

3.6.1 Uniform-Cost Search

Uniform-cost search (UCS) [60] is a general-purpose, uninformed search algorithm that can be used to find the shortest path from an initial node to a goal node. Other simple uninformed methods such as breadth-first search (BFS) [228] and depth-first search (DFS) [229] are applicable only to search trees where all step costs are equal. Therefore, BFS and DFS are appropriate for tree search problems where the objective is to reach a goal node in the least number of actions. By contrast, UCS is also applicable to search trees with non-uniform step costs, which is an essential property of the search tree formulation of the UC problem, which we describe in Section 4.4.1.

UCS is derived from Dijkstra's algorithm [60]. Dijkstra's algorithm is used to find the shortest path from a root node to *all* other nodes, whereas UCS finds the shortest path to a goal node only, and terminates once this is found. The UCS algorithm is described in Algorithm 1. UCS relies on a *priority queue* of nodes (sometimes known as the frontier) that orders nodes by their *path cost* $g(n)$, the sum of costs of reaching n from the root. At each iteration of the main loop, the node with lowest path cost is removed from the queue, and the goal test is applied. If the

Algorithm 1 Uniform-cost search algorithm for finding shortest path from root node r to a goal node.

```
function UniformCostSearch(r)
    q ← priority queue containing root node r
    loop
        remove first node from queue and assign to n
        if n is a goal node then
            return path to n
        end if
        for action a available from n do
            c ← child node following a from n
            g(c) ← cost of path to c
            add c to priority queue q with cost g(c)
        end for
    end loop
end function
```

Iteration	Selected Node n	Queue q
0 (start)	-	[A(0)]
1	A	[B(3), C(5)]
2	B	[D(4), C(5)]
3	D	[C(5), I(6), H(9), G(13)]
4	C	[I(6), E(7), H(9), F(11), G(13)]
5	**I**	[E(7), H(9), F(11), G(13)]

Table 3.1: Uniform-cost search solution to the search tree in Figure 3.2. Each row is an iteration of the main loop in Algorithm 1. The selected node n corresponds to the node removed from the priority queue q at the beginning of the looped routine. q represents the priority queue at the end of the routine.

node n is a goal node, the algorithm terminates, returning the path to n. Otherwise the node is *expanded* by adding all subsequent child nodes from n to the priority queue. The first goal node that is reached using UCS is guaranteed to return the optimal (i.e. lowest cost) path. Hence, UCS is an *optimal* search algorithm [61].

As an example, we will apply UCS to the search tree in Figure 3.2. Table 3.1 shows the selected node n at the beginning of each iteration in Algorithm 1, and the priority queue q at the end of the iteration. Elements in q are ordered by their path cost (in brackets). The algorithm begins with a priority queue consisting of only the root node q = [A(0)]. At each iteration, the first element of the queue is removed and assigned to n. The children of n are added to the tree and to q, ordered by their path cost. Once node I is reached, the algorithm terminates as this is a goal node. The path to I is returned: [A → B → D → I], which is the lowest cost path.

In Section 4.5 we present Guided UCS, the first guided tree search algorithm presented in this thesis. Guided UCS is based on the UCS algorithm described in this section, using a RL-trained policy to reduce the branching factor of the search

Algorithm 2 A* search for finding shortest path from root node r to a goal node.

```
function A*Search(r)
    q ← priority queue containing root node r
    loop
        remove first node from queue and assign to n
        if n is a goal node then
            return path to n
        end if
        for action a available from n do
            c ← child node following a from n
            h(c) ← heuristic estimate of optimal cost-to-go
            g(c) ← cost of path to c
            add c to priority queue q with cost g(c) + h(c)
        end for
    end loop
end function
```

tree.

3.6.2 A* Search

A* search [62] is an *informed* search method (see Section 3.5.3) that uses a problem-specific heuristic to improve the search efficiency. The heuristic $h(n)$ estimates the optimal *cost-to-go* from node n to a goal node $h^*(n)$. A* search is very similar to UCS, except that the priority queue is ordered by $f(n)$, the sum of the path cost $g(n)$ and the heuristic estimate of the cost-to-go $h(n)$:

$$f(n) = g(n) + h(n) \tag{3.29}$$

Nodes are expanded in an order that considers both the observed path cost $g(n)$ and the anticipated future costs $h(n)$. By contrast, UCS always chooses the node with the lowest observed path cost $g(n)$ to expand first and is more short-sighted as compared with A*. Note that UCS is a special case of A* search where $h(n) = 0$ for all n. Pseudocode for the A* search algorithm is shown in Algorithm 2.

A* search is optimal if the heuristic $h(n)$ is *admissible*, meaning it strictly underestimates the optimal cost-to-go $h^*(n)$ [230]:

$$h(n) \leq h^*(n) \tag{3.30}$$

With an admissible heuristic, A* search will return the same solution as UCS, but can do so more efficiently if an accurate heuristic is used. To demonstrate this, we will apply A* search to the example search tree in Figure 3.2. Table 3.2 shows the iterations of the main loop in A* search, as we showed for UCS in Table 3.1. Illustrative values for the estimated cost-to-go $h(n)$ of each node in the search

Iteration	Selected Node n	Queue q	h(n)	
0 (start)	-	A(5)	A: 5	F: 2
1	A	B(6), C(7)	B: 3	G: 3
2	B	D(5), C(7)	C: 2	H: 4
3	D	I(6), C(7), H(13), G(16)	D:1	I: 0
4	**I**	C(7), H(13) G(16)	E: 0	J: 1

Table 3.2: A* search solution to the search tree in Figure 3.2, as well as a lookup table for heuristic values $h(n)$. Note that A* search requires one fewer iteration to reach the same optimal solution as UCS (Table 3.1).

tree are shown as a lookup table. Note that due to the large state space of many problems, a lookup table is not practical and the function $h(n)$ must be evaluated as-and-when it is needed. Compared with UCS, A* search requires one fewer iteration of the main loop, as node C is never expanded, but reaches the same solution. In terms of number of nodes expanded, A* is more efficient. However, an important consideration is the run time of the heuristic itself: if the heuristic function is slow to evaluate, then the run time reduction from fewer node expansions can be cancelled out by running the heuristic function. We discuss the heuristic properties impacting efficiency improvements of A* search relative to UCS in Section 3.6.4.

In Section 5.2.1, we present Guided A* search, a novel RL-aided algorithm based on A* search.

3.6.3 Iterative Deepening Algorithms

Iterative deepening [224] is a general search strategy, whereby a tree search is conducted repeatedly up to a limited horizon, with the horizon increasing at each iteration. At the first iteration, a search is conducted up to a depth of $H = 1$. At each subsequent iteration, the depth H is incremented and the search tree is discarded and rebuilt from scratch. The solution proceeds until some stopping criterion (such as a run time limit or based on solution quality) is met. Psuedocode for applying iterative deepening to a generic tree search function f such as UCS or A* search is shown in Algorithm 3. By stopping the algorithm according to a time budget, iterative deepening algorithms can be made anytime (Section 3.5.3). This approach has been widely used to create anytime games playing algorithms [231]. Section 5.2.2 describes Guided IDA* search.

3.6.4 Properties of Heuristics for A* Search

The main characteristic of informed search methods such as A* search described in Section 3.6.2 is their use of heuristics. The choice of heuristic has a significant impact on performance [232] as well as the optimality of informed algorithms due to the admissibility criterion in Equation 3.30. In this section we discuss the properties

Algorithm 3 General purpose iterative deepening for a tree search function f, for a problem with initial state r. At each iteration, f is used to solve the search tree up to a depth H and the search horizon is incremented.

```
function IterativeDeepeningSearch(r, f)
    H ← 1
    repeat
        solution ← f(r, H)
        H ← H + 1
    until stopping criterion is met
    return solution
end function
```

of effective heuristics for informed search.

Many problems have well-established heuristics. In route planning problems, a common heuristic is the straight-line distance from n to the destination [233]. Expert pattern databases have been used in some problems, such as the Rubik's cube puzzle [222]. Supervised learning has also been used to learn $h(n)$ for route planning problems [234, 235]. In unstudied domains without established heuristics, heuristic design is an important research topic that can have significantly impact performance in practice. In Section 5.3, we develop problem-specific heuristics for the UC problem in order to apply A* search.

The effectiveness of heuristics in improving search efficiency of A* search relative to UCS is dependent on the following heuristic properties:

Run time Heuristic run time is the average time taken to evaluate $h(n)$. Complex methods may be used to accurately estimate $h(n)$, but may be impractically slow to calculate. Since informed search methods are used to improve search efficiency by reducing the number of node evaluations required to reach a solution, a large heuristic run time may offset efficiency improvements achieved from fewer node evaluations. Run time is therefore an important heuristic property impacting the efficiency improvements achieved in practice by informed search methods.

Admissibility As described in Section 3.6.2, heuristic admissibility is a necessary condition for A* search to be optimal [230]. The admissibility criterion states that the heuristic must not overestimate the optimal cost-to-go $h^*(n)$:

$$h(n) \leq h^*(n) \quad \forall n \tag{3.31}$$

While admissibility is necessary to guarantee the optimality of A* search, in some contexts an inadmissible heuristic may still be effective in practice if optimal solutions are not required [235]. This is due to efficiency improvements resulting in lower execution time of A* search, which may be more valuable than higher solution quality in practice.

Accuracy Heuristic accuracy measures how well $h(n)$ is able to approximate $h^*(n)$ [61] and is an important factor influencing the efficiency improvement achieved by A* search relative to UCS, its uninformed counterpart. Accuracy is measured by an error metric, such as mean absolute error or root mean squared error. Perfect heuristics where $h(n) = h^*(n)$ are oracles that can be used in A* to immediately find the optimal solution without exploring any sub-optimal sub-trees, yielding maximal efficiency improvement. By contrast, in the case where $h(n) = 0$, A* is equivalent to UCS and no efficiency improvement is achieved.

In conclusion, efficiency improvements achieved in practice by informed search relative to uninformed search depends collectively on all three properties: run time, admissibility and accuracy. Accurate and admissible heuristics can be impractical if the heuristic is slow to compute. Inadmissible heuristics can be effective if accuracy is high and run time is low due to the practical value of lower run times. In designing an effective heuristic, the three properties must usually be traded off and heuristics evaluated experimentally to determine effectiveness in a given use case.

We use the methods described in this section as the basis of guided tree search algorithms developed in this thesis. In the following section, we describe mathematical optimisation methods for the UC problem that are used to benchmark the performance of guided tree search methods.

3.7 Mathematical Optimisation for Unit Commitment

RL and tree search covered in Sections 3.2–3.6 form the basis of the original solution methods developed in this thesis. In this section we will cover conventional mathematical optimisation methods to solve the UC problem. We begin by covering priority list methods, heuristic algorithms for UC. In Section 3.7.2, we cover mixed-integer linear programming (MILP), which is the predominant method for solving the UC problem, as discussed in Section 2.2.2. We use MILP in Section 4.2 to produce benchmark solutions to UC problem instances. Finally, in Section 3.7.3 we will describe the lambda-iteration for economic dispatch, the task of determining the lowest cost setpoints of generators to satisfy demand. The ED problem is an important component of the power systems simulation environment described in Section 4.2.

3.7.1 Priority List

Priority list (PL) methods for the UC problem are based on heuristics and use a simple ordering of generators (known as the PL) to commit generators, typically in order of start cost, fuel cost or capacity, initially ignoring inter-temporal constraints [1]. We

Algorithm 4 Priority list algorithm for the UC problem with demand forecast $\boldsymbol{D}$ and reserve volumes $\boldsymbol{R}$ (T decision periods), and generators $\mathcal{G}$.

```
function PriorityList(D, R, G)
    x ← [0]_{N×T} (empty solution matrix)
    for t in {1..T} do
        p ← priority list ordering of generators G
        while committed capacity ≤ D_t + R_t do
            remove generator with index g from p
            x_{g,t} ← 1
        end while
    end for
    return x
end function
```

reviewed literature applying PL methods to the UC problem in Section 2.2.2. The generator with highest priority (i.e. cheapest or largest capacity) is committed first at each time period, and further generators are committed until demand (sometimes with a reserve constraint) is met. Algorithm 4 shows this procedure in pseudocode. The drawback of this algorithm is that it does not consider inter-temporal constraints such as minimum up/down time constraints, usually resulting in an infeasible schedule. An illustrative schedule produced by a PL algorithm was shown in Figure 2.1, with generators committed in decreasing priority (blue to red). PL-produced schedules are typically 'fixed' to obey constraints using expert rules or heuristics [1].

As discussed in Section 2.2.2, PL solutions to the UC problem are quick to calculate, but generally have higher operating costs than more advanced methods such as MILP. In Section 5.3, we develop three PL-based heuristics for the UC problem to deploy in informed guided tree search methods based on A* search. Thanks to their simplicity and short run times, these heuristics are able to provide a rapid estimation of the optimal operating costs for a given power system state, which significantly improves search efficiency when applied in A* search. In the following section, we provide a background to MILP for the UC problem, which we use to benchmark the guided tree search methods applied in this thesis.

3.7.2 Mixed-Integer Linear Programming for Unit Commitment

In order to provide high quality benchmark solutions for the UC problem instances used in this thesis, we use mixed-integer linear programming (MILP) to solve the deterministic UC problem. This is the dominant solution method for practical UC applications [11]. MILP benefits from very efficient solution methods such as branch-and-bound, as well as highly optimised software implementations in commercial solvers such as CPLEX. We will compare MILP solutions to the UC problem with

guided tree search in Chapters 4–6. In Section 4.2.3, we use MILP to produce benchmark solutions to 20 UC problem instances for power systems of different sizes. This section provides a truncated description of the MILP formulation of the deterministic UC problem that we will use in this thesis. For a comprehensive introduction to MILP, we refer the reader to seminal texts [174, 236].

Unit Commitment Formulation

In this thesis we formulate the deterministic UC problem as an MILP, using the model given in [140]. The extensive review of MILP formulations of the UC problem in [11] identified this formulation as computationally efficient, capable of solving large-scale problems of 1000s of generators with modest hardware in practical run times. The formulation in Equations 3.32–3.34 uses piecewise linear approximations of the fuel cost curves, which are typically quadratic. A full description of the model is given in [140]; here we will provide a truncated version for brevity:

$$\min \sum_{g \in \mathcal{G}} \sum_{t \in \mathcal{T}} (c_g(t) + CP_g^1 u_g(t) + CS_g \delta(t)) \tag{3.32}$$

subject to (see [140] for full constraints):

$$\sum_{g \in \mathcal{G}} \Big(p_g(t) + \underline{P}_g u_g(t) \Big) = D(t) \qquad \forall t \in \mathcal{T} \tag{3.33}$$

$$\sum_{g \in \mathcal{G}} r_g(t) \geq R(t) \qquad \forall t \in \mathcal{T} \tag{3.34}$$

Parameters:

- $g \in \mathcal{G}$: the set of thermal generators
- $t \in \mathcal{T}$: the set of time periods
- $\underline{P}_g$: minimum power output of generator g
- CP_g^l: cost of operating at piecewise generation point l for generator g
- CS_g: startup cost for generator g at time t
- $D(t)$: demand net of wind generation at time t
- $R(t)$: reserve requirement at time t

Variables:

- $c_g(t)$: cost of power produced over minimum for generator g at time t
- $p_g(t)$: power output above minimum for generator g at time t

- $u_g(t)$: commitment of generator g at time t
- $\delta(t)$: startup status of generator g at time t
- $r_g(t)$: reserve provided by generator g at time t

The dominant solution technique for solving UC models such as that in Equations 3.32–3.34 is branch-and-bound [237], which is implemented in several open-source and commercial solvers. Improvements in MILP solvers such as the use of cutting planes, parallelism and branching heuristics, have contributed to branch-and-bound becoming the dominant solution method for solving the UC problem [11]. In Section 4.2.3 we use branch-and-bound to produce benchmark solutions to UC problem instances, using the open-source COIN-OR software [238]. A useful property of branch-and-bound is that at any point, a lower bound on the objective function (in the minimisation case) is given. The difference between the objective function and this lower bound is the gap, which is commonly used to define stopping criteria. Once the gap falls below a threshold (e.g. 1%), the current best solution is returned. However, in many contexts branch-and-bound may take many iterations to find a feasible solution and is therefore not a fully anytime algorithm.

The MILP formulation of the UC problem simultaneously solves both the UC problem setting the integer decision variables, as well as the economic dispatch (ED) problem setting the real-valued generator setpoints. However, in some contexts, the ED problem is considered as a separate problem, such as in the context of real-time balancing or re-dispatch of generators in response to deviations in demand from forecasts. In the following section, we describe the lambda-iteration method for solving the ED problem.

3.7.3 Economic Dispatch with Lambda-Iteration

While the UC problem is concerned with integer-valued commitment decisions, optimising the real-valued power outputs of online generators is known as the economic dispatch (ED) problem [2]. As can be seen in the MILP formulation in Equations 3.32–3.34, solutions to the UC problem typically solve both the UC problem and the ED problem simultaneously, since the objective function depends on the power outputs. However, in practice, the ED problem is often solved independently of the UC problem in the real-time operation of power systems. Generator dispatch may be required to change throughout the day in order to meet deviations in demand or renewables generation from forecasts, or other contingencies such as generator outages. The ED problem is an important component of the power system simulation environment described in Section 4.2 as it is used to determine the total fuel costs of generators for a given commitment decision under realisations of demand and wind generation.

The ED problem is concerned with finding the lowest-cost dispatch of online generators to meet a given demand, subject to generator operating limits. Mathematically, the ED problem ignoring ramping constraints and transmission losses can be described as follows:

$$\text{minimise} \qquad F = \sum_{i \in \mathcal{G}} F_i(p_i) \tag{3.35}$$

$$\text{subject to} \qquad \phi = D - \sum_{i \in \mathcal{G}} p_i = 0 \tag{3.36}$$

$$p_i^{\min} \leq p_i \leq p_i^{\max} \tag{3.37}$$

where $\mathcal{G}$ is the set of online (committed) generators; F_i is the fuel cost function for generator i; p_i is the power output; $p_i^{\min}$ and $p_i^{\max}$ are minimum and maximum operating limits and D is the demand.

Whereas the UC problem is NP-hard, the ED problem is typically formulated as a constrained convex optimisation problem and can be solved quickly with a wide range of methods [239]. A common numerical method for solving the ED problem is the lambda-iteration method. In this section, we will describe the lambda-iteration method for solving the economic dispatch as described in [2]. We consider quadratic fuel cost curves of the form:

$$F_i(p_i) = a_i p_i^2 + b_i p_i + c_i \tag{3.38}$$

where a_i, b_i, c_i are constant coefficients. Fuel costs are often modelled using quadratics [5], and we use this model in our UC problem setting in Section 4.2.

The lambda-iteration method begins by constructing a Lagrange function:

$$\mathcal{L} = F + \lambda \phi \tag{3.39}$$

where λ is a Lagrange multiplier and ϕ is the load balance equality in Equation 3.36. Finding the stationary points of $\mathcal{L}$ with respect to the variables p_i and the Lagrange multiplier is equivalent to finding the extreme of the objective function F while observing the constraint ϕ [240]. Using this fact, taking the partial derivative of $\mathcal{L}$ with respect to p_i yields the following set of equations:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \frac{dF_i(p_i)}{dp_i} - \lambda = 0 \tag{3.40}$$

$$\lambda = \frac{dF_i}{dp_i} \tag{3.41}$$

To account for the inequality constraints 3.37, the following conditions are added:

$$\lambda \geq \frac{dF_i}{dp_i} \quad \text{when } p_i = p_i^{\max} \tag{3.42}$$

$$\lambda \leq \frac{dF_i}{dp_i} \quad \text{when } p_i = p_i^{\min} \tag{3.43}$$

Differentiating the quadratic fuel cost curve in Equation 3.38 and substituting into Equation 3.41:

$$\lambda = a_i p_i + b_i \tag{3.44}$$

$$p_i = \frac{\lambda - b_i}{a_i} \tag{3.45}$$

For a given value of λ, Equation 3.45 can be used to calculate the power output p_i for each generator. Then, using the constraints in Equations 3.42, and 3.43, p_i violating operating limit constraints in Equation 3.37 are updated without loss of optimality. That is, where $p_i \geq p_i^{\max}$, we set $p_i = p_i^{\max}$ and correspondingly for $p_i^{\min}$. For fixed λ, these powers given by Equation 3.45 will provide the lowest fuel cost, being an extremum of the Lagrange equation (Equation 3.39). However, not all values of λ will satisfy the load balance constraint in Equation 3.36. Since the ED problem is convex, a local minimum is also a global minimum. It follows that the optimal solution to the ED problem can be found by searching for a value of λ which yields generator outputs p_i that satisfy the load balance constraint in Equation 3.36.

The lambda-iteration method uses the Newton-Raphson method [241] to search for λ, terminating when the difference between supply and demand $|\sum_i p_i - D| < \epsilon$, where ϵ is a small tolerance value (e.g. 1 MW). At each iteration, λ is updated by interpolation between upper and lower bounds from previous estimates as shown in Figure 3.3. The power outputs are calculated using Equation 3.45 and the inequalities in Equations 3.42 and 3.43. If $\sum_i p_i - D > 0$, then λ is set to the midpoint of the two smallest values of λ (in this case λ_2 and λ_3), otherwise the midpoint of the two largest values is used.

The lambda-iteration is an efficient solution method for the ED problem and is an integral component of the power systems simulation environment described in Section 4.2, used to solve the ED problem for commitment decisions made by the decision-making agent. The resulting fuel costs are then use to evaluate the reward function in the UC MDP described in Section 4.3.

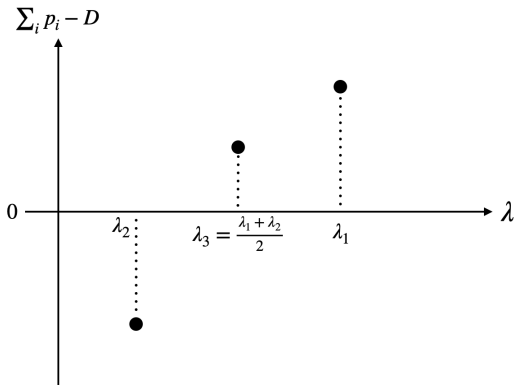


Figure 3.3: Linear interpolation of λ in the lambda-iteration algorithm, adapted from [2]. The next value λ_4 can be found by interpolating between λ_2 and λ_3. At each iteration, the difference between supply and demand $|\sum_i p_i - D|$ reduces. The algorithm terminates when the difference is below a tolerance ϵ.

3.8 Conclusion

This chapter forms the theoretical basis for the material presented in Chapters 4–6, where we use tree search and RL to develop a scalable solution method for the UC problem. We use the actor-critic method PPO described in Section 3.4.3 to train an RL agent to solve the UC problem and use the trained policy to improve the efficiency of the tree search methods described in Section 3.6. Furthermore, throughout Chapters 4–6 we benchmark the RL-based solutions against traditional deterministic approaches using MILP, based on the methods described in Section 3.7.2. The following chapter describes the power system environment used in this thesis and presents Guided UCS, an RL-aided tree search algorithm which is used to solve the UC problem.

Chapter 4

Guided Tree Search

4.1 Introduction

Real-world UC problems in large central-dispatching power markets may involve 1000s of generators [14], requiring scalable solution methods. Such large problem sizes cannot not be easily solved by the tree search methods described in Section 3.6 due to exponential growth of the search tree in the number of generators. As a result, existing tree search methods have been limited to small UC problems of up to 12 generators [55]. Practical challenges also limit the ability of model-free RL to scale to larger power systems, as shown by relatively small-scale studies reviewed in Section 2.5. Recent research has shown that RL can be exploited to significantly improve the efficiency of tree search algorithms, achieving tractability and superior performance in a number of challenging problem domains with large branching factors [27, 28, 44, 207, 208]. In this chapter we present a scalable UC solution method using a novel RL-guided tree search algorithm. We show that whereas conventional tree search has exponential run time complexity in the number of generators, solution times are roughly constant in the number of generators when using guided tree search. We find that guided tree search results in negligible increases in operating costs as compared with conventional tree search without RL and outperforms industry-standard deterministic methods based on mixed-integer linear programming (MILP).

In order to apply RL to the UC problem, this chapter describes an open-source power system simulation environment developed for this research. The UC problem is then formulated as an MDP, which can be simulated using the power system environment, allowing for RL agents to be trained to solve the UC problem by trial-and-error. In this chapter, we apply the tree search algorithm uniform-cost search (UCS), described in Section 3.6.1, to the UC problem, evaluating performance in terms of operating costs and run time using the simulation environment. We show experimentally that UCS, while achieving low operating costs, exhibits exponential time complexity in the number of generators, limiting its application to real-world power systems. We then present *Guided UCS*, which uses an 'expansion policy' trained

by model-free RL to reduce the breadth of the search tree. We show that operating costs do not significantly increase when applying an expansion policy trained by model-free RL despite removing branches, while the run time remains stable in the number of generators. We demonstrate the capability of RL to intelligently reduce the branching factor without degrading solution quality, making our approach scalable to larger power systems than studied in existing literature. Furthermore, Guided UCS is compared with industry-standard deterministic UC approaches, solved with MILP, and shown to achieve lower operating costs and better security of supply.

4.1.1 Contributions

This chapter makes the following contributions:

1. The UC problem is formulated as an MDP and realised in an open-source simulation environment. The environment, which is used throughout this thesis, uses real demand and wind data from the GB power system and is described in detail in Section 4.2.

2. Uniform-cost search (UCS) is used to solve the UC problem and shown to achieve operating costs that are competitive with deterministic MILP benchmarks. To the best of our knowledge, this is the first application of UCS to the UC problem.

3. To improve the run time complexity of UCS in the number of generators, we present Guided UCS, an RL-aided tree search algorithm which uses a guiding 'expansion policy' trained by model-free RL to reduce the branching factor.

4. We train expansion policies using proximal policy optimisation (PPO) and a sequential feed-forward neural network architecture in the simulation environment for systems of between 5–30 generators. We use the trained policies to solve 20 unseen test UC problems model-free. To the best of our knowledge, this is the first application of policy gradient RL to solve the UC problem.

5. Using Guided UCS to solve 20 unseen test problems, we conduct a parameter analysis, investigating the impact of breadth and depth parameters on run time, operating costs and schedule characteristics.

6. Guided UCS is compared with UCS and shown to exhibit constant time complexity in the number of generators with no significant degradation of solution quality.

7. Compared with the MILP benchmarks, Guided UCS is shown to achieve lower operating costs, loss of load probability and exhibit novel operational strategies.

The rest of this chapter is organised as follows. Section 4.2 describes the problem setup and power system environment used in this research. In Section 4.3, the UC problem is formulated as a Markov Decision Process, suitable for applying RL methods. In Section 4.4 UCS is described and applied to solve the UC problem. In Section 4.5, we describe the RL-aided tree search algorithm, Guided UCS, and train expansion policies with model-free RL. The policies are applied in Section 4.6, where Guided UCS is used to the solve the UC problem. We conduct a parameter analysis of guided tree search, and compare performance with UCS and MILP benchmarks. In Section 4.7 we discuss our findings and Section 4.8 concludes the chapter.

4.2 Problem Setup & Simulation Environment

This section describes the power system simulation environment (henceforth 'environment') developed for this research. In contrast with existing power system environments such as PandaPower [242], ours is specifically designed for the UC problem and models a simplified, single-bus model of a power system, in line with most existing UC research. This comes at the cost of less accurate representation of transmission-related impacts of UC, such as transmission contingencies, reactive power support and line overloadings. The benefit of the single-bus model is that the computational cost of evaluating the environment is much lower. Our decision to neglect transmission constraints is in line with most existing research and industry practice, where post-solve AC load flow analyses are conducted to determine the feasibility and security of a UC solution [11].

The environment models stochastic demand and wind and can be used to generate scenarios reflecting the uncertainty and variability of real world power systems, making it suitable for stochastic UC research. The main function of the environment is to simulate the dispatch of generators and resulting operating costs given UC decisions inputted by the user (or RL agent), under scenarios of demand and wind generation. The simulation environment is available as an open-source Python package[1].

The cost-minimising UC problem represented in the environment is reflective of centrally-dispatching power markets such as those of North America. By contrast, GB's self-dispatching market structure means that no central operator solves a UC problem to determine the commitment of all generators. However, cost-minimising UC remains the most widely-studied setup, and the environment and solution methods presented in this thesis can be modified for profit-maximising problem setups [243].

This section begins by describing the main routine executed in the environment. We then describe the data used to realise the environment, defining generator specifications and forecasts for demand and wind. Lastly, the test problems used to

[1] `https://github.com/pwdemars/rl4uc`

evaluate the relative performance of UC methods are shown, along with a description of benchmark solutions to these problems using MILP.

4.2.1 Overview

The environment models a power system of N generators with 48 30-minute settlement periods per day, reflecting GB power market structure. Given data inputs of generator specifications and demand and wind generation forecasts, the environment repeats the following routine for each of the 48 settlement periods of the day (shown as a flowchart in Figure 4.1).

First, the user (or agent) inputs a commitment decision $\{0,1\}^N$, determining the on/off statuses of generators. This decision must satisfy generator constraints (see Generator Specifications below). The environment then updates the generator up/down times according to the commitment decision and samples forecast errors for demand and wind from stochastic processes. The 'real' demand and wind generation are the sum of the forecast and forecast errors, and their difference is the net demand (wind is treated as negative demand). The environment solves the *economic dispatch problem* [2] to calculate the lowest-cost real-valued power outputs (set points) for the generators. The set points must meet the net demand if possible within the constraints of the online generators. If it is not possible to meet net demand, for instance if there is not enough capacity committed, then lost load is incurred. Finally, the operating costs are calculated as the sum of fuel costs, startup costs and lost load costs. This routine is repeated for each of the 48 decision periods, with forecasts rolling forward by 1 timestep at each iteration.

At the end of the episode, the environment returns the total operating cost for the entire day. For the same data inputs and commitment decisions, the environment may output different operating costs as different forecast errors are sampled. The environment can be used to evaluate operating costs of a unit commitment schedule under different scenarios of demand and wind.

We will now describe the key environment components in more detail.

Generator Specifications

Generators in the simulation environment are specified by the following variables:

- $p_{\min}, p_{\max}$ (MW): minimum and maximum operating limits.
- a (\$/MWh^2), b (\$/MWh), c (\$): quadratic coefficients of the fuel cost curve.
- u_0: initial up/down time in settlement periods.
- $t^{\text{down}}_{\min}$ $t^{\text{up}}_{\min}$: minimum up/down times in settlement periods.
- c^s (\$): startup cost.

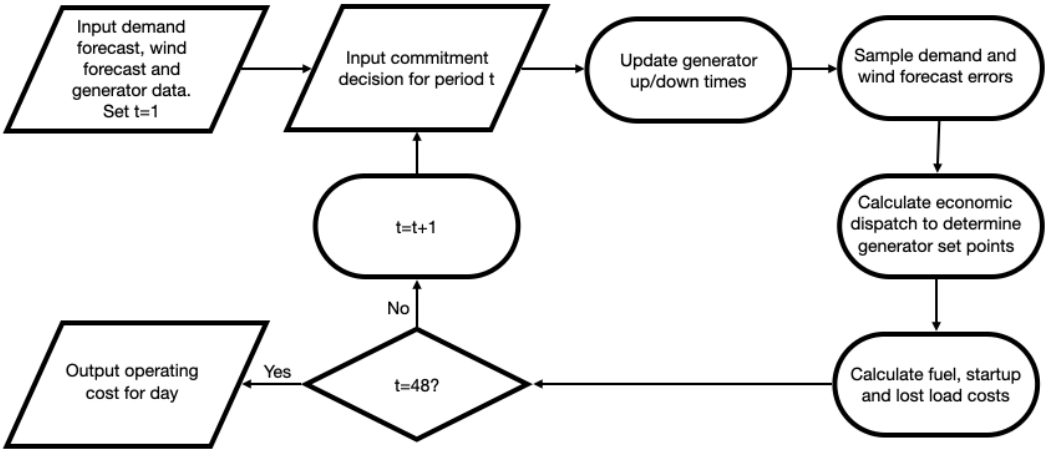


Figure 4.1: Flowchart of the simulation environment. The user inputs forecasts and generator data, and unit commitment decisions at each timestep of a 48-period day. The environment samples demand and wind scenarios and simulates dispatch by solving the economic dispatch problem. The environment outputs total operating costs at the end of the day.

The generators in the environment do not have ramping constraints; in this respect, the simulated system studied is more flexible than real-world power systems. However, the solution methods presented in this thesis are equally applicable to generators with ramping constraints.

The fuel cost for generator i is a quadratic function of the variables a, b, c:

$$C_i^f(p) = \frac{1}{2} z_i (a_i (p_i)^2 + b_i (p_i) + c_i) \tag{4.1}$$

where $z_i \in \{0, 1\}$ is the generator commitment (on or off) and where p_i is the real-valued power output in MW. The factor of $\frac{1}{2}$ is used to convert MW to MWh, given a settlement period length of 30 minutes.

At the beginning of each day, generators are initialised with up/down times (number of periods spent on/offline) defined by u_0. Positive values indicate up time, negative values indicate down time. The minimum up/down time constraints limit the extent to which generators can be committed/decommitted. An offline generator (that is with up/down time $u < 0$) cannot be committed until $u \leq -t_{\text{min}}^{\text{down}}$. Similarly, when $u > 0$, the generator must remain online until $u \geq t_{\text{min}}^{\text{up}}$. UC solutions that violate the minimum up/down time constraints are infeasible.

Startup costs are incurred whenever a generator is committed (that is, in any settlement period when $u = 1$).

Forecast Errors

For each day, forecasts $\boldsymbol{d}$ and $\boldsymbol{w}$ for demand and wind are inputs into the environment and hence pre-determined. The environment is stochastic due to the inclusion of forecast errors, which are sampled from stochastic processes. A *scenario* of demand

and wind generation depends on demand forecast errors $\boldsymbol{x} \in \mathbb{R}^{48}$ and wind forecast errors $\boldsymbol{y} \in \mathbb{R}^{48}$, sampled from stochastic processes. The demand scenario is then $\bar{\boldsymbol{d}} = \boldsymbol{d} + \boldsymbol{x}$, and the wind scenario is $\bar{\boldsymbol{w}} = \boldsymbol{w} + \boldsymbol{y}$. Finally, given that wind is treated as negative demand, a *net demand* is scenario $\bar{\boldsymbol{d}}_{\text{net}} = \bar{\boldsymbol{d}} - \bar{\boldsymbol{w}}$.

Forecast errors are modelled using autoregressive moving average (ARMA) processes, an approach that has previously been used in [244, 245]. At settlement period t, the demand forecast error $x_t \sim X_t$ is sampled from:

$$X_t = \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{d,t-i} + \epsilon_{d,t} \tag{4.2}$$

where p is the order of the autoregressive component and q the order of the moving average component. α_i and β_i are constant parameters and $\epsilon_{d,t}$ is a normally distributed random variable with mean 0 and standard deviation σ (white noise). Similarly, wind forecast errors $y_t \sim Y_t$ are sampled from an ARMA process with different parameters and with white noise $\epsilon_{w,t}$.

Economic Dispatch

The UC problem deals only with binary decision variables, giving the on/off schedules of generators. In order to determine fuel costs, it is necessary to solve the economic dispatch (ED) problem, determining the real power outputs $\boldsymbol{p} \in \mathbb{R}^N$ that meet net demand at lowest cost. The ED problem is a convex optimisation problem and is solved with the lambda-iteration method [2] in the simulation environment. We described the economic dispatch from and the lambda-iteration method in detail in Section 3.7.3. The generator outputs $\boldsymbol{p}$ are used in Equation 4.1 to calculate the fuel costs. In some cases, an ED solution that meets net demand $\bar{d}_{\text{net}}$ is not possible due to the operating limits $p_{\min}, p_{\max}$ of the online generators. In these cases, we set $p = \{p_{\min}, p_{\max}\}$ for all online generators, depending on whether there is insufficient footroom (when net demand is low) or headroom (when net demand is high). The difference between net demand and generation from thermal plants is penalised at a value of lost load (see Equation 4.8 below).

Operating Costs

For each settlement period, total operating costs C are calculated as the sum of fuel costs C^f ; startup costs C^s; lost load costs C^l:

$$C = C^f + C^s + C^l \tag{4.3}$$

Total operating costs for the day are calculated as the sum of period operating costs.

Fuel costs Fuel costs for each generator are calculated according to the quadratic cost curves defined in Equation 4.1:

$$C^f = \sum_{i=1}^{N} C_i^f \tag{4.4}$$

Startup Costs Startup costs are incurred whenever a generator is committed:

$$C^s = \sum_{i=1}^{N} \lambda_i c_i^s \tag{4.5}$$

where:

$$\lambda_i = \begin{cases} 1 & \text{if } u_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Lost load costs When available generators are unable to meet demand, a lost load cost is incurred:

$$C^l = Vl \tag{4.7}$$

$$l = |\bar{d}_{\text{net}} - \sum_{i=1}^{N} p_i| \tag{4.8}$$

where V (\$/MWh) is the value of lost load and l (MWh) is the *lost load*, the difference between supply $\sum_{i=1}^{N} p_i$ and net demand $\bar{d}_{\text{net}}$. Note that in this setup there is equal penalty for over-commitment and under-commitment of generation; in practice, the costs of over-commitment are likely to be substantially lower, and could be managed by wind shedding or other means apart from load shedding. As a result, the optimal commitments and reserve allocation strategies in this setup are likely to give greater priority to generator footroom than would be common in real power systems.

4.2.2 Data

To define the generator variables described in Section 4.2.1, we used data from Kazarlis et al. [5]. While this data is not recent (published in 1996), it is still widely-used as a benchmark power system [64, 88, 246]; we reviewed deterministic UC research which has used the Kazarlis benchmark power system in Table 2.1. In addition, the data provides complete descriptions of generator cost curves, whereas other data sources use piecewise linear approximations or assume constant efficiencies.

Gen ID	$p_{\min}$	$p_{\max}$	u_0	a	b	c	$t^{\text{down}}_{\min}$	$t^{\text{up}}_{\min}$	c^s
1	150	455	16	0.00048	16.19	1000.0	16	16	4500
2	150	455	16	0.00031	17.26	970.0	16	16	5000
3	20	130	-10	0.00200	16.60	700.0	10	10	550
4	20	130	-10	0.00211	16.50	680.0	10	10	560
5	25	162	-12	0.00398	19.70	450.0	12	12	900
6	20	80	-6	0.00712	22.26	370.0	6	6	170
7	25	85	-6	0.00079	27.74	480.0	6	6	260
8	10	55	-2	0.00413	25.92	660.0	2	2	30
9	10	55	-2	0.00222	27.27	665.0	2	2	30
10	10	55	-2	0.00173	27.79	670.0	2	2	30

Table 4.1: Generator specifications for the 10 generator problem, from [5].

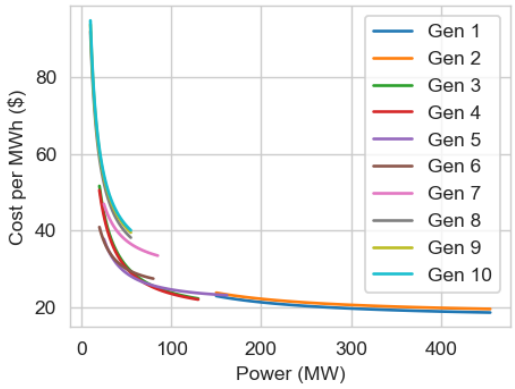


Figure 4.2: Quadratic cost curves for the 10 generators described in Table 4.1 in $ per MWh. Efficiency improves as load factor $\frac{p}{p_{\max}}$ increases.

The generator specifications from [5] are shown in Table 4.1. The quadratic cost curves for the 10 generators are shown in Figure 4.2. Generators 1 and 2 most closely reflect baseload generation, having the lowest fuel costs, largest capacities and the most restrictive minimum up/down times (8 hours). Generators 8–10 are peaking plants, with small capacity, high fuel costs and short minimum up/down times (2 hours). Larger systems are created in the simulation environment by duplicating the 10 generators, an approach followed in existing research using this power system [5, 64, 88, 246].

We set the value of lost load (VOLL) V (Equation 4.8) to be $10,000 per MWh for both training and testing, set to represent the approximate VOLL for a range of customer types [247]. This large penalty reflects the potentially catastrophic outcomes of failing to meet demand in the real world.

For demand forecasts we used National Grid Demand Data [3] from 2016–2019. Demand is scaled linearly as a function of number of generators to be between 40–100% of the total capacity $\sum_{i=1}^{N} p_{\max,i}$. For wind forecasts, we used openly available data for Whitelee onshore wind farm [4], chosen as a relatively large wind farm that operated continuously between 2016–2019. The output of a single wind farm is more

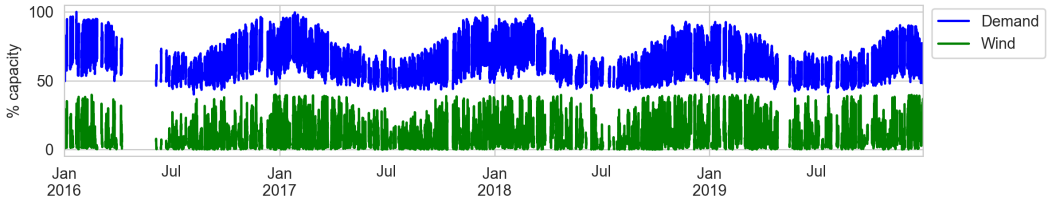


Figure 4.3: National Grid demand [3] and Whitelee wind generation data [4] used to define forecasts in the simulation environment. Demand and wind generation are scaled depending on the number of generators so are shown in terms of % of total generator capacity. Incomplete days were removed, leaving 806 complete forecasts.

volatile than the national wind generation, providing a diverse set of wind profiles for UC problems. In addition, using a single farm keeps the overall wind penetration roughly constant across the period 2016–2019, while GB-wide wind penetration increased significantly over the period. The wind generation data was scaled to be between 0–40% of the total capacity of the generation mix. We found a significant number of incomplete days in the source data for either demand or wind. We omitted data for these days entirely, resulting in a total of 806 unique days of demand and wind forecasts. The full time series for both demand and wind, showing the omitted days, is displayed in Figure 4.3.

The α and β parameters of the ARMA processes (defined in Equation 4.2) for demand and wind forecast errors were set manually and are identical for all power system sizes. Both were tuned to decay exponentially, such that there is a stronger correlation with more recent forecast errors. In both cases we set $p = q = 5$, allowing 5 steps of history to be accounted for in both autoregressive and moving average components of the ARMA process. The standard deviation parameters σ were scaled proportionally to the number of generators, such that relative to the demand and wind generation, the level of uncertainty remained roughly constant for all problems.

4.2.3 Test Problems and Benchmarks

In order to compare UC solution methods, we created a dataset of 20 test problems, sampled from the 806 complete forecasts. The remaining days form a training set of forecasts that were used to train RL agents. Whereas most UC research reports results for a single test problem such as the widely-studied demand profile in [5], more statistically robust results can be achieved by evaluating solution methods on multiple test problems. 20 test problems ensures a wide range of days with different characteristics across multiple seasons while preserving a large set of training episodes. The 20 test problems (scaled for the 10 generator power system) are described in Table 4.2 and visualised in Figure 4.4. The test problems exhibit a range of characteristics and daily wind penetration (wind generation as proportion of demand) ranges from

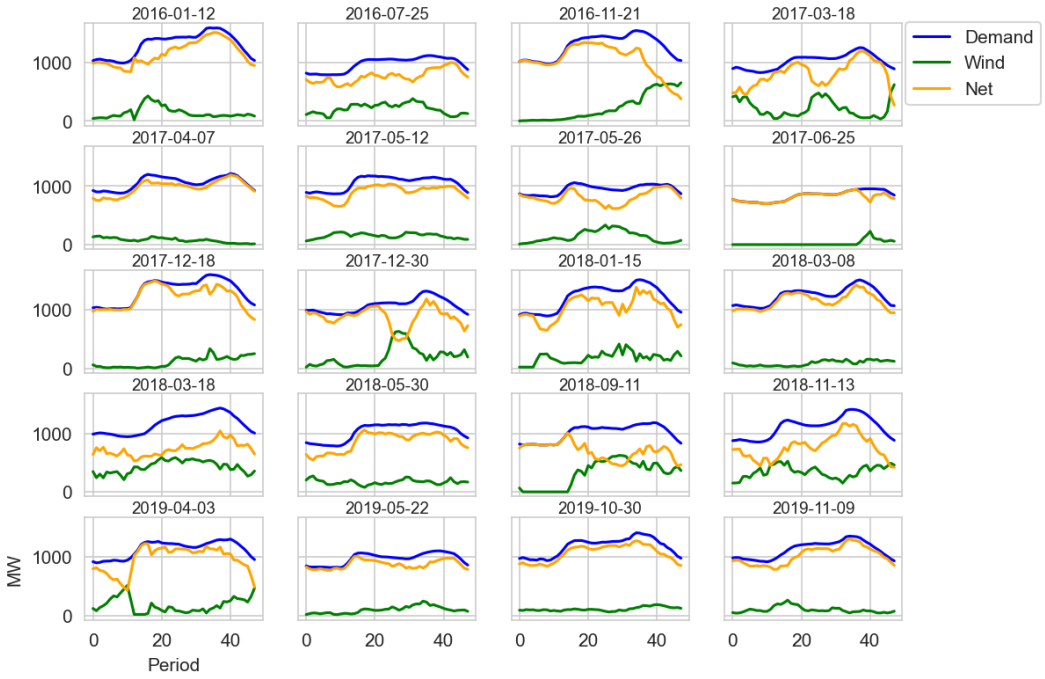


Figure 4.4: Unseen test problems, shown for the 10 generator problem.

Date	Day	Wind (%)	$D_{\min}$ (MW)	$D_{\max}$ (MW)
2016-01-12	Tue	11.0	990.9	1593.1
2016-07-25	Mon	21.7	790.2	1118.5
2016-11-21	Mon	18.1	972.5	1545.3
2017-03-18	Sat	21.3	827.5	1253.9
2017-04-07	Fri	7.4	880.2	1214.2
2017-05-12	Fri	14.4	867.5	1176.9
2017-05-26	Fri	14.8	817.4	1058.7
2017-06-25	Sun	2.8	700.3	954.1
2017-12-18	Mon	8.1	1011.6	1599.9
2017-12-30	Sat	18.4	916.4	1318.7
2018-01-15	Mon	15.0	893.3	1513.2
2018-03-08	Thu	7.0	1024.1	1509.8
2018-03-18	Sun	36.8	943.6	1432.6
2018-05-30	Wed	16.5	782.9	1177.1
2018-09-11	Tue	30.8	799.6	1183.8
2018-11-13	Tue	31.1	852.6	1409.8
2019-04-03	Wed	15.6	901.0	1306.3
2019-05-22	Wed	10.1	819.0	1104.8
2019-10-30	Wed	9.5	938.8	1414.2
2019-11-09	Sat	8.9	918.5	1356.7

Table 4.2: Summary of test profiles for the 10 generator setting, visualised in Figure 4.4

2.8% to 36.8%.

In order to provide strong benchmark solutions to the 20 test problems, we used mixed-integer linear programming (MILP) to solve a deterministic formulation [140] of the UC problem, as described in Section 3.7.2. Deterministic UC solved with MILP is widely used in industry [13]. The Power Grid Lib software package[2] was used to formulate the deterministic UC problem and we used the open-source COIN-OR library using the branch-and-cut algorithm [238] to solve the MILP. The specific formulation combines the MILP formulation described in [140] with the method for piecewise linear approximation of quadratic cost curves described in [248].

We implemented two benchmarks, the first using a reserve constraint to manage uncertainties, the second assuming perfect foresight (i.e. all forecast errors being zero) and no reserve constraint. For the first benchmark, the reserve constraint was set to be 4 times the standard deviation of the net demand forecast errors, a common industry approach described in [249]. We refer to this benchmark as MILP(4σ). The standard deviation was determined empirically by sampling from the ARMA processes for demand and wind forecast errors.

To probabilistically evaluate the quality of MILP(4σ) solutions to the test problems, we applied the following Monte Carlo approach, which has also been employed in [13, 22, 154, 173]:

1. Calculate the UC schedule using solution method (e.g. MILP, UCS, Guided UCS).

2. Use the environment to calculate operating costs for 1000 scenarios of demand and wind.

Step 2 involves repeatedly passing the UC schedule as an input to the environment (step 2 of the flowchart in Figure 4.1), and recording the operating costs on each iteration. This returns a distribution of operating costs over scenarios.

For the second, perfect foresight benchmark, the reserve was set to 0. We call this benchmark MILP(perfect). This approach only considers the point forecast, with zero forecast errors. Hence, we do not evaluate this solution over the 1000 scenarios.

The results for MILP benchmarks for the 10 generator problem are plotted in Figure 4.5, shown in terms of cost per MWh net demand to control for variations in net demand under different scenarios. Also shown are loss of load probability (LOLP; average probability of a lost load event in any settlement period) for the MILP(4σ) solution and the wind penetration of each test problem. There is significant variation in solution quality of MILP(4σ) relative to MILP(perfect). For instance, MILP(4σ) performs considerably worse on 2016-11-21, due to a high LOLP (roughly 0.5%). This problem is characterised by a sharp increase in wind generation towards the end of the day, coinciding with decreasing demand, resulting in a rapid decrease in net

[2] `https://github.com/power-grid-lib/pglib-uc`

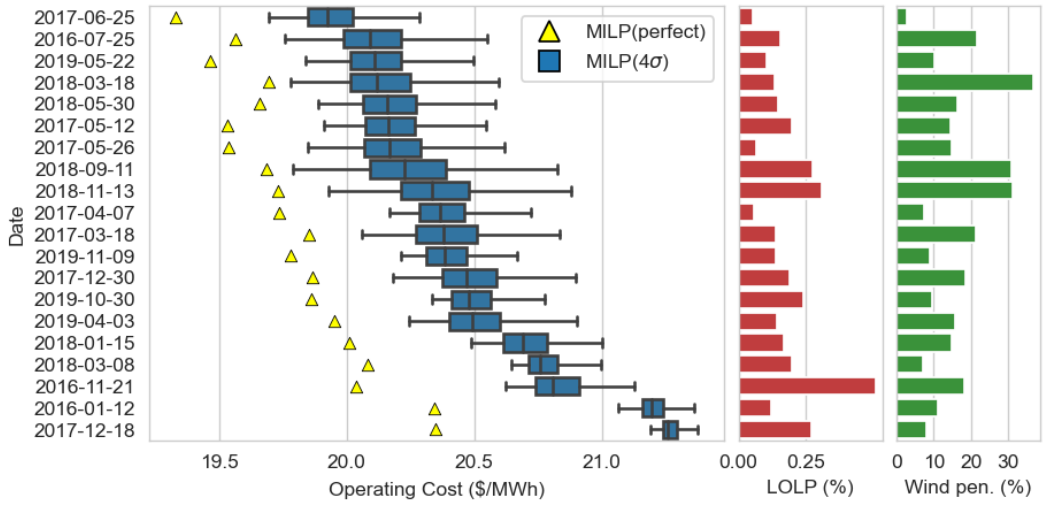


Figure 4.5: Operating costs for MILP benchmarks on the 20 test problems. The distribution of operating costs for MILP(4σ), evaluated under 1000 scenarios of demand and wind generation are shown with outliers removed. The operating costs for the MILP(perfect) solution, which considers only the point forecast, is shown in yellow. Loss of load probability (LOLP) is shown for the MILP(4σ) for each day, as well as the daily wind penetration.

demand (Figure 4.4). As a result, the UC solution is likely to encounter insufficient footroom constraints, with large lost load penalties according to Equation 4.8.

Note that the solver's optimality gap was set at 1% for the experiments. Reducing the gap to 0.01% caused a slight reduction in operating costs for the 10 generator problem (0.2%), but increased operating costs by approximately 1% for 20 and 30 generator cases. As both solutions employ the same reserve strategy and lost load contributes significantly to total operating costs, it is not surprising that reducing the gap does not strictly reduce expected costs. As a result, we adopt an optimality gap of 1% throughout this thesis.

The power system simulation environment can also be used to represent a Markov Decision Process (MDP) formulation of the UC problem, with states, actions and rewards. In this context, an agent may be trained using RL to take actions in the environment that maximise the long-run expected reward, based on contextual information about the state. In the following section, we will describe the MDP formulation of the UC problem.

4.3 Markov Decision Process Formulation

In Section 4.2 we described the power system simulation environment used to train and evaluate RL agents. We also showed how the environment is used to evaluate MILP benchmark solutions to the 20 test problems. The environment can also

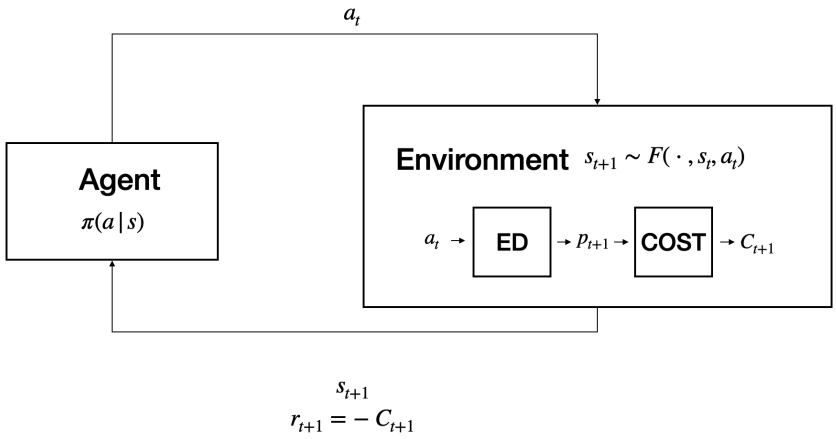


Figure 4.6: Agent-environment interaction in the UC MDP. The agent takes an action a_t, which is processed by the environment, returning a new state s_{t+1} sampled from the transition function $F(s_{t+1}, s_t, a_t)$ and a reward r_{t+1}. The action is sampled from a policy $\pi(a_t|s_t)$, considering a partial observation of the state. To calculate the reward, the environment solves the economic dispatch (ED) and evaluates the cost function, as described in Section 4.2.1. The reward is the negative operating cost: $r_{t+1} = -C_{t+1}$.

be used to represent the UC problem as a partially-observable Markov Decision Process (MDP), which we will describe in this section. MDPs, which consist of states, observations, action, rewards and a transition function, were described in detail in Section 3.2.2. Formally describing the UC problem as an MDP allows for the application of RL methods, with an agent learning by trial and error in the simulation environment described in Section 4.2. The agent interacts with the environment by scheduling generators based on uncertain forecasts for demand and wind generation, aiming to minimise operating costs. A schematic of this agent-environment interaction in the UC MDP is shown in Figure 4.6.

Using definitions of power system variables described in Section 4.2, we will now describe the components of the UC MDP.

4.3.1 MDP Components

The components of the UC MDP are shown in Table 4.3. Here we will describe each component in more detail.

States The state s_t includes: generator up/down times $\boldsymbol{u}_t$; demand forecast $\boldsymbol{d}$; wind forecast $\boldsymbol{w}$; historical demand and wind forecast errors $\boldsymbol{x}_t, \boldsymbol{w}_t \in \mathbb{R}^p$; historical white noise samples for demand and wind forecast errors $\boldsymbol{\epsilon}_{d,t}\boldsymbol{\epsilon}_{w,t} \in \mathbb{R}^q$; the timestep t. p and q are the orders of autoregressive and moving average components of the ARMA(p, q) processes. A state is terminal when $t = T$.

Observations Observations include all elements of the state except forecast errors $\boldsymbol{x}_t, \boldsymbol{w}_t$ and white noise components $\boldsymbol{\epsilon}_{d,t}\boldsymbol{\epsilon}_{w,t}$. The MDP is therefore partially-

States	$\boldsymbol{u}_t$: generator up/down times $\in \mathbb{Z}^N$ $\boldsymbol{d}$: demand forecast $\in \mathbb{R}^T$ $\boldsymbol{w}$: wind forecast $\in \mathbb{R}^T$ $\boldsymbol{x}_t$, $\boldsymbol{y}_t$: preceding demand and wind forecast errors for ARMA(p, q) processes $\in \mathbb{R}^p$ $\boldsymbol{\epsilon}_{d,t}$, $\boldsymbol{\epsilon}_{w,t}$: preceding white noise samples for ARMA(p, q) processes $\in \mathbb{R}^q$ t: timestep $0 \leq t \leq T \in \mathbb{Z}$
Observations	$\{\boldsymbol{u}_t, \boldsymbol{d}, \boldsymbol{w}, t\}$
Actions	a_t: commitment decisions $\{0, 1\}^N$
Rewards	r_t: negative operating cost $\in \mathbb{R}$
Transitions	$u_{i,t+1} = \begin{cases} u_{i,t} + 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} > 0 \\ 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} < 0 \\ -1, & \text{if } a_{i,t} = 0 \text{ and } u_{i,t} > 0 \\ u_{i,t} - 1, & \text{if } a_{i,t} = 0t \text{ and } u_{i,t} < 0 \end{cases}$ $x_t \sim X_t$: sample demand forecast error (from ARMA) $y_t \sim Y_t$: sample wind forecast error (from ARMA)

Table 4.3: MDP components for the UC problem with N generators and T decision periods.

observable. Omitting forecast errors from the observation preserves the day-ahead properties of the problem: the decision-maker cannot observe the forecast errors until after the UC problem has been solved. However, forecast errors are included in s_t in order to preserve the Markov property in the MDP, and for completeness to ensure that the reward distribution $R(s_t)$ does not depend on a latent variable that is not included in the MDP.

Actions An action is a commitment decision that determines the on/off statuses of generators for the next timestep. An action is defined as an array $\boldsymbol{a}_t = [a_{1,t}, a_{2,t}..., a_{N,t}]$, $a_{i,t} \in \{0, 1\}$ for N generators. The action space is therefore combinatorial, and has a total of 2^N unique actions. However, for a given state the set of of legal actions $A(s)$ is limited to those meeting the minimum up/down time constraints described in Section 4.2.1.

Rewards The reward is the negative total operating cost of the system:

$$r_t = -C_t \tag{4.9}$$

where C_t is the sum of fuel costs, startup costs and lost load costs (Equation 4.3). The fuel cost depends on the real-valued power outputs of the generators (Equation 4.1), which are determined by solving the economic dispatch (ED) problem (see Section 4.2.1) to satisfy net demand (which is stochastic).

Transitions Transitions consist of updating the generator up/down times and sampling demand and wind forecast errors using the ARMA processes. Given a commitment decision $a_{i,t}$ and generator status $u_{i,t}$ for generator i, the transition function for the generator status is:

$$u_{i,t+1} = \begin{cases} u_{i,t} + 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} > 0 \\ 1, & \text{if } a_{i,t} = 1 \text{ and } u_{i,t} < 0 \\ -1, & \text{if } a_{i,t} = 0 \text{ and } u_{i,t} > 0 \\ u_{i,t} - 1, & \text{if } a_{i,t} = 0 \text{ and } u_{i,t} < 0 \end{cases} \tag{4.10}$$

Forecast errors are sampled using the ARMA process described in Equation 4.2.

The transition function is stochastic due to the inclusion of forecast errors in the state. However, since these are not observed by the agent, the transition function can be considered deterministic with respect to the observations. This means that the MDP can be easily expressed as a search tree in the observation space, with nodes representing observations and edges representing actions. The search tree formulation is required in order to apply tree search methods such as uniform-cost search, which we describe in the next section.

4.4 Uniform-Cost Search

Having formulated the UC problem as an MDP, we will now apply tree search methods to solve the UC problem. In this section, a variation of the well-known uniform-cost search (UCS) algorithm [61], described in Section 3.6.1 is applied to the UC problem. We begin by formulating the UC MDP as a search tree. The altered UCS algorithm is then described, before being applied to the UC test problems described in Section 4.2.3.

4.4.1 Search Tree Representation of the UC MDP

As stated in the Section 4.3, the MDP can be expressed as a simple search tree in the observation space, which we illustrate in Figure 4.7. A similar formulation is used in [55], although only does not consider uncertain demand or wind generation in the MDP formulation. Each node in the search tree represents an observation (that is, forecasts $\boldsymbol{d}, \boldsymbol{w}$ and generator up/down times $\boldsymbol{u}_t$, and edges represent actions. Since

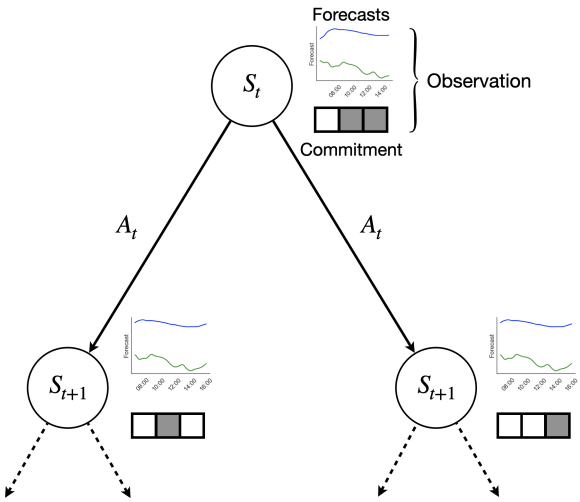


Figure 4.7: Search tree representing the UC MDP. Nodes represent observations, and edges represent actions. The cost of traversing an edge is the expected operating cost, estimated by a Monte Carlo method, simulating each transition N_s times and calculating the mean. The time series at each node represent the demand and wind forecasts at that state, while the commitment is represented by blocks representing the commitment of three generators where grey/white refer to offline/online.

the transition function is deterministic in this space, there is a one-to-one mapping from $(s_t, a_t) \rightarrow s_{t+1}$, which is required for the search tree representation. Traversing an edge on the tree incurs a cost, which is the negative reward in the UC MDP. A UC problem of T decision periods and N generators can be expressed as a search tree with depth T and maximum branching factor $b = 2^N$. The branching factor is a theoretical maximum as in practice, some of the 2^N actions from any node will be illegal due to the minimum up/down time constraints.

Maximising return in the UC MDP amounts to finding the cheapest cost path through the search tree, which can be achieved through tree search algorithms. The cost, however, is stochastic, as it depends on x_t and y_t, the forecast errors. Hence, we are interested in determining the path of least *expected* cost. To achieve this, we set the step costs $C(s)$ (the cost of traversing edge to state s) to an estimate for the mean cost of that transition. This is achieved by a Monte Carlo method: using a set of demand and wind forecast error scenarios $\mathcal{S} \in \mathbb{R}^{T \times N_s}$, sampled from the ARMA processes, we calculate the mean operating cost (negative reward) over $\mathcal{S}$:

$$C(s_t) = -\frac{1}{N_s} \sum_{x \in \mathcal{S}_t} R(s_t, x) \tag{4.11}$$

where $\mathcal{S}_t$ is the set of scenarios for timestep t, x is a scenario in $\mathcal{S}_t$ and $R(s_t, x)$ is the reward function evaluated for state s_{t+1} under scenario x.

Each reward function evaluation requires solving the ED problem for a different scenario x, which may be computationally expensive if N_s is large. However, the

Algorithm 5 Real-time method for solving the UC problem beginning in initial state s_0 using a tree search algorithm f, such as UCS. The algorithm f solves the search tree up to a lookahead horizon H, returning a solution path (sequence of actions). The first action in the path is used to determine the root for the next sub-tree. The algorithm loops for T settlement periods.

function RealTimeTreeSearch(s_0, f, H)
  initialise solution schedule $\boldsymbol{a} \in \{0,1\}^{T \times N}$
  $s \leftarrow s_0$
  for t in $0, 1, \ldots, T-1$ **do**
    $p \leftarrow$ solve tree rooted at s with algorithm f up to horizon H
    $a \leftarrow$ first action in solution path p
    $s \leftarrow$ state following a from s
    $\boldsymbol{a}_t \leftarrow a$
  end for
  return $\boldsymbol{a}$
end function

ED problem is a convex optimisation which can be solved very rapidly using the lambda-iteration method described in Section 3.7.3 and is parallelisable over scenarios. As a result, tree search algorithms using this Monte Carlo method have only $\mathcal{O}(N_s)$ complexity, which is preferable to the super-linear complexity in the number of scenarios of the stochastic optimisation methods reviewed in Section 2.3.

4.4.2 Algorithm: Real-Time UCS

Having formulated the UC MDP as a search tree, we use the uniform-cost search (UCS) algorithm [61] described in Section 3.6.1 to find the path of least expected cost through the tree. Numerous tree search algorithms exist which can be used to solve the UC problem. UCS is appropriate for being simple, heuristic-free and optimal [61]. Furthermore, unlike other simple tree search algorithms like breadth-first search and depth-first search, UCS is naturally applicable to trees with non-uniform step costs, such as in the UC problem.

A notable candidate algorithm for solving the UC problem is Monte Carlo tree search (MCTS), which has been used in model-based RL methods including AlphaGo [27,44]. MCTS is well-suited to zero-sum games domains where the objective is to maximise the probability of winning given an opponent's strategy, but is not a natural fit for conventional optimisation problems. MCTS has rarely been applied to stochastic optimisation problems, and is outperformed by mixed-integer approaches in [250]. We opt to use UCS as a more appropriate search algorithm designed for least cost path problems.

For a search tree with a constant branching factor b nodes and goal node at depth k, UCS has time complexity $\mathcal{O}(b^k)$. For the UC problem with T settlement periods, which has a maximum branching factor of $b = 2^N$, UCS algorithm has

worst-case $\mathcal{O}(2^{NT})$ time complexity making it intractable even for very small power systems with 48 settlement periods. Hence, we use a *real-time* [225] approach to solve each UC problem. Real-time algorithms, which were discussed in Section 3.5.3, do not exhaustively search to the end of the problem, but instead repeatedly solve reduced sub-problems, and take the best first action with respect to this sub-optimal solution. Real-time methods, while not optimal, are significantly less computationally expensive in many problems. In the real-time approach, UCS is used to solve T sub-problems, one for each settlement period. Each sub-problem is limited by a fixed lookahead horizon $H \in \mathbb{N}, 1 \leq H \leq (T-t)$, where H is a constant depth parameter. Once the sub-problem rooted at period t is solved, the first action in the solution path is taken, determining the root node for the sub-problem at time $t+1$. This is repeated until the end of the day $T = t$. This routine is shown as pseudo-code in Algorithm 5. The RealTimeTreeSearch algorithm is modular and can be used with any tree search algorithm for finding the lowest cost path. Real-time UCS has $\mathcal{O}(2^{NH}T)$ complexity: it is linear in the number of settlement periods T and exponential in both H and N. Henceforth we refer to real-time UCS simply as UCS.

Pseudocode for UCS was shown in Algorithm 1. In order to implement real-time UCS, we assign all nodes at timestep $t+H$ to be goal nodes. This ensures that UCS will return the cheapest cost path to a node at the lookahead horizon. In addition, we use the Monte Carlo estimates of expected step costs described in Section 4.4.1 when calculating the path cost $c(n)$. In the limit of H and N_s, UCS will minimise the expected operating cost over all scenarios by searching for the least expected cost path through the MDP. However, the exponential time complexity in the number of generators N limits application of this approach to larger power systems.

4.4.3 Application to Test Problems

We will now apply UCS to the test problems described in Section 4.2.3 to empirically demonstrate exponential run time complexity and evaluate solution quality in comparison with MILP benchmarks described in Section 4.2.3. We set the depth parameter $H = 2$ and applied UCS to power systems of 5–10 generators inclusive. Larger values of H were not possible due to the exponential time complexity in this parameter.

A comparison of the costs and loss of load probability (LOLP) of the UCS and MILP approaches is shown in Table 4.4. We find that UCS achieves similar operating costs to MILP(4σ), being 1.3% more expensive in the 5 generator case, and 0.3% in the 10 generator case. Compared with MILP(perfect), the solution for the point forecast with no reserve constraint, costs are 5.7% and 6.9% higher for 5 and 10 generator problems, respectively. Figure 4.8 shows that UCS has an exponential run time complexity in the number of generators. Extrapolating the relationship to a system of 20 generators, the anticipated average run time is approximately 1×10^6

Num. gens	Version	Mean cost ($M)	Std. cost	LOLP (%)
5	MILP(perfect)	4.38	0.00	0.000
5	MILP(4σ)	4.57	0.25	0.079
5	UCS	4.63	0.31	0.074
10	MILP(perfect)	8.82	0.00	0.000
10	MILP(4σ)	9.40	1.02	0.180
10	UCS	9.43	1.11	0.177

Table 4.4: Comparison of UCS with $H = 2$ with MILP benchmarks from Section 4.2.3. UCS achieves similar operating costs and loss of load probability (LOLP) as compared with MILP(4σ).

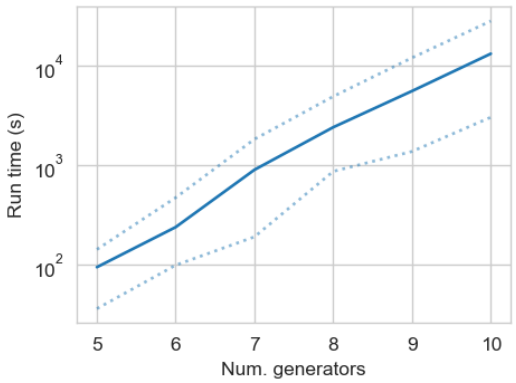


Figure 4.8: Run time of UCS with $H = 2$ with increasing numbers of generators. Solid line shows mean daily run time; dotted lines show minimum and maximum. The straight line on the logged time axis indicates exponential run time complexity in the number of generators.

seconds per problem instance.

The relatively strong performance of UCS with a limited time horizon of $H = 2$ (1 hour) indicates that tree search and direct sampling of the cost function under realisations of uncertainty to estimate expected costs (Equation 4.11) is a strong method for managing uncertainty. However, the short-sightedness of this approach results in worse performance overall as compared with MILP(4σ). With a less flexible generation mix, such as with longer minimum up/down times, the difference between short-sighted tree search and mathematical programming approaches is likely to be larger still. Above all, UCS is not a practical solution method due to the exponential complexity in the number of generators, preventing applications to larger power systems. In the next section, we will describe Guided UCS, which uses an RL-trained policy to reduce the branching factor, allowing for application to larger power systems and greater search depth.

4.5 Guided Uniform-Cost Search

The results in Section 4.4.3 showed exponential time complexity of UCS in the number of generators. As a result, the application of this method to the UC problem is restricted to small power systems only. Even for the small systems of 5–10 generators, it was necessary to limit the search depth to $H = 2$ as higher settings would not be complete in practical run times. While UCS was shown to be competitive with the MILP benchmarks, improvements to operating costs could be achieved by increasing H. In order to improve the run time complexity in number of generators, in this section we present Guided UCS, an RL-aided tree search algorithm that uses an RL-trained *expansion policy* to reduce the breadth of the search tree. Learning by trial-and-error in the simulation environment, the expansion policy offers a rapid approximation of promising regions of the action space. The action space at any given node typically contains a large number of expensive or insecure commitment decisions, such as those which decommit baseload and those which do not commit enough capacity to meet the forecast net demand. The expansion policy is trained by RL to identify these actions as well as more complex, time-dependent properties of the action space as a function of the state variables. Guided by the expansion policy, the search tree of large power systems can be reduced to a much smaller size while preserving optimal or near-optimal solution paths, enabling tree search methods to be applied in larger problem instances.

In this section we will begin by describing *guided expansion*, the method by which an expansion policy is incorporated into UCS to reduce the branching factor. We will then present our approach for training the expansion policy by model-free RL in the power system simulation environment. Finally, we will present the details of policies trained with model-free RL, which are used in experiments applying Guided UCS to test problems in Section 4.6.

4.5.1 Guided Expansion

Guided UCS uses an *expansion policy* $\pi(a|s)$ giving probabilities for the actions from state s to prune low probability actions. We call this routine *guided expansion*. Figure 4.9 illustrates the difference between UCS and Guided UCS, showing how guided expansion is used to reduce the branching factor of the search tree. The expansion policy is used in Guided UCS to prune the left branch from the root. Note that in this illustrative example, both UCS and Guided UCS reach the same solution path, but Guided UCS is more efficient, requiring fewer nodes to be evaluated and expanded. However, Guided UCS may remove an optimal branch if the expansion policy $\pi(a|s)$ is inaccurate. By training the expansion policy with RL, we aim to ensure that $\pi(a|s)$ prunes only sub-optimal branches.

Formally, Guided UCS follows Algorithm 1, but only considers a subset of

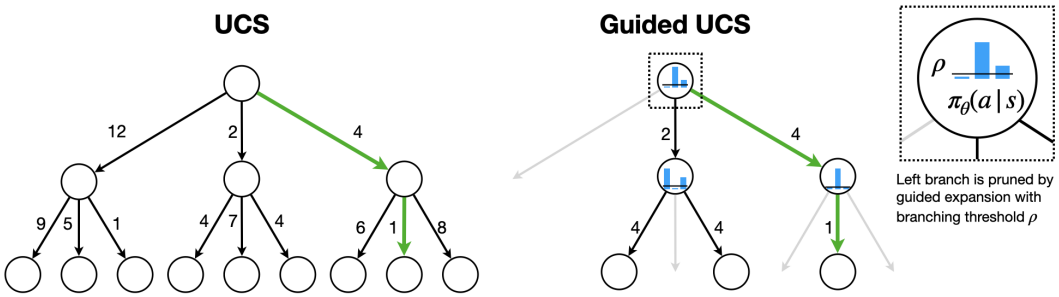


Figure 4.9: Comparison between UCS and Guided UCS algorithms, with a search depth $H = 2$. While UCS considers the full search tree, Guided UCS uses a reduced search tree, with branches pruned by guided expansion (Equation 4.12). The histogram represents the distribution estimated by $\pi_\theta(a|s)$. From the root node, the left branch is pruned as its probability is less than the branching threshold ρ (represents by a horizontal line on the histogram). Green line represents the lowest cost path.

actions in the inner for loop over actions. Guided expansion reduces the complete set of actions $A(s)$ available from state s to a subset of actions $A_\pi(s)$:

$$A_\pi(s) = \{a \in A(s) | \pi(a|s) \geq \rho\} \tag{4.12}$$

where ρ is a branching threshold that controls the search breadth. The maximum number of nodes M that can be added to the tree is therefore limited to $M \leq \frac{1}{\rho}$, since $\sum_{a \in A} \pi(a|s) = 1$. By preventing an exponential explosion in the branching factor with increasing number of generators, the worst case time complexity of Guided UCS is $\mathcal{O}(T(\frac{1}{\rho})^H)$, compared with $\mathcal{O}(2^{NH}T)$ for unguided UCS, presented in Section 4.4. As a result, Guided UCS does not exhibit exponential time complexity in the number of generators, making it feasible to consider real world application to larger power systems. For application to the UC problem, we additionally add the action that keeps all current generator commitments the same to $A_\pi(s)$. This 'do nothing' action is always feasible as it does not change any generator commitments.

The breadth and depth of the search tree are controlled by ρ and H respectively. There is a trade-off between these two parameters, as reducing ρ and increasing H both increase the run time of Guided UCS. Setting ρ to be large results in a narrow search that can cause operating costs to increase. Similarly, a shallow search with low H can also degrade performance due to short-sighted decision-making. In Section 4.6.1 we demonstrate the impact of H and ρ parameter settings on run time, operating costs, and schedule characteristics for the UC problem.

The expansion policy can be defined with expert rules, trained by supervised learning on existing UC solutions, or trained by RL as in this research. Next, we will describe our approach for training the expansion policy with RL.

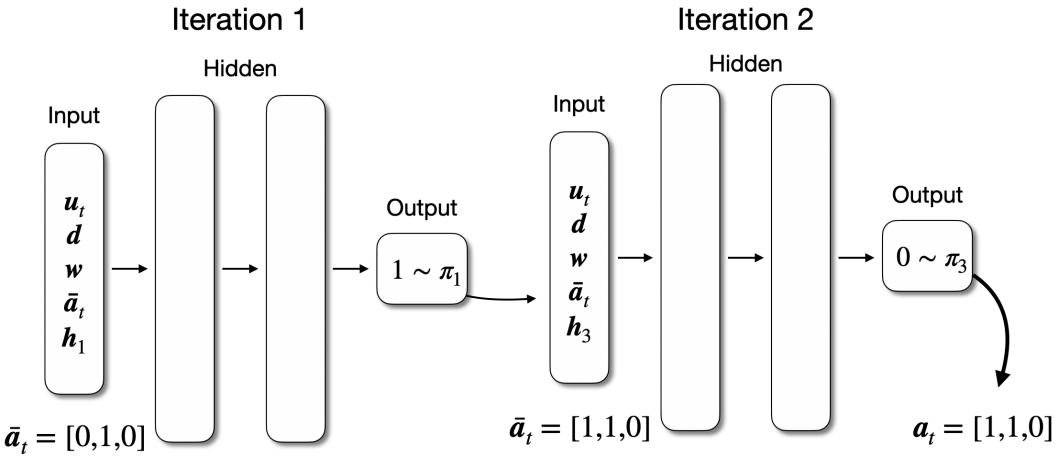


Figure 4.10: Sequential feed-forward neural network architecture used to parameterise the expansion policy. Each generator commitment is classified sequentially with the current action sequence $\bar{a}_t$ used to estimate the following commitment. In the example, the second generator is constrained to remain on, so at the first iteration, $\bar{a}_t = [0, 1, 0]$. Commitment decisions for the unconstrained generators $i = \{1, 3\}$ are made in sequence, sampling from the distribution π_i calculated using the neural network.

4.5.2 Expansion Policy

The expansion policy is trained using the policy gradient RL method proximal policy optimisation (PPO) [35], described in detail in Section 3.4.3. The RL agent (i.e. the expansion policy) interacts with the environment described in Section 4.2 to improve performance with respect to the reward function. As discussed in Section 3.3.1 policy gradient methods have several advantages over value-based methods in certain problem domains. Policy gradient methods can naturally learn stochastic policies and are better suited to large action spaces [29]. The ability to learn a stochastic policy is essential in the context of training an expansion policy for Guided UCS, as the expansion policy should propose a diverse range of actions to add to the search tree in guided expansion. Entropy regularisation, discussed in Section 3.4.3, can be used in policy gradient methods to further promote stochastic policies. PPO was chosen as a state-of-the-art policy gradient method that incorporates a clipped loss function which helps prevent catastrophic performance decreases. We optimise the entropy-regularised PPO objective given in Equation 3.27.

In order to train an expansion policy in large action spaces, we use a sequential feed-forward neural network architecture, predicting a sequence of individual generator actions to create the joint commitment action. While recurrent neural network (RNN) architectures such as long short-term memory (LSTM) networks have excelled in sequence-based learning, most notably natural langauge processing (NLP) tasks such as speech-recognition [251], simpler feed-forward neural networks are also competitive in language modelling [252] and speech synthesis [253] and benefit from

lower computational cost and generally better training stability [254].

Fully enumerating the actions at the output layer of the policy is not feasible due to the size of the action space. Parametrising the multi-dimensional action space with N output nodes is also not appropriate due to the strong dependency of each generator's action probability on that of the other generators. Instead, the policy is parametrised as a binary classifier which sequentially predicts each value in the sequence $\boldsymbol{a} = [a_1, a_2, ..., a_N]$ representing a commitment decision where $a_i \in \{0, 1\}$ are sub-actions giving the commitment for generator i (Figure 4.10). The output of the classifier at each iteration is passed as an input into the next forward-pass through the network, thus maintaining the history of generator commitments already decided. In addition, the input vector includes a one-hot encoding indicating the a_i being classified on each forward pass as well as the observation. This parametrisation succeeds in preserving the interdependencies between generators while remaining tractable for larger power systems.

A disadvantage of this approach is that it is not possible to analytically compute the distribution $\pi(a|s)$. It is necessary to approximate this distribution in order to determine which actions meet the branching threshold ρ in guided expansion (Equation 4.5.1). We use a Monte Carlo method to approximate the distribution: we estimate $\pi(a|s) = \frac{n_a}{N}$ where n_a is the number of times a was sampled and N is the total number of samples.

4.5.3 Training Details

Policies were trained for power systems of $N \in \{5, 6, 7, 8, 9, 10, 20, 30\}$ generators using the method described in Section 4.5.2. Each power system is an instance of the environment described in Section 4.2, using the generator data from [5]. The trained policies were used as expansion policies in Guided UCS in the experiments described in the next section, Section 4.6. Here we provide technical details of the policy training.

Training Episodes During training, the RL agent samples days at random from the training data, defining the demand forecast $\boldsymbol{d}$ and wind forecast $\boldsymbol{w}$ for the episode. The maximum daily wind penetration in training was 58%; the minimum was 0.1%. The RL agent therefore experiences more extreme levels of wind penetration than observed in the test problems where wind penetration is between 3%–37%. The initial generator up down times $\boldsymbol{u}_0$ are set randomly to encourage exploration of the state space. To speed up the early stages of training, an episode ends if the agent encounters lost load.

Actor-Critic Parameters A summary of the parameters used to train the 10, 20 and 30 generator policies is given in Table 4.5. Each policy was trained with 8 workers, with weights of the actor-critic neural network updated asynchronously

Variable	Generators 10	20	30
Update	PPO		
Reward transformation	$\hat{r} = \log(m/r)$		
Clip ratio [35]	0.1		
Entropy coefficient [53]	0.05	0.001	0.0
Actor architecture	100, 50, 25	64, 64	64, 64
Critic architecture	64, 64	100, 50, 25	64, 64
Epochs	100,000	200,000	300,000
Forecast window (periods)	24		
Gamma	0.95		

Table 4.5: Parameter settings for training the expansion policy using PPO.

after every 2000 forward passes through the policy network. Every update is an *epoch*. This implementation differs from standard implementations of PPO due to the sequential parametristation of the policy: each forward pass through the network is recorded separately in the replay buffer, along with the complete encoding of the input vector (that is, including the one-hot encoding $\boldsymbol{h}$ and action draft $\bar{\boldsymbol{a}}$), such that for each timestep, multiple entries can be added to the replay buffer. As the input vector dimensions are different for the actor and critic (the actor uses the one-hot encoding $\boldsymbol{h}$ and draft action $\bar{\boldsymbol{a}}$), we used different architectures and replay buffers for the actor and critic networks. A grid search approach was used to determine the best performing combination of architectures, testing all combinations of the three architectures used in [255]: (64, 64); (100, 50, 25); (400, 300), where (x_i) indicates the number of nodes in hidden layer i. In addition, we set the discount factor $\gamma = 0.95$. This assigns more credit to actions which are temporally close to the reward as compared with setting $\gamma = 1$. We included an entropy bonus [53] for two reasons: first to encourage exploration in training; second, to prevent the policy from converging to a policy which strongly favours a single action, so as to maintain a diverse set of actions in Guided UCS.

Reward Transformation We also transformed the reward function using $\hat{r} = \log(m/r)$, where m is a constant used to scale the reward to around the range $[-1, 1]$. The log transformation dampens extreme negative rewards resulting from lost load events, which can otherwise result in overly conservative behaviour.

Observation Pre-Processing Observations were pre-processed before passing through the policy network to improve training stability. First, we capped the generator up/down times u_i at the minimum up/down time, and scaled this to between $[-1, 1]$ such that -1 indicates that the generator is offline and has satisfied its minimum down time constraint and likewise for the minimum up time constraint. This bounds the state space while exploiting symmetries in the state space such that there is no loss of information for the expansion policy. Second, the timestep

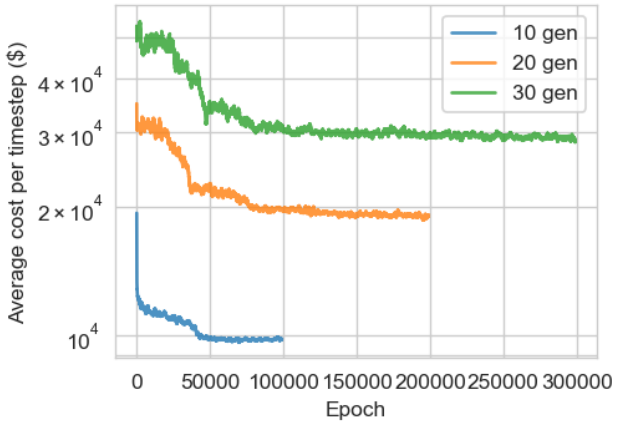


Figure 4.11: Average cost per timestep for 10, 20 and 30 generator policies during training. Plot shows a moving average over 1000 epochs. The 10 and 20 generator problems converged more quickly than the 30 generator problem.

was normalised by the episode length $\hat{t} = t/T$. Third, we scaled demand and wind variables by a constant factor, $1/\sum_i p_{\max}$. This aims to keep all state variables within roughly the same order of magnitude which is known to improve neural network training. Lastly, we truncated the forecasts $\boldsymbol{d}, \boldsymbol{w}$ to a forecast window of k periods ahead of the decision period. This method benefits from reducing the size of the state vector and only presenting the most 'relevant' forecast information.

Convergence Results

Figure 4.11 shows the convergence of 10, 20 and 30 generator policies in terms of operating cost per timestep. The agents improve rapidly at the beginning of training, learning to avoid lost load events. However, the average episode length is around 42 periods in all three problems by the end of training, indicating significant loss of load probability (episodes end when lost load is observed).

We used the 10, 20 and 30 generator expansion policies to solve the 20 test problems 'model-free', sampling directly from the distribution $\pi(a|s)$ for each state s, to evaluate the potential of the policies to solve the unseen problems without tree search. The results are compared with MILP benchmarks in Table 4.6. The model-free solutions are significantly more expensive, and have higher loss of load probability. This indicates that the policies are unable to provide good solutions to the UC test problems without tree search. In the next section, the expansion policies are applied in Guided UCS to solve the test problems.

4.6 Evaluating Guided UCS

Section 4.5 described Guided UCS, an RL-aided tree search algorithm that incorporates an expansion policy to reduce the branching factor of the search tree. In Section

Num. gens	Version	Mean cost ($M)	Std. cost	Mean time (s)	LOLP (%)
10	$\pi(a\|s)$	22.00	5.54	0.2	4.045
10	MILP(4σ)	9.40	1.02	19.1	0.180
10	MILP(perfect)	8.82	0.00	2.3	0.000
20	$\pi(a\|s)$	24.19	4.45	0.5	1.097
20	MILP(4σ)	18.90	2.92	5.5	0.244
20	MILP(perfect)	17.58	0.00	11.9	0.000
30	$\pi(a\|s)$	34.87	7.81	0.8	1.020
30	MILP(4σ)	28.53	4.94	8.0	0.291
30	MILP(perfect)	26.31	0.00	6.8	0.000

Table 4.6: Comparison of model-free solutions using expansion policies $\pi(a|s)$, with MILP benchmarks for the 20 test problems. The model-free solutions have much higher loss of load probability and hence higher operating costs. Average run times are at least one order of magnitude lower than MILP.

4.5.3 we trained expansion policies for systems of between 5–30 generators. Now, we will evaluate the performance of Guided UCS on the 20 test problems described in Section 4.2.3. As stated in Section 4.5.1, the depth and breadth parameters H and ρ in Guided UCS are important variables impacting the quality of the solution and the run time. First, we conduct a simulation study varying H and ρ to determine suitable parameters for the next experiments. We then compare Guided UCS with traditional (unguided) UCS for 5–10 generator problem instances, demonstrating that Guided UCS has better run time complexity in the number of generators while achieving similar solution costs. Last, we compare Guided UCS with the MILP benchmarks for 10, 20 and 30 generator problems in Section 4.6.3, showing that Guided UCS achieves lower operating costs overall and exhibits novel schedule characteristics.

4.6.1 Parameter Analysis

The first experiment investigates the impact of search breadth and depth on performance. We considered the 5 generator problem only and set $H = \{1, 2, 4, 6, 8, 12, 16, 24\}$ and $\rho = \{0.01, 0.05, 0.1, 0.25, 0.33\}$. Under some parameter combinations, a subset of the 20 test problems did not complete within a 24 hour time budget and these results are not reported. Figure 4.12 shows the parameter combinations and those that did not complete within 24 hours (shaded in grey).

In all experiments, we set $N_s = 100$, the number of scenarios used to calculated expected edge costs in Equation 4.11. Here we will analyse performance as a function of H and ρ in terms of costs, run time and characteristics of the schedules. Using these results, we aim to determine parameter settings which achieve good solution quality within practical run times.

Figure 4.13 summarises the performance of Guided UCS with different settings of H and ρ along side the UCS solutions with $H = 2$ from Section 4.4.3. Figure 4.13a verifies the exponential time complexity in H for fixed ρ (linear fits on log scale) and clearly shows the run time reduction achieved by Guided UCS at $H = 2$ for all

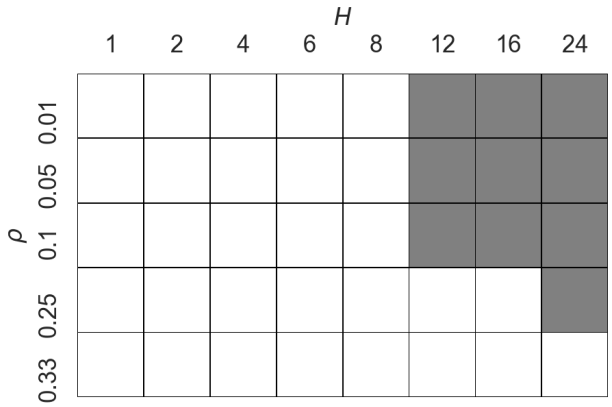


Figure 4.12: Parameter combinations for parameter analysis experiment. Those which did not complete in 24 hours are shaded in grey.

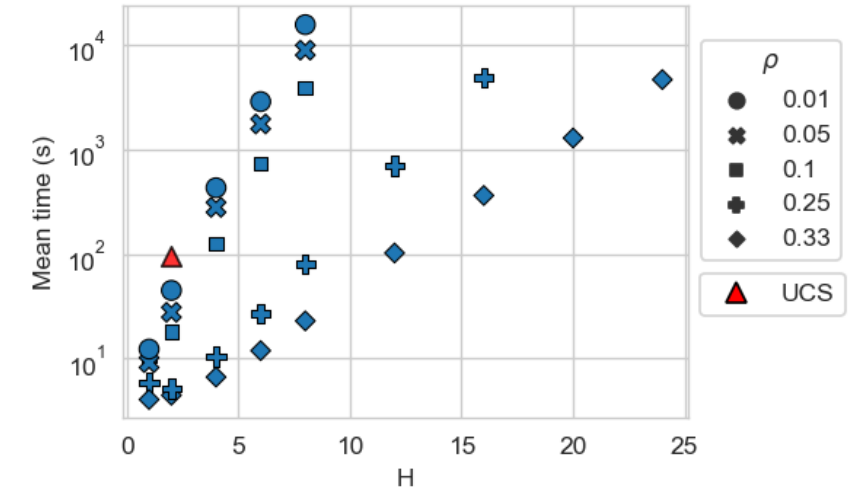
settings of ρ. The lowest settings of ρ exhibit the steepest rise in run time with increasing search depth.

Figure 4.13b compares operating costs as a function H and ρ. The largest values of $\rho = \{0.25, 0.33\}$ (maximum branching factor of 4 and 3, respectively) consistently have among the highest operating costs consistently regardless of the depth H. Overall, the lowest costs were produced by setting $H = 8$ and $\rho = 0.01$ (maximum branching factor of 100), which also had the longest run time as shown in Figure 4.13b. While the longer running parameter settings generally had lower costs, there is considerable variation between parameter combinations that result in similar run times: for instance, the (H, ρ) parameter settings $(4, 0.01)$ and $(16, 0.33)$ both have run times of around 400 seconds, the former's operating costs are around 4% lower than the latter's.

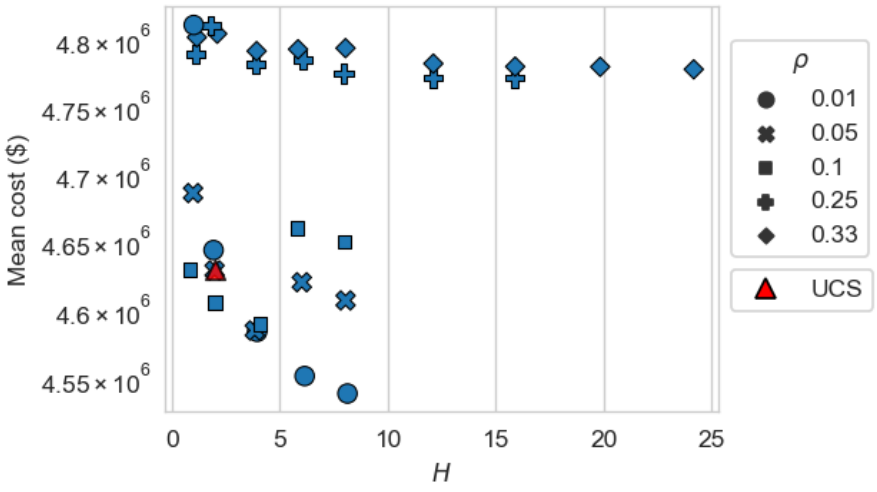
Guided UCS with (H, ρ) settings of $(2, 0.05)$ and $(1, 0.1)$ both have similar operating costs to the UCS solution, but with lower run times (roughly 10 times faster in the latter case). Settings of $\rho = \{0.01, 0.05, 0.1\}$ all achieve consistently similar or lower costs to UCS for $H > 1$, although $\rho = 0.01$ is the only setting which improves monotonically with increasing depth H. However, the run time of $\rho = 0.01$ is most sensitive to H, as shown in Figure 4.13a.

Varying ρ and H results in different schedule characteristics such as startup frequency and up times of generators. Startups were found to generally decrease with increasing H, shown in Figure 4.14. Small settings of ρ usually have larger numbers of startups for fixed H, as a wider branching factor allows for greedier decision-making. Generator utilisation (proportion of periods spent online) also varies with ρ and H, such as reduced utilisation of peaking plants when the depth H is increased.

We determined that $\rho = 0.05$ (maximum branching factor of 20) was the most suitable value to use for subsequent experiments, as it achieves consistently low operating costs and scales well with search depth relative to $\rho = 0.01$. Using $\rho = 0.05$,



(a)



(b)

Figure 4.13: Comparison of run time and cost for settings of ρ and H for the 5 generator problem. Figure 4.13a verifies that run time grows exponentially with H for a fixed setting of ρ. The two largest settings of ρ perform worst, even with large H. Figure 4.13b shows that costs generally decrease with H for fixed ρ. Performance of UCS (results from Section 4.4.3) is also shown to have similar nearly identical costs to Guided UCS with $H = 2, \rho = 0.05$. The lowest setting of $\rho = 0.01$ achieves the lowest costs for fixed H due to the wider search breadth, but Figure 4.13a shows this scales most quickly with run time.

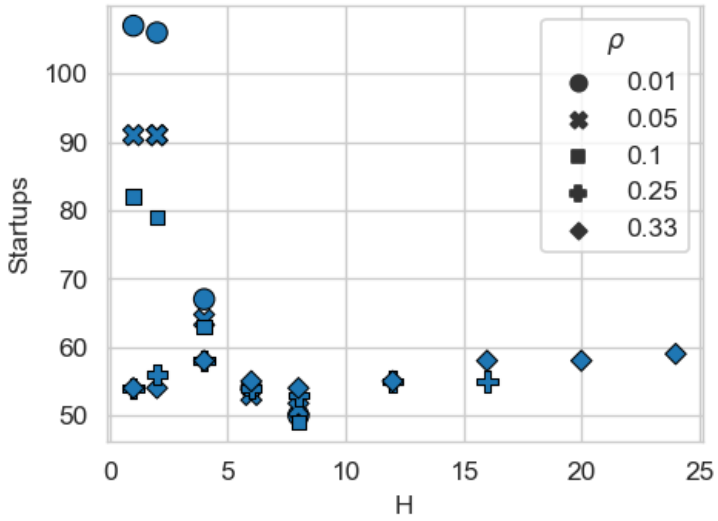


Figure 4.14: Total number of startups for the 20 unseen test problems with parameter settings of H and ρ. Startups generally decrease with search depth up to $H = 8$, after which we observe a small increase in startups.

increasing H as far as possible in practical computing times is likely to improve solution quality.

4.6.2 UCS Comparison

Having conducted experiments to determine parameter settings for Guided UCS, in the second experiment we compare Guided UCS with unguided UCS (which does not use an expansion policy). In particular, we aim to determine whether Guided UCS succeeds in achieving sub-exponential time complexity in the number of generators, while achieving similar operating costs to UCS which exhaustively solves the search tree with no branches removed. Guided UCS with $\rho = 0.05$ and $H = 2$ (which was found to result in similar operating costs to UCS) was used to solve the test problems for power systems of 5–10 generators. The results are compared with UCS with $H = 2$, as reported in Section 4.4.3.

The run times of guided and UCS for systems of 5–10 generators are compared in Figure 4.15. Whereas the mean run time of UCS rises exponentially with the number of generators, run time for Guided UCS remains stable. For the 10 generator problem, the run time for Guided UCS is around 0.2% that of UCS. By limiting the branching factor to $\frac{1}{\rho}$, the run time of Guided UCS slightly decreases after $N > 6$, enabling application to larger power systems.

Given the constant run time complexity of Guided UCS, the apparent downward trend in run time may be explained by challenges in tuning policy entropy for larger power systems. For larger action spaces, the number of actions meeting the branching threshold ρ may be highly sensitive to the entropy of the expansion policy. If policy

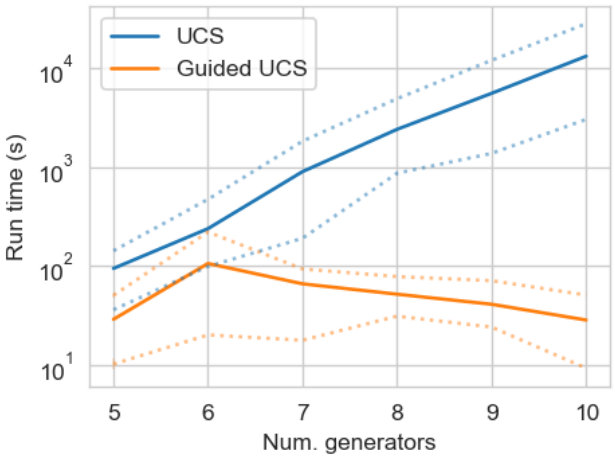


Figure 4.15: Mean computation time for guided and UCS from 5–10 generators. Dotted lines show the maximum and minimum time taken for a single problem. UCS run time increases exponentially with the number of generators for a fixed search depth, while Guided UCS shows no significant increase in run time.

Table 4.7: Comparison of guided and unguided search for 5 and 10 generator problems.

Num. gens	Version	Mean cost ($M)	Std. cost	Mean time (s)	Max. time	Min. time	LOLP (%)
5	Guided UCS	4.63	0.32	27.7	52.5	8.9	0.077
5	UCS	4.63	0.31	94.1	141.8	35.9	0.074
10	Guided UCS	9.44	0.83	28.3	50.6	9.2	0.152
10	UCS	9.43	1.11	13249.9	28272.4	3037.9	0.177

entropy is low, the policy approaches determinism and only a small number of actions are likely to be added to the search tree at each node. Similarly, if the policy entropy is too high, very few actions will be added, also resulting in low branching factors and hence shorter average run times. We address this issue explicitly in the next chapter using target entropy regularisation to solve the 100-generator problems in Section 5.5.

The operating cost results for the 5 and 10 generator problems are summarised in Table 4.7. Mean operating costs are very similar for both problem settings. Guided UCS achieves lower loss of load probability in the 10 generator problem. Guided UCS is successful in substantially reducing run time without notable increase in operating costs. Both methods exhibit substantial run time variability of roughly one order of magnitude between minimum and maximum run times.

Schedules produced by guided and UCS were usually similar, but there were notable differences on some problems as demonstrated by Figure 4.16. In this example for the 5 generator problem, unguided search makes more frequent commitment changes and operates a tighter reserve margin. Guided UCS has longer periods of no commitment changes. Overall, Guided UCS used 11% and 15% fewer startups than UCS for 5 and 10 generator problems, respectively.

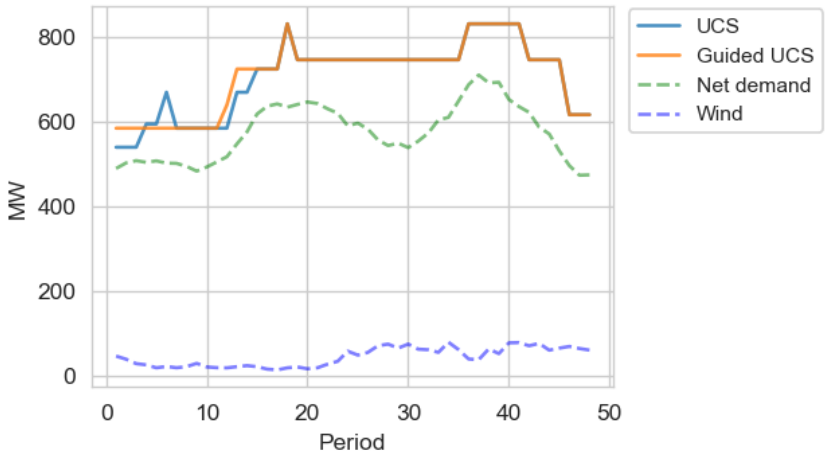


Figure 4.16: Comparison of committed capacity of Guided UCS and UCS schedules for the 2018-03-08 test problem with 5 generators. The generation floor is the sum of minimum operating outputs $p_{\min}$ of committed generators. UCS makes more frequent commitment changes and operates tighter reserve margins.

4.6.3 MILP Comparison

The results in Section 4.6.2 show that Guided UCS with $H = 2$ and $\rho = 0.05$ has constant time complexity in the number of generators and achieves similar operating costs to exhaustive UCS with $H = 2$. In our final experiment, Guided UCS is compared with the MILP(4σ) and MILP(perfect) benchmarks for systems of 10, 20 and 30 generators. This experiment shows that Guided UCS is competitive with industry standard approaches, and performance in terms of operating costs does not deteriorate with increasing problem size. We set the breadth parameter $\rho = 0.05$, allowing for a maximum branching factor of 20. While it was only possible to run UCS with $H = 2$ due to the high computational cost of this approach, the lower run times of Guided UCS allowed the search depth to be increased to $H = 4$ to achieve further reductions in operating costs. This decision was made on the basis of the parameter study in Section 4.6.1, where Figure 4.13a showed setting $\rho = 0.05$ and $H = 4$ gave a mean run time in the order of 100 seconds for the 5 generator problem, which is a practical time budget for real-world UC problems. Given constant run time complexity of Guided UCS in the number of generators, mean run times for the larger systems are likely to be similar with these parameter settings. Furthermore, given the run time variability of Guided UCS observed in Table 4.7, maximum episode run times may be impractically large for larger values of H.

The results are presented in Table 4.8. Guided UCS achieves lower operating costs than MILP(4σ) for all three problem instances, with improvements of 0.33%, 0.87% and 0.45% for 10, 20 and 30 generator problems respectively. Guided UCS has notably lower LOLP in all cases, which is reflected in lower standard deviation of operating costs over the 1000 realisations of demand and wind. The worst

Num. gens	Version	Mean cost ($M)	Max. cost	Std. cost	Mean time (s)	Max. time	Min. time	LOLP (%)
10	MILP(perfect)	8.82	8.82	0.00	2.3	5.6	1.5	0.000
10	Guided UCS	9.37	18.15	0.84	807.3	1992.1	76.9	0.128
10	MILP(4σ)	9.40	19.39	1.02	19.1	177.2	1.8	0.180
20	MILP(perfect)	17.58	17.58	0.00	11.9	150.8	3.3	0.000
20	Guided UCS	18.73	37.72	1.41	117.3	374.5	10.4	0.107
20	MILP(4σ)	18.90	46.15	2.92	5.5	8.4	3.8	0.244
30	MILP(perfect)	26.31	26.31	0.00	6.8	18.0	4.9	0.000
30	Guided UCS	28.41	49.93	2.04	391.3	976.8	129.4	0.142
30	MILP(4σ)	28.53	75.03	4.94	8.0	12.4	5.6	0.291

Table 4.8: Comparison of MILP and Guided UCS solutions for 10, 20 and 30 generator problems.

case (maximum) costs are also significantly lower, with the margin of improvement increasing from 6.3% with 10 generators to 33% for the 30 generator case. This is indicative of superior uncertainty management using Guided UCS as compared with the 4σ reserve strategy. Compared with MILP(perfect), which adopts no reserve constraint and is evaluated on the single point forecast scenario, Guided UCS is 6.2%, 6.6% and 8.0% more expensive for 10, 20 and 30 generator problems, respectively.

Mean run time of Guided UCS is of the order of 100 seconds in all cases, as predicted. Guided UCS is substantially slower than the MILP solutions, by around 1–2 orders of magnitude. As discussed in Section 3.7.2, MILP solvers have benefited from a number of efficiency improving innovations in commercial and open-source solvers. For this research we used the open-source COIN-OR library's branch-and-cut algorithm [238], which implements methods such as cutting planes, parallelism and branching heuristics to very efficiently solve the MILP. Optimising the implementation of Guided UCS is beyond the scope of this thesis, but there are several avenues for improvement including parallel node evaluations, more efficient economic dispatch calculation [239] and implementation in a compiled language such as C++. As a result, it is not possible to address differences in absolute computational demands of MILP and Guided UCS by run time alone.

There were several differences in terms of schedule characteristics comparing MILP(4σ) with Guided UCS. Figure 4.17 shows example solutions for Guided UCS and MILP for the 20-generator test problem, 2019-11-09. Compared with the MILP solution, Guided UCS is characterised by longer periods of no commitment changes, and larger reserve margins at the end of the day when net demand uncertainty is greater. Actions taken by Guided UCS were more concentrated towards those which change the commitment of multiple generators at once, as illustrated in Figure 4.18. Guided UCS exhibits a 'long tail' of actions changing multiple generator commitments at once, whereas MILP is concentrated towards actions with fewer simultaneous startups/shutdowns. For the 20 and 30 generator problems, we also found that the Guided UCS made more frequent use of the 'do nothing' action, making no commitment changes.

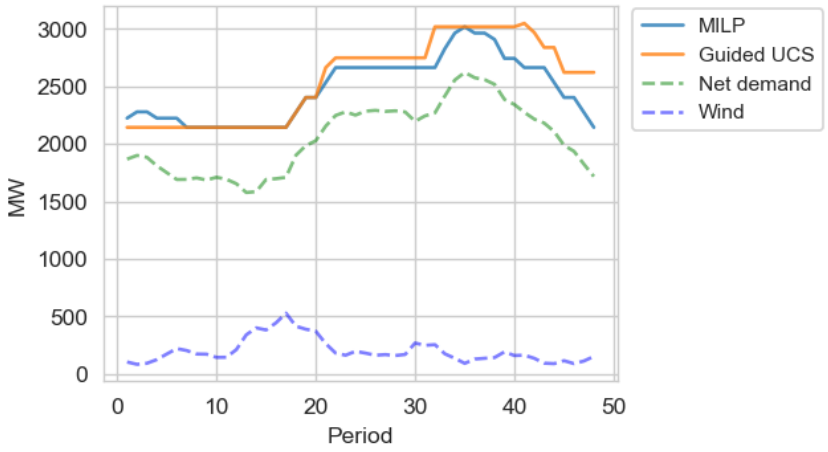


Figure 4.17: Committed capacity of Guided UCS and MILP(4σ) solutions to the 2019-11-09 test problem (20 generators). Guided UCS makes more frequent use of actions making no commitment changes, thereby avoiding startup costs. The Guided UCS solution also employs larger reserve margins at the end of the day when forecast errors can be larger.

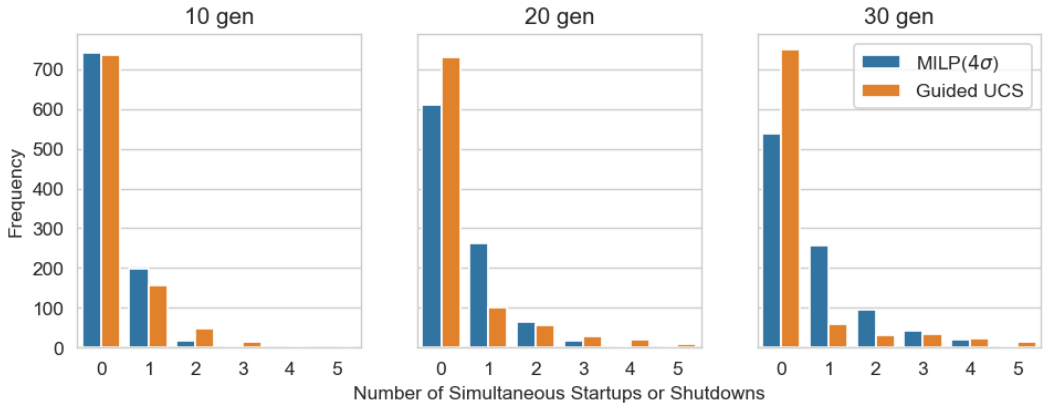


Figure 4.18: Frequency of actions by number of simultaneous startups or shutdowns, comparing Guided UCS and MILP(4σ). The Guided UCS solutions have a longer tail of actions with multiple simultaneous commitment changes. For 20 and 30 generator problems, Guided UCS uses the 'do nothing' action (0 commitment changes) more frequently than MILP.

4.7 Discussion

Our results found that whereas UCS has exponential run time complexity in the number of generators, Guided UCS exhibited no significant increase in run time when increasing from systems of 5 to 10 generators. In addition, we found no significant deterioration in solution quality as measured by operating costs when using Guided UCS. Comparing with MILP, Guided UCS outperformed MILP(4σ) for all problem sizes. Despite not employing a reserve constraint, Guided UCS achieved roughly half the LOLP as compared with MILP(4σ), representing more secure operation.

While there are multiple possible sources of the improved solution quality of Guided UCS relative to mathematical programming, the most important in our experiments was improved management of uncertainty afforded using RL as compared with deterministic mathematical programming. Sampling directly from the environment enables the RL agent to develop a rich representation of uncertainty, whereas the deterministic MILP(4σ) formulation is reliant on heuristics. The RL agent thus optimises directly for expected operating costs, providing more rigorous consideration of extreme scenarios that was evidenced by lower LOLP and better worst case operating costs.

An additional advantage of RL is the ability to represent generator cost functions and constraints accurately, without linear approximations that are required in linear programming approaches. While this is unlikely to have contributed significantly to improved solution quality in this instance, in some cases linear functions may provide an inadequate representation of generator fuel cost curves, such as those exhibiting non-convex valve point effects [256].

The results in this chapter show that the policy learned by model-free RL was effective in intelligently selecting promising actions to add to the search tree. This was evident in the qualitative differences between schedules produced by Guided UCS and UCS. Guided UCS used fewer startups and took fewer greedy actions to minimise short-term costs; these actions were pruned by guided expansion. Guided UCS used novel strategies that differed qualitatively from those produced using MILP. Guided UCS tended to use more extreme actions, changing multiple generator commitments at once, with longer periods of no commitment changes. These actions may be difficult for human operators to identify and indicates scope for Guided UCS to be used as part of a decision support tool for system operators.

Guided UCS achieved lower LOLP as compared with MILP(4σ), and lower worst case costs. The improvement in security of supply relative to MILP(4σ) can be attributed in part to the Monte Carlo approach used to estimate the expected edge costs on the search tree described in Section 4.4.1. This allows for costs to be evaluated under scenarios of demand and wind, thus anticipating possible lost load events. Whereas the MILP benchmark uses a heuristic reserve constraint, Guided UCS includes security of supply as part of the cost function, with a parameter (value

of lost load V in Equation 4.8) that weights security relative to other costs. The ability to shape rewards to reflect societal value of security of supply, economic affordability and environmental sustainability is an important property of RL and tree search approaches to the UC problem, that is not easily afforded by mathematical programming methods. In Section 6.2, we further investigate reward shaping in the UC problem by introducing carbon pricing.

A hybrid methodology combining strategies learned using RL with the advantages of deterministic mathematical programming such as a measurable optimality gap and fast solve times is a worthy topic of further research. Comparison of schedules produced by Guided UCS and MILP(4σ) indicated that Guided UCS dynamically allocated reserves, reflecting increasing uncertainty throughout the day as the decision horizon increased. Incorporating the reserve margins of Guided UCS solutions into MILP formulations could provide a means of improving solution quality of deterministic mathematical programming under high uncertainty.

Guided UCS possesses attractive run time properties, with constant complexity in the number of generators N, linear complexity in the number of periods T and linear complexity in the number of scenarios used to evaluate expected edge costs N_s. However, performance depends considerably on the quality of the expansion policy $\pi(a|s)$ and its ability to intelligently select promising actions to retain in the search tree. Nonetheless, our results did not find a deterioration in performance with increasing numbers of generators (Guided UCS outperformed MILP(4σ) by the greatest margin in the 20 generator case), suggesting that the model-free RL approach employed to train the expansion policies was effective even in very large action spaces (up to roughly 1 billion actions in the 30 generator case). The sequential parametrisation of the policy was an effective approach in enabling scaling to larger power systems. The linear complexity of Guided UCS in T may be valuable in future power markets as many transition towards higher frequency settlements. Finally, linear complexity in N_s may also be a valuable characteristic in more complex stochastic environments which cannot be effectively reduced to a few scenarios for stochastic programming approaches reviewed in Section 2.3.

4.7.1 Related Work

Despite recent successes of RL in numerous challenging domains, until now only a small body of research has investigated applications of RL to the UC problem. Existing research [40–43, 55], reviewed in detail in Section 2.5, has focused on small numbers of generators, in part due to the combinatorial action space that limits the application of existing RL methods 'out-of-the-box'. Fuzzy Q-learning is used in [42] to solve the widely-studied 10-generator Kazaris [5] UC problem. The results of this study are not directly comparable to those in this chapter, due to its use of a single demand profile and absence of uncertainty. In the most similar research [55],

tree search methods are applied to a system of 12 generators, which, to the best of our knowledge, is the largest prior study in this area. However, the problem considered is deterministic and does not consider generalisability to unseen problems. In subsequent related research, a larger power system is considered but the UC problem is simplified to a single commitment decision per day [56]. To the best of our knowledge, the work presented in this chapter is unique in considering generalisability to unseen profiles and training on multiple episodes, and is the largest simulation study of its kind.

4.8 Conclusion

In this chapter we formally described the UC problem as an MDP, and presented a power system simulation environment suitable for RL research in this area. We then showed how the UC problem can be formulated as a search tree and solved using the traditional planning algorithm uniform-cost search (UCS). This method is competitive in terms of cost with industry-standard MILP benchmarks and is suited to stochastic problems, but suffers from exponential time complexity in the number of generators. To improve the run time complexity in the number of generators we presented guided expansion, a method by which an RL-trained policy can be used to reduce the branching factor of a search tree. We applied this in Guided UCS, a guided tree search algorithm, with a policy trained with proximal policy optimisation (PPO). We conducted a parameter analysis to determine suitable values of the depth and breadth parameters for Guided UCS, considering run time, operating costs and schedule characteristics.

Guided UCS was found to exhibit constant run time complexity in the number of generators, and achieved similar operating costs to UCS. Guided UCS was also shown to be competitive with the MILP benchmark employing a reserve constraint, resulting in lower operating costs and improvements to security of supply. Whereas existing research applying RL to the UC problem has been limited to small power systems [40–43, 55], guided tree search was successful in outperforming deterministic MILP for problems of up to 30 generators. To the best of our knowledge, this is the largest application of RL to the UC problem in the literature. In addition, we found qualitative differences between schedules produced by Guided UCS and MILP, with Guided UCS using complex and unusual strategies that may be difficult for human operators to identify.

The principle of guided expansion is applicable in other tree search algorithms. In Chapter 5, we apply informed and anytime methods that leverage domain-specific knowledge of the UC problem to improve performance.

Chapter 5

Informed and Anytime Search

5.1 Introduction

Guided expansion, the key innovation of guided uniform-cost search (UCS) presented in the previous chapter, can be applied in a modular fashion to any tree search algorithm, creating a broader class of guided tree search methods. We focused on UCS as a simple, general-purpose algorithm that can be applied in any problem domain. As discussed in Section 3.5.3, there exists a broad taxonomy of tree search algorithms with properties suiting different applications. Informed search methods can benefit from greater search efficiency by exploiting problem-specific knowledge but cannot be generally applied across different problem domains. In addition, informed search methods may lack the optimality guarantees of uninformed search methods. Similarly, anytime (interruptible) methods can mitigate run time variability and improve performance in time-constrained contexts at the expense of optimality guarantees. In this chapter we demonstrate that the application of informed and anytime search methods to the UC problem using the guided tree search approach improves performance relative to Guided UCS in terms of run time and operating costs.

Using the power system simulation environment introduced in Section 4.2, we demonstrate that guided tree search algorithms designed more specifically for the UC problem can achieve lower operating costs in similar run times, and exhibit practical benefits compared with the more general-purpose algorithm, Guided UCS. In particular, we show that heuristics based on simple priority list UC solution methods can be leveraged in informed search methods to substantially improve search efficiency, saving computational budget that can be used to increase the depth of search. Furthermore, an anytime algorithm exhibits practical benefits in mitigating the run time variability associated with Guided UCS, and allowing for computational resources to be fully exploited.

These improvements, culminating in the informed, anytime algorithm Guided IDA* search, enable the application of guided tree search to larger problem instances of 100 generators. The larger action space of the 100 generator power system poses

further challenges for policy training as policy entropy must converge at a suitable level for guided expansion. To manage this problem, we use a novel method of *target entropy regularisation*, which penalises deviations of policy entropy from a specified value. Setting the target entropy as a function of the desired branching threshold ρ and the number of generators unifies the elements of policy training and guided tree search, and is shown to improve policy convergence in practice.

5.1.1 Contributions

This chapter makes the following contributions.

1. We introduce two new guided tree search algorithms, Guided A* search and Guided IDA* search, applying the principle of guided expansion introduced in Section 4.5. Both algorithms are informed, using a problem-specific heuristic to improve search efficiency. Guided IDA* is also anytime and we replace the depth parameter H with a time budget, preventing high run time variability.

2. Three heuristics based on a PL algorithm are introduced for application in Guided A* and IDA*: Naive Minimum Marginal Fuel Cost (MMFC); Naive Economic Dispatch (ED) and Constrained ED. The heuristics are analysed in terms of average run time, accuracy and admissibility and applied in Guided A* search to solve the 20 test problems from Section 4.2.3 to evaluate improvements to search efficiency improvements as compared with Guided UCS. Constrained ED is found to be the most effective heuristic, reducing mean run time by between 64–94% as compared with Guided UCS, without significant changes in operating costs.

3. Using the strongest heuristic (Constrained ED), the anytime algorithm Guided IDA* search is applied to the test problems and found to allow for deeper search on average by more consistently exploiting the computational budget. Costs are found to be between 0.4–1.0% lower than Guided UCS while completing in similar run time.

4. Guided IDA* search is applied to a 100-generator system, a significantly larger problem than previous research. A novel method of entropy regularisation based on *target entropy* is used which is shown to improve training convergence and promote policies with a suitable level of entropy for guided expansion in high dimensional actions spaces. We find that operating costs are 0.14% lower on average than the MILP benchmark using a reserve constraint. Guided IDA* achieves lower loss of load probability, and achieves lower expected operating costs than MILP in 15 out of 20 problems.

In the next section, Guided A* search and Guided IDA* search algorithms are presented. In Section 5.3, the three PL-based heuristics Naive MMFC, Naive

ED and Constrained ED are described and analysed. In Section 5.4 we compare Guided A* search using each of the three heuristics and apply the strongest heuristic (Constrained ED) in Guided IDA* search. In Section 5.5, Guided IDA* is applied to a larger power system of 100 generators. We discuss our findings in Section 5.6 and Section 5.7 concludes the chapter.

5.2 Informed and Anytime Algorithms

In Section 4.5 we presented Guided UCS, an RL-aided tree search algorithm. Guided UCS uses an expansion policy to determine a subset of branches to add to the search tree at each node. This process of guided expansion (Equation 4.12) can be applied in a modular fashion during the expansion phase of any tree search algorithm. We previously focused on UCS as a simple, heuristic-free algorithm that can be applied to search trees with non-uniform costs [61]. However, exploiting domain knowledge through *informed* search algorithms can significantly improve the efficiency of tree search algorithms in practice while remaining optimal [61]. Furthermore, the significant run time variability of Guided UCS that was shown in Section 4.6.2 and is investigated further for Guided A* in Section 5.4.1 motivates the development of an *anytime* algorithm - that is one which can be interrupted and return a solution, with the solution quality improving over time. We discussed the taxonomy of tree search algorithms, including informed and anytime algorithms, in Section 3.5.3.

In this section we apply guided expansion to A* search [62] (Guided A* search) and iterative-deepening A* search (Guided IDA* search) [224]. Guided A* is an informed search algorithm, while Guided IDA* is both informed and anytime.

5.2.1 Guided A* Search

First we present Guided A* search, in which guided expansion is applied to A* search [62]. A* search was described in detail in Section 3.6.2, with pseudocode shown in Algorithm 2. The A* search algorithm is similar to UCS, differing only in its ordering of the priority queue giving the next node to expand. Whereas UCS orders nodes by their path costs $g(n)$, A* orders nodes by:

$$f(n) = g(n) + h(n) \tag{5.1}$$

where $h(n)$ is a heuristic estimate of the optimal path cost from n to a goal node (cost-to-go). In other words, while UCS chooses the next node to expand based on the cost to reach that node alone, A* search chooses based on the path cost plus an estimate for the remaining path cost to a goal node. A comparison between UCS and A* search, applied to a search tree with depth $H = 2$, is shown in Figure 5.1. By expanding nodes in order of $f(n)$, A* search requires one fewer node evaluation

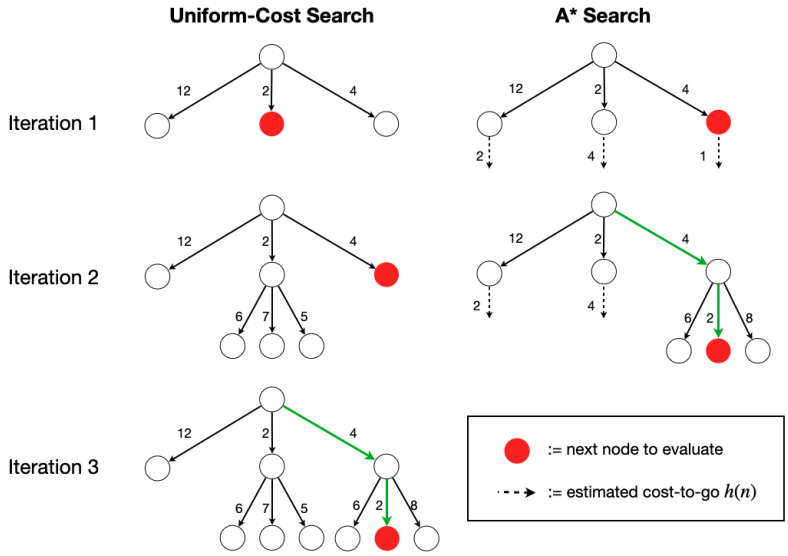


Figure 5.1: Comparison of uniform-cost search (UCS) and A* algorithms for a problem of depth $H = 2$. Values on the search tree branches correspond to the step costs, while dotted show estimates of the cost-to-go $h(n)$. UCS takes three iterations to reach the solution path, while A* requires two. By expanding nodes in order of $g(n) + h(n)$, one fewer node evaluation is required for A* search.

while still finding the optimal solution.

In order to make A* search a guided tree search algorithm, guided expansion (Equation 4.12) is applied to select a subset of actions $A_\pi(s)$ for each state s, using an expansion policy $\pi(a|s)$. We discussed guided expansion in detail in Section 4.12. Using guided expansion, a reduced search tree is created with a smaller branching factor than the exhaustive (unguided) tree. A* search is then used to find the shortest path through the reduced tree.

Unlike UCS, A* is not a general-purpose algorithm that is applicable 'out-of-the-box' to any tree search problem as the heuristic $h(n)$ is problem-specific.[1] In order to apply Guided A* search to the UC problem, in Section 5.3 we develop three heuristics which estimate the cost-to-go (that is, the remaining operating costs from node n to the search horizon).

We apply the same real-time strategy described in Section 4.4.2, Algorithm 5, whereby Guided A* is used repeatedly to solve T sub-problems, with the first action in the solution being used to determine the root of the next sub-problem. As discussed in Section 4.4.2, this prevents exponential run time complexity in the number of decision periods T. A* search is least efficient when using no heuristic, i.e. $h(n) = 0$ for all n, which is equivalent to UCS. As a result, the worst-case time complexities of A* and UCS are the same: $\mathcal{O}(2^{NH}T)$ without guided expansion, and $\mathcal{O}(T(\frac{1}{\rho})^H)$ with guided expansion using branching threshold ρ. However, we show in Section 5.4.1 that with an appropriate heuristic, the absolute run times of Guided A*

[1] Excluding the case where $h(n) = 0$ for all n, where A* is equivalent to UCS.

search are lower than Guided UCS (on average) due to improved search efficiency.

5.2.2 Guided IDA* Search

In this section we describe an *anytime* algorithm, based on iterative-deepening A* (IDA*) search. As described in Section 3.5.3, anytime (or interruptible) algorithms can be terminated at any point and return a solution. The development of an anytime algorithm for the UC problem is motivated by the results in Section 4.6.2, where we found the run time of Guided UCS is highly variable between episodes. In Section 5.4.1, we show that Guided A* exhibits similar characteristics, with over an order of magnitude separating the shortest and longest episode run times for fixed parameter settings. Furthermore, in Section 4.6.1 we found that average run time of Guided UCS is highly sensitive to parameter choices. Run time depends on characteristics of the episode (such as demand variation) and cannot be easily predicted given a set of parameters. The depth parameter H is difficult to tune as it has a significant impact on both solution quality and run time (to which it is exponentially related). Given the time-constrained nature of UC problems, there is motivation to develop an anytime algorithm for the UC problem which can be terminated when a time budget is spent (such as before market closure), rather than running to completion.

Iterative deepening [224], discussed in Section 3.6.3, is a general strategy that has been applied to a wide range of tree search algorithms and can be used to create anytime algorithms. The principle behind iterative deepening is to use a search algorithm to solve search trees of increasing search depth. A sub-optimal first action is found almost immediately by looking only one timestep ahead. Thereafter, the depth is increased at each iteration and the search is conducted again. The solution quality improves the longer the algorithm is run. The algorithm terminates once a stopping criterion (based on run time or solution quality) is met.

We apply iterative-deepening to the A* search algorithm described in Section 5.2.2 (IDA* [224]). Pseudocode for IDA* is shown in Algorithm 6. Our implementation of IDA* replaces the depth parameter H in A* and UCS with a time budget parameter b (seconds). A* search is used to iteratively solve the sub-problem rooted at r with a gradually increasing depth H. When the time budget b has elapsed, the last solution is returned. A significant advantage of the Guided IDA* algorithm is that the depth parameter H is replaced by the time budget b. In the context of UC, the time budget b may easily be determined by market constraints, such as settlement period length. Using an anytime algorithm like Guided IDA* ensures that computational resources are fully exploited within time constraints. Note that Algorithm 6 does not follow the same implementation as the popular IDA* algorithm described in [224], where each iteration corresponds to a gradually increasing cost-related cutoff bound.

Both Guided A* and Guided IDA* require a problem-specific heuristic $h(n)$

Algorithm 6 Anytime IDA* search algorithm for the UC problem from intial state r. A* search is run with progressively increasing search horizon H until the time budget b is spent.

```
function IDAStar(r, b)
    H ← 1
    repeat
        solution ← AStar(r, H)
        H ← H + 1
    until time budget b is spent
    return solution
end function
```

to be applied to the UC problem. In the next section we present three heuristics based on a priority list algorithm for estimating the cost-to-go $h(n)$ in Guided A* and Guided IDA*.

5.3 Heuristics for Unit Commitment

As informed search algorithms, the A* and IDA* search algorithms described in Section 5.2 require a problem-specific heuristic $h(n)$, which is used to estimate the lowest cost from node n to a goal node. The choice of heuristic is an important decision that has significant impact on the effectiveness of informed search relative to uninformed methods [61].

In this section, we will begin by justifying our approach of using priority list (PL) algorithms as the basis for UC heuristics, with reference to the heuristic properties of run time, admissibility and accuracy outlined in Section 3.6.4. We will then present three PL-based heuristics: Naive Minimum Marginal Fuel Cost (MMFC), Naive Economic Dispatch (ED) and Constrained ED. Finally, we will evaluate the run time, admissibility and accuracy of the three heuristic methods.

5.3.1 Choice of Heuristic Approach

In Section 3.6.4 we gave run time, admissibility and accuracy as important properties of heuristics impacting the efficiency improvements achieved by informed search (A* search) compared with uninformed search (UCS). Run time is significant since this impacts the extent to which efficiency improvements from fewer node evaluations are offset by the heuristic computation itself. Admissibility, which requires that the heuristic $h(n)$ should not over-estimate $h^*(n)$ is important as it is a criterion of optimality for A* search [230]. Finally, accuracy (that is, the average error between the estimated $h(n)$ and optimal $h^*(n)$) measures the heuristic's ability to effectively prune sub-optimal sub-trees. A perfectly accurate heuristic (where $h(n) = h^*(n)$) functions as an oracle, yielding maximal efficiency improvement by immediately

identifying the optimal path; a naive heuristic (where $h(n) = 0$) yields no efficiency improvements as A* is equivalent to UCS. Heuristic run time, admissibility and accuracy are often traded-off in practice. For instance, more complex heuristics may achieve higher levels of accuracy at the expense of run time or admissibility [61].

There is no all-purpose approach to designing effective heuristics for a particular problem domain. Some widely-studied problems have well-established heuristics. For instance, in route planning problems, a common heuristic is the straight-line distance from n to the destination [61, 62, 257, 258]. Alternatively, expert pattern databases may be used in some problems, such as the Rubik's cube puzzle [222]. Supervised learning has also been used to learn $h(n)$ for route planning problems [234]. The choice of heuristic has a significant impact on the efficiency of informed search algorithms [61].

Due to the lack of research in applying tree search methods to the UC problem, no established UC-specific heuristics exist. In the UC problem, $h(n)$ aims to estimate the lowest expected operating costs from n to a node at the search horizon. There is no 'distance' measure that is analogous to the straight-line distance in route planning which can be applied in the UC problem. Furthermore, unlike problem like the Rubik's cube, there are innumerable states in the UC problem due to its continuous state space, so pattern database approaches cannot be easily applied. Supervised learning methods are possible in principle, but generating a large dataset would be computationally expensive. In addition, the supervised learning task of predicting future operating costs given a state would be very challenging, due to the highly non-linear and non-continuous operating cost function and the admissibility criterion would be difficult to satisfy.

Our approach is based on the priority list (PL) algorithms described in Section 3.7.1. Improvements in MILP have made PL methods largely obsolete for practical UC problems, due to PL methods' lack of optimality guarantees and reliance on complex rules to fix constraints. However, PL methods have appropriate characteristics for designing heuristics. PL algorithms are very fast to compute, satisfying the run time property. In addition, the ability to relax constraints can be exploited to further reduce run time and increase admissibility by making cost estimates more optimistic. Lastly, PL algorithms are well-studied solution methods to the UC problem in their own right and are therefore capable of achieving a high degree of accuracy by approximating optimal schedules.

In the following section we develop three PL-based heuristics and compare their run time, admissibility and accuracy properties in Section 5.3.3. In Section 5.4.1 we evaluate efficiency improvements achieved in practice when each is applied in Guided A* search.

5.3.2 Priority List Heuristics

We will now present the three heuristic methods developed for the UC problem. The heuristics use different PL algorithms to generate an approximate UC schedule for the following periods up to the search horizon H. The operating costs of this schedule are then evaluated and used to approximate the optimal cost-to-go $h^*(n)$. All three heuristics are based on a PL ordering of generators by their economic efficiency at $p^{\max}$, which is the minimum marginal fuel cost (MMFC) ($ per MWh). The MMFC (in $ per MWh) which we denote q_i is the lowest point on the generator's fuel cost curve (e.g. Figure 4.2) and found by evaluating the cost function for generator i at $p_i^{\max}$ and dividing by capacity:

$$q_i = \frac{a_i(p_i^{\max})^2 + b_i(p_i^{\max}) + c_i}{p_i^{\max}} \tag{5.2}$$

The PL orders generators in increasing order of MMFC q_i. The highest priority generator (first to be commmitted) is that which has the lowest q_i. In order to reduce computational costs and increase the heuristics' optimism (admissibility), we only consider fuel costs, ignoring start costs and lost load costs. With the PL ordered by q_i, we present three heuristics for estimating the cost-to-go $h(n)$.

Heuristic I: Naive MMFC

The first heuristic, Naive MMFC, follows Algorithm 4 described in Section 3.7.1, committing generators without consideration for inter-temporal (only minimum up/down time in our problem setup) constraints in order of q_i (Equation 5.2). For each period t up to the search horizon (that is up to the depth of the search tree), generators are committed in increasing order of q_i until:

$$\sum_{i \in K_t} p_i^{\max} \geq D_t \tag{5.3}$$

where K_t is the set of generators committed at period t and D_t is the forecast demand. Note that no reserve constraint is enforced. By ignoring minimum up/down time constraints, the UC schedule produced by Naive MMFC is not guaranteed to be feasible but can be calculated very quickly.

To calculate the operating costs, we assume that all generators are fully-loaded ($p_i = p_i^{\max}$) except the last committed (marginal) generator which is part-loaded and satisfies the remaining load. This means the operating level constraints $p_i^{\min} \leq p_i \leq p_i^{\max}$ of the marginal generator can be violated and the dispatch may be infeasible for this generator. As a result, we do not evaluate the quadratic fuel cost curves in Equation 4.1 to calculate operating costs, instead we take the dot product of generator dispatch vector $\boldsymbol{p}_t$ and the MMFC $\boldsymbol{q}$:

$$C_t^f = \frac{1}{2}(\boldsymbol{q} \cdot \boldsymbol{p}_t) \tag{5.4}$$

where the factor $\frac{1}{2}$ accounts for the settlement period interval of 30 minutes. By assuming all generators operate at maximum efficiency, this operating cost estimate is optimistic and hence likely to be admissible. The heuristic $h(n)$ is calculated as the sum of C_t^f up to the search horizon H.

Heuristic II: Naive ED

The second heuristic method, Naive ED, is slightly more advanced than Naive MMFC, since we use the lambda-iteration method described in Section 3.7.3 to solve the economic dispatch (ED) problem. This gives a more accurate estimate of operating costs, at the expense of increased run time. Like Naive MMFC, by ignoring the minimum up/down time constraints, the UC schedule is not guaranteed to be feasible.

Naive ED generates a commitment schedule using the same method as for Naive MMFC, ignoring minimum up/down time constraints. Given this schedule, the ED problem is solved for each period t, and the resulting operating costs calculated using the quadratic fuel cost function (Equation 4.1).

Heuristic III: Constrained ED

The final heuristic, constrained ED, partially considers the minimum up/down time constraints of generators. Generators are committed in PL order, but must first serve their minimum up/down time constraints. That is, any generator that has been offline for less than its minimum down time in the state represented by the node n must remain offline until it becomes available, with the same applying for minimum up time constraints.

Once the schedule has been produced, the ED problem is solved for each period, in the same way as Naive ED. Compared with Naive MMFC and Naive ED, which do not consider up/down time constraints, Constrained ED is more expensive to compute due to the additional logic rules, but is able to recognise states where constraints are likely to be encountered. For instance, states where a base-load generator has recently been switched off will be recognised as having reduced capacity for the following periods, and less efficient generators may be required. Constrained ED is likely to give the most realistic UC schedule and the most accurate estimate of $h^*(n)$, the optimal cost-to-go. However, it is also the most complex and hence the slowest heuristic to calculate. As discussed in Section 3.6.4, heuristic run time is an important property as it partially offsets efficiency improvements achieved by using an informed search method.

In summary, it should be emphasised that none of the three heuristics are guaranteed to offer feasible UC solutions as minimum up/down time constraints are not fully observed in any case. Constrained ED only considers the initial up/down

time constraints, preventing generators which are constrained to remain on/off at node n from being decommitted/committed before satisfying those constraints. Thereafter, generators can be cycled without obeying these inter-temporal constraints. As a result, none of the heuristics can be used on their own to solve the UC commitment problem without further modifications to satisfy constraints.

Having presented the three heuristics, next we will analyse their properties in terms of run time, accuracy and admissibility properties described in Section 3.6.4.

5.3.3 Analysis of Heuristics

We will now analyse the Naive MMFC, Naive ED and Constrained ED heuristics presented in the previous subsection in terms of run time, admissibility and accuracy. These properties were discussed in Section 3.6.4 as being important factors impact the efficiency improvements achieved by informed search relative to uninformed search.

In order to evaluate the admissibility and accuracy of the three priority list heuristic methods, we compare the predicted cost-to-go $h(n)$ with the optimal cost-to-go $h^*(n)$. Unguided UCS, which is guaranteed to produce optimal solutions, is used to calculate $h^*(n)$. The 20 test UC problem instances for the 5 generator problem described in Section 4.2.3 were solved with UCS with $H = 4$ (a larger value of H was not practical due to exponential run time complexity of UCS), and the heuristics evaluated for each root node n in the solution path. In total, this produced $48 \cdot 20 = 960$ values for $h(n)$ (for each heuristic) and $h^*(n)$. We then calculated admissibility as the proportion of nodes satisfying the admissibility criterion:

$$h(n) \leq h^*(n) \tag{5.5}$$

We measured accuracy for the 960 estimates using the root-mean squared error. For N estimates $h(n)$ and optimal $h^*(n)$, the RMSE is calculated using:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(h^*(n_i) - h(n_i))^2} \tag{5.6}$$

In addition, we measured run time by sampling 1000 random nodes, and measuring the mean time taken to calculate $h(n)$ with a depth $H = 4$.

The run time measurements, along with admissibility and accuracy results are summarised in Table 5.1. The Naive MMFC heuristic (which is the simplest PL heuristic) had the lowest run time, by roughly a factor of 3. Constrained ED was the slowest, with 11% higher mean run time than Naive ED. Naive MMFC was 100% admissible in our experiment, but was the least accurate, having the highest RMSE. All three heuristics succeed in making mostly admissible estimates: for Naive MMFC, 100% of estimates are admissible; Naive ED achieves 95% and Constrained ED 98%

Heuristic	Mean time (ms)	RMSE	Admissibility (%)
Naive MMFC	0.59	1528.27	100.00
Naive ED	1.70	1134.88	95.00
Constrained ED	1.89	992.75	98.12

Table 5.1: Summary of run time, root mean squared error and admissibility (proportion of estimates where $h(n) \leq h^*(n)$) for the three PL heuristics.

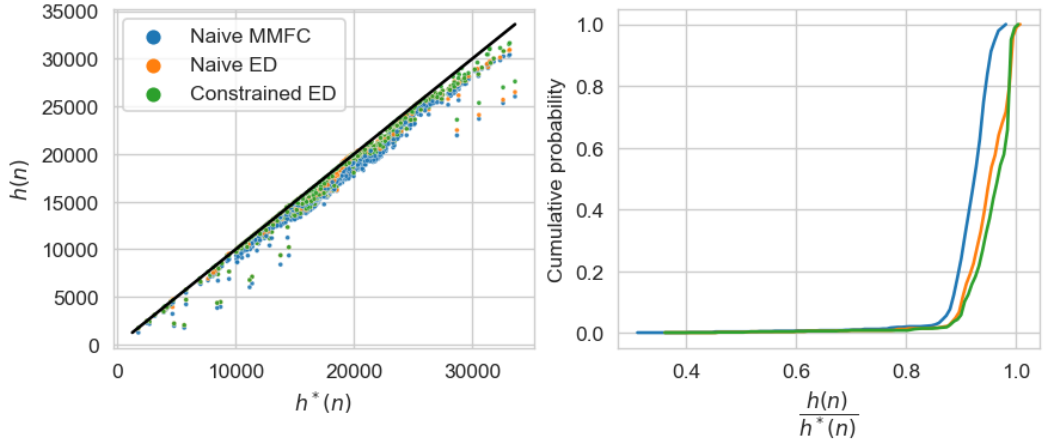


Figure 5.2: Admissibility of the three PL-based heuristics. Both plots use the same data: optimal cost-to-go $h^*(n)$ versus the heuristic $h(n)$ heuristic estimate (left) and cumulative distributions showing the proportion of admissible estimates (right).

admissibility.

The estimated and optimal cost-to-go $h(n)$ and $h^*(n)$ are plotted in Figure 5.2. In the left-hand plot, points below the black line $h(n) = h^*(n)$ are admissible estimates $h(n) \leq h^*(n)$. Nodes with the highest $h^*(n)$ are generally under-estimated by the largest margin. It is likely that the optimal paths at these nodes include a relatively large loss of load probability (LOLP) whose costs are not considered by any heuristic and may be the cause of this gap. The same data is used to calculate cumulative distributions of $\frac{h(n)}{h^*(n)}$ in the right-hand plot. This plot clearly shows that Naive MMFC is the most optimistic and least accurate heuristic, and is only within 5% of $h^*(n)$ (that is, $\frac{h(n)}{h^*(n)} > 0.95$) in 23% of cases. By contrast, $\frac{h(n)}{h^*(n)} > 0.95$ in 60% of cases for ED, and 70% of cases for Constrained ED.

In summary, the three heuristics exhibit different properties, and there is a clear trade-off between run time and accuracy with accuracy improving with increased run time. Naive MMFC is significantly faster but less accurate than the other two heuristics. Naive ED is the least promising heuristic as it has the lowest admissibility and lower accuracy than Constrained ED, with only a 0.19ms lower run time. Constrained ED is the most accurate and has high admissibility of 98%, but is also more than 3 times slower than Naive MMFC. Having analysed the heuristic properties, in Section 5.4.1 we evaluate the efficiency improvements achieved in

practice by each heuristic when applied in Guided A* search.

5.4 Experiments

We now conduct two experiments applying heuristics developed in Section 5.3 to Guided A* search and Guided IDA* search using the power system defined in Section 4.2. We solve the test problems described in Section 4.2.3 and compare performance with Guided UCS, which was shown in Section 4.6.3 to outperform MILP benchmarks.

In the first experiment, each of the three heuristic methods is applied in A* search. We compare the run time of Guided A* using each heuristic with that of Guided UCS, finding Constrained ED to result in the largest run time reduction as compared with Guided UCS with no significant deterioration in solution quality. In the second experiment, we use IDA* with the Constrained ED heuristic and a varying time budget. We compare the performance of the time-limited anytime algorithm IDA* with the depth-limited A* and UCS search algorithms, finding IDA* to result in lower operating costs in comparable run times. Guided IDA* is shown to be the strongest guided tree search algorithm for the UC problem developed in this thesis, with practical benefits stemming from its anytime property. The algorithm achieves significant operating cost improvements as compared with MILP benchmarks and can be applied more effectively to larger power systems than fixed-depth search methods. We demonstrate this by solving 100-generator UC problems in Section 5.5.

5.4.1 Guided A* Search

In the first experiment, Guided A* with $H = 4$ was used to solve the 10, 20 and 30 generator test problems used in the previous chapter, described in Section 4.2.3. Each of the three PL heuristics (Naive MMFC, Naive ED and Constrained ED) described in Section 5.3 was used. We used the same expansion policies trained in Section 4.5.3 for guided expansion. We compare run times and operating costs with the Guided UCS ($H = 4$) results from Section 4.6.3.

The results are shown in Table 5.2 and Figure 5.3. All of the heuristics achieve significant run time reductions as compared with Guided UCS: greater than 86%, 53%, 75% improvement for 10, 20 and 30 generator problems, respectively, with small deviations in operating costs ($\leq 0.09\%$). The most effective heuristic for improving search efficiency in Guided A* was Constrained ED, where run times are reduced by 94%, 64% and 82% for 10, 20 and 30 generator problems, respectively. Despite being the fastest heuristic to compute, Naive MMFC achieves the smallest efficiency improvements in all cases except the 30 generator problem, where Naive ED is the least effective. Due to some heuristic estimates being inadmissible, there are small cost differences between UCS and A* search with Constrained ED for 20 and 30

Generators	Heuristic	Time (% of UCS)	Cost (% of UCS)
10	Naive MMFC	13.55	100.00
10	Naive ED	11.03	100.00
10	Constrained ED	6.41	100.00
20	Naive MMFC	46.64	99.96
20	Naive ED	45.20	100.08
20	Constrained ED	35.62	100.03
30	Naive MMFC	24.19	100.01
30	Naive ED	24.55	100.09
30	Constrained ED	17.74	100.08

Table 5.2: Difference in mean run time and operating cost using Guided A* search with each of the three heuristic methods, compared with Guided UCS. Guided A* with all three heuristics achieves significant run time reductions, with only small changes in operating costs ($< 0.1\%$.)

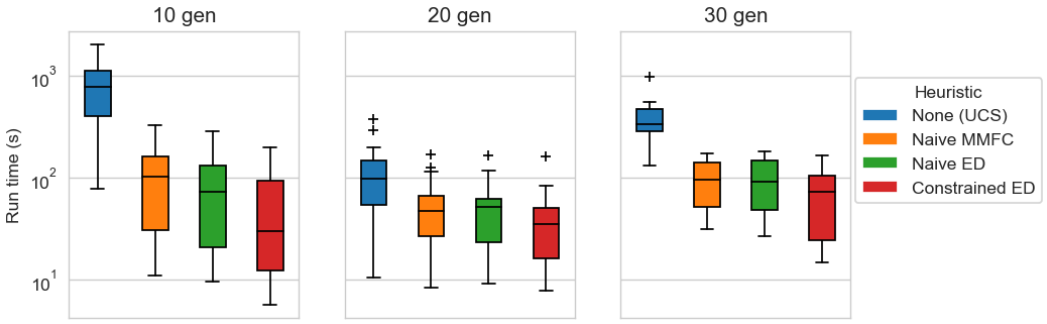


Figure 5.3: Run times (log-axis) of Guided A* search ($H = 4$) with the three heuristic methods and using no heuristic (i.e. uniform-cost search, results in Section 4.8). All three heuristics achieve significant run time improvements relative to Guided UCS, with Constrained ED providing the largest speed-up.

generator problems, indicating that unlike UCS, Guided A* search is not guaranteed to be optimal over the lookahead horizon. However, the deterioration in solution quality is small, with a maximum increase of 0.09% in operating costs. The run times savings afforded by A* search as compared with UCS free computational budget that can be used to increase the search depth H and compensate for the small deterioration in operating costs.

Run Time Variability

In Section 4.6.2, Figure 4.8 showed significant variation in episode run time using Guided UCS. Simple problem instances (those with a low branching factor using guided expansion) are solved quickly, while more challenging problems may take more than an order of magnitude longer. Figure 5.3 shows that Guided A* search exhibits similar characteristics, with episode run times typically varying by over an order of magnitude.

The variation in episode run time can be further explained by variation in period run times due to variable search breadth throughout the day. Figure 5.4 shows the

variation in period run time and average search breadth (measured at the root of the search tree) with respect to decision period for different settings of H, using A* search with the Constrained ED heuristic. Search breadth is generally lower in the early morning periods, indicating the expansion policy used in guided expansion is relatively 'certain' during these periods, with probability mass concentrated in a small number of actions. Later in the day the search breadth increases in all cases, although not uniformly between the 10, 20 and 30 generator problems. Mean search breadth in the 10 generator case roughly follows a typical demand profile, increasing in the morning (around period 14 or 7:00), decreasing in the middle of the day and increasing for the evening peak. By contrast, 20 and 30 generator problems had higher search breadth at the end of the day. This may be due to the larger uncertainties at the end of the day, due to the propagation of forecast errors. In addition, the 20 and 30 generator problems have larger action spaces as well as symmetries deriving from duplicate generators, which may lead to a more uniform distribution of probability mass over actions as compared with the 10 generator problem.

The search breadth trends are reflected in period run time trends, shown in the bottom row of Figure 5.4. Periods with larger search breadth are solved more slowly due to the computational expense of using A* search to solve a broader search tree. As the number of nodes in the search tree scales with b^H, where b is the branching factor, the period run time is highly sensitive to changes in b caused by varying certainty of the expansion policy $\pi(a|s)$. Several orders of magnitude separate the mean run time in the early morning periods with the periods in the middle of the day. Solutions to the UC problems are usually characterised by more frequent commitment changes during peak periods to meet demand, contributing to greater uncertainty in decision-making and larger branching factors. The period run time variability is ultimately the cause of high variability in episode run time.

The run time variability of Guided A* motivates the anytime algorithm Guided IDA* developed in Section 5.2.2. In the following section, we apply Guided IDA* search with the Constrained ED heuristic to the test problems.

5.4.2 Guided IDA* Search

Using the strongest heuristic, Constrained ED, we applied Guided IDA* to the test problems of 10, 20 and 30 generators with a time budget of $b = \{1, 2, 5, 10, 30, 60\}$ seconds per period, constraining episode run time to a maximum of 48 minutes (2880 seconds). As with the previous experiment using Guided A* search, we used the expansion policies trained in Section 4.5.3.

Operating cost savings relative to Guided UCS solutions from Section 4.6.3 are shown in Figure 5.5. Costs were generally found to decrease with increasing time budget. For all three problems, budgets $b \geq 10$ seconds outperform Guided

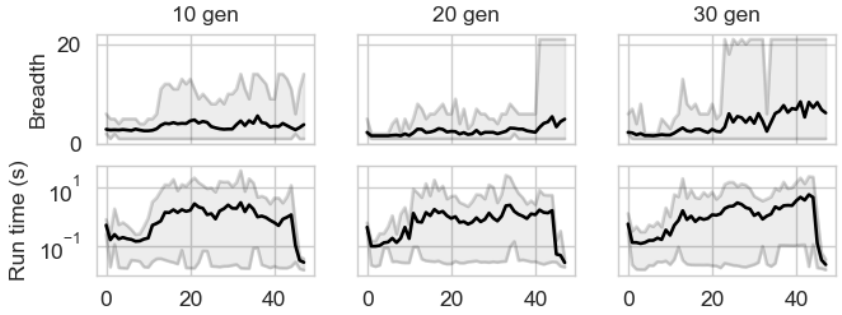


Figure 5.4: Mean, minimum and maximum search breadth at the root node (top row) and run time (bottom row) by period for A* search using the Constrained ED heuristic. Search breadth and run time are lower during early morning periods in all problem sizes, and larger later in the day. The sharp decline in run time at the end of the day in all instances is due to the truncated search horizon ($H < 4$).

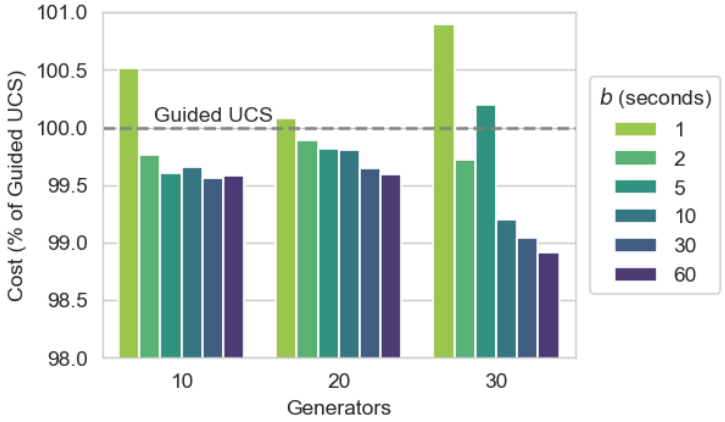


Figure 5.5: Cost saving of Guided IDA* with Constrained ED compared to Guided UCS. Operating costs generally decrease with increasing time budget. The largest improvements are found in the 30 generator case, where IDA* is 1.1% cheaper than Guided UCS when $b = 60$ seconds.

UCS. The largest savings were achieved in the 30 generator case, where costs were 1.1% lower with $b = 60$. However, in this case there was less consistent performance for lower time budgets, with operating costs increasing between $b = 2$ and $b = 5$. Compared with the deterministic MILP(4σ) benchmarks from Section 4.2.3, IDA* achieved lower operating costs for time budgets $b \geq 2$ in all problem sizes.

IDA* with $b = 30$ seconds is compared with Guided A* (using Constrained ED) and Guided UCS (from Section 4.8), both with $H = 4$, in Table 5.3. We show $b = 30$ (maximum run time of 24 minutes per problem) as this budget is most comparable in run time to Guided UCS. The results show that IDA* achieves lower operating costs than both Guided UCS and Guided A*, while mitigating the run time variability.

Improvements in operating costs can be attributed to greater average search depths using Guided IDA* compared with Guided UCS and Guided A*, where

Num. gens	Method	Heuristic	Mean cost ($M)	Std. cost	Mean time (s)	Max. time	Min. time	LOLP (%)
10	IDA*	Constrained ED	9.33	0.75	1086.3	1294.1	892.0	0.115
	A*	Constrained ED	9.37	0.84	51.8	195.8	5.7	0.128
	UCS	None	9.37	0.84	807.3	1992.1	76.9	0.128
	MILP(4σ)	None	9.40	1.02	19.1	177.2	1.8	0.180
20	IDA*	Constrained ED	18.67	1.46	1099.6	1267.6	797.0	0.116
	A*	Constrained ED	18.74	1.44	41.8	159.3	7.7	0.112
	UCS	None	18.73	1.41	117.3	374.5	10.4	0.107
	MILP(4σ)	None	18.90	2.92	5.5	8.4	3.8	0.244
30	IDA*	Constrained ED	28.14	1.96	1210.0	1359.7	986.0	0.110
	A*	Constrained ED	28.43	2.07	69.4	164.4	14.5	0.142
	UCS	None	28.41	2.04	391.3	976.8	129.4	0.142
	MILP(4σ)	None	28.53	4.94	8.0	12.4	5.6	0.291

Table 5.3: Comparison of IDA* ($b = 30$ seconds), A* ($H = 4$) and UCS ($H = 4$) for 10, 20 and 30 generator problems.

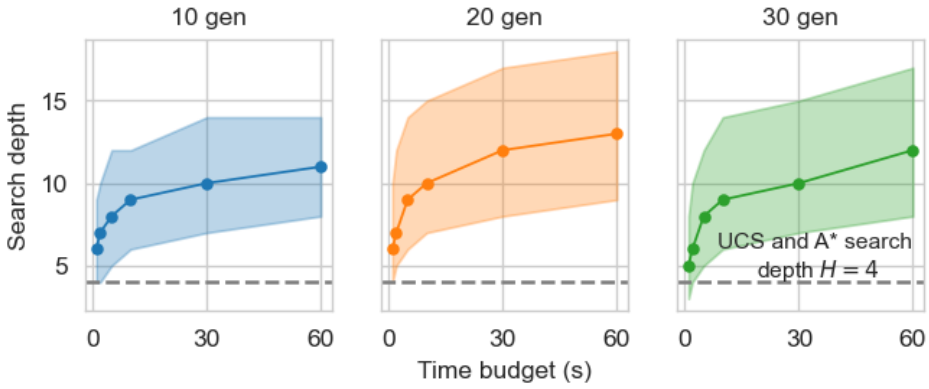


Figure 5.6: Variation of Guided IDA* search using Constrained ED heuristic. Solid line and points show the median search depth; shaded area indicates inter-quartile range. Dotted line shows $H = 4$, the search depth used in Guided UCS and Guided A* search. The average search depth of Guided IDA* is significantly greater than these methods for all time budgets.

search depth was fixed at $H = 4$. Figure 5.6 shows the median search depth H using Guided IDA* with varying time budget b. Even with the lowest time budget of $b = 1$ second per period, the median search depth is $H \geq 5$ and at larger time budgets is significantly higher than $H = 4$ which was used for UCS and A*. Median search depth increases logarithmically with respect to the budget b, increasing by one for approximately each doubling of b. Retaining the search tree after each timestep means that search depth is inherited in subsequent periods, yielding significantly higher search depths on average for Guided IDA*. While search tree retention is not unique to Guided IDA*, the retained search tree in Guided A* and Guided UCS is used to reduce run time of solving subsequent periods rather than allow for deeper search. Whereas Guided A* adaptively spends computational resources (reflected in run time) on planning from more complex states, Guided IDA* adaptively reduces search depth.

Figure 5.7 shows the variation in median search depth throughout the day for a budget $b = 30$s. Similar to search *breadth* of Guided A*, shown in Figure 5.4, the

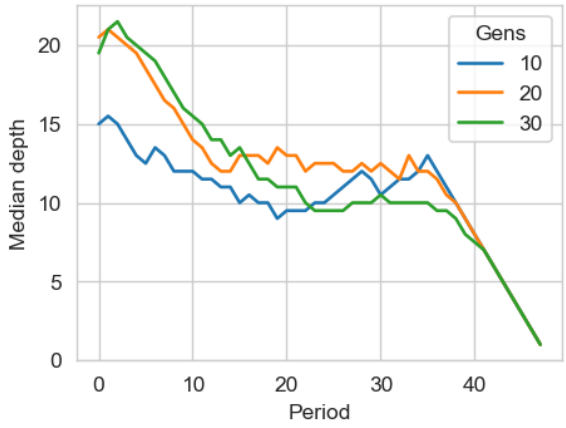


Figure 5.7: Median search depth for IDA* with b = 30 seconds and Constrained ED heuristic. Deeper search is achieved in the early morning periods, where search breadth is comparatively narrow.

figure shows that during the simpler early morning periods, search depth is relatively large, due to a lower branching factor and greater certainty in these periods. The large search tree created in early morning periods where median depth is greatest benefits periods later in the day due to search tree retention.

Redundant Computation in Iterative Deepening

Iterative deepening algorithms necessarily involve a certain amount of additional computation compared with their non-iterative counterparts (such as IDA* and A*) due to the search tree being rebuilt at each iteration. As noted in [61], in practice this is usually not an important consideration due to the exponential relationship between the depth of a generation and the number of nodes in that generation.

Furthermore, applying Guided IDA* to the UC problem, repeated evaluation of nodes is not a significant computational concern, as most of the computational cost of a node evaluation can be retained after the first iteration. Figure 5.8 shows the run time composition of a node evaluation consisting of 4 elements. The step cost (solving the ED problem for each of the net demand scenarios in $\mathcal{S}$ (see Section 4.4.1) and evaluation of the expansion policy (required for guided expansion) constitute the majority of total run time. These components, as well as the transition function (i.e. taking a step in the simulation environment), only need to be computed on the first node evaluation in IDA*. The heuristic is the only component which needs to be re-evaluated after the first visit, which accounts for $< 10\%$ of the total run time.

5.5 100-Generator Problem

In Section 5.4, we showed that Guided A* search achieved better search efficiency by leveraging domain knowledge through a heuristic. We then showed that the

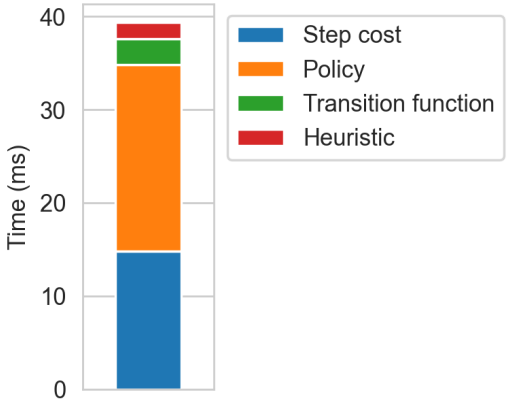


Figure 5.8: Composition of run time of the major routines of initial node evaluation in Guided IDA*. *Step cost* is the economic dispatch calculations required to determine expected operating costs over net demand scenarios. *Policy* is the neural network evaluation for guided expansion. *Transition function* evaluates the system dynamics, advancing to a new state. *Heuristic* (here using Constrained ED) is the only component which is evaluated when a node is revisited. When a node is revisited in IDA*, the computational cost is around 8% of the first visit.

anytime property of Guided IDA* means that further operating cost reductions can be achieved in practice by more effectively exploiting computational resources. These improvements invite the application of Guided IDA* to larger power systems in order to evaluate its potential for practical applications in regional or national-scale transmission networks.

In this section we apply Guided IDA* to a power system of 100 generators, which to the best of our knowledge is the largest simulation study applying RL and/or tree search to the UC problem. Due to the high dimensionality of the action space, we use a novel form of entropy regularisation during training which aims to promote policies which match a pre-determined *target entropy*. As opposed to maximum entropy RL techniques used previously in this research, the target entropy regularisation technique used here penalises policies whose entropy differs from a pre-determined target entropy. This ensures that policies achieves a level of stochasticity that is appropriate for guided tree search. In this section we show that Guided IDA* is capable of achieving operating costs that are competitive with industry-standard MILP approaches.

5.5.1 Target Entropy Regularisation

Guided expansion requires a stochastic policy $\pi(a|s)$ in order to build a search tree with several branches from each node. As described in Section 3.4.3, policy entropy quantifies the randomness of a policy: high entropy policies are more stochastic than low entropy policies. In high dimensional action spaces, there are practical

challenges in training a policy with a suitable level of policy entropy such that several actions satisfy the branching threshold ρ in guided expansion, while ensuring the policy does not converge to a deterministic one. In Section 4.5.3, the entropy-regularised PPO objective function $J^{\text{PPO+}H}$ (Equation 3.27) was used to promote stochastic policies via an *entropy bonus*. This method was found to be a successful approach for systems of up to 30 generators, with Guided UCS, A* and IDA* all outperforming MILP benchmarks when trained with this technique. As shown in Table 4.5, we decreased the level of entropy regularisation via the entropy coefficient β in the entropy-regularised PPO objective function (Equation 3.27) as the number of generators increased. This reflected the increasing dimensionality of the action space, and thus the requirement for a lower entropy policy for guided expansion. However, as the dimensionality of the action space grows, tuning the level of entropy regularisation becomes increasingly difficult, and may even demand negative entropy bonus (i.e. $\beta < 0$) to produce satisfactory expansion policies. Furthermore, the final level of policy entropy for a given level of entropy regularisation may be difficult to predict, depending on the exploration of the policy and state spaces during training. Therefore, it is difficult to ensure a usable policy with a suitable entropy is produced during training.

To mitigate this problem of tuning β in high-dimensional action spaces and control the final policy entropy level, we implement a novel entropy regularisation technique based on *target entropy*. Rather than employing an entropy bonus, we introduce a term to the objective function based on the squared error of the policy entropy and the target entropy. Specifically, we use the following modified PPO objective function:

$$J^{\text{PPO+}H_T}(\theta) = \mathbb{E}[J^{\text{PPO}}(\theta) + \beta(H(\pi_\theta) - H_T)^2] \tag{5.7}$$

where $J^{\text{PPO}}(\theta)$ is the PPO objective taken from Equation 3.25 in Chapter 3, β is a constant entropy coefficient controlling the level of regularisation and H_T is the target entropy. Higher levels of H_T promote more stochastic policies, and higher settings of β penalise deviations from target entropy more strongly.

A heuristic method can be used to set the target entropy H_T as a function of the number of generators N and the branching threshold ρ, uniting the elements of policy training and guided tree search. For a given ρ, we aim to achieve an expected entropy such that the joint action probability $\pi(a = [a_1, a_2, ..., a_N]|s) = \rho$ on average, where $a_i \in \{0, 1\}$ is the commitment decision for generator i. As we use a sequential policy architecture, described in Section 4.5.2, each sub-action $p = \pi(a_i|s)$ should be $p = \rho^{\frac{1}{N}}$ on average. Using the definition of the entropy of a random variable in Equation 3.28, the target entropy as a function of ρ and N is:

$$H_T = -p \log_2(p) - (1-p) \log_2(1-p) \tag{5.8}$$

$$p = \rho^{\frac{1}{N}} \tag{5.9}$$

In the following section we investigate the impact of H_T and β on policy training and convergence for the 100 generator problem.

5.5.2 Training Details

Expansion policies were trained for the 100-generator system using the method described in Section 4.5.3, using the target entropy regularisation described in Section 5.5.1. Using a grid search approach, we studied the impact of parameters H_T and β on policy training.

The training parameters are shown in Table 5.4. The buffer size (representing the number of policy evaluations per epoch) was increased from 2000 in previous experiments to 5000. This reflects the fact that more forward passes are required to generate a single action using the sequential policy parametrisation described in Section 4.5.2. We also used a network architecture of two hidden layers with 128 and 64 nodes for both actor and critic architectures. The total policy training time was approximately 30 hours over 8 CPU workers.

We demonstrate the impact of two target entropy regularisation parameters, target entropy H_T and the entropy coefficient β, on policy training using a grid search approach. For target entropy, we set $H_T \in \{0.1567, 0.1919, 0.2648\}$, corresponding to $\rho \in \{0.01, 0.05, 0.1\}$ respectively according to Equation 5.8. For each setting of H_T, we trained a policy with entropy coefficient $\beta \in \{0.1, 1.0\}$, which was scaled linearly beginning at $\beta = 0$ over 100,000 epochs to prevent premature convergence. The policies with target entropy regularisation were compared to a baseline trained without entropy regularisation (i.e. $\beta = 0$).

Figure 5.9 shows the convergence of mean operating costs and entropy during training of the 7 expansion policies (6 using target entropy regularisation, 1 baseline with no regularisation). The policies with $\beta = 1$ (solid lines) converge to a policy with stable entropy H_T, although there are significant variations in performance, such as the drop in performance towards the end of training for $H_T = 0.2648$. Those with a lower level of entropy regularisation $\beta = 0.1$ (dotted lines) display gradually increasing entropy throughout training, and achieve lower average reward than those with $\beta = 1$. The policy with $H_T = 0.1919$ and $\beta = 1.0$ was found to converge to the highest average reward, while the policy with no entropy regularisation was the worst performing. In addition, this baseline policy with no entropy regularisation converges to the lowest level of policy entropy, indicating a tendency in large power systems to converge to near-deterministic policies with entropy regularisation, which

Variable	Value
Clip ratio	0.1
Actor architecture	128, 64
Critic architecture	128, 64
Entropy coefficient β	$\{0.1, 1.0\}$
Target entropy H_T	$\{0.1567, 0.1919, 0.2648\}$
Buffer size	5000
Epochs	300,000
Gamma	0.95

Table 5.4: Parameters used to train 100-generator policies. Combinations of the target entropy regularisation variables β and H_T are used in a grid search studying policy convergence with respect to these parameters.

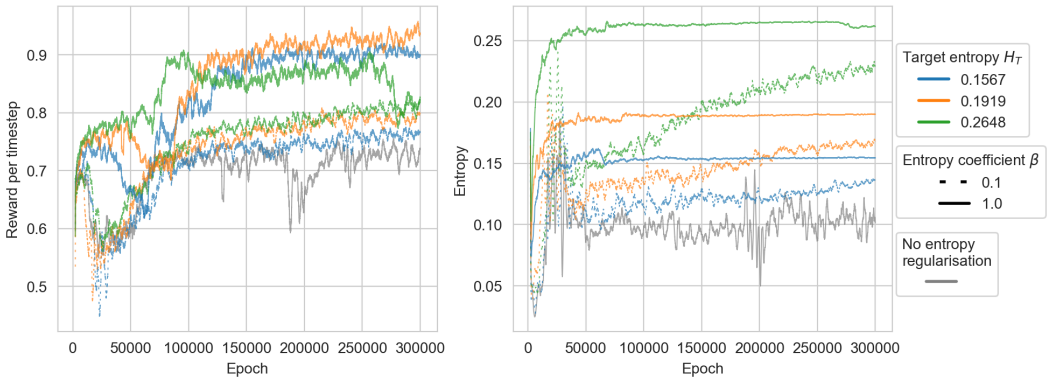


Figure 5.9: Convergence of 100 generator expansion policies using target entropy regularisation with varying H_T and β. The grey line shows policy training with no entropy regularisation. The left plot shows reward per timestep while the right plot shows policy entropy. Both plots display a moving average over 2000 epochs.

is not suitable for guided tree search.

Overall, the policy trained with $H_T = 0.1919$ and $\beta = 1.0$ was found to achieve the highest average reward at the end of training. In the following section, we use this policy in Guided IDA* search to solve the 20 unseen test problems for the 100-generator problem.

5.5.3 Results

Using the expansion policy trained with $H_T = 0.1919$ and $\beta = 1.0$ and using the Constrained ED heuristic, Guided IDA* search was used to solve the 20 unseen UC problem instances. The branching threshold was set to $\rho = 0.05$, corresponding to the target entropy $H_T = 0.1919$ set with Equation 5.8, and the time budget was set to $b = 60$ seconds as in previous experiments. Comparison is made with the deterministic UC benchmarks MILP(4σ) and MILP(perfect).

Version	Mean cost ($M)	Std. cost	Mean time (s)	Max. time	Min. time	LOLP (%)
MILP(perfect)	87.45	0.00	30.0	47.5	24.6	0.000
Guided IDA*	96.65	9.73	2737.1	2782.0	2563.1	0.188
MILP(4σ)	96.78	20.47	40.2	59.5	27.7	0.397

Table 5.5: Comparison of Guided IDA*, MILP(4σ) and MILP(perfect) solutions to 100 generator test problems.

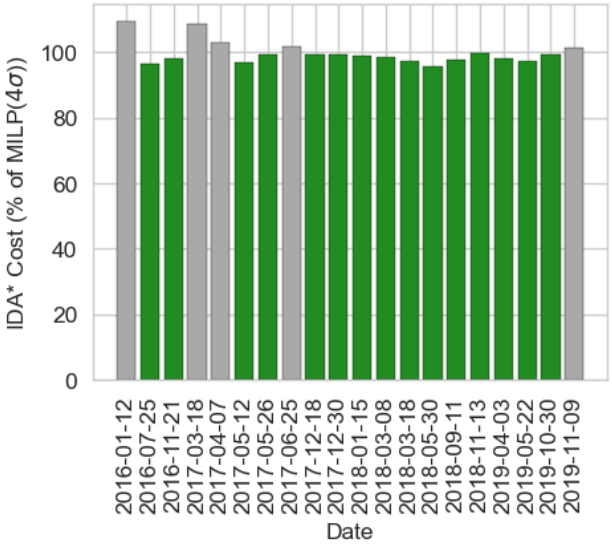


Figure 5.10: Day-by-day comparison of IDA* operating costs with MILP(4σ). IDA* has lower mean costs than MILP on 15 out of 20 days (those in green).

The results are shown in Table 5.5. Operating costs were 0.14% lower using Guided IDA* search as compared with MILP(4σ), and 11% higher than MILP(perfect). Guided IDA* had greater security of supply than the reserve-constrained MILP(4σ), achieving lower LOLP (0.19% compared with 0.40%) and reducing standard deviation in operating costs by a factor of 2. A comparison of mean operating costs between IDA* and MILP(4σ) broken down into individual test problems is shown in Figure 5.10. There is significant variation in expected Guided IDA* costs as a percentage of MILP(4σ), ranging between 96% to 109%. Overall, IDA* has lower operating costs than MILP in 15 of the 20 days, marked in green in Figure 5.10.

Guided IDA* generally adopts more flexible operating patterns than MILP(4σ). Figure 5.11 shows the distribution of up times (proportion of periods spent online) for Guided IDA* and MILP(4σ). Guided IDA* adopts lower utilisation rates for the 20 baseload generators and higher rates for peaking plants and also uses roughly twice as many startups as compared with MILP(4σ). The generation patterns of MILP(4σ) shown in Figure 5.11 are more stratified as compared with Guided IDA*. This indicates similarities with a merit order or priority list commitment approach, whereas Guided IDA* adopts more nuanced strategies. Furthermore,

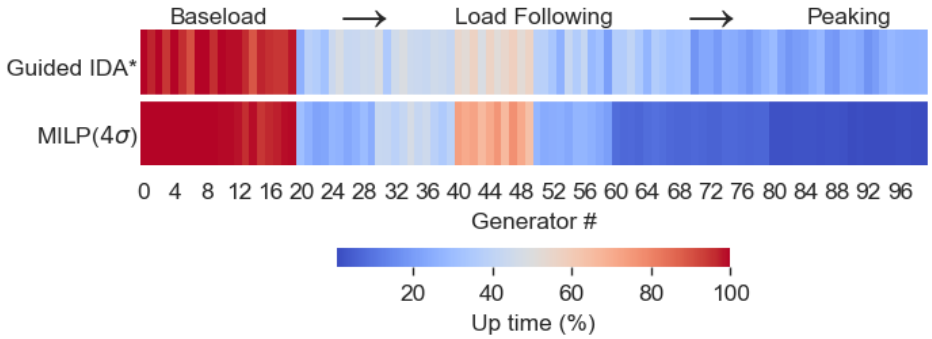


Figure 5.11: Proportion of periods spent online for generators in the 100-generator problem using Guided IDA* and MILP(4σ). Small capacity peaking plants are shown at the right of the graph, with base-load at the left.

Guided IDA* search uses more extreme actions than MILP(4σ), changing up to 41 generator commitments simultaneously, compared with a maximum of 20 for MILP(4σ). In addition, Guided IDA* made more use of the 'do nothing' action, keeping all generator statuses the same. This follows similar results found in Section 4.6.3, where we observed Guided UCS made more extreme commitment changes, followed by longer periods of no commitment changes.

5.6 Discussion

Using guided expansion in a modular way, any tree search algorithm can be enhanced using an RL-trained agent as a guide. This chapter showed that the choice of tree search algorithm (UCS, A*, IDA*) is an important design decision in the broader class of guided tree search methods, and has a significant impact on performance. Using informed and anytime search methods yielded substantial performance improvements and practical benefits as compared with Guided UCS, and allowed for the successful application of Guided IDA* to larger UC problems of 100 generators.

5.6.1 Advantages of Informed Search

In the context of informed search, we showed that the heuristic is an important design decision that can substantially impact search efficiency. Since there is no single heuristic approach that is suitable for all problem domains, designing the heuristic for the UC problem requires domain expertise: our approach using priority list algorithms was based on UC literature that showed that these methods have properties that are suited to heuristic design such as short run time and the ability to relax constraints [1]. There is inevitably a trade-off between run time, accuracy and admissibility, but our results found that the most accurate and slowest heuristic (Constrained ED) was clearly the best performing in practice, reducing run time by

up to 94% as compared with Guided UCS with negligible impact on solution quality. There is scope for further research into more accurate heuristics that can further improve search efficiency.

5.6.2 Advantages of Anytime Search

Anytime search with Guided IDA* was shown to outperform Guided A* for similar computational budgets, by maximising use of computational resources. Whereas some days were solved with very short run times by Guided A* due to narrow search breadth, Guided IDA* compensates by adaptively increasing the search depth in such situations. In practice, Guided IDA* achieves far greater search depths on average, as shown in Figure 5.6, resulting in lower operating costs for similar computational budgets. The anytime property of Guided IDA* is a significant practical advantage over UCS and A* for the UC problem, allowing for schedules to be reliably produced in time-constrained contexts. Both Guided UCS and Guided A* search methods have highly variable and unpredictable run times (Figure 5.3) which makes the branching threshold and search depth parameters ρ and H difficult to tune in practice. In particular, due to the exponential complexity in H, the run times of Guided UCS and Guided A* are very sensitive to the search depth. In IDA*, H is replaced with a time budget b, which can be set by knowledge of market constraints, such as the time to gate closure when bids and offers must be submitted.

Increasing the time budget in IDA* generally resulted in operating cost reductions (Figure 5.5). However, in all three problems, even small time budgets $b \geq 2$ seconds per period were enough to outperform the MILP(4σ) benchmark and $b \geq 10$ outperformed Guided UCS. Further increasing the budget has a stabilising effect in addition to reducing costs. This is most clear for the 30 generator case, where costs fluctuate significantly between budgets of 1, 2 and 5 seconds per period, but increase steadily for larger budgets.

5.6.3 Planning in Complex Decision Periods

We observed analogous trends between the variations in period run time observed for Guided A* search (Figure 5.4) and the depth variations when using IDA* (Figure 5.7), which reflect the relative uncertainty in decision-making during these periods. For Guided A*, run times were higher around the morning peak. By contrast, Guided IDA* conducts shallower search in periods with higher decision-making uncertainty where the branching factor is greater. There may be contexts where methods with adaptive run time, such as Guided A* search, are more appropriate than methods with adaptive depth (Guided IDA*); shallow search in complex decision periods using Guided IDA* may result in unreliable or insecure decision making. A hybrid approach, for instance enforcing a minimum search depth in IDA*, could also perform

better in some contexts.

5.6.4 Scaling to Larger Power Systems

Applying IDA* to the 100 generator problem, which has a very large action space of up to 2^{100} actions, we found Guided IDA* achieved operating costs that were 0.14% lower than the MILP(4σ) benchmark. While expected operating costs were similar to the MILP(4σ) benchmark, Guided IDA* schedules were significantly more secure, with lower LOLP and variation in total operating costs.

In previous experiments on smaller power systems, we used traditional entropy regularisation with an entropy bonus, using the objective function $J^{\text{PPO+}H}$ (Equation 3.27), to encourage exploration during training and an appropriate level of policy entropy for guided tree search. This was shown to be an effective strategy for systems of up to 30 generators but was not so effective for the 100-generator problem. Introducing the target entropy term to the PPO loss function, as described in Section 5.5.1, had a significant impact on policy training, with policies converging to higher average rewards when target entropy regularisation was used. Setting the target entropy H_T based on the number of generators and the branching threshold ρ was also shown to be a useful heuristic for uniting policy training and policy testing in large discrete action spaces.

In Figure 5.9, the smoothest convergence of mean reward was observed for policies with target entropy regularisation and $\beta = 0.1$, although convergence was faster with $\beta = 1.0$. All other policies exhibited sharp, temporary performance losses that indicate relatively unstable and unpredictable convergence properties which were more pronounced than for smaller systems. The practical challenges of training expansion policies in large action spaces invites further research, and improvements in expansion policy training could yield significant performance improvements. Overall, the application of Guided IDA* to the 100 generator problem is a significant milestone, and our results are promising for practical applications of RL-based methods to regional or national-scale electricity network operation. To the best of our knowledge, this is the largest application of RL and/or tree search to the UC problem in the existing literature.

5.7 Conclusion

In this chapter we presented two guided tree search algorithms, building on the method of guided expansion developed in Section 4.5.1: Guided A* search and Guided IDA* search. Both algorithms are informed search methods, using problem-specific knowledge to improve search efficiency at the expense of generality across problem domains. Guided IDA* is additionally anytime, which is practically advantageous in

time-constrained contexts such as electricity markets.

We developed three heuristics based on priority list UC solution methods, which are used to rapidly approximate the optimal solution cost and are used by both algorithms to improve search efficiency relative to Guided UCS. Guided IDA* was shown to mitigate the run time variability of non-anytime algorithms, reaching greater search depths and reducing operating costs by up to 1.0% relative to Guided UCS within similar time budgets.

The improvements afforded by the informed and anytime search algorithms in this chapter allowed for Guided IDA* search to be applied to a larger power system of 100 generators. A novel technique of target entropy regularisation was used to improve policy convergence and unite policy training with tree search. Expected total operating costs were found to be lower than the MILP benchmark with a reserve constraint and Guided IDA* outperformed MILP on 15 out of 20 days, with a lower loss of load probability overall. These results demonstrate the potential for guided tree search methods to outperform existing industry methods in large-scale contexts.

Chapter 6

Case Studies: Curtailment and Outages

6.1 Introduction

Existing research has shown that RL algorithms can be used to achieve expert performance over a broad range of tasks, with little or no adaptation of the algorithm itself [26, 28, 35, 36]. Compared with MILP approaches, which often require expert domain knowledge to develop a suitable mathematical formulation, flexibility across problem domains is a valuable property of RL that is beneficial for operating power systems with heterogeneous technologies for power generation, consumption, transmission and storage. Furthermore, RL has been shown to provide insights into solution techniques for complex problems [27, 44, 214, 218], most famously shown in novel gameplay strategies used by AlphaGo [44]. These studies have shown that RL is able to learn fundamentals of problem-solving across many problem domains without human intervention. Having demonstrated that Guided IDA* is an effective solution method for the UC problem in Chapter 5, in this chapter we extend the RL approach to solve two more advanced variants of the UC problem, considering carbon intensity and power system security.

The case studies reflect current challenges in the development of power systems and the integration of variable renewable energy [259]. First, we consider carbon pricing and an additional action to curtail wind generation. Increasing curtailment of variable renewable energy [260] poses a challenge to climate goals as the curtailed energy is usually replaced by fossil-fuel generators, but is now an essential part of power systems operation that must be optimised alongside other decisions. Additionally, carbon pricing is an important policy mechanism to achieve CO_2 emissions reductions, but entails changes to the objective function, thus requiring different operational strategies favouring lower carbon generation. In the second case study, we introduce generator outages to the environment. Outages are a crucial security consideration for system operators, currently handled by $N-x$ reserve criteria that protect against the largest loss of generation. However, current security practices must adapt to increasing penetration of variable renewables [261] and coincident outages [52] which have been the cause of recent major blackouts such as the GB power network outage

of 9 August 2019, which impacted over 1 million customers [59].

The experiments in this chapter show that Guided IDA* is a highly flexible methodology that can be applied to environments of arbitrary complexity without significant modifications beyond parameter tuning. It is also highly scalable in the number of scenarios N_s, which allows problems with large numbers of uncertain parameters to be approached probabilistically without heuristics.

6.1.1 Contributions

This chapter makes the following contributions:

1. We present two variations on the power system simulation environment presented in Section 4.2 and formulate corresponding MDPs for each. In the first environment we implement a **curtailment action** and a **carbon price**. In the second we introduce **generator outages** to the environment. These environments present novel and challenging tasks that can be used to benchmark RL approaches to the UC problem considering heterogeneous actions and stochastic generation.

2. Guided IDA* is used to solve unseen test problems with three levels of carbon price in the curtailment environment. We observe changes in schedule characteristics as the carbon price is increased, such as a reduction in coal generation, employment of gas as base-load and fewer startups. Our results show that Guided IDA* generalises across MDPs with different action spaces and is sensitive to changes in the reward function.

3. In the generator outages case study, Guided IDA* is shown to significantly outperform MILP benchmarks with $N-x$ reserve criteria. Guided IDA* adaptively allocates reserve margins that vary between $N-1$ and $N-4$ levels and achieves operating costs that are up to 1.9% lower than the best performing MILP benchmark. Security of supply remains similar to that shown in previous experiments without outages. We show that Guided IDA* can be used to discover optimal reserve allocation strategies with uncertain generation availability without the use of heuristics.

4. We investigate the impact of the number of scenarios N_s used to build the search tree on Guided IDA* solution quality in the outages case study. Our results show that with large numbers of uncertain parameters, increasing N_s can be used to improve solution quality. Due to the linear time complexity in N_s and anytime quality of Guided IDA*, we show that a large number of scenarios can be considered while constraining run time and maintaining deep search.

This chapter is organised as follows. Section 6.2 covers the curtailment and carbon price case study. In this section we describe the problem environment, MDP formulation, experimental setup and the results of our experiments applying Guided IDA* to solve unseen test problems. Section 6.3 follows the same structure for the generator outages case study. In Section 6.4 we reflect on the results of both experiments, discussing implications for power systems operation. Section 6.5 concludes the chapter.

6.2 Case I: Wind Curtailment and Carbon Price

In this case study, we modify the power system simulation environment described in Section 4.2 to include an additional wind curtailment action, which can be used to temporarily reduce wind generation. In addition, we incorporate a carbon price into the environment's cost function and investigate the impact of carbon price levels on the operating strategies and CO_2 emissions of Guided IDA* schedules.

The effects of carbon pricing and curtailment are important topics of research for current and future power systems. Increasing levels of wind penetration have been accompanied by high rates of wind curtailment in several countries and transmission networks including Great Britain [262], China, Texas and Italy [260]. While in general the system operator aims to maximise wind penetration as it has no marginal cost, in some cases it is necessary to curtail wind generation in order to manage transmission network congestion or to ensure an adequate level of controllable reserve generation is available [51]. As a result, curtailment is an important power systems operational decision that can be used to benefit overall system security and can in some cases reduce operating costs and carbon emissions compared with other measures [263, 264]. Currently, curtailment decisions are often made by the system operator on an ad-hoc, manual basis [265]. Studies have investigated automating short horizon curtailment decisions as remedial actions to maintain grid security [265, 266]. In addition, some studies have investigated co-optimisation of curtailment alongside UC in a day-ahead context, with the aim of reducing total system operating costs [267–271]. However, curtailment decisions have not been introduced in RL studies for the UC problem.

This case study shows that Guided IDA* is applicable to MDPs and environments with different characteristics to those studied in Chapters 4 and 5. Compared with MILP approaches, introducing new actions to the MDP does not require modification of the solution method or manual reformulation of the problem. Furthermore, we analyse the impact of carbon pricing on operating strategies and curtailment rates of Guided IDA* solutions. We first describe the setup of the environment, modified from that described in Section 4.2 to incorporate the wind curtailment action and carbon price. We then formalise by describing the modified MDP. Finally we present the results of experiments applying Guided IDA* to solve the 20 test problems.

6.2.1 Environment Setup

To incorporate wind curtailment and carbon price, we modified the power system environment described in Section 4.2. We made two changes: first, we adjusted the fuel cost curves based on a carbon price. Second, we implemented an additional curtailment action, which is scheduled in a day-ahead context alongside generator commitment. The curtailment action and carbon price can be activated in a modular fashion in the open-source Python package developed for this research[1].

We adopt the approach used in [270] of treating a wind curtailment action as a decision variable (i.e. action in the MDP). As we do not consider transmission network constraints, the purpose of the curtailment action is to maintain grid security by: (1) ensuring there is sufficient downward ramping capacity in cases of high wind generation and low demand; (2) reducing the net demand uncertainty to reduce reserve requirements. As wind is considered as negative demand in our problem setup, using the curtailment action has the effect of increasing net demand. The carbon price is reflected in adjusted fuel cost curves for the thermal power stations. The following sections describe the environment modifications in more detail.

Carbon Price-Adjusted Fuel Cost Curves

In order to incorporate a carbon price, we adjusted the quadratic fuel cost curves (Equation 3.38) for each generator to reflect the increased cost per unit of fuel combusted. Fuel types were not included in the original Kazarlis data [5] (Table 4.1) and were manually assigned to either coal, oil or gas.

The generator fuel assignments are shown in Table 6.1 and were decided based on properties of generators in the original data, which naturally formed three groups. The first group comprises the largest capacity (455 MW) generators 1 and 2, and was assigned the coal fuel type. These generators have properties that are most characteristic of base-load, with the largest startup costs, most restrictive up/down time constraints and lowest marginal costs. The second group includes the medium capacity (80–162 MW) load-following generators 3–7, which we assigned to gas. The final group includes generators 8–10 which are the smallest capacity (55 MW) and most flexible (shortest minimum up/down time constraints); this group was assigned to oil. As in previous experiments, to create the 30 generator power system used in this case study, we duplicated each generator three times.

Having assigned fuel types, we adjusted the original fuel cost curves C^f (Equation 4.1) to account for the carbon price using the following method. The adjusted fuel cost curves for a carbon price of \$50/tCO_2 are shown in Figure 6.1, which also shows the capacity and fuel cost curve characteristics of the three fuel types. Each curve is transposed upwards when the carbon price is applied, indicating higher total fuel costs.

[1] `https://github.com/pwdemars/rl4uc`

Generator	Fuel type	Characteristics	EF $[\frac{\text{kgCO}_2}{\text{MMBTU}}]$	FP $[\frac{\$}{\text{MMBTU}}]$
1–2	Coal	Base-load	95	1.30
3–7	Gas	Load-following	54	2.69
8–10	Oil	Peaking	73	3.18

Table 6.1: Generator fuel assignments, operational characteristics, carbon emissions factors EF and fuel prices FP.

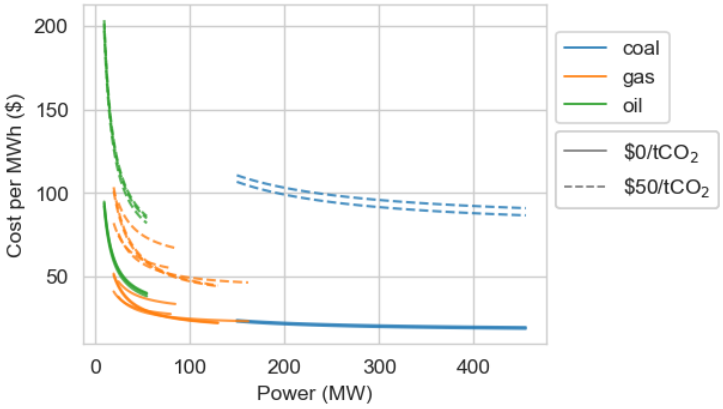


Figure 6.1: Fuel cost curves with a without a carbon price of \$50 per tCO_2 applied.

To calculate the adjusted fuel cost curves, original fuel cost curves $C^f(p)$ are decomposed into the product of heat rate $H(p)$ [BTU] (amount of fuel combusted as a function of generator output p) and the fuel price FP $(\frac{\$}{\text{BTU}})$:

$$C^f(p)[\$] = H(p)[\text{BTU}] \times FP\left[\frac{\$}{\text{BTU}}\right] \tag{6.1}$$

Rearranging Equation 6.1 and subsituting the original quadratic fuel cost curves from Equation 3.38, the heat rate curve is expressed in terms of the original cost curve coefficients a, b, c and FP:

$$H(p) = \frac{C^f(p)}{FP} \tag{6.2}$$

$$= \frac{ap^2 + bp + c}{FP} \tag{6.3}$$

The heat rate curve can be used to calculate the carbon-adjusted price by multiplying by the sum of fuel price FP and fuel-specific carbon price. The fuel-specific carbon price is the product of the fuel's CO_2 emissions factor EF and the carbon price:

$$EF\left[\frac{\text{tCO}_2}{\text{BTU}}\right] \times CP\left[\frac{\$}{\text{tCO}_2}\right] \tag{6.4}$$

In summary, for a generator i with initial fuel cost curve coefficients a_i, b_i, c_i (included in the original data [5], Table 4.1), fuel price FP_i, emissions factor EF_i and carbon price CP (which are all manually-set constants), the carbon-adjusted fuel cost curve $C^f_{c,i}(p)$ can be calculated using the following equation:

$$C^f_{c,i}(p) = H_i(p)(FP_i + EF_i \times CP) \tag{6.5}$$

$$C^f_{c,i}(p) = \frac{a_i m_i p^2 + b_i m_i p + c_i m_i}{FP} \tag{6.6}$$

$$m_i = FP_i + EF_i \times CP \tag{6.7}$$

For a given carbon price CP, the following variables (which are not defined in the original source data [5]) are required to adjust the fuel cost curves:

- Fuel price FP for coal, gas and oil (\$ per BTU)
- Carbon emissions factors EF for coal, gas and oil (tCO_2 per BTU)

We used 1996 (year of publication of [5]) average annual fuel prices for coal, gas and oil from [272] to set the fuel prices FP shown in Table 6.1 (in \$/MMBTU). The emissions factors EF are also shown in Table 6.1 and are properties of the fuels. With these constants, CP is included as a parameter in the power system environment and can be used to simulate operating costs under different levels of carbon pricing. When $CP = 0$, the cost function is identical to that described in Section 4.2.1.

The adjusted fuel cost curves for carbon prices of \$0/tCO_2 and \$50/tCO_2 are shown in Figure 6.1. The original fuel cost curves CP=\$0/tCO_2 show the three distinct generator groups that were identified. When the carbon price is applied, all curves are transposed vertically to reflect larger marginal fuel costs. Coal generators experience the largest increase in fuel costs due to having the largest CO_2 emissions factor EF. Furthermore, cost curves overlap significantly in the vertical axis with CP=\$50/tCO_2. For instance, oil units operating at maximum capacity $p = p_{\max}$ have strictly lower marginal fuel costs than both coal power stations, but are more expensive when part-loaded. Therefore, optimal UC decisions are more sensitive to generator dispatch when the carbon price is applied.

Curtailment Action

To model wind curtailment scheduling, we added an additional curtailment action. The curtailment action is a binary decision which is scheduled in the same way as a generator commitment. As a result, UC schedules have increased dimensionality to $T \times (N + 1)$, with the additional column indicating the curtailment decision. The curtailment action reduces wind generation by 50%. A single discrete reduction was chosen to enable a straightforward search tree representation of the modified UC problem. Further discrete curtailment amounts could be used to enable greater

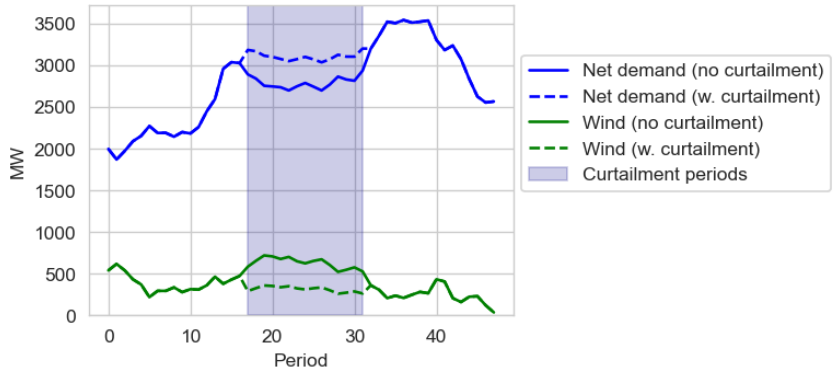


Figure 6.2: Example use of curtailment action. Curtailing wind during the afternoon increases net demand, preventing a reduction that might demand and shutdown and later startup of a generator before the evening peak.

flexibility. For continuous curtailment actions, tree search methods can be applied with techniques including progressive widening [273].

Figure 6.2 shows the change in the point forecasts for net demand and wind when the curtailment action is applied. The curtailment action smoothes the net demand profile which can be used to prevent a shutdown or keep generators running at higher load factors.

As the curtailment action reduces wind generation by 50%, the forecast errors are also assumed to reduce by the same factor, leading to a narrower distribution of wind generation and net demand realisations. The reduction in the 5–95% quantile interval for net demand is shown in Figure 6.3, where the curtailment action is applied uniformly throughout the day. The mean 5–95% interval for net demand decreases by 18% when the curtailment action is applied. In addition, the profile is noticeably smoother when the curtailment action is applied, due to the reduced wind generation variability. The curtailment action can therefore be used to reduce net demand variability and thus reduce spinning reserve requirements.

6.2.2 MDP Formulation

Having described the changes made to the simulation environment, we will now formalise the changes by making necessary modifications to the UC the MDP described in Section 4.3. The updated MDP contains two significant changes: the reward function is updated to reflect changes to the cost function, and the action definition is changed to incorporate the new curtailment action. Since there are no inter-temporal constraints on curtailment actions, the state and transition function definitions can be defined similarly to the original MDP in Section 4.3. We will now describe each MDP modification in turn.

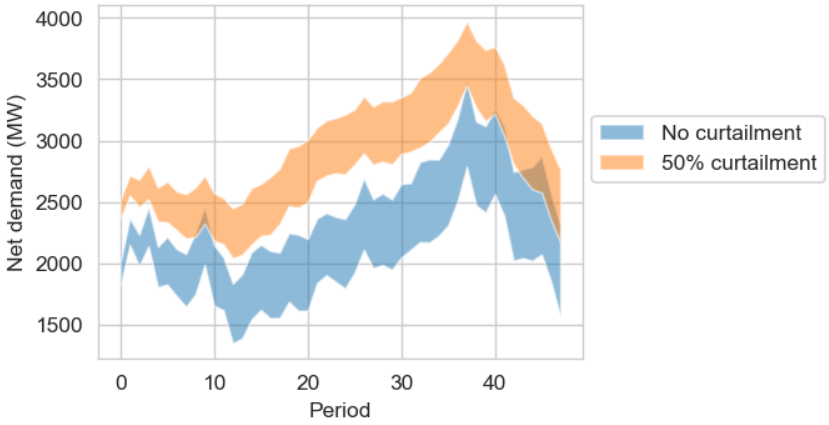


Figure 6.3: 5–95% quantile interval of net demand, with and without curtailment. The mean width of this interval is reduced by 18% in this case when the curtailment action is applied.

Reward Function

The reward function reflects the updated cost function which includes the carbon-adjusted fuel costs C_c^f :

$$r = -C \tag{6.8}$$

$$C = C_c^f + C^s + C^l \tag{6.9}$$

where C^s and C^l are startup costs and lost load costs respectively.

Action Definition

The action definition is updated to include the curtailment action, and is a binary vector of length $N+1$ for N generators:

$$\boldsymbol{a} = [a_1, a_2..., a_N, a_c] \quad a_i \in \{0, 1\} \tag{6.10}$$

where a_c indicates the curtailment action.

State and Transition Function Definitions

We will briefly describe the state and transition function definitions, although these are very similar to the original MDP. To reflect the impact of curtailment on demand, the wind generation forecast w_t and forecast error y_t are reduced by 50% when the curtailment action is applied at the previous timestep.

The MDP can be represented as a search tree using the method described in Section 4.4.1. The size of the action space $|\mathcal{A}|$ increases by a factor of two to 2^{N+1} with the additional curtailment dimension. In the following section, we will describe our experiments applying Guided IDA* to solve UC problems with carbon price and

curtailment.

6.2.3 Experimental Setup and Policy Training

The adjustments made to the environment described previously enable us to study the impact of carbon price on the solutions produced by Guided A* Search. Carbon pricing impacts the relative operating costs of different fuel types, leading to different optimal operating strategies. In addition, carbon price impacts the relative value of system security and renewables integeration, thus affecting the optimal curtailment rate.

Our results focus on the schedule characteristics at different levels of carbon price, rather than on operating costs in comparison with existing methods. Developing an MILP formulation of the UC problem including the curtailment action is beyond the scope of this thesis, and no MILP benchmark solutions were calculated. We investigate three carbon price levels $CP = \{0, 25, 50\}$ \$/tCO_2 and analyse solutions in terms of curtailment rate, CO_2 emissions, utilisation of fuel types and other system variables. Whereas previous chapters focused on scaling characteristics with the number of generators, in this chapter we exclusively examine the 30 generator problem. With the curtailment action, there are $2^{31} \approx 2$ billion actions.

The changes to the environment and MDP mean that new expansion policies are required in order to apply Guided IDA*. We trained an expansion policy for each carbon price level using model-free RL using PPO, as described in Section 4.5.3. All parameters were the same as described in Table 4.5 for the 30 generator problem except the number of epochs, which was increased to 500,000 in light of the increased problem complexity and larger action space. The wall clock training time over 500,000 epochs was approximately 26 hours, trained over 8 CPU workers as in previous experiments. We used the sequential policy parametrisation described in Section 4.5.3 represented by a feed forward neural network. When predicting the action sequence as visualised in Figure 4.10, the curtailment action a_c was predicted as the first value in the action sequence followed by the generator commitments $a_i, i \in \{1 \dots N\}$.

The convergence profiles of expansion policies for the three levels of carbon price are shown in Figure 6.4. The convergence to higher mean operating costs per timestep when $CP > 0$ is explained by the additional carbon costs incurred. As with the policies trained in Section 4.5.3, large improvements are made in the early stages of training within the first 1000 epochs, as the policy learns to avoid lost load events. Thereafter, comparatively small improvements are made as the policy is fine-tuned to minimise fuel and startup costs.

Using the trained expansion policies, we solved the 20 test episodes using Guided IDA* search, which was described in detail in Section 5.2.2. The test episodes are identical to those described in Section 4.2.3, with the same demand and wind profiles.

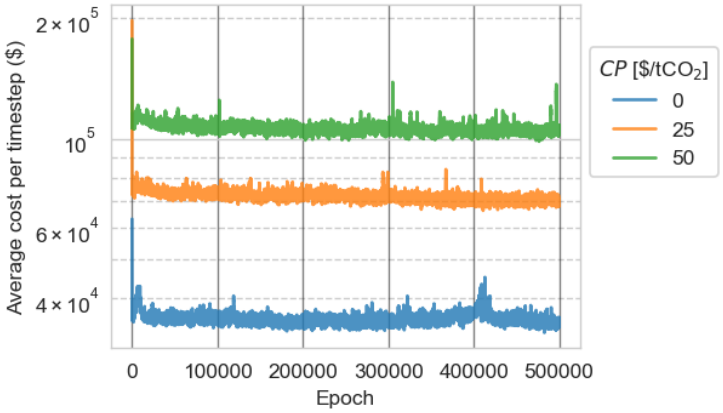


Figure 6.4: Convergence of expansion policies for the three carbon price levels. Each epoch represents 2000 policy evaluations. The policies trained with $CP > 0$ converge to higher average operating costs due to the additional carbon costs.

We set the time budget $b = 60$s, the branching threshold $\rho = 0.05$, number of scenarios $N_s = 100$ and used the Constrained ED heuristic as in previous experiments (Section 5.4.2).

6.2.4 Results

Table 6.2 summarises the results of our experiments using Guided IDA* to solve the 20 unseen test problems for each carbon price level. Increasing the carbon price from \$0 to \$25/tCO$_2$, there is a 22% reduction in carbon emissions and a 115% increase in total operating costs due to the additional carbon emissions costs. Higher total operating costs are inevitable under the carbon price rise in our problem setup, reflecting the cost burden of carbon pricing on consumers [274]. Further increasing the carbon price to \$50/tCO$_2$ yields comparatively minor reductions in CO$_2$ emissions, whereas total operating costs increase by a further 49%. Loss of load probability (LOLP) did not change substantially between \$0–25/tCO$_2$, but increased from 0.09% to 0.22% when increasing from \$25–50/tCO$_2$. The curtailment rate (defined as curtailed volume as a proportion of available volume) is lower at \$25 and \$50/tCO$_2$ due to the greater incentive to integrate wind generation and lower the average carbon emissions factor. The largest drop was observed between \$0–\$25/tCO$_2$, while there was a small relative increase between \$25–50/tCO$_2$. In the following sections we will analyse the changes to operational patterns under different levels of carbon price.

Changes to Generator Usage Patterns

The utilisation rates and operational patterns of different fuel types were found to vary with carbon price CP. Average committed capacity disaggregated by fuel type is shown in Figure 6.5. Total committed capacity increased consistently with increasing

						Avg. committed (MW)		
$/tCO$_2$	Cost ($M)	LOLP (%)	ktCO$_2$	Curtailment (%)	Startups	Coal	Gas	Oil
0	30.89	0.10	1638.27	2.01	426	2617	1273	276
25	66.39	0.09	1282.70	1.21	280	2329	1604	372
50	98.80	0.22	1253.66	1.30	121	2284	1726	451

Table 6.2: Comparison of Guided IDA* solutions to 30 generator test problems with curtailment action and varying carbon price.

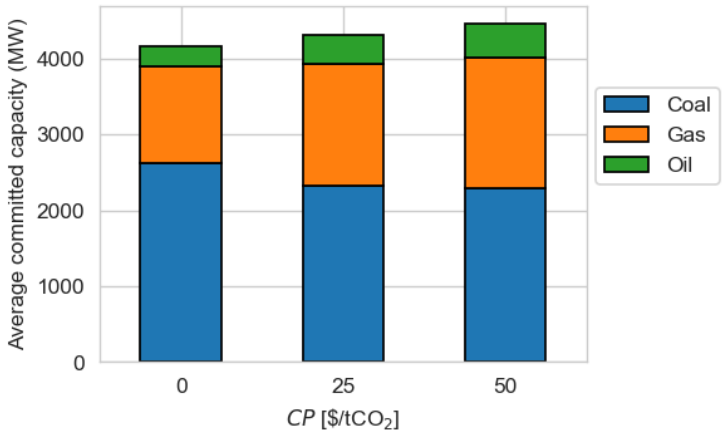


Figure 6.5: Committed capacity by fuel type for the three levels of carbon price. Gas and oil displace coal in terms of committed capacity as the carbon price increases. Total committed capacity also increases with carbon price.

CP, indicating larger reserve margins. This result is unexpected, as lost load costs are devalued relative to fuel-related costs when the carbon price is increased. The preference for different fuel types varied as expected with increasing CP: coal is displaced by gas and oil which have lower emissions factors. Figure 6.6 shows the generator-level utilisation, demonstrating the shift from coal to gas supplying base-load generation. In both the \$25 and \$50/tCO$_2$ settings, 2 of the 6 coal-fired power stations operate with significantly lower utilisation rates, whereas at \$0/tCO$_2$ all coal plants are online for >80% of periods. At the highest carbon price level, all gas units have near 100% utilisation rates. Similarly, oil is operated much more consistently when the carbon price is increased, representing a shift away from peaking use to load-following and even base-load in some cases.

Figure 6.7 shows the decline in total startups with carbon price, split by fuel type. Coal startups remain low for all three carbon price levels, with less than 10 startups. Gas startups decrease from 225 to 90 to 30 as the carbon price increases, adopting base-load operation patterns. Oil startups are not significantly reduced at a carbon price of \$25/tCO$_2$, but decrease substantially at \$50/tCO$_2$. Whereas Figure 6.6 shows that coal units are displaced in the merit order as the carbon price increases, the startups data show that coal still does not act flexibly at higher carbon prices. Fewer coal plants are committed on average, but these commitments tend to

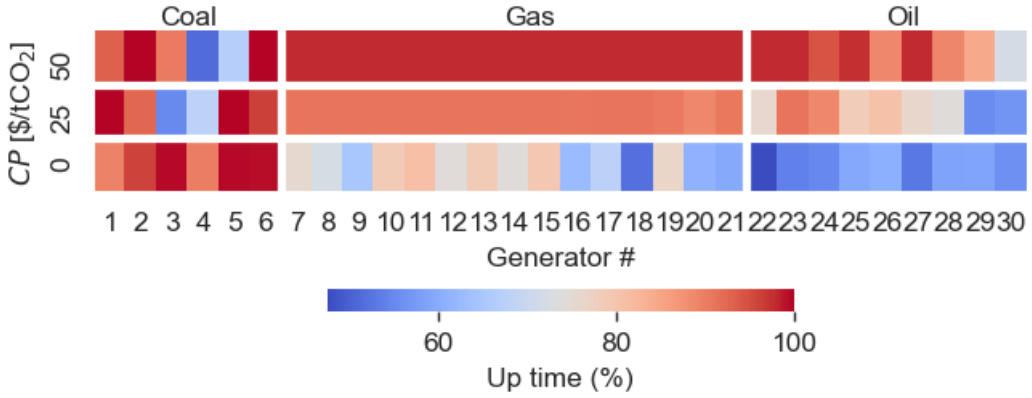


Figure 6.6: Up time of generators in the curtailment case study. Gas replaces coal as base-load generation, while oil shifts from peaking usage to predominantly load-following as the carbon price increases.

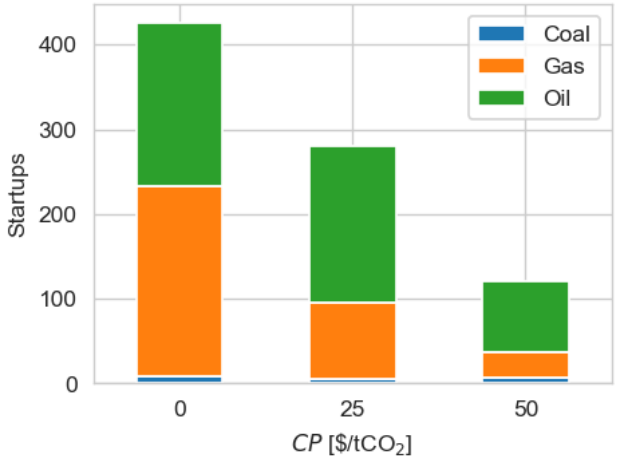


Figure 6.7: Total startups across the 20 test problems, disaggregated by fuel type. Coal startups remain roughly stable with <10 startups in all cases. Gas and oil startups decrease substantially as the carbon price is increases, as these fuel replaces coal as base-load generation.

remain fixed for each episode. This is in part due to the large startup costs for these generators, as well long minimum up/down times of 8 hours which prevents them being committed for short periods such as the evening peak.

Impact of Carbon Price on Curtailment Rate

Table 6.2 shows that the curtailment rate varied with the carbon price, and was lowest at CP = \$25/tCO$_2$. Combined with the relatively small decrease in CO$_2$ emissions between \$25–50/tCO$_2$, this indicates that there are diminishing returns from increasing CP beyond \$25/tCO$_2$ in this problem setup.

Furthermore, we observed differences in the circumstances under which the curtailment action was used at varying levels of CP. Figure 6.8 compares the

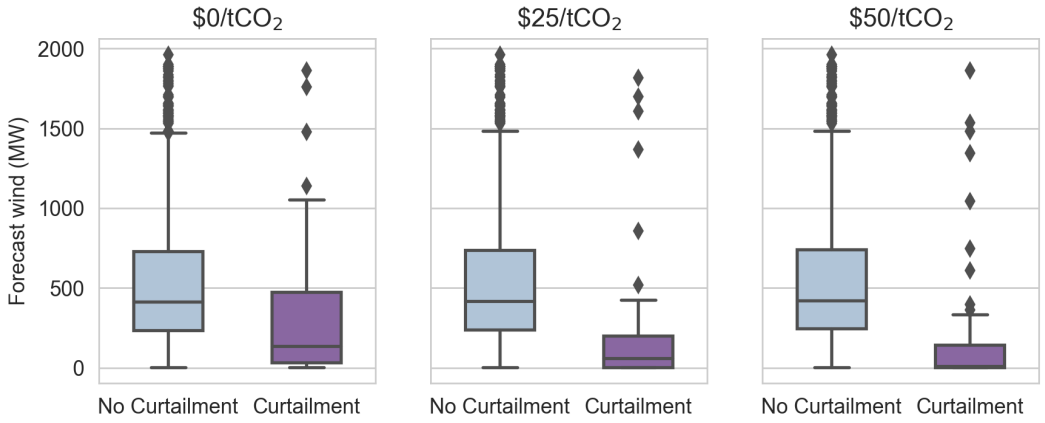


Figure 6.8: Distribution of forecast wind generation for periods when curtailment action is used versus when it is not. As the carbon price is increased, the curtailment action is used less frequently in periods of high wind generation.

distribution of forecast wind generation when the curtailment action is used versus when it is not for each of the carbon price levels. In the lowest carbon price case, there is a wider range of wind generation levels over which wind is curtailed, including periods of very high forecast wind generation. At carbon prices of \$25 and \$50/tCO$_2$, the curtailment action is used much more sparingly at higher wind penetration due to the greater value of zero-carbon generation relative to other sources. The median wind generation level at which the curtailment action is used thus decreases from 135 MW to 58 MW to 8 MW as CP increases. Curtailment of wind at low penetrations is the least costly and is used to reduce net demand variability and probability of lost load due to suppressed net demand. However, the overall curtailment rate does not decrease after \$25/tCO$_2$.

Extreme Curtailment Events: Example of 2017-03-18

Figure 6.8 indicates that at higher CP, curtailment volumes are dominated by a few extreme events, where forecast wind generation is > 1 GW. Two of the three largest curtailment events with CP =\$50/tCO$_2$ occur at the end of the test problem 2017-03-18, visualised in Figure 6.9. Curtailment is used over two periods with forecast wind generation of 1.5 GW and 1.9 GW in order to dampen the large drop in net demand that occurs at the end of the day. A counter-factual analysis was conducted for this day, comparing the operating costs with and without the curtailment action. Without the curtailment action during these periods, mean operating costs for this schedule would be \$5,108,605, compared to \$4,424,294 when curtailment is used due to the lower LOLP. This shows economic benefits to using curtailment thanks to improved system security.

In summary, Guided IDA* exhibits significant changes in operating patterns as the carbon price CP is increased. Fuel types play different roles in satisfying demand,

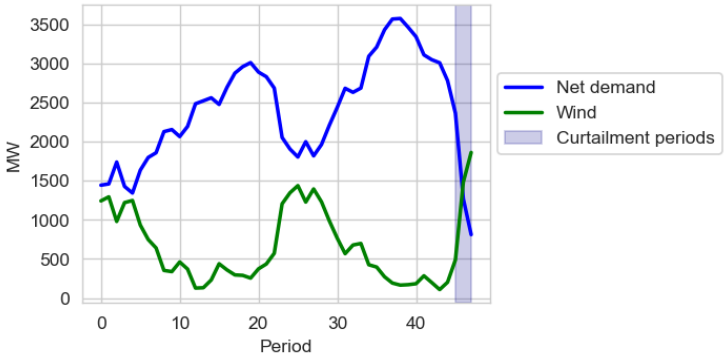


Figure 6.9: Curtailment action used by Guided IDA* in test problem 2017-03-18, with CP =\$50/tCO_2. Curtailment is used over 2 periods to mitigate the rapid decrease in net demand resulting from a spike in wind generation at the end of the day. Without the curtailment action during these periods, total operating costs for this schedule would be \$5108605, compared to \$4424294 when curtailment is used.

with gas gradually transitioning to full base-load operation when CP=\$50/tCO_2. The curtailment action is used less frequently at higher CP, but is still used sparingly to manage uncertainties and improve system security.

6.3 Case II: Generator Outages

In the second case study we introduced generator outages to the simulation environment described in Section 4.2 in order to study the ability of Guided IDA* to learn robust strategies to handle generation losses. Protection against generator outages is typically achieved by including a criterion that is based on the *single largest loss of infeed* [275], commonly known as the $N-1$ criterion. The $N-1$ criterion generally offers acceptable levels of system security but may be insufficient to prevent lost load in cases of multiple coincident outages, which was the cause of several blackouts across North America and European power systems in 2003 [276] and the 9 August 2019 blackout in the GB power system [59]. Similar $N-x$ approaches can be applied to protect against x simultaneous outages, but may be overly conservative or encounter constraints around minimum operating levels of generators which requires the curtailment of variable renewable energy. Probabilistic approaches have been proposed to allocate reserve constraints in deterministic UC [20, 277], but heuristic approaches such as $N-1$ are still widely-used [59, 278].

This case study investigates the effectiveness of Guided IDA* for developing UC solutions that are robust to generator outages, a critical factor in power system stability. We begin by describing the simulation environment including modelling of generator outages and formulate the problem as an MDP. We then apply Guided IDA* and MILP to solve unseen test problems with outages.

6.3.1 Environment Setup

In order to investigate the impact of generator outages, we modified the simulation environment described in Section 4.2. As with previous simulation environments, the environment with outages can be activated in the Python package developed for this research[2]. Each generator has a probability $\psi_i(u_{i,t-1})$ of failing when it transitioning from time $t-1$ to t, which is a function of the generator's up time $u_{i,t-1}$ at the previous timestep. Every generator begins the the episode available, and once it has failed it cannot be dispatched for the remainder of the episode. Furthermore, generators cannot fail during the first period of commitment (that is, when $u_{i,t-1} < 0$).

We use the Weibull distribution to model the failure rates of generators. The Weibull distribution is widely used for reliability analysis to represent failure rates [279–281] due to its generality, accommodating increasing, decreasing, and non-monotone failure functions. It is therefore well-suited to modelling failures of a heterogeneous generation mix. The two-parameter Weibull distribution is defined by a shape parameter k and scale parameter λ. Failure rates are scaled to a realistic order of magnitude by setting $\lambda = 100$ for all generators, leaving a single parameter k which is unique for each generator. The k parameters were assigned for generators to ensure a suitable level of heterogeneity and to achieve a weighted-equivalent forced outage rate (WEFOR) that approximately corresponds to real-world power systems. WEFOR measures the long-run average proportion of total generation capacity that is unavailable due to forced outages. The 5-year average WEFOR for North America, reported by the North American Electric Reliability Corporation, was 7.16% between 2015–2019 [282]. The outage rates as a $\psi_{i,t}$ function of generator up time $u_{i,t-1} > 0$ for the 10 generators described in Section 4.2.1 are shown in Figure 6.10.

We calculated the WEFOR by simulating outages over 1 year (365 episodes), where all generators are committed in every period. The WEFOR as a function of settlement period is shown in Figure 6.11. Since generators cannot be repaired and must remain offline for the remainder of the episode after an outage, the WEFOR increases linearly throughout the day. In the final period of the day the WEFOR is 9.67%. The average WEFOR over all periods was 4.68% indicated by the dotted line in Figure 6.11. While this is lower than 7.16% reported by NERC [282], the comparatively large WEFOR in later periods indicates the potential for extreme outage scenarios under our modelling approach, posing a significant challenge for managing system security in the context of UC.

In the following section, we formalise the environment changes by modifying the UC MDP from Section 4.3.

[2] `https://github.com/pwdemars/rl4uc`

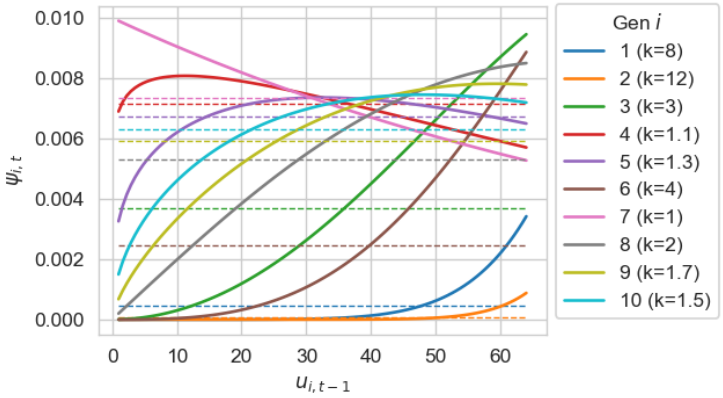


Figure 6.10: Outage rates ψ_i as function of up time $u_{i,t-1}$. Outage rates are modelled with a Weibull with fixed scale parameter $\lambda = 100$ and variable shape parameter k. The dotted lines show the mean outage rate over all periods. The two base-load generators 1 and 2 have the lowest forced outage rates. Most generators have generally increasing forced outage rates, while some have higher failure rates at the beginning of operation.

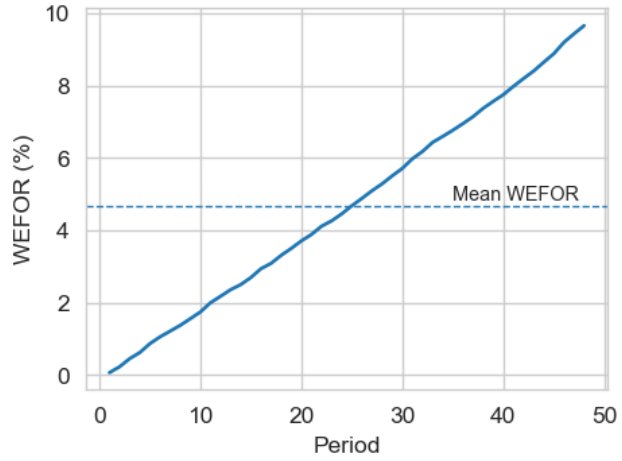


Figure 6.11: Weighted equivalent forced outage rate (WEFOR) as a function of decision period. Later periods have higher WEFOR,

6.3.2 MDP Formulation

To formulate the UC MDP for the power system environment with outages, we modified the state and transition function definitions in the UC MDP from Section 4.3. Actions and rewards remain unchanged.

State Definition

The state definition s_t is modified to include an outages vector $\boldsymbol{z}_t \in \{0,1\}^N$, indicating which generators experience an outage at time t. We then include an additional availability vector $\boldsymbol{v}_t \in \{0,1\}^N$, indicating which generators have experienced an outage so far in the episode up to and including timestep t. Generators which are unavailable $v_{i,t} = 0$ cannot be dispatched. Like the forecast errors, neither $\boldsymbol{z}_t$ nor $\boldsymbol{v}_t$ are observed by the agent in the day-ahead setting.

Transition Definition

The transition function $F(s_{t+1}, s_t, a_t)$ is updated in two steps. For each online generator, we sample an outage $z_{i,t}$ for each generator i, independently of previous periods or the current generator status:

$$z_{i,t} \sim \text{Bern}(\psi_i(u_{i,t-1})) \tag{6.11}$$

Second, the generator availability $v_{i,t}$ is updated:

$$v_{t,i} = \begin{cases} 0, & \text{if } v_{t-1,i} = 0 \\ 0, & \text{if } v_{t-1,i} = 1 \text{ and } z_{i,t} = 1 \\ 1, & \text{if } v_{t-1,i} = 1 \text{ and } z_{i,t} = 0 \end{cases} \tag{6.12}$$

When $v_{t,i} = 0$, the generator cannot be dispatched. In addition, it remains unavailable for the rest of the episode. The remaining components of the UC MDP are identical to those described in Section 4.3. In the following subsection, we will describe how the UC MDP with outages can be represented as a search tree in order to apply tree search methods.

6.3.3 Search Tree Formulation

In Section 4.4.1 we described how the original UC MDP can be formulated as a search tree, with nodes representing states and edges representing actions. The step costs in the search tree are evaluated using a Monte Carlo approach described in Equation 4.11, calculating mean operating costs over N_s scenarios of net demand (demand minus wind generation). While the UC MDP with curtailment and carbon price could be represented as a search tree using precisely this method, the search tree representation for the outages case study must be modified to include outage scenarios.

The general approach to calculating the expected step cost $C(s)$ of transitioning to state s remains the same. We generate N_s scenarios for the uncertain processes (net demand and outages), calculate the dispatch costs under each scenario and take the mean. However, we now must account for additional uncertain parameters corresponding to outages $\boldsymbol{z}$ and availability $\boldsymbol{v}$.

Step costs are calculated using the following equation:

$$C(s) = \frac{-1}{N_s} \sum_{k=1}^{N_s} R(s, x_k, \boldsymbol{z}_k(\boldsymbol{u}_{t-1}), \boldsymbol{v}_{t-1,k}) \tag{6.13}$$

$$x_k = \mathcal{S}_{t,k} \quad \boldsymbol{z}_k(\boldsymbol{u}) = \mathcal{Z}_{*,\boldsymbol{u},k} \tag{6.14}$$

The reward function $R(\cdot)$ represents the reward function evaluated for state s, net demand x_k scenario, outage scenario $\boldsymbol{z}_k(\boldsymbol{u}_{t-1})$ and previous generator availabilities $\boldsymbol{v}_{t-1,k}$. $\mathcal{S} \in \mathbb{R}^{T \times N_s}$ represents the net demand scenarios, as described in Section 4.4.1. $\mathcal{Z} \in \{0,1\}^{N \times u^{\max} \times N_s}$ represents the outage scenarios.

Generator outage scenarios are generated offline at the beginning of each episode and stored in a lookup table in the same way as net demand scenarios. This ensures that outages are fairly distributed across nodes. This is particularly important for the outages case study, as the extreme nature of outages and the large number of random parameters ($N + 2$ including the wind and demand forecast errors) means that the distribution of outages over scenarios is likely to be highly variable. Whereas demand and wind forecast error distributions were a function of the timestep t, the outage probability distributions are functions of generator up time $u_{i,t-1}$. Hence, for each generator we create a lookup table with N_s rows and $\max(T + u_{0,i}, T)$ columns, where u_0, i is the initial up/down time of generator i (this is the maximum up time of generator i in a given episode). Each column is populated with $\{0, 1\}$ sampled from the Bernoulli outage distribution in Equation 6.11. The set of outage scenarios $\mathcal{Z}$ is the stack of all lookup tables for outage scenarios.

Equation 6.14 calculates the empirical mean of operating costs over N_s scenarios for a node at time t. For scenario $k \in \{1..N_s\}$, we use net demand scenario k at timestep t. Then, for each generator, we use outage scenario k at generator up/down time $u_{t-1,i}$, the up/down time of generator i at the previous timestep. This scenario is retrieved from row k, column $u_{t-1,i}$ of the lookup table for generator i. The operating costs for this joint scenario of net demand and outages are then calculated. The expected costs are calculated by repeating this process for each of the N_s scenarios and calculating the mean operating costs.

Having formalised the environment and MDP modifications and the search tree formulation, in the following section we will describe our experiments training and applying Guided IDA* to solve the UC problem instances.

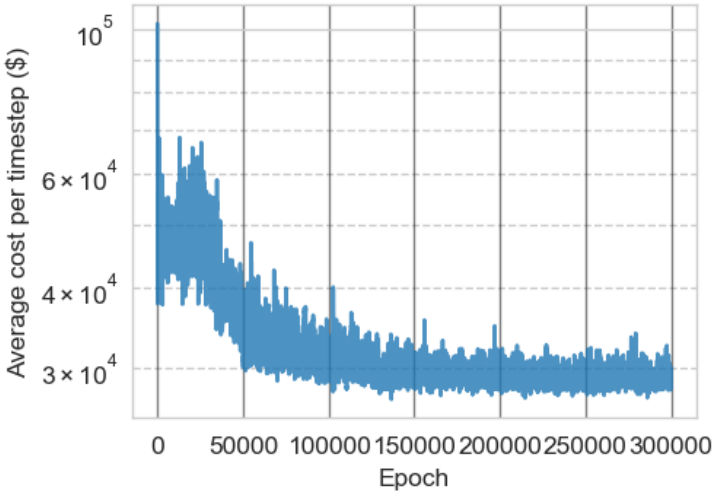


Figure 6.12: Convergence of expansion policy for the 30 generator environment with generator outages. The figure shows a moving average of operating costs per timestep over 100 epochs. The policy was trained by PPO using the method described in Section 4.5.3.

6.3.4 Experimental Setup and Policy Training

We will now describe the setup of experiments applying Guided IDA* and MILP with $N-x$ reserve criteria to solve unseen test problems. As with the curtailment experiments, in the generator outages problem we exclusively used the 30 generator system. We used the same split of training and test episodes as in previous experiments, holding out the 20 test episodes described in Table 4.2 from the training data.

In order to apply Guided IDA*, we trained a new expansion policy by model-free RL with PPO, using the method described in Section 4.5.3. We used the same parameters used to train expansion policies in Chapter 4, described in Table 4.5. Convergence of the expansion policy for the outages environment is shown in Figure 6.12. Compared with the expansion policies for the original MDP (Figure 4.11) and with curtailment and carbon prices (Figure 6.4), convergence is significantly slower, caused by the noisier reward function with respect to observations. Convergence speed may also be impacted by differences in the reward distribution, caused by greater frequency of lost load events due to generator outages.

Guided IDA* was then used to solve the 20 unseen UC problem instances. To evaluate the schedules, we simulated the dispatch of 1000 scenarios as in previous experiments. Scenarios include random realisations of demand, wind generation and outages. We set the branching threshold $\rho = 0.05$, the time budget $b = 60$s and used the Constrained ED heuristic described in Section 5.3.2. In Section 6.3.3 we discussed the method of calculating expected step costs in the search tree considering scenarios of outages and net demand. In previous experiments which considered only

Method	Total cost ($M)	Std. cost ($M)	Fuel ($M)	Lost load ($M)	Startups ($M)	LOLP (%)
Guided IDA*($N_s = 100$)	28.69	0.86	28.22	0.32	0.15	0.08
Guided IDA*($N_s = 200$)	28.69	1.71	27.99	0.56	0.14	0.13
Guided IDA*($N_s = 500$)	28.63	0.95	28.13	0.35	0.15	0.09
Guided IDA*($N_s = 1000$)	28.58	0.91	28.11	0.33	0.15	0.09
Guided IDA*($N_s = 2000$)	28.71	0.92	28.17	0.40	0.15	0.09
Guided IDA*($N_s = 5000$)	28.84	0.86	28.23	0.45	0.15	0.10
MILP($N-1$)	43.83	15.67	26.70	17.00	0.13	2.62
MILP($N-2$)	29.20	2.94	27.65	1.39	0.16	0.26
MILP($N-3$)	29.74	1.15	28.82	0.74	0.18	0.11
MILP($N-4$)	31.32	1.35	30.12	1.02	0.17	0.16

Table 6.3: Comparison of Guided IDA* and MILP($N-x$) solutions in the generator outages case study.

demand and wind generation uncertainty, the number of scenarios N_s was fixed at $N_s = 100$. This sufficiently approximated the distribution of net demand. In the outages experiments, motivated by the greater number of random parameters, we ran Guided IDA* with $N_s = \{100, 200, 500, 1000, 2000, 5000\}$. Higher N_s allows for the expected step costs in the search tree to be more accurately estimated but increases the overall run time of each node evaluation, thus decreasing the search depth for fixed time budget b.

Guided IDA* was compared with MILP($N-x$) benchmarks which use $N-x$ reserve criteria. $N-1$ is a widely used reserve criterion that assures that enough capacity is available to protect against an outage of the largest generator. We implement $N-x$ criteria up to $x = 4$. For the 30 generator problem, this corresponds to reserve margins of $\{455, 910, 1365, 1820\}$ MW for $x = \{1, 2, 3, 4\}$, respectively.

6.3.5 Results

The operating costs and loss of load probability of Guided IDA* and MILP($N-x$) are compared in Table 6.3. Operating costs are disaggregated into fuel costs, lost load costs and startup costs.

MILP Benchmarks

Of the MILP benchmarks, MILP($N-2$) is found to achieve the lowest total operating costs. Fuel costs increase consistently with the reserve size as generators are forced to operate at lower efficiencies on average to satisfy the increased reserve requirements. Lost load costs decrease up to $N-3$, but increase thereafter due to increased probability of encountering minimum generation constraints (events of insufficient footroom to manage high wind generation outturn and/or low demand) with large reserve margins. The worst performing MILP benchmarks are those with the smallest and largest reserve margins: MILP($N-1$) has very high LOLP (2.62%) and thus the highest lost load costs; MILP($N-4$) is the most conservative solution and thus has the highest fuel costs.

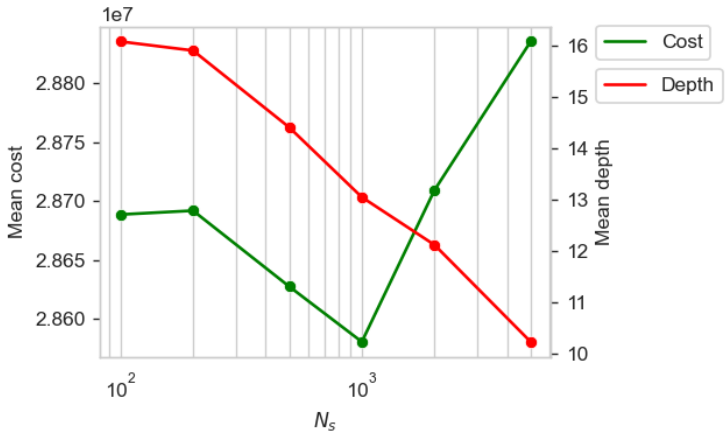


Figure 6.13: Mean total operating costs and search depth of Guided IDA* solutions with varying N_s. Search depth decreases with increasing N_s due to the increased node evaluation time. The lowest operating costs are achieved with $N_s = 1000$. At higher values of N_s, total operating costs increase due to shallower search depth.

Impact of N_s on Operating Costs and Search Depth

All Guided IDA* versions achieve lower operating costs than MILP$(N-2)$, the best performing MILP benchmark. Figure 6.13 shows the change in operating cost and search depth with varying N_s. Search depth decreases monotonically with increasing N_s, due to the greater node evaluation time. The figure shows that for $N_s > 1000$, improvements stemming from more accurate estimates of expected step cost begin to be outweighed by the detrimental impact of shallower search depth. The highest settings of $N_s = \{2000, 5000\}$ had the greatest total operating costs. Guided IDA*$(N_s = 1000)$ achieves the lowest total operating costs, which are 2.1% lower than MILP$(N-2)$ and 0.4% lower than Guided IDA*$(N_s = 100)$. These cost savings are larger than those found in experiments without outages, reported in Section 5.4.2, where Guided IDA* achieved operating costs that were 1.4% lower than MILP(4σ). Our results show comparable savings to those reported in the stochastic UC literature reviewed in Section 2.3.5 which found operating cost savings of approximately 1% compared with deterministic methods [18, 22, 161].

Reserve Margins

Guided IDA* adaptively allocates reserve capacity to achieve low levels of LOLP without producing overly conservative schedules. The average reserve margins of Guided IDA*$(N_s = 1000)$ for each settlement period are compared with the MILP benchmarks in Figure 6.14. The reserve margins of all methods follow a similar daily pattern, with the lowest reserve margins coinciding with morning and evening demand peaks. However, Guided IDA* increases its reserve commitment relative to the other methods throughout the day, beginning with similar margins to $N-1$, and

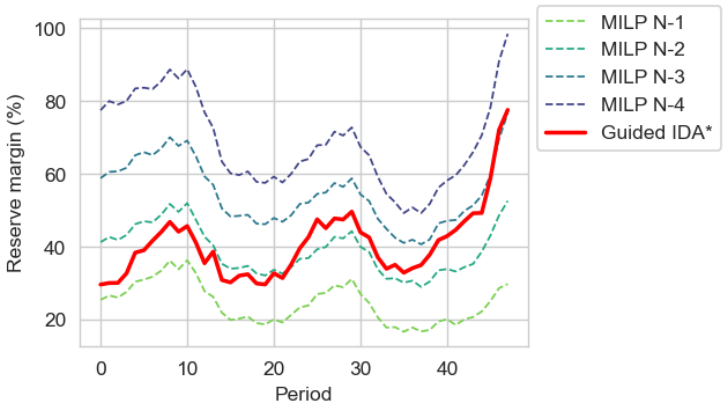


Figure 6.14: Average reserve margins by period for MILP with $N-x$ reserve criteria, and Guided IDA*($N_s = 1000$). The figure shows average reserve as a proportion of net demand. Guided IDA* begins the day with reserve margins similar to MILP($N-1$), and finishes with reserve margins close to MILP($N-3$).

finishing with roughly $N-3$ reserve margins. This reflects the greater probability of multiple coincident outages at the end of the scheduling period.

Allocating constant reserve margins, the approach taken by MILP($N-x$) benchmarks, results in relatively poor performance for this problem setup. Our results show that Guided IDA* learns an adaptable reserve commitment that reflects properties of the environment.

Overall, Guided IDA* outperforms the MILP($N-x$) benchmarks for all settings of N_s, and allocates reserves in a more efficient way to protect against generator outages. Solution quality improved with increasing number of scenarios N_s up to $N_s = 1000$, but further increases had a detrimental impact on operating costs due to the shallower search depth.

6.4 Discussion

Guided tree search is a highly adaptive framework for solving the UC problem. This chapter showed that complex problem settings incorporating co-optimisation of curtailment, carbon pricing and generator outages, can be reflected in the simulation environment and solved with Guided IDA* without significant changes to the solution method beyond limited parameter tuning. Guided tree search and RL approaches are not limited to linear objective functions or constraints, and can easily incorporate large numbers of stochastic variables in the environment. By contrast, MILP approaches would require complex, manual reformulations to incorporate co-optimisation of curtailment. As discussed in Section 2.2.2, designing effective MILP formulations for the UC problem is an active research topic, and can have significant impact on run time and/or solution quality. In addition, the traditional MILP deterministic

approach of allocating reserve using $N-x$ criteria in the outages case was shown to be inadequate at trading off security of supply with expensive over-commitment of reserve. By contrast, Guided IDA* search is capable of handling stochastic decision making without explicit reserve constraints.

Curtailment and Carbon Price

Including the curtailment action in the first case study demonstrated the potential for Guided IDA* to incorporate more heterogeneous decision-making alongside UC. This approach could be extended to other operational tasks such as demand-side response instructions and charge/discharge of storage assets. However, the results of this experiment indicate that including the curtailment action added considerable complexity that made the problem harder to solve. Comparing the results in Table 6.2 for the $\$0/tCO_2$ case with those of experiments from the previous chapter (Table 5.3), overall operating costs were higher when the curtailment action was introduced, despite the reward function remaining the same. Improving the policy training component in this case study by implementing a different RL algorithm or policy network architecture could yield further cost reductions for the curtailment case.

The Guided IDA* solutions employed unexpected operational strategies at higher carbon prices, which were used to develop insights regarding the problem setup. The increase in reserve margins with carbon price was surprising, and could be partially explained by the lack of flexible peaking capability when gas is employed as base-load, which could be the cause of greater reserve margins. We found that coal startups remained low, indicating that it was not used flexibly for peaking, such as during the evening peak. Furthermore, the curtailment action was not phased out completely even at the highest carbon price. Most of the curtailment volume at higher carbon prices was concentrated in a few extreme actions, such as the instance shown in Figure 6.9, where curtailment is used to manage a sharp drop in demand and increase in wind generation. Such extreme net demand swings are difficult to manage securely and the curtailment action is a useful and economic option in this instance.

Outages

In the outages case study, MILP($N-x$) were shown to be outperformed by Guided IDA* which allocated reserve more dynamically. We experimented with varying the number of scenarios N_s used to calculate expected step costs as a response to the increased number of random parameters in the outages problem. Our results showed that increasing N_s to 1000 yielded operating cost reductions for the outages case study relative to lower values. Further increasing N_s resulted in worsening of solution quality due to the shallower search depth. Linear time complexity in N_s is a valuable property of Guided IDA*, allowing N_s to be increased without large reductions in search depth for the same computational budget; average search depth was 10.2 at $N_s = 5000$ compared to 16.1 with $N_s = 100$. By contrast, large

numbers of uncertain parameters pose problems for both scenario-based stochastic UC (reviewed in Section 2.3) and robust UC (reviewed in Section 2.4) approaches. For scenario-based methods, solutions to stochastic programs with large numbers of scenarios are typically very computationally expensive and must usually be reduced substantially by scenario reduction methods [166]. For robust optimisation methods, outages (which are integer-valued) cannot be represented in a polyhedral uncertainty set. As a result, existing robust UC research has either used $N-x$ approaches, where a suitable value for x must be assigned manually [189] or omitted outage-related security constraints completely [177]. By contrast, the sampling approach adopted in Guided IDA* is highly scalable in the number of scenarios N_s, allowing rare contingencies to be considered probabilistically, without heuristic reserve constraints.

Figure 6.12 showed slower convergence of the expansion policy in the outages case study as compared with previous problems, reflecting the greater reward uncertainty. Further increasing the number of uncertain parameters (such as by considering disaggregated loads or wind farms) would be likely to further increase the training time required for Guided IDA*. As with the curtailment problem, there is scope to improve the policy training component for more complex problem domains. A useful property of Guided IDA* is that policy training algorithms can be substituted in a modular fashion, allowing for new state-of-the-art RL methods to be exploited in future.

In both case studies, Guided IDA* uncovered properties of the problem through its use of unexpected or novel operational strategies. There is scope to use such insights in hybrid methodologies or as a decision support tool. The reserve margins learned by Guided IDA* in the outages case study could easily be used to inform reserve constraints for deterministic UC approaches. Furthermore, Guided IDA* could be used to identify periods of low system security. Our results from the curtailment experiment found that Guided IDA* solutions exposed periods where curtailment significantly improved system security.

6.5 Conclusion

In this chapter we applied Guided IDA* search in two case studies, demonstrating the flexibility of this approach in heterogeneous power system contexts. In the first case study, we introduced a curtailment action and carbon price, finding that Guided IDA* responded dynamically to increasing carbon price in terms of curtailment rates and utilisation of different fuel types. We found that carbon emissions decreased by 22% when increasing the carbon price from \$0–25/tCO_2, driven by lower curtailment rates and displacement of coal with gas and oil-fired power stations. In the second case study, we introduced generator outages and showed that Guided IDA* achieved

up to 2.1% lower total operating costs than MILP benchmarks with $N - x$ reserve criteria. Guided IDA* intelligently allocates reserves, reaching levels of loss of load probability that are similar to that achieved in the case with no generator outages.

Our experiments show that guided tree search is a highly adaptive framework for solving UC problems in heterogeneous power systems. Other power systems decisions such as curtailment can be included in the simulation environment and co-optimised alongside UC and the reward function can be shaped to reflect societal relative values of security of supply and environmental considerations using a carbon price. Furthermore, the UC solutions produced by Guided IDA* exhibited unexpected properties that improved our understanding of the problem itself.

Chapter 7

Conclusion

7.1 Summary

This thesis has shown that RL is a viable methodology for solving the UC problem, offering solution quality that is competitive with and often outperforms the state-of-the-art in deterministic mathematical programming methods. Novel guided tree search methods were used to solve UC problem instances with greater numbers of generators and more complex stochastic processes than have been addressed in the existing literature.

In Chapter 4 we developed guided tree search, a framework for applying RL to the UC problem using model-based planning and model-free RL methods, which is the methodological base for this thesis. In addition, we developed a novel benchmark simulation environment based on data from the GB power system, enabling the application of RL to solve the UC problem. Using guided tree search, the search space of generator commitments is intelligently reduced to a subset of promising actions, enabling the application of conventional tree search methods in practical computing times. We applied this methodology to uniform-cost search (UCS) for systems of up to 30 generators. Our results showed that Guided UCS achieved lower operating costs than deterministic UC approaches using MILP. Furthermore, we showed that the run time of Guided UCS remained roughly constant in the number of generators, with negligible impact of operating costs as compared with UCS without RL, despite the reduced branching factor of the search tree. In addition, the security of supply was significantly improved, with reserve commitments learned *tabula rasa*. These experiments are the first to compare RL with MILP, the current state-of-the-art, and demonstrate the competitiveness of our approach in solving practical UC problems under uncertainty.

In Chapter 5 we explored using more advanced search methods to address the variability in run time across UC problem instances and achieve greater search efficiency by employing domain-specific knowledge. A heuristic based on priority list solution methods [1] was employed in the informed search method Guided A* search, and found to reduce run times by up to 94% as compared with Guided UCS with

negligible impact on operating costs. This large efficiency improvement confirms the potential value of incorporating domain knowledge in RL algorithms to achieve tractability for challenging real-world problems [30, 45]. Guided IDA* employed the anytime strategy iterative deepening to tackle run time variability, enabling solution quality to be maximised within a fixed time budget. This enabled the application of Guided IDA* to a 100-generator problem, the largest in the existing literature. The only other study of comparable size used a simplified problem setup that eliminated intra-day commitment changes to achieve tractability for a problem of 99 generators, and cannot be directly compared with our experiments [56]. Our results show that guided tree search is competitive with MILP approaches for problems of this size, achieving similar (0.1% lower) operating costs over 20 UC problem instances. In comparison with existing literature, this represents a significant step forward in the application of RL to practical UC problem instances at scale. In experiments of up to 30 generators, we found cost reductions of approximately 1%, comparable to the improvements of the deterministic MILP methods currently used in practical contexts over Lagrangian relaxation [64], which were found to yield large absolute cost savings following their adoption [9]. Stochastic optimisation methods have also been shown to achieve similar improvements in operating costs of around 1% [13, 18, 22] but are impractically expensive to run [17]. By off-loading most of the computational expense to training, RL offers the potential to substantially improve UC solution quality within practical computing times.

In Chapter 6 we adapted the simulation environment to incorporate advanced challenges in UC, introducing a carbon price, wind curtailment and generator outages. Our results showed that guided tree search methods can be applied to UC problem variants without reformulating the solution method. This allows for the application of RL to UC problems of arbitrary complexity, including stochastic environments with large numbers of uncertain parameters. In the generator outages case study, we showed that RL provides lower operating operating costs and higher levels of system security than MILP methods using $N-x$ reserve criteria. While there are concerns of trust surrounding safe RL [283] and the application of RL in critical infrastructure such as power systems, employing model-based methods guided tree search enables forward planning through consultation of a model, improving robustness and explainability [45]. In addition, our results showed that modifications to the reward function through a carbon price can incentivise operating patterns with lower carbon emissions. Using RL, decision-makers are able to adjust scheduling behaviour in line with current system priorities. In general, a qualitative analysis of RL solutions can be used to analyse properties of UC problems under consideration. The use of curtailment was indicative of periods of high uncertainty and possibility of load shedding; and the reserve margins employed by RL in the outages case study indicated robust commitments where heuristic methods were inadequate. The ability of RL to learn

fundamentals of the problem domain can add value in decision support contexts where AI methods are used to inform human operators.

Machine learning methods have long been recognised as having significant potential to profoundly change and improve the operation of power systems [30, 284, 285]. The results of this thesis have shown that RL is capable of learning optimal control strategies in challenging power system contexts. As electricity networks become increasingly complex and uncertain, the optimality gap of traditional deterministic methods is likely to grow and the value of AI-augmented decision-making will further increase. This thesis has shown that RL is a competitive methodology for UC with the ability to bridge this gap and provide substantial cost reductions and practical benefits in the operation of future power systems.

7.2 Limitations and Further Work

This thesis has shown that RL can be applied in combination with tree search to solve the UC problem and outperform traditional mathematical optimisation approaches. However, there remain further opportunities to improve and extend this approach and address the limitations of our research.

Comparisons with a broader array of mathematical programming methods would be beneficial for further understanding the limitations and advantages of RL-based approaches relative to other optimisation methods. The performance improvement of guided tree search as compared with the deterministic MILP formulations used in this thesis was due in large part to superior management of uncertainties, evidenced by lower levels of load shedding. Stochastic formulations, although relatively expensive to solve, may produce superior solution quality to guided tree search by capturing uncertainty across multiple scenarios. Furthermore, by sampling directly from the environment, RL and guided tree search do not require the objective function or constraints to be linear, as in linear programming methods. The advantages of this may be significant in contexts where the objective function cannot be well-approximated with piecewise linear functions. Our problem setup and open-source environment described in Section 4.2 provides a valuable test-bed for further comparisons of UC solution methods.

We found no deterioration of solution quality in scaling from 5–30 generators relative to MILP. However, the margins of improvement in the 100-generator case were markedly smaller: 0.1% lower operating costs as compared with c. 1% in smaller problem instances. Target entropy regularisation was effective in unifying policy training and guided tree search and improving training convergence, but practical challenges remain in order to stabilise performance on large problems. Further improvements to Guided IDA* and applications to larger power systems of greater than 100 generators could be achieved by a more optimised implementation

of this algorithm and greater computing resources. Using the anytime algorithm Guided IDA* ensures that run times do not become impractical for larger systems, but further experiments are required to assess the impact of increasing numbers of generators on solution quality. Since the guided tree search framework is modular in both the RL algorithm and the search algorithm, there is also potential for further variations which exploit new RL and planning methods which could also improve solution quality. A more efficient implementation of Guided IDA* would also help address the challenge of comparing computational requirements of guided tree search and MILP approaches, discussed in Section 4.6.3.

This thesis focused exclusively on the day-ahead UC problem as the most widely-studied of UC problem variants. There is growing interest in UC for intraday markets, whose prominence as a share of total power traded is increasing [286]. The real-time algorithm presented in Algorithm 5 that is used to implement all of the guided tree search approaches for the UC problem is well-suited to scheduling tasks with a rolling decision horizon, such as in intraday settings or system balancing. Intraday scheduling has also been shown to improve the overall power system efficiency and security through rolling horizon stochastic optimisation [18]. Compared with MILP approaches, guided tree search would exhibit more apparent run-time advantages in these contexts.

There are several avenues for improvement to the environment presented in this thesis. Adherence with the OpenAI Gym API [287] could broaden access to the wider RL research community and enable straightforward benchmarking of state-of-the-art RL algorithms. As demonstrated in Chapter 6, RL is well-suited to solving UC problem variants through modification of the environment. Further extensions could accelerate research into pertinent topics in power systems operation, including the introduction of storage assets, transmission constraints, distributed renewables generation and demand-side response.

Bibliography

[1] Tomonobu Senjyu, Kai Shimabukuro, Katsumi Uezato, and Toshihisa Funabashi. A fast technique for unit commitment problem by extended priority list. *IEEE Transactions on Power Systems*, 18(2):882–888, 2003.

[2] Allen J Wood, Bruce F Wollenberg, and Gerald B Sheblé. *Power generation, operation, and control*. John Wiley & Sons, 2013.

[3] National Grid Demand Data. `https://www.nationalgrideso.com/data-explorer`.

[4] Balancing Mechanism Reporting Service. `https://www.bmreports.com`.

[5] Spyros A Kazarlis, AG Bakirtzis, and Vassilios Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE transactions on power systems*, 11(1):83–92, 1996.

[6] Steven J Davis, Nathan S Lewis, Matthew Shaner, Sonia Aggarwal, Doug Arent, Inês L Azevedo, Sally M Benson, Thomas Bradley, Jack Brouwer, Yet-Ming Chiang, et al. Net-zero emissions energy systems. *Science*, 360(6396):eaas9793, 2018.

[7] Gunnar Luderer, Silvia Madeddu, Leon Merfort, Falko Ueckerdt, Michaja Pehl, Robert Pietzcker, Marianna Rottoli, Felix Schreyer, Nico Bauer, Lavinia Baumstark, et al. Impact of declining renewable energy costs on electrification in low-emission scenarios. *Nature Energy*, 7(1):32–42, 2022.

[8] James H Williams, Andrew DeBenedictis, Rebecca Ghanadan, Amber Mahone, Jack Moore, William R Morrow, Snuller Price, and Margaret S Torn. The technology path to deep greenhouse gas emissions cuts by 2050: the pivotal role of electricity. *science*, 335(6064):53–59, 2012.

[9] RP O'Neill. Computational issues in iso market models. In *workshop on energy systems and optimization*, 2017.

[10] Pascale Bendotti, Pierre Fouilhoux, Cécile Rottner, et al. On the complexity of the unit commitment problem. *Ann. Oper. Res.*, 274(1-2):119–130, 2019.

[11] Bernard Knueven, James Ostrowski, and Jean-Paul Watson. On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing*, 32(4):857–876, 2020.

[12] Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205, 2004.

[13] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems*, 28(1):52–63, 2012.

[14] Yonghong Chen, Aaron Casto, Fengyu Wang, Qianfan Wang, Xing Wang, and Jie Wan. Improving large scale day-ahead security constrained unit commitment performance. *IEEE Transactions on Power Systems*, 31(6):4732–4743, 2016.

[15] Juha Kiviluoma, Mark O'Malley, Aidan Tuohy, Peter Meibom, Michael Milligan, Bernhard Lange, Hannele Holttinen, and Madeleine Gibescu. Impact of wind power on the unit commitment, operating reserves, and market design. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE, 2011.

[16] Hannele Holttinen, Peter Meibom, Antje Orths, Bernhard Lange, Mark O'Malley, John Olav Tande, Ana Estanqueiro, Emilio Gomez, Lennart Söder, Goran Strbac, et al. Impacts of large amounts of wind power on design and operation of power systems, results of iea collaboration. *Wind Energy*, 14(2):179–192, 2011.

[17] Anthony Papavasiliou, Shmuel S Oren, and Barry Rountree. Applying high performance computing to transmission-constrained stochastic unit commitment for renewable energy integration. *IEEE Transactions on Power Systems*, 30(3):1109–1120, 2014.

[18] Aidan Tuohy, Peter Meibom, Eleanor Denny, and Mark O'Malley. Unit commitment for systems with significant wind penetration. *IEEE Transactions on power systems*, 24(2):592–601, 2009.

[19] Rüdiger Barth, Heike Brand, Peter Meibom, and Christoph Weber. A stochastic unit-commitment model for the evaluation of the impacts of integration of large amounts of intermittent wind power. In *2006 International Conference on Probabilistic Methods Applied to Power Systems*, pages 1–8. IEEE, 2006.

[20] Francois Bouffard and Francisco D Galiana. Stochastic security for operations planning with significant wind power generation. In *2008 IEEE Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century*, pages 1–11. IEEE, 2008.

[21] Farrokh Aminifar, Mahmud Fotuhi-Firuzabad, and Mohammad Shahidehpour. Unit commitment with probabilistic spinning reserve and interruptible load considerations. *IEEE Transactions on Power Systems*, 24(1):388–397, 2009.

[22] Pablo A Ruiz, C Russ Philbrick, Eugene Zak, Kwok W Cheung, and Peter W Sauer. Uncertainty management in the unit commitment problem. *IEEE Transactions on Power Systems*, 24(2):642–651, 2009.

[23] Zihang Dai, Hanxiao Liu, Quoc Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in Neural Information Processing Systems*, 34, 2021.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[25] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[27] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[28] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[29] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[30] Mevludin Glavic, Raphaël Fonteneau, and Damien Ernst. Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, 50(1):6918–6927, 2017.

[31] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *arXiv preprint arXiv:1906.05433*, 2019.

[32] ATD Perera and Parameswaran Kamalaruban. Applications of reinforcement learning in energy systems. *Renewable and Sustainable Energy Reviews*, 137:110618, 2021.

[33] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[34] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[35] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[36] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[37] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[38] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[39] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[40] EA Jasmin and Imthias Ahamed TP. Reinforcement learning solution for unit commitment problem through pursuit method. In *2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pages 324–327. IEEE, 2009.

[41] EA Jasmin, TP Imthias Ahamed, and T Remani. A function approximation approach to reinforcement learning for solving unit commitment problem with photo voltaic sources. In *2016 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, pages 1–6. IEEE, 2016.

[42] Nandan Kumar Navin and Rajneesh Sharma. A fuzzy reinforcement learning approach to thermal unit commitment problem. *Neural Computing and Applications*, 31(3):737–750, 2019.

[43] Fangyuan Li, Jiahu Qin, and Wei Xing Zheng. Distributed q-learning-based online optimization algorithm for unit commitment and dispatch in smart grid. *IEEE transactions on cybernetics*, 50(9):4146–4156, 2019.

[44] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[45] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[46] Giuseppe Pinto, Marco Savino Piscitelli, José Ramón Vázquez-Canteli, Zoltán Nagy, and Alfonso Capozzoli. Coordinated energy management for a cluster of buildings through deep reinforcement learning. *Energy*, 229:120725, 2021.

[47] José R Vázquez-Canteli, Jérôme Kämpf, Gregor Henze, and Zoltan Nagy. Citylearn v1. 0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 356–357, 2019.

[48] Deunsol Yoon, Sunghoon Hong, Byung-Jun Lee, and Kee-Eung Kim. Winning the l2rpn challenge: Power grid management via semi-markov afterstate actor-critic. In *International Conference on Learning Representations*, 2020.

[49] Antoine Marot, Benjamin Donnot, Camilo Romero, Balthazar Donon, Marvin Lerousseau, Luca Veyrin-Forrer, and Isabelle Guyon. Learning to run a power network challenge for training topology controllers. *Electric Power Systems Research*, 189:106635, 2020.

[50] Anne Sjoerd Brouwer, Machteld van den Broek, Ad Seebregts, and André Faaij. Operational flexibility and economics of power plants in future low-carbon power systems. *Applied Energy*, 156:107–128, 2015.

[51] Daniel J Burke and Mark J O'Malley. Factors influencing wind energy curtailment. *IEEE Transactions on Sustainable Energy*, 2(2):185–193, 2011.

[52] Sinnott Murphy, Jay Apt, John Moura, and Fallaw Sowell. Resource adequacy risks to the bulk power system in north america. *Applied energy*, 212:1360–1376, 2018.

[53] 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 *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[54] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *arXiv preprint arXiv:1705.08439*, 2017.

[55] Gal Dalal and Shie Mannor. Reinforcement learning for the unit commitment problem. In *2015 IEEE Eindhoven PowerTech*, pages 1–6. IEEE, 2015.

[56] Gal Dalal, Elad Gilboa, and Shie Mannor. Hierarchical decision making in electricity grid management. In *International Conference on Machine Learning*, pages 2197–2206. PMLR, 2016.

[57] David Klenert, Linus Mattauch, Emmanuel Combet, Ottmar Edenhofer, Cameron Hepburn, Ryan Rafaty, and Nicholas Stern. Making carbon pricing work for citizens. *Nature Climate Change*, 8(8):669–677, 2018.

[58] JR Minkel. The 2003 northeast blackout–five years later. *Scientific American*, 13, 2008.

[59] Janusz Bialek. What does the gb power outage on 9 august 2019 tell us about the current state of decarbonised power systems? *Energy Policy*, 146:111821, 2020.

[60] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[61] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition, 2009.

[62] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[63] Miguel F Anjos and Antonio J Conejo. *Unit commitment in electric energy systems*. Now Foundations and Trends, 2017.

[64] Miguel Carrión and José M Arroyo. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on power systems*, 21(3):1371–1378, 2006.

[65] Samer Takriti, John R Birge, and Erik Long. A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems*, 11(3):1497–1508, 1996.

[66] RC Johnson, HH Happ, and WJ Wright. Large scale hydro-thermal unit commitment-method and results. *IEEE Transactions on Power Apparatus and Systems*, (3):1373–1384, 1971.

[67] Arthur I Cohen and Miki Yoshimura. A branch-and-bound algorithm for unit commitment. *IEEE Transactions on Power Apparatus and Systems*, (2):444–451, 1983.

[68] André Merlin and Patrick Sandrin. A new method for unit commitment at electricite de france. *IEEE transactions on power apparatus and systems*, (5):1218–1225, 1983.

[69] Walter L Snyder, H David Powell, and John C Rayburn. Dynamic programming approach to unit commitment. *IEEE Transactions on Power Systems*, 2(2):339–348, 1987.

[70] Andrew L Ott. Evolution of computing requirements in the pjm market: Past and future. In *IEEE PES General Meeting*, pages 1–4. IEEE, 2010.

[71] Germán Morales-España, Álvaro Lorca, and Mathijs M de Weerdt. Robust unit commitment with dispatchable wind power. *Electric Power Systems Research*, 155:58–66, 2018.

[72] Lei Wu. A tighter piecewise linear approximation of quadratic cost curves for unit commitment problems. *IEEE Transactions on Power Systems*, 26(4):2581–2583, 2011.

[73] Alberto Borghetti, Claudia D'Ambrosio, Andrea Lodi, and Silvano Martello. An milp approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *IEEE Transactions on power systems*, 23(3):1115–1124, 2008.

[74] Nima Amjady and Mohammad Reza Ansari. Hydrothermal unit commitment with ac constraints by a new solution method based on benders decomposition. *Energy Conversion and Management*, 65:57–65, 2013.

[75] Hrvoje Pandžić, Ting Qiu, and Daniel S Kirschen. Comparison of state-of-the-art transmission constrained unit commitment formulations. In *2013 IEEE power & energy society general meeting*, pages 1–5. IEEE, 2013.

[76] S Patra, SK Goswami, and B Goswami. Differential evolution algorithm for solving unit commitment with ramp constraints. *Electric Power Components and Systems*, 36(8):771–787, 2008.

[77] Ana Viana and João Pedro Pedroso. A new milp-based approach for unit commitment in power production planning. *International Journal of Electrical Power & Energy Systems*, 44(1):997–1005, 2013.

[78] PG Lowery. Generating unit commitment by dynamic programming. *IEEE Transactions on Power Apparatus and Systems*, (5):422–426, 1966.

[79] James Ostrowski, Miguel F Anjos, and Anthony Vannelli. Tight mixed integer linear programming formulations for the unit commitment problem. *IEEE Transactions on Power Systems*, 27(1):39–46, 2011.

[80] Chuan-Ping Cheng, Chih-Wen Liu, and Chun-Chang Liu. Unit commitment by lagrangian relaxation and genetic algorithms. *IEEE transactions on power systems*, 15(2):707–714, 2000.

[81] KS Swarup and S Yamashiro. Unit commitment solution methodology using genetic algorithm. *IEEE Transactions on power systems*, 17(1):87–91, 2002.

[82] Weerakorn Ongsakul and Nit Petcharaks. Unit commitment by enhanced adaptive lagrangian relaxation. *IEEE Transactions on Power Systems*, 19(1):620–628, 2004.

[83] B Zhao, CX Guo, BR Bai, and YJ Cao. An improved particle swarm optimization algorithm for unit commitment. *International Journal of Electrical Power & Energy Systems*, 28(7):482–490, 2006.

[84] TO Ting, MVC Rao, and CK Loo. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *IEEE transactions on power systems*, 21(1):411–418, 2006.

[85] Dimitris N Simopoulos, Stavroula D Kavatza, and Costas D Vournas. Unit commitment by an enhanced simulated annealing algorithm. *IEEE Transactions on Power Systems*, 21(1):68–76, 2006.

[86] Yun-Won Jeong, Jong-Bae Park, Se-Hwan Jang, and Kwang Y Lee. A new quantum-inspired binary pso: application to unit commitment problems for power systems. *IEEE Transactions on Power Systems*, 25(3):1486–1495, 2010.

[87] Prateek Kumar Singhal and R Naresh Sharma. Dynamic programming approach for solving power generating unit commitment problem. In *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)*, pages 298–303. IEEE, 2011.

[88] Ran Quan, Jinbao Jian, and Linfeng Yang. An improved priority list and neighborhood search method for unit commitment. *International Journal of Electrical Power & Energy Systems*, 67:278–285, 2015.

[89] Eric Krall, Michael Higgins, and Richard P O'Neill. Rto unit commitment test system. *Federal Energy Regulatory Commission*, 98, 2012.

[90] Clayton Barrows, Aaron Bloom, Ali Ehlen, Jussi Ikäheimo, Jennie Jorgenson, Dheepak Krishnamurthy, Jessica Lau, Brendan McBennett, Matthew O'Connell, Eugene Preston, et al. The ieee reliability test system: A proposed 2019 update. *IEEE Transactions on Power Systems*, 35(1):119–127, 2019.

[91] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[92] Leopold Kuttner, Martin Scheffler, Udo Buscher, and Dominik Möst. Ramping constraint formulations under consideration of reserve activation in unit commitment problems. *Zeitschrift für Energiewirtschaft*, pages 1–24, 2021.

[93] KA Juste, H Kita, E Tanaka, and J Hasegawa. An evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power systems*, 14(4):1452–1459, 1999.

[94] Tomonobu Senjyu, Hirohito Yamashiro, Katsumi Uezato, and Toshihisa Funabashi. A unit commitment problem by using genetic algorithm based on unit characteristic classification. In *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No. 02CH37309)*, volume 1, pages 58–63. IEEE, 2002.

[95] Huseyin Hakan Balci and Jorge F Valenzuela. Scheduling electric power generators using particle swarm optimization combined with the lagrangian relaxation method. *International Journal of Applied Mathematics and Computer Science*, 14:411–421, 2004.

[96] D Srinivasan and J Chazelas. A priority list-based evolutionary algorithm to solve large scale unit commitment problem. In *2004 International Conference on Power System Technology, 2004. PowerCon 2004.*, volume 2, pages 1746–1751. IEEE, 2004.

[97] Dimitris N Simopoulos, Stavroula D Kavatza, and Costas D Vournas. Reliability constrained unit commitment using simulated annealing. *IEEE Transactions on Power Systems*, 21(4):1699–1706, 2006.

[98] Tomonobu Senjyu, Tsukasa Miyagi, Ahmed Yousuf Saber, Naomitsu Urasaki, and Toshihisa Funabashi. Emerging solution of large-scale unit commitment problem by stochastic priority list. *Electric Power Systems Research*, 76(5):283–292, 2006.

[99] Tomonobu Senjyu, Ahmed Yousuf Saber, Tsukasa Miyagi, Naomitsu Urasaki, and Toshihisa Funabashi. Absolutely stochastic simulated annealing approach to large scale unit commitment problem. *Electric Power Components and Systems*, 34(6):619–637, 2006.

[100] Chuangyin Dang and Minqiang Li. A floating-point genetic algorithm for solving the unit commitment problem. *European Journal of Operational Research*, 181(3):1370–1395, 2007.

[101] Sarmila Patra, SK Goswami, and B Goswami. Fuzzy and simulated annealing based dynamic programming for the unit commitment problem. *Expert Systems with Applications*, 36(3):5081–5086, 2009.

[102] TW Lau, CY Chung, KP Wong, TS Chung, and Siu Lau Ho. Quantum-inspired evolutionary algorithm approach for unit commitment. *IEEE Transactions on Power Systems*, 24(3):1503–1512, 2009.

[103] Xiaohui Yuan, Anjun Su, Hao Nie, Yanbin Yuan, and Liang Wang. Unit commitment problem using enhanced particle swarm optimization algorithm. *Soft Computing*, 15(1):139–148, 2011.

[104] CY Chung, Han Yu, and Kit Po Wong. An advanced quantum-inspired evolutionary algorithm for unit commitment. *IEEE Transactions on Power Systems*, 26(2):847–854, 2010.

[105] Shantanu Chakraborty, Takayuki Ito, Tomonobu Senjyu, and Ahmed Yousuf Saber. Unit commitment strategy of thermal generators by using advanced fuzzy controlled binary particle swarm optimization algorithm. *International Journal of Electrical Power & Energy Systems*, 43(1):1072–1080, 2012.

[106] Anupam Trivedi, Dipti Srinivasan, Subhodip Biswas, and Thomas Reindl. Hybridizing genetic algorithm with differential evolution for solving the unit commitment scheduling problem. *Swarm and Evolutionary Computation*, 23:50–64, 2015.

[107] Anup Shukla and SN Singh. Advanced three-stage pseudo-inspired weight-improved crazy particle swarm optimization for unit commitment problem. *Energy*, 96:23–36, 2016.

[108] Kyu-Hyung Jo and Mun-Kyeom Kim. Improved genetic algorithm-based unit commitment considering uncertainty integration method. *Energies*, 11(6):1387, 2018.

[109] Jatinder Singh Dhaliwal and Jaspreet Singh Dhillon. Modified binary differential evolution algorithm to solve unit commitment problem. *Electric Power Components and Systems*, 46(8):900–918, 2018.

[110] CJ Baldwin, KM Dale, and RF Dittrich. A study of the economic shutdown of generating units in daily dispatch. *Transactions of the American Institute of Electrical Engineers. Part III: Power Apparatus and Systems*, 78(4):1272–1282, 1959.

[111] RH Kerr, JL Scheidt, AJ Fontanna, and JK Wiley. Unit commitment. *IEEE Transactions on Power Apparatus and Systems*, (5):417–421, 1966.

[112] Raymond R Shoults, Show Kang Chang, Steve Helmick, and W Mack Grady. A practical approach to unit commitment, economic dispatch and savings allocation for multiple-area pool operation with import/export constraints. *IEEE Transactions on Power Apparatus and Systems*, (2):625–635, 1980.

[113] Gerald B Sheble. Solution of the unit commitment problem by the method of unit periods. *IEEE Transactions on Power Systems*, 5(1):257–260, 1990.

[114] RM Burns. Optimization of priority lists for a unit commitment program. In *Proc. IEEE Power Eng. Soc. Summer Meeting, 1975*, 1975.

[115] Fred N Lee. Short-term thermal unit commitment-a new method. *IEEE Transactions on Power Systems*, 3(2):421–428, 1988.

[116] Abdullah M Elsayed, Ahmed M Maklad, and Sobhy M Farrag. A new priority list unit commitment method for large-scale power systems. In *2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*, pages 359–367. IEEE, 2017.

[117] Hao Quan, Dipti Srinivasan, Ashwin M Khambadkone, and Abbas Khosravi. A computational framework for uncertainty integration in stochastic unit commitment with intermittent renewable energy sources. *Applied energy*, 152:71–82, 2015.

[118] Saleh Y Abujarad, Mohammad Wazir Mustafa, and Jasrul Jamani Jamian. Recent approaches of unit commitment in the presence of intermittent renewable energy resources: A review. *Renewable and Sustainable Energy Reviews*, 70:215–223, 2017.

[119] Jimmie D Guy. Security constrained unit commitment. *IEEE Transactions on Power apparatus and Systems*, (3):1385–1390, 1971.

[120] CK Pang, Gerald B Sheblé, and F Albuyeh. Evaluation of dynamic programming based methods and multiple area representation for thermal unit commitments. *IEEE Transactions on Power Apparatus and Systems*, (3):1212–1218, 1981.

[121] Z Ouyang and SM Shahidehpour. An intelligent dynamic programming for unit commitment application. *IEEE Transactions on power systems*, 6(3):1203–1209, 1991.

[122] Stephen T Lee and Zia A Yamayee. Load-following and spinning-reserve penalties for intermittent generation. *IEEE Transactions on Power Apparatus and Systems*, (3):1203–1211, 1981.

[123] John A Muckstadt and Sherri A Koenig. An application of lagrangian relaxation to scheduling in power-generation systems. *Operations research*, 25(3):387–403, 1977.

[124] Arthur I Cohen and SH Wan. A method for solving the fuel constrained unit commitment problem. *IEEE Transactions on Power Systems*, 2(3):608–614, 1987.

[125] Fulin Zhuang and Frank D Galiana. Towards a more rigorous and practical unit commitment by lagrangian relaxation. *IEEE Transactions on Power Systems*, 3(2):763–773, 1988.

[126] Sudhir Virmani, Eugene C Adrian, Karl Imhof, and Shishir Mukherjee. Implementation of a lagrangian relaxation based unit commitment problem. *IEEE Transactions on Power Systems*, 4(4):1373–1380, 1989.

[127] Yong Fu, Mohammad Shahidehpour, and Zuyi Li. Security-constrained unit commitment with ac constraints. *IEEE transactions on power systems*, 20(2):1001–1013, 2005.

[128] Tao Li and Mohammad Shahidehpour. Price-based unit commitment: A case of lagrangian relaxation versus mixed integer programming. *IEEE transactions on power systems*, 20(4):2015–2025, 2005.

[129] Hugh Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research*, 11(3):399–417, 1963.

[130] Jonathan F Bard. Short-term scheduling of thermal-electric generators using lagrangian relaxation. *Operations Research*, 36(5):756–766, 1988.

[131] Ioannis G Damousis, Anastasios G Bakirtzis, and Petros S Dokopoulos. A solution to the unit-commitment problem using integer-coded genetic algorithm. *IEEE Transactions on Power systems*, 19(2):1165–1172, 2004.

[132] Nima Amjady and Ali Shirzadi. Unit commitment using a new integer coded genetic algorithm. *European Transactions on Electrical Power*, 19(8):1161–1176, 2009.

[133] Po-Hung Chen. Two-level hierarchical approach to unit commitment using expert system and elite pso. *IEEE Transactions on Power Systems*, 27(2):780–789, 2011.

[134] T Logenthiran and Dipti Srinivasan. Particle swarm optimization for unit commitment problem. In *2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems*, pages 642–647. IEEE, 2010.

[135] F Zhuang and FD Galiana. Unit commitment by simulated annealing. *IEEE Transactions on Power Systems*, 5(1):311–318, 1990.

[136] AH Mantawy, Youssef L Abdel-Magid, and Shokri Z Selim. A simulated annealing algorithm for unit commitment. *IEEE Transactions on Power Systems*, 13(1):197–204, 1998.

[137] Grzegorz Dudek. Adaptive simulated annealing schedule to the unit commitment problem. *Electric Power Systems Research*, 80(4):465–472, 2010.

[138] Tharam S Dillon, Kurt W Edwin, H-D Kochs, and RJ Taud. Integer programming approach to the problem of optimal unit commitment with probabilistic reserve determination. *IEEE Transactions on Power Apparatus and Systems*, (6):2154–2166, 1978.

[139] Chern-Lin Chen and Shun-Chung Wang. Branch-and-bound scheduling for thermal generating units. *IEEE transactions on energy conversion*, 8(2):184–189, 1993.

[140] Germán Morales-España, Jesus M Latorre, and Andres Ramos. Tight and compact milp formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems*, 28(4):4897–4908, 2013.

[141] Kai Pan and Yongpei Guan. Convex hulls for the unit commitment polytope. *arXiv preprint arXiv:1701.08943*, 2017.

[142] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning*, pages 344–353. PMLR, 2018.

[143] Sumit Mitra, Lige Sun, and Ignacio E Grossmann. Optimal scheduling of industrial combined heat and power plants under time-sensitive electricity prices. *Energy*, 54:194–211, 2013.

[144] AD Hawkes and MA Leach. Modelling high level system design and unit commitment for a microgrid. *Applied energy*, 86(7-8):1253–1265, 2009.

[145] Wolf-Peter Schill, Michael Pahle, and Christian Gambardella. Start-up costs of thermal power plants in markets with increasing shares of variable renewable generation. *Nature Energy*, 2(6):1–6, 2017.

[146] Edward V Mc Garrigle, John Paul Deane, and Paul G Leahy. How much wind energy will be curtailed on the 2020 irish power system? *Renewable Energy*, 55:544–553, 2013.

[147] Nikolaos E Koltsaklis, Athanasios S Dagoumas, Michael C Georgiadis, George Papaioannou, and Christos Dikaiakos. A mid-term, market-based power systems planning model. *Applied Energy*, 179:17–35, 2016.

[148] Kibaek Kim and Victor M Zavala. Large-scale stochastic mixed-integer programming algorithms for power generation scheduling. In *Alternative Energy Sources and Technologies*, pages 493–512. Springer, 2016.

[149] Claus C Carøe, Andrzej Ruszczynski, and Rüdiger Schultz. Unit commitment under uncertainty via two-stage stochastic programming. 1997.

[150] Matthias P Nowak, Rüdiger Schultz, and Markus Westphalen. A stochastic integer programming model for incorporating day-ahead trading of electricity into hydro-thermal unit commitment. *Optimization and Engineering*, 6(2):163–176, 2005.

[151] Darinka Dentcheva and Werner Römisch. Optimal power generation under uncertainty via stochastic programming. In *Stochastic programming methods and technical applications*, pages 22–56. Springer, 1998.

[152] Amin Nasri, S Jalal Kazempour, Antonio J Conejo, and Mehrdad Ghandhari. Network-constrained ac unit commitment under uncertainty: a benders' decomposition approach. *IEEE transactions on power systems*, 31(1):412–422, 2015.

[153] Ignacio Blanco and Juan M Morales. An efficient robust solution to the two-stage stochastic unit commitment problem. *IEEE Transactions on Power Systems*, 32(6):4477–4488, 2017.

[154] Yury Dvorkin, Yishen Wang, Hrvoje Pandzic, and Daniel Kirschen. Comparison of scenario reduction techniques for the stochastic unit commitment. In *2014 IEEE PES General Meeting— Conference & Exposition*, pages 1–5. IEEE, 2014.

[155] Ping Che, Lixin Tang, and Jianhui Wang. Two-stage minimax stochastic unit commitment. *IET Generation, Transmission & Distribution*, 12(4):947–956, 2018.

[156] Qianfan Wang, Jianhui Wang, and Yongpei Guan. Stochastic unit commitment with uncertain demand response. *IEEE Transactions on power systems*, 28(1):562–563, 2012.

[157] Chunheng Wang and Yong Fu. Fully parallel stochastic security-constrained unit commitment. *IEEE Transactions on power systems*, 31(5):3561–3571, 2015.

[158] Peter Meibom, Rüdiger Barth, Bernhard Hasche, Heike Brand, Christoph Weber, and Mark O'Malley. Stochastic optimization model to study the operational impacts of high wind penetrations in ireland. *IEEE Transactions on Power Systems*, 26(3):1367–1379, 2010.

[159] Colm Lowery and Mark OMalley. Reserves in stochastic unit commitment: An irish system case study. *IEEE Transactions on Sustainable Energy*, 6(3):1029–1038, 2014.

[160] EM Constantinescu, VM Zavala, M Rocklin, S Lee, and M Anitescu. Unit commitment with wind power generation: integrating wind forecast uncertainty and stochastic programming. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2009.

[161] Tim Schulze and Ken McKinnon. The value of stochastic programming in day-ahead and intra-day generation unit commitment. *Energy*, 101:592–605, 2016.

[162] Alexander Sturt and Goran Strbac. Efficient stochastic scheduling for simulation of wind-integrated power systems. *IEEE transactions on Power Systems*, 27(1):323–334, 2011.

[163] Eleanor Denny, A Tuohy, Peter Meibom, A Keane, D Flynn, A Mullane, and M O'malley. The impact of increased interconnection on electricity systems with large penetrations of wind generation: A case study of ireland and great britain. *Energy Policy*, 38(11):6946–6954, 2010.

[164] Katarzyna Maciejowska, Weronika Nitka, and Tomasz Weron. Day-ahead vs. intraday—forecasting the price spread to maximize economic benefits. *Energies*, 12(4):631, 2019.

[165] Matthias P Nowak and Werner Römisch. Stochastic lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty. *Annals of Operations Research*, 100(1):251–272, 2000.

[166] Lei Wu, Mohammad Shahidehpour, and Tao Li. Stochastic security-constrained unit commitment. *IEEE Transactions on power systems*, 22(2):800–811, 2007.

[167] Qipeng P Zheng, Jianhui Wang, Panos M Pardalos, and Yongpei Guan. A decomposition approach to the two-stage stochastic unit commitment problem. *Annals of Operations Research*, 210(1):387–410, 2013.

[168] Takayuki Shiina and John R Birge. Stochastic unit commitment problem. *International Transactions in Operational Research*, 11(1):19–32, 2004.

[169] Juan M Morales, Salvador Pineda, Antonio J Conejo, and Miguel Carrion. Scenario reduction for futures market trading in electricity markets. *IEEE Transactions on Power Systems*, 24(2):878–888, 2009.

[170] Miguel Asensio and Javier Contreras. Stochastic unit commitment in isolated systems with renewable penetration under cvar assessment. *IEEE Transactions on Smart Grid*, 7(3):1356–1367, 2015.

[171] Pierre Carpentier, Guy Gohen, J-C Culioli, and Arnaud Renaud. Stochastic optimization of unit commitment: a new decomposition framework. *IEEE Transactions on Power Systems*, 11(2):1067–1073, 1996.

[172] Yang Wang, Qing Xia, and Chongqing Kang. Unit commitment with volatile node injections by using interval optimization. *IEEE Transactions on Power Systems*, 26(3):1705–1713, 2011.

[173] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Incorporating wind power forecast uncertainties into stochastic unit commitment using neural network-based prediction intervals. *IEEE transactions on neural networks and learning systems*, 26(9):2123–2135, 2014.

[174] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[175] Aharon Ben-Tal and Arkadi Nemirovski. Robust convex optimization. *Mathematics of operations research*, 23(4):769–805, 1998.

[176] Ruiwei Jiang, Muhong Zhang, Guang Li, and Yongpei Guan. Two-stage robust power grid optimization problem. *submitted to Journal of Operations Research*, pages 1–34, 2010.

[177] Ruiwei Jiang, Jianhui Wang, and Yongpei Guan. Robust unit commitment with wind power and pumped storage hydro. *IEEE Transactions on Power Systems*, 27(2):800–810, 2011.

[178] Ruiwei Jiang, Jianhui Wang, Muhong Zhang, and Yongpei Guan. Two-stage minimax regret robust unit commitment. *IEEE Transactions on Power Systems*, 28(3):2271–2282, 2013.

[179] Ruiwei Jiang, Muhong Zhang, Guang Li, and Yongpei Guan. Two-stage network constrained robust unit commitment problem. *European Journal of Operational Research*, 234(3):751–762, 2014.

[180] Changhyeok Lee, Cong Liu, Sanjay Mehrotra, and Mohammad Shahidehpour. Modeling transmission line constraints in two-stage robust unit commitment problem. *IEEE Transactions on Power Systems*, 29(3):1221–1231, 2013.

[181] Chao-An Li, Raymond B Johnson, Alva J Svoboda, Chung-Li Tseng, and Eric Hsu. A robust unit commitment algorithm for hydro-thermal optimization. In *Proceedings of the 20th International Conference on Power Industry Computer Applications*, pages 186–191. IEEE, 1997.

[182] Guodong Liu and Kevin Tomsovic. Robust unit commitment considering uncertain demand response. *Electric Power Systems Research*, 119:126–137, 2015.

[183] Alvaro Lorca and Xu Andy Sun. Adaptive robust optimization with dynamic uncertainty sets for multi-period economic dispatch under significant wind. *IEEE Transactions on Power Systems*, 30(4):1702–1713, 2014.

[184] Alvaro Lorca, X Andy Sun, Eugene Litvinov, and Tongxin Zheng. Multistage adaptive robust optimization for the unit commitment problem. *Operations Research*, 64(1):32–51, 2016.

[185] Alexandre Velloso, Alexandre Street, David Pozo, José M Arroyo, and Noemi G Cobos. Two-stage robust unit commitment for co-optimized electricity markets: An adaptive data-driven approach for scenario-based uncertainty sets. *IEEE Transactions on Sustainable Energy*, 11(2):958–969, 2019.

[186] Qianfan Wang, Jean-Paul Watson, and Yongpei Guan. Two-stage robust optimization for nk contingency-constrained unit commitment. *IEEE Transactions on Power Systems*, 28(3):2366–2375, 2013.

[187] Hongxing Ye, Yinyin Ge, Mohammad Shahidehpour, and Zuyi Li. Uncertainty marginal price, transmission reserve, and day-ahead market clearing with robust unit commitment. *IEEE Transactions on Power Systems*, 32(3):1782–1795, 2016.

[188] Long Zhao and Bo Zeng. Robust unit commitment problem with demand response and wind energy. In *2012 IEEE power and energy society general meeting*, pages 1–8. IEEE, 2012.

[189] Alexandre Street, Fabrício Oliveira, and José M Arroyo. Contingency-constrained unit commitment with $n - k$ security criterion: A robust op-

timization approach. *IEEE Transactions on Power Systems*, 26(3):1581–1590, 2010.

[190] Qipeng P Zheng, Jianhui Wang, and Andrew L Liu. Stochastic optimization for unit commitment—a review. *IEEE Transactions on Power Systems*, 30(4):1913–1924, 2014.

[191] Chaoyue Zhao and Yongpei Guan. Unified stochastic and robust unit commitment. *IEEE Transactions on Power Systems*, 28(3):3353–3361, 2013.

[192] Chaoyue Zhao and Yongpei Guan. Data-driven stochastic unit commitment for integrating wind generation. *IEEE Transactions on Power Systems*, 31(4):2587–2596, 2015.

[193] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

[194] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[195] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[196] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pages 167–176, 2018.

[197] Elizaveta Kuznetsova, Yan-Fu Li, Carlos Ruiz, Enrico Zio, Graham Ault, and Keith Bell. Reinforcement learning for microgrid energy management. *Energy*, 59:133–146, 2013.

[198] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[199] Robin Henry and Damien Ernst. Gym-anm: Reinforcement learning environments for active network management tasks in electricity distribution systems. *Energy and AI*, page 100092, 2021.

[200] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[201] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[202] 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. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[203] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[204] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[205] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[206] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[207] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

[208] David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pages 3191–3199. PMLR, 2017.

[209] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[210] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[211] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[212] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[213] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[214] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[215] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[216] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[217] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

[218] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[219] George F Luger. *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education, 2005.

[220] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, volume 5, pages 156–165. Citeseer, 2005.

[221] Alan M Frieze. Shortest path algorithms for knapsack type problems. *Mathematical Programming*, 11(1):150–157, 1976.

[222] Richard E Korf, Michael Reid, and Stefan Edelkamp. Time complexity of iterative-deepening-a*. *Artificial Intelligence*, 129(1-2):199–218, 2001.

[223] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[224] Richard E Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial intelligence*, 27(1):97–109, 1985.

[225] Richard E Korf. Real-time heuristic search. *Artificial intelligence*, 42(2-3):189–211, 1990.

[226] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[227] Thomas L Dean and Mark S Boddy. An analysis of time-dependent planning. In *AAAI*, volume 88, pages 49–54, 1988.

[228] Edward F Moore. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959*, pages 285–292, 1959.

[229] Shimon Even. *Graph algorithms*. Cambridge University Press, 2011.

[230] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[231] David J Slate and Lawrence R Atkin. Chess 4.5—the northwestern university chess program. In *Chess skill in Man and Machine*, pages 82–118. Springer, 1983.

[232] Dusan Sormaz, Jing Huang, and Chandrasekhar Ganduri. Comparison of various heuristic evaluation functions for tsp space-search optimization. In *IIE Annual Conference. Proceedings*, page 1. Institute of Industrial and Systems Engineers (IISE), 2010.

[233] Bo Huang, Q Wu, and FB Zhan. A shortest path algorithm with novel heuristics for dynamic transportation networks. *International Journal of Geographical Information Science*, 21(6):625–644, 2007.

[234] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. Empowering a* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 539–547, 2019.

[235] Marco Ernandes and Marco Gori. Likely-admissible and sub-symbolic heuristics. In *ECAI*, volume 16, page 613. Citeseer, 2004.

[236] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

[237] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[238] johnjforrest, Stefan Vigerske, Haroldo Gambini Santos, Ted Ralphs, Lou Hafer, Bjarni Kristjansson, jpfasano, EdwinStraver, Miles Lubin, rlougee, jpgoncal1,

h-i gassmann, and Matthew Saltzman. coin-or/cbc: Version 2.10.5, March 2020.

[239] Badrul H Chowdhury and Saifur Rahman. A review of recent advances in economic dispatch. *IEEE transactions on power systems*, 5(4):1248–1259, 1990.

[240] Brian Beavis and Ian Dobbs. *Optimisation and stability theory for economic analysis*. Cambridge university press, 1990.

[241] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[242] Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, 2018.

[243] Hamdi Abdi. Profit-based unit commitment problem: A review of models, methods, challenges, and future directions. *Renewable and Sustainable Energy Reviews*, 138:110504, 2021.

[244] Lennart Soder. Simulation of wind speed forecast errors for operation planning of multiarea power systems. In *2004 International Conference on Probabilistic Methods Applied to Power Systems*, pages 723–728. IEEE, 2004.

[245] Christoph Weber, Peter Meibom, Rüdiger Barth, and Heike Brand. Wilmar: a stochastic programming tool to analyze the large-scale integration of wind energy. In *Optimization in the energy industry*, pages 437–458. Springer, 2009.

[246] Lokesh Kumar Panwar, Srikanth Reddy, Ashu Verma, Bijaya K Panigrahi, and Rajesh Kumar. Binary grey wolf optimizer for large scale unit commitment problem. *Swarm and Evolutionary Computation*, 38:251–266, 2018.

[247] Thomas Schröder and Wilhelm Kuckshinrichs. Value of lost load: An efficient economic indicator for power supply security? a literature review. *Frontiers in energy research*, 3:55, 2015.

[248] Srikrishna Sridhar, Jeff Linderoth, and James Luedtke. Locally ideal formulations for piecewise linear functions with indicator variables. *Operations Research Letters*, 41(6):627–632, 2013.

[249] Hannele Holttinen, Michael Milligan, Brendan Kirby, Tom Acker, Viktoria Neimane, and Tom Molinski. Using standard deviation as a measure of increased operational reserve requirement for wind power. *Wind Engineering*, 32(4):355–377, 2008.

[250] Dimitris Bertsimas, J Daniel Griffith, Vishal Gupta, Mykel J Kochenderfer, and Velibor V Mišić. A comparison of monte carlo tree search and rolling horizon optimization for large-scale dynamic resource allocation problems. *European Journal of Operational Research*, 263(2):664–678, 2017.

[251] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[252] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[253] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[254] John Miller and Moritz Hardt. Stable recurrent models. *arXiv preprint arXiv:1805.10369*, 2018.

[255] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[256] T Aruldoss Albert Victoire and A Ebenezer Jeyakumar. Reserve constrained dynamic dispatch of units with valve-point effects. *IEEE Transactions on Power Systems*, 20(3):1273–1282, 2005.

[257] Robert Sedgewick and Jeffrey Scott Vitter. Shortest paths in euclidean graphs. *Algorithmica*, 1(1-4):31–48, 1986.

[258] Bruce L Golden and Michael Ball. Shortest paths with euclidean distances: An explanatory model. *Networks*, 8(4):297–314, 1978.

[259] Philip J Heptonstall and Robert JK Gross. A systematic review of the costs and impacts of integrating variable renewables into power grids. *nature energy*, 6(1):72–83, 2021.

[260] Ndamulelo Mararakanye and Bernard Bekker. Renewable energy integration impacts within the context of generator type, penetration level and grid characteristics. *Renewable and Sustainable Energy Reviews*, 108:441–451, 2019.

[261] Erik Ela, Michael Milligan, and Brendan Kirby. Operating reserves and variable generation. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2011.

[262] Michael Joos and Iain Staffell. Short-term integration costs of variable renewable energy: Wind curtailment and balancing in britain and germany. *Renewable and Sustainable Energy Reviews*, 86:45–65, 2018.

[263] Germán Morales-España, Elis Nycander, and Jos Sijm. Reducing co2 emissions by curtailing renewables: Examples from optimal power system operation. *Energy Economics*, 99:105277, 2021.

[264] Henrik Klinge Jacobsen and Sascha Thorsten Schröder. Curtailment of renewable generation: Economic optimality and incentives. *Energy Policy*, 49:663–675, 2012.

[265] Vignesh Venkata Gopala Krishnan, Shyam Gopal, Ren Liu, Alex Askerman, Anurag Srivastava, David Bakken, and Patrick Panciatici. Resilient cyber infrastructure for the minimum wind curtailment remedial control scheme. *IEEE Transactions on Industry Applications*, 55(1):943–953, 2018.

[266] Ren Liu, Anurag K Srivastava, David E Bakken, Alexander Askerman, and Patrick Panciatici. Decentralized state estimation and remedial control action for minimum wind curtailment using distributed computing platform. *IEEE Transactions on Industry Applications*, 53(6):5915–5926, 2017.

[267] Georgios N Psarros and Stavros A Papathanassiou. Comparative assessment of priority listing and mixed integer linear programming unit commitment methods for non-interconnected island systems. *Energies*, 12(4):657, 2019.

[268] Cheng Wang, Feng Liu, W Wei, S Mei, F Qiu, and J Wang. Robust unit commitment considering strategic wind generation curtailment. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2016.

[269] Rui Alves, FS Reis, and Hong Shen. Wind power curtailment optimization for day-ahead operational planning. In *2016 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2016.

[270] Yury Dvorkin, Miguel A Ortega-Vazuqez, and Daniel S Kirschen. Wind generation as a reserve provider. *IET Generation, Transmission & Distribution*, 9(8):779–787, 2015.

[271] Mohammad Amin Hozouri, Ali Abbaspour, Mahmud Fotuhi-Firuzabad, and Moein Moeini-Aghtaie. On the use of pumped storage for wind energy maximization in transmission-constrained power systems. *IEEE Transactions on Power Systems*, 30(2):1017–1025, 2014.

[272] U.S. Energy Information Administration. Annual Energy Review. `https://www.eia.gov/totalenergy/data/annual/index.php`.

[273] Adrien Couëtoux, Jean-Baptiste Hoock, Nataliya Sokolovska, Olivier Teytaud, and Nicolas Bonnard. Continuous upper confidence trees. In *International Conference on Learning and Intelligent Optimization*, pages 433–445. Springer, 2011.

[274] Corbett A Grainger and Charles D Kolstad. Who pays a price on carbon? *Environmental and Resource Economics*, 46(3):359–376, 2010.

[275] National Grid ESO. NETS Security and Quality of Supply Standard v2.5. Technical report, 2021.

[276] Göran Andersson, Peter Donalek, Richard Farmer, Nikos Hatziargyriou, Innocent Kamwa, Prabhashankar Kundur, Nelson Martins, John Paserba, Pouyan Pourbeik, Juan Sanchez-Gasca, et al. Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance. *IEEE transactions on Power Systems*, 20(4):1922–1928, 2005.

[277] Ronan Doherty and Mark O'malley. A new approach to quantify reserve demand in systems with significant installed wind capacity. *IEEE Transactions on Power Systems*, 20(2):587–595, 2005.

[278] North American Electric Reliability Corporation. Reliability Guidelines. Technical report, September 2018.

[279] Govind S Mudholkar and Deo Kumar Srivastava. Exponentiated Weibull family for analyzing bathtub failure-rate data. *IEEE Transactions on Reliability*, 42(2):299–302, 1993.

[280] Ming Xie and Chin Diew Lai. Reliability analysis using an additive weibull model with bathtub-shaped failure rate function. *Reliability Engineering & System Safety*, 52(1):87–93, 1996.

[281] CE Love and R Guo. Utilizing weibull failure rates in repair limit analysis for equipment replacement/preventive maintenance decisions. *Journal of the operational Research Society*, 47(11):1366–1376, 1996.

[282] North American Electric Reliability Corporation. State of Reliability. Technical report, July 2020.

[283] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[284] Louis A Wehenkel. *Automatic learning techniques in power systems*. Number 429. Springer Science & Business Media, 1998.

[285] Adrian Kelly, Aidan O'Sullivan, Patrick de Mars, and Antoine Marot. Reinforcement learning for electricity network operation. *arXiv preprint arXiv:2003.07339*, 2020.

[286] Karsten Neuhoff, Nolan Ritter, Aymen Salah-Abou-El-Enien, and Philippe Vassilopoulos. Intraday markets for power: Discretizing the continuous trading. 2016.

[287] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.