LANCASTER UNIVERSITY

APPROXIMATE POLICY ITERATION (API) WITH NEURAL NETWORKS FOR THE GENERALIZED SINGLE NODE ENERGY STORAGE PROBLEM

BY

RICHLOVE AMPOFOWAA FRIMPONG

This thesis is submitted to the Department of Entrepreneurship and Strategy and the Board of Examiners of Lancaster University in fulfilment of the requirements for the degree of Master of Science by Research in Innovation.

JANUARY, 2019

DECLARATION

I, RICHLOVE AMPOFOWAA FRIMPONG, hereby declare that this thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:... Date: ....

Name: Richlove Ampofowaa Frimpong

ABSTRACT

Energy storage problems are hard sequential decision-making problems often modelled as markov decision processes. Exact solution using dynamic programming quickly becomes implausible with large state spaces hence approximate dynamic programming using policy iteration (API) is often employed in such cases. API does not always work, one reason being that the approximation architectures used are often linear for computational tractability reasons. We propose a mathematical model which allows easier implementation of non-linear approximations with API. We use neural networks along with monte-carlo simulation to predict the future values for the generated states during the improvement step of the API algorithm. Our initial experiments suggest that the proposed method provides good results which can be further improved with more fine tuning of the neural network parameters.

ACKNOWLEDGEMENTS

I express my sincere gratitude to my supervisors Dr Denes Csala and Dr Soetanto for their guidance during this dissertation period. Special thanks to the Centre for Global Eco-Innovation (CGE) and Extreme Low Energy (ELe) for the opportunity they gave me to embark on this Masters by Research Course. I would also like to thank Dr. Trivikram Dokka, whose immense support and invaluable inputs made this research project a great learning experience. I am forever grateful.

Finally, I must express my profound gratitude to my parents and to all my friends for providing me with the unfailing support and continuous encouragement throughout this year of study. This accomplishment would not have been possible without them. Thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background and Motivation

The traditional electricity market enables the flow of electricity from huge power stations through national or regional transmission networks to local distribution networks which then supply the electricity to the consumers. In other words, generators trade their produced electricity from their power plants on the wholesale market from which suppliers buy and sell electricity to their customers at competitive prices. The electricity market is however undergoing major transformation with the continuous prevalence of smart grids which provide a more optimized way of monitoring and adjusting electricity flows and enable easier integration of distributed generation systems and renewable energy sources (e.g. wind farms and solar panels).

Also, there has been an increased interest in providing feasible regulations, policies and cost-incentives by some countries to increase the use of renewable energy technologies in electricity generation and supply due to benefits such as energy security, reliability and reduced emissions. For instance, renewable energy sources provided about a quarter of the total electricity generation in the UK compared to 5% in 2016 (Ofgem, Accessed December 2018). Renewable energy sources such as wind farms or solar panels are much less controllable and highly volatile as they depend on external factors like weather, seasons and time of the day. These resources can be paired with energy storage devices like batteries to increase the value of the energy generated and optimize existing grid connections. Battery storage provides many benefits such as peak shaving, time shifting, the provision of operating reserve and the reduction of curtailment, electricity price arbitrage and balancing costs (Eyer et al., 2004; Eyer and Corey, 2010; Zhou et al., 2018). It is therefore no surprise that there are projections of likely increment in the global deployment of battery technology as a grid-scale energy storage device with the 70% decrease in the prices for battery cells between 2012 and 2017 (PV Magazine, Accessed February 2019).

The combination of smart metering systems, smart grids, decentralised renewable generation and energy storage systems provide end-users the opportunity to reduce their reliance on the national grid, produce their own energy, manage their electricity usage in response to different prices throughout the day (also known as demand response) and sell the excess to the grid in order to reduce their energy bills. For example, customers may decide to reduce the amount of energy they purchase during peak demand hours and rather use excess stored energy from the renewable source or battery to satisfy demand (also known as peak shaving). They may also decide to shift the demand (say do the laundry at another time) to less expensive, off-peak periods or even make use of lower tariffs to buy energy to store for peak demand periods or sell stored energy at higher tariffs. Aside this, the rapid penetration of electric vehicles is projected to have a major impact on the electricity market since they can be considered as short-term mobile battery storage for the grid. Electric vehicle owners may choose to recharge their batteries during lower demand periods (typically at night) and then potentially sell power back into the grid when the demand is high. With all these new technologies and options, the grid can be regarded as omnidirectional hence permitting the flow of electricity from the distribution networks to consumers and vice versa. Therefore, we see the emergence and development of new market models and algorithms such as storage-as-service business models, peer-to-peer trading (P2P) electricity trading models, machine learning energy trading algorithms (e.g. auto-trading and algo-trading) and blockchain applications in energy trading.

Electricity customers are faced with the issue of making real-time trading decisions due to these new and exciting opportunities which allow them to become active participants in the electricity market. For instance, households that own energy storage systems such as batteries or electric vehicles and renewable energy sources like solar panels must be able to determine the optimal energy resource management and operation policy for the distribution of energy in order to satisfy the total household electricity demand and reduce their bills. This problem of optimally managing a storage device (e.g. battery) that interacts with both the grid and an uncertain renewable energy source (e.g. solar or wind) to satisfy demand whilst considering the electricity spot prices can also be referred to as the energy storage problem. In this regard, the energy storage problem is that of deciding when to buy or sell electricity from or to the grid, when to use the the renewable energy source and when to charge or discharge the battery.

Since the energy storage problem is a very complex decision problem, we concentrate on its basic sub-problem (also known as the single-node energy storage (SNES) problem). Halman et al. (2018) described the SNES problem as the decision problem faced by a single energy-producing node in a smart grid which does not explicitly consider the goals of the system operator. The SNES problem has garnered a lot of attention from researchers from fields such as operations research, computer science and electrical engineering (Löhndorf and Minner, 2010; Powell, 2011; Hannah and Dunson, 2012; Powell, George, et al., 2012; Jiang et al., 2014; Natarajan et al., 2014; Xiaomin et al., 2014; Zhou et al., 2016; Halman et al., 2018) and can be modelled as a stochastic dynamic program due to the uncertainties involved in the decision-making process. Many of these existing studies in literature have suggested a variety of approaches to solve their proposed stochastic dynamic programming mathematical models as well as performed numerical analysis on the resulting optimal or heuristic policies. The most popular approach encountered so far in these literature is the Approximate Dynamic Programming (ADP) approach which can be employed with several techniques and approximation architectures. An example technique with ADP is the Approximate Policy Iteration (API) method and we propose a novel API algorithm which employs neural networks to approximately solve the SNES problem in this work. To the best of our knowledge, the only other paper which combines neural networks within API is that of Liu and Wei (2014) but the neural network structure and implementation employed by them differs from ours.

1.2 Research Aim

The main aim of our research is to evaluate the performance of our novel mathematical model and API algorithm against benchmark problems to ensure that our proposed approach works and can produce optimal or close-to-optimal results.

1.3 Research Contribution

Most literature mainly used the mathematical model described in Section 4.2 with minor variations depending on the problem definition and assumptions. These papers usually assumed equal buying and selling prices on the electricity spot market so as to make the problem easy to solve. To the best of our knowledge, no research has been conducted to see the effect of different buying and selling prices on the proposed mathematical models and algorithms of this problem even though the buying and selling prices are less likely to be equal in real world applications. Therefore in this work, we consider a more general version of the SNES problem with different buying and selling prices with the presumption that a good performance of our proposed novel algorithm on this harder version of the SNES problem may lead to a much better performance in the easier version with equal prices.

Also, we capture the decisions using much simpler decision variables, mainly inspired by Secomandi (2010) rather than representing them as possible energy transfers that can occur from the main components of the single node energy storage problem as shown in Figure 1. This reduces the dimensionality of the feasible decision space and enables the determination of the optimal decision within reasonable time.

In addition to the use of fewer and simpler decision variables, we employ a relaxation approach by capturing the transmission losses associated with the battery energy transfers in the objective function as costs rather than in the constraint equation as energy units as done in most previous studies (Halman et al., 2018; Jiang et al., 2014; Salas and Powell, 2013). In optimization literature, relaxations are often useful in obtaining very close-to-optimal solutions and also serve as useful indicators of the quality of a solution. We highlight here that valuing losses in monetary units rather than in energy units is plausible especially when the battery operation is outsourced. Hence, our proposed model is also easily usable in the scenario where the decision-maker uses a storage-as-a-service business model and does not directly own the energy storage system.

In his work on the commercial management of a commodity storage asset, Secomandi (2010) proposed that the feasible decision space structure can be divided into three possible optimal regions: buy and inject (BI), do nothing (DN) and withdraw and sell (WS). Extending Secomandi's work to our case, we identify and divide the decision space structure from our proposed model into eight possible optimal regions (buy and inject (BI), buy and withdraw (BW), sell and inject (SI), sell and withdraw (SW), sell (S), buy (B), inject (I) withdraw (W)) and also describe the scenarios under which we may observe these decision regions as optimal.

1.4 Thesis Structure and Outline

The thesis is structured as follows: Chapter 2 provides a literature review of various papers in the energy storage field, with emphasis on energy storage modelling techniques. Topics such as Dynamic Programming (DP), Markov Decision Processes (MDPs) and Approximate Dynamic Programming (ADPs) are discussed as part of our methodology in Chapter 3. In chapter 4, we first present and highlight the differences between an example mathematical model from literature and our proposed model as well as describe our novel API algorithm. Chapter 5 then describes the experimental set-up which includes the data, benchmark problem instances and metrics used in evaluating the performance of our proposed API algorithm. We also discuss our findings from the conducted numerical experiments. Finally, we outline some ideas for further work that can be conducted to improve this research in Chapter 6.

CHAPTER TWO

LITERATURE REVIEW

Evangelopoulos et al. (2016) provided an extensive review of the models, optimization methods and areas of further research in the optimal operation of smart distribution networks (OOSDN) field. Much work has also been done investigating the economic prospects and value of energy storage. Staffell and Rustomji (2016) showed that battery storage in the UK could triple their profits by participating in the reserve market in addition to providing arbitrage. Also, results from Granado et al. (2016) indicated that the integration of energy storage and renewable energy sources with smart grids reduces energy costs by 7-10% in electricity and 3% in gas charges, a further proof of the value of energy storage. Barbour et al. (2018) argued the need for the development of specific market mechanisms and energy policies for the deployment of community energy storage systems due to the higher internal rate of return (IRR) associated with community energy storage investments (El-Batawy and Morsi, 2018). As such, Lüth et al. (2018) proposed two market designs for peer-to-peer (P2P) trading in the presence of storage devices and Zhang et al. (2018) developed a P2P system architecture and energy trading platform. Results from Lüth et al. (2018) showed that there is a potential end-user savings of 31% due to P2P trade and storage. Furthermore, results from Zhang et al. (2018) proved that P2P energy trading facilitates local power and energy balance.

Since this thesis concentrates more on the modelling and optimization of energy storage problems, we provide a review on some of the problems, solution approaches and policies that have been considered in the next sections of the literature review. This chapter is then concluded with a review on the work that has been done on Approximate Dynamic Programming approaches because of its frequent application as a solution method for energy storage problems.

2.1 Energy Storage Problem Variations and Solution Approaches

The energy storage problem is strongly linked to inventory optimization problems as they both deal with determining the decisions for meeting demand from various supply sources. The demand and supply could either be deterministic or stochastic and these have been well studied under the inventory optimization theory concepts (Zipkin, 2000; Porteus, 2002). This problem can also be related to research conducted in commodity trading literature such as Rempala (1994) and Secomandi (2010) due to the possibility of trading energy with the grid. In particular, Secomandi (2010) focused on the commercial management of a commodity storage asset and determining the optimal inventory-trading policy under capacity constraints and stochastic spot prices.

Similar to the settings of the problem discussed in this thesis, Zhou et al. (2018) investigated the management of a merchant wind farm co-located with a grid-level storage facility and connected to the market via a transmission line. In contrast to majority of research conducted on this problem, they considered negative electricity prices which happens in most deregulated markets and quantified their effect on the value of energy storage. They formulated the problem as a finite-horizon Markov decision process and conducted a numerical study to assess the performance of their suggested heuristics against the optimal policy.

According to Halman et al. (2018), the assumption of the same buying and selling price on the spot market makes the stochastic version of the SNES problem solvable in polynomial time via dynamic programming. However, the stochastic version of the problem becomes *#P-hard* when different buying and selling prices are used in the model. This may be the reason why most literature consider the same buying and selling price in their models. They also showed that the deterministic case of the single-node energy storage problem can be solved strongly in polynomial time. They provided a Fully Polynomial-Time Approximation Scheme (FPTAs) which is an extension of the framework by Halman et al. (2009) to consider continuous state and action spaces for the case in which energy can only be bought from the grid. Halman et al. (2018) also suggested that their traditional FPTAs used with a piecewise linear convex objective function (for a minimization problem) provided stronger approximation guarantees than the $(\sum, \prod)$-FPTAs developed in Halman et al. (2009).

Teleke et al. (2010) considered a rule-based dispatch scheme for the finite-horizon energy storage problem without taking into account the effect of prices or the variability of wind. Moazeni et al. (2015) suggested the need to consider the effect of risk on the optimal policy following the results from the risk analysis they conducted on an optimal deterministic risk-neutral policy and a simple myopic policy.

In the case of problems with infinite horizon, Harsha and Dahleh

(2015) utilized a stochastic dynamic program formulation to minimize the average cost derived from the installation and management of a storage device integrated with a renewable energy source to meet uncertain demand under dynamic pricing. They also showed that the optimal management policy has a dual threshold structure which is also discussed in Rempala (1994) and Secomandi (2010) under the commodity trading context.

2.2 Energy Storage Problem Policies

Since the single-node energy storage problem is seen as a stochastic optimization problem, a solution requires the computation of the optimal policy and the optimal profit. A policy is simply defined as any mapping, rule or function that determines a decision x_t based on a state S_t. A policy that depends on only the available information in a state S_t at time t is referred to as an admissible policy or $\mathcal{F}_t$-measurable (Powell and Meisel, 2016).

The choice of a policy requires a good balance between its computational complexity and solution quality and robustness. Most papers tend to focus on one class of policy even when the problem structure suggests that other policies could be potential solutions due to the challenge of analysing policies. Therefore in Powell and Meisel (2016), researchers are advised to simulate variations of their selected policy class and compare to a standard benchmark such as a deterministic lookahead to ensure that they have the best policy within their selected policy class.

Powell (2011) explained four fundamental classes of policies which have also been reviewed by Powell and Meisel (2016) and Durante et al. (2017). These policy classes are discussed below.

2.2.1 Myopic Policies/ Cost Function Approximations (CFA)

These are widely used in resource allocation problems as elementary policies which optimize the current costs or rewards without explicitly using any forecasted information.

They are however replaced in Powell and Meisel (2016) and Durante et al. (2017) with Cost Function Approximations (CFA) policies. CFA-based policies could be seen as either myopic policies with tunable parameters or deterministic lookahead (which can accommodate forecasts) with tunable parameters to handle uncertainty. According to Durante et al. (2017), CFAs rely on policy search to optimize any parameters involved in maximizing its parameterized cost function approximation. Simão et al. (2017) employed the CFA approach in their robust power system control to specifically tune the reserves in order to cater for the intermittent nature of renewable energy sources.

2.2.2 Lookahead Policies

These policies make decisions now by maximizing over some horizon both current and future actions based on an approximate model of the problem. They are typically used for non-stationary problems with available good forecasts such as unit commitment by independent system operators (ISOs) (Powell and Meisel, 2016). They can either be deterministic lookahead policies which have long been used as heuristics in planning natural gas storage (Lai et al., 2010) or stochastic lookahead models which have been over the years used for hydroelectric power planning (Jacobs et al., 1995). Arnold and Andersson (2011) used deterministic lookahead policies also known as Model Predictive Control (MPC) policies to operate a storage hub comprising of battery, hot storage devices and uncertain renewable resources, whilst considering electricity prices and natural gas prices.

2.2.3 Policy Function Approximations (PFA)

PFAs map states to actions without the use of any form of optimization method. They may come in the form of lookup tables (e.g. if temperature = 80, increase generating reserves), parametric functions (e.g. charge the battery when the electricity price is below a threshold θ^{charge} and discharge when it is above $\theta^{discharge}$) or through non-parametric models such as kernel regression. Han and Weinan (2016) used an artificial neural network (ANN) which is a form of a non-linear PFA to optimize the flow of power between a wind farm, the grid and a storage device to satisfy a time varying load.

2.2.4 Value Function Approximations (VFA)

These policies capture the future value (cost or reward) of being in a state at a point in time. In other words, VFA-based policies approximate the impact of current decisions on the future by maximizing the one-step contribution of the decision in addition to the approximation of the future contribution of that decision (Powell and Meisel, 2016; Durante et al., 2017). VFAs like PFAs may also use lookup tables (e.g. state space aggregation, hierarchical aggregation, representatives), parametric models (e.g. basis functions, piecewise linear functions, neural networks) and non-parametric models (e.g. kernel regression, support vector machines) as their approximation strategies. Value function approximation is widely used in approximate dynamic programming research papers such as Jiang et al. (2014), Natarajan et al. (2014) and Xiaomin et al. (2014).

2.3 Approximate Dynamic Programming (ADP) In Energy Storage

Approximate dynamic programming (ADP) has been used extensively as a modelling framework for most energy storage problems.

For instance, Jiang et al. (2014) considered the use of ADP since the implementation of exact backward dynamic programming especially for large-scale problems can quickly become intractable and computationally intensive. However, for the same wind-farm battery storage system configuration, whereas Jiang et al. (2014) utilized forward approximate dynamic programming methods, Durante et al. (2017) applied the backward approximate dynamic programming method also seen in Senn et al. (2014) and Cheng et al. (2018) to overcome the computational problems associated with exact backward dynamic programming. In contrast to forward ADP, which estimates the value functions while stepping forward in time, backward ADP like dynamic programming performs a single backward pass but fits an approximate model based on a small sample of states. Aside Durante et al. (2017) and Cheng et al. (2018), backward ADP has been seldom used in the energy storage field. Durante et al. (2017) showed that his backward ADP methodology consistently produces higher quality solutions than forward ADP methods such as the ones used in Jiang et al. (2014).

Powell, George, et al. (2012) proposed another type of modelling and algorithmic strategy based on the ADP framework, known as the Stochastic Multiscale model for the Analysis of energy Resources, Technology, and policy (SMART) to model long-term investment decisions and economic analyses of portfolios for energy technologies in the presence of uncertainty. They showed that their algorithm exhibits robust behaviours when applied to stochastic problems. SMART can also be applied to deterministic problems which makes it comparable to the solution from a deterministic linear model.

2.3.1 Value Function Approximation Methods In ADP

Value function approximations in approximate dynamic programming enables the creation of a policy by approximating the value of being in a state. Some academic papers have focused on the different methodologies used in developing the value function approximations employed in ADP.

In particular, Jiang et al. (2014) compared the performance of various approximation architectures implemented with ADP approaches such as Approximate Policy Iteration (API), Approximate Value Iteration(AVI) and Direct Policy Search on benchmark instances for the finite-horizon energy storage problem. They used non-parametric models such as Support Vector Regression (SVR), Gaussian Process Regression (GPR), Local Polynomial Regression (LPR) and Dirichlet Cloud-Radial Basis Function (DCR) in their API method whereas Liu and Wei (2014) applied neural networks in their proposed policy iteration method for solving the infinite horizon optimal control problem for non-linear systems.

Scott and Powell (2012) also focused on the use of a parametric linear model with pre-specified basis functions, least-square temporal difference (LSTD), and Bellman error minimization with approximate policy iteration to solve the same energy allocation problems considered by Jiang et al. (2014). A similar approach is used by Löhndorf and Minner (2010) to find the optimal infinite horizon storage and bidding strategy in the day-ahead market for a renewable power generation and energy storage system.

Jiang et al. (2014) presented a version of the approximate value iteration algorithm (AVI) known as Monotone-ADP which exploits the monotone nature of the problem. Their AVI algorithm required a lookup table representation of the state space unlike their proposed API method. In his PhD thesis, Dong (2010) stated that the resulting lookup table for a problem with a fairly large state space under optimal policies will be huge even though the values in many states are unlikely to be used. He suggested the use of simulation based value iteration algorithm that updates the approximation iteratively and reduces the computational efforts needed. Another commonly used strategy in ADP is to aggregate the state space so as to reduce its size. In aggregation, the value of aggregated states are rather estimated instead of the value of individual states as each individual state is made to belong to an aggregated state. Examples of aggregation methods used are fixed level aggregation (Bean et al., 1987), adaptive state aggregation (Bertsekas and Castanon, 1989; Singh et al., 1995) and hierarchical aggregation (Mes and Rivera, 2016).

It can be deduced from Nascimento and Powell (2010), Jiang et al. (2014) and Jiang and Powell (2015) that pure lookup AVI table perform poorly in practice due to very slow convergence rate despite the existence of convergence theory. However results from Jiang et al. (2014) show that structured lookup table AVI outperform other more general approaches like API paired with a generic approximation technique but is limited to moderately sized-problems. Therefore Hannah and Dunson (2012) approximated the value functions by employing Dirichlet process mixture models which scales well to large state spaces.

The ADP proposed by Nascimento (2008) iteratively constructs piecewise linear and concave value function approximations to help determine the solution of complex storage problem. He also proved that his algorithm converges to an optimal policy by learning the optimal value functions for important regions of the state space selected by the algorithm. Nascimento and Powell (2013) provided an extension to the algorithm and results presented in Nascimento (2008) by considering problems with highly-dimensional continuous decision vectors. Similar to Nascimento (2008) and Nascimento and Powell (2013), Salas and Powell (2013) exploited the concavity nature of the Value Function Approximations to speed up the convergence of their proposed finite-horizon ADP algorithm. Their work unlike most papers was specifically aimed at handling multiple storage devices for grid-level storage since results from recent research conducted in the vehicular electronics and energy systems field have shown the potential benefits of using integrated energy storage systems (Vazquez et al., 2010; Kraning et al., 2011; Kuperman and Aharon, 2011; El-Batawy and Morsi, 2018).

A fundamental challenge with ADP which is discussed in Powell (2011) and Mes and Rivera (2016) is the exploration versus exploitation problem. This refers to the issue of deciding whether to visit a state for no apparent reason or to make use of current estimates of the value function in order to select the best decision. For instance, the ADP used in Nascimento (2008) employed a pure exploitation scheme in which the states that were visited depended on the decisions produced by solving the approximate problem.

CHAPTER THREE

METHODOLOGY

Like many sequential decision problems, energy storage problems are often modelled in most academic literature as Markov Decision Processes (MDPs) and solved using the Approximate Dynamic Programming (ADP) method (Halman et al., 2009; Jiang et al., 2014; Jiang and Powell, 2015; Zhou et al., 2018). We explain the theories behind MDPs and ADPs in this chapter since we also take a similar approach in our work.

3.1 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are discrete time stochastic control processes which are useful for studying optimization problems solved via dynamic programming and reinforcement learning. MDPs like any other sequential (multi-stage) stochastic optimization model consist of five fundamental elements: states, actions or decisions, exogenous information, transition function and cost/contribution function. We provide a brief description of each element below.

3.1.1 State Variable

Definition 3.1.2 The state variable S_t is the minimally dimensioned function of history that is necessary and sufficient to compute the decision function, the transition function, and the cost (or reward or contribution) function (Powell, 2011).

In resource management problems, state variables can be categorized into three types:

- The resource/physical state, R_t: depicts the status and attributes of the physical resources being managed. In the case of the SNES problem, this could be the amount of energy stored in the battery.
- The information state, I_t: refers to any other information apart from the physical state that is needed to make a decision, compute the transition, cost/contribution function. Examples in our case are the spot prices of electricity, the demand and wind profiles.
- The belief/knowledge state, K_t: is a set of probability distributions which describes one's knowledge of unobservable parameters. The remaining lifetime of a battery is an example of a knowledge state.

For most resource allocation problems, R_t and I_t are often used as the state variables. Hence a typical representation of the state variable is given by $S_t = (R_t, I_t)$.

3.1.3 Decision Variable

Decision variables refer to the quantities that need to be determined in order to solve the problem. In stochastic programming or markov decision processes, they are often represented by two notational systems: x_t for decisions in the form of vectors and a_t for scalar, discrete actions.

3.1.4 Exogenous Information

Exogenous information are random variables whose information become available over time from some external source. In our case, our exogenous information can be the energy demand, the electricity spot prices or the amount of energy produced by our renewable source.

It is important to distinguish the modelling of the *flow of exogenous information* from the *flow of physical resources.* Therefore Powell, Simao, and Bouzaiene-Ayari (2012) advise modellers to think of decisions as occurring in discrete time, while exogenous information arrive in continuous time using W_t = new information that first becomes available between $t-1$ and t. The random exogenous information flow could either be modelled using some probability distribution or through sample realizations.

3.1.5 Transition Function

The transition function, $S_{t+1} = S^M(S_t, x_t, W_{t+1})$, describes the evolution of the system from one state to another based on the selected decisions and exogenous information.

For many problems, the exogenous information flow can be described as a Markov information process since the exogenous information W_t arriving during time interval $t+1$ depends on the state S_t at the end of the time interval t, but is conditionally independent of all prior history given S_t. In this case the transition function can be stored in the form of a one-step matrix using $\mathbb{P}(S_{t+1}|S_t, x)$ which is the probability of transitioning to state

S_{t+1} if the system is in state S_t and decision x_t is taken.

The transition probability only depends on the current state and is independent of all the previous states and actions. This property is known as the Markov property, hence the name Markov Decision Processes. The one-step transition matrix could either be given as data or derived using assumptions from the underlying exogenous information process of the problem. Similar to Powell (2011), the techniques considered in this thesis will directly use the transition function since the one-step transition function can be computationally intractable for a very large state space,.

3.1.6 Contribution Function

The contribution function $C(S_t, x_t)$ at time t is the reward of making decision x_t in state S_t.

In summary, at each time step in the markov decision process, the system with a state space $\mathcal{S}$ is in a particular state $S_t \in \mathcal{S}$. In that state S_t, a decision x_t is selected from the feasible decision region $\mathcal{X}_t$ which results in rewards or costs, by computing the contribution or cost function $C(S_t, x_t)$. Using the transition function directly or the one-step transition matrix, the system is then sent to a new state S_{t+1} which is conditionally independent of all the previous states and actions.

Definition 3.1.7 A policy $\pi \in \Pi$ can be seen as a decision function $X^{\pi}(S_t)$ that returns a decision $x_t \in \mathcal{X}_t$ for all states $S_t \in \mathcal{S}$, with Π being a set of potential decision functions (Mes and Rivera, 2016).

The goal is to find the policy that determines the best actions and maximizes the objective function.

3.2 Determining The Optimal Policy

Dynamic programming solves complex MDPs by breaking them into smaller sub-problems. According to Bellman (1957), the optimal policy for a problem modelled as a markov decision process is the one that provides an optimal solution to all the sub-problems of the MDP. The optimal policy therefore can be found by solving the *Bellman equation* which enables the computation of the *value function* in a recursive approach.

$$\begin{aligned} V_t(S_t) &= \max_{x_t \in \mathcal{X}_t} \Big(C(S_t, x_t) + \\ & \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(S_t + 1 = s' | S_t, x_t) V_{t+1}(s') \Big) \forall t \in T \end{aligned} \quad (3.2.1)$$

The value function defined in Equation (3.2.1) consists of two parts: the reward for being in state S_t and taking decision x_t, and reward at the next state S_{t+1} derived from the decisions that were taken at state S_t. With dynamic programming, the value of V_{t+1} is assumed to be known and using the Bellman equation, we step back in time to compute the value function V_t for all the states in the state space. The optimal decision is therefore the decision that gives the maximum V_t for each state.

However, the computation of the Bellman equation via dynamic programming is generally difficult and possibly intractable for many large problems due to a particular drawback of DP known as the *curses of dimensionality.* Powell (2011) states the three main *curses of dimensionality* in DP as follows:

- the possibility of a large state space $\mathcal{S}$ which may affect the evaluation of the value function $V_t(S_t)$ for all the states within reasonable time.

- the possibility of a large feasible decision space $\mathcal{X}_t$ which may hinder the determination of the optimal decision for all the states within reasonable time.
- the possibility of a large outcome space which may cause the computation of the expectation of the 'future' profits to be intractable.

The size of the outcome space depends on the dimension of the random information W_t. Using the transition function directly, Equation (3.2.1) can be rewritten as:

$$\begin{aligned} V_t(S_t) &= \max_{x_t \in \mathcal{X}_t} \Big(C(S_t, x_t) + \\ &\quad \gamma \sum_{\omega \in \Omega_{t+1}} \mathbb{P}(W_t + 1 = \omega) V_{t+1}(S_{t+1} | S_t, x_t, \omega) \Big) \forall t \in T \end{aligned} \tag{3.2.2}$$

where Ω_{t+1} is the set of all possible outcomes of W_{t+1} and $\mathbb{P}(W_t + 1 = \omega)$ is the probability of outcome $\omega \in \Omega_{t+1}$.

The Approximate Dynamic Programming (ADP) modelling framework, which is based on Markov Decision Processes (MDP) can be used to tackle the issue of *curses of dimensionality* associated with employing Dynamic Programming (DP) in large, multi-period stochastic optimization problems. In the next section, the basic idea behind ADP is explained.

3.3 Approximate Dynamic Programming (ADP)

The central idea of ADP is to use an approximation of the value function to make decisions. ADP steps forward in time and then updates the estimated value functions iteratively.

The Bellman equation (3.2.2) is often expressed in its expectational form (Equation (3.3.1)) when used in ADP.

$$V_t(S_t) = \max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \gamma \mathbb{E}^{\omega}\{V_{t+1}(S_{t+1}|S_t, x_t, \omega)\}\right) \forall t \in T \qquad (3.3.1)$$

It is often difficult to deal with an expectation operator within a maximum operator especially in the case of simulation. This problem can however be solved in ADP by using the post-decision state formulation of the Bellman equation.

Definition 3.3.1 The post-decision state S_t^x is the state immediately after decision x_t is taken and before the arrival of new information W_{t+1}.

Substituting Equation (3.3.3) into Equation (3.3.2) gives Equation (3.3.4).

$$V_{t-1}^x(S_{t-1}^x) = \mathbb{E}^{\omega}\{V_t(S_t|S_{t-1}^x, \omega)\} \forall t \in T \qquad (3.3.2)$$

$$V_t(S_t) = \max_{x_t \in \mathcal{X}_t} \left(C(S_t, x_t) + \gamma V_t^x(S_t^x)\right) \forall t \in T \qquad (3.3.3)$$

$$V_{t-1}^x(S_{t-1}^x) = \mathbb{E}^{\omega}\left\{\max_{x_t \in \mathcal{X}_t}(C(S_t, x_t) + \gamma V_t^x(S_t^x|S_{t-1}^x, \omega))\right\} \qquad (3.3.4)$$

According to Mes and Rivera (2016), solving Equation (3.3.3) as a deterministic optimization problem provides decisions that can be used to update the estimates $\overline{V}_{t-1}^{n-1}(S_{t-1}^x)$ of $V_{t-1}^x(S_{t-1}^x)$ iteratively over a number of iterations N, in which the random information ω are generated using Monte Carlo simulation.

The steps mentioned above is a simple description of the Approximate Value Iteration (AVI) algorithm. However, a known problem associated with AVI algorithms, which has been shown in some research papers such as Farias and Roy (2000) is their lack of convergence. Therefore, Approximate

Policy Iteration (API) Algorithm is employed in this thesis instead of the AVI algorithm because of its ability to offer convergence guarantees (Powell, 2011).

Whereas AVI algorithms aim at approximating the value functions associated with an unknown policy, Approximate policy iteration (API) starts with a chosen policy function, fixes an approximation architecture to the selected policy and finally constructs an updated policy at each iteration. A detailed explanation of our proposed API algorithm will be presented in Section 4.5.

3.3.2 Why ADP?

ADP is highly recommended and considered for stochastic problems with very large-scale instances such as the energy storage problems because of its ability to handle two of the three dimensionality issues.

Firstly, the construction of the post-decision state $S_t^{x,n}$ can be used to handle the problem of having a large outcome space. This is because the presence of the post-decision state eliminates the need to evaluate the Bellman's optimality equation for all possible sample realization outcomes of the exogenous information, $\omega \in \Omega$.

Secondly, the approximate value functions $\overline{V}_t^n(S_t^{x,n})$ that are learned during the iteration process can allow generalization across the states so that instead of learning the values of each state individually, the already visited states may be able to provide some information about the value of the states that are yet to be visited. This helps in dealing with the problem of the large state space.

CHAPTER FOUR

MATHEMATICAL MODELS AND API ALGORITHM

In this chapter, we describe the SNES problem, present and highlight the differences between an example mathematical model in literature and our proposed mathematical model and analyse the feasible decision space of our model. We then conclude this chapter with a detailed description of our novel Approximate Policy Iteration with Neural Network (APINN) algorithm.

4.1 SNES Problem Description and Notation

A household which has an energy storage system (battery) colocated with a renewable energy source like solar panels or wind farm has to ensure that their total electricity demand for a fixed number of time periods in the future is fully satisfied either from the solar panel, battery or the electricity spot market (national grid). Aside this, energy from the renewable energy source can also be used to directly charge the battery system. The battery's energy may be sold to the power grid at any given time to yield some revenue and electricity from the grid may be bought to replenish the battery at a cost. The energy left over in the battery at the end of each time period in the finite time horizon is assumed to be lost. At every time step, we must decide how to allocate the energy from these available sources to ensure the entire demand is met at a reduced electricity cost.

We formulate the problem as a finite-time horizon markov decision process. The decision epochs of the MDP is represented by a discrete time index, $t \in T$, where t could be measured in hours or days in our case. The objective is to find a policy that maximizes the expected profits over the finite time horizon.

4.1.1 The state of the system

- R_t: amount of energy in the battery at time t
- E_t: newly available energy from the renewable source at time t
- D_t: energy demand of the household (or some other type of energy sink) at time t
- C_t: buying price of electricity at time t
- P_t: sale price of electricity at time t

R_t represents the physical state since it is the main physical resource being managed. D_t, E_t, C_t and P_t are examples of information states from the definition given in Section 3.1.1. Hence, the state of our system is $S_t = (R_t, D_t, E_t, C_t, P_t)$. Since the information states are from external sources, they can be referred to as the exogenous information for our system. The exogenous information process is denoted by the vector $W_t = (\hat{D}_t, \hat{E}_t, \hat{C}_t, \hat{P}_t)$ where:

- $\hat{E}_t$ refers to the change in the renewable energy amount between times $t - \Delta t$ and t
- $\hat{D}_t$ refers to the change in the demand between times $t - \Delta t$ and t
- $\hat{C}_t$ refers to the change in the buying price between times $t - \Delta t$ and t
- $\hat{P}_t$ refers to the change in the selling price between times $t - \Delta t$ and t

W_t can be characterized by different stochastic and deterministic models and some of these models are given in Section 5.3.2.

Additionally, we provide the general assumptions which hold for both the example mathematical model in literature and our proposed mathematical model described in Sections 4.2 and 4.3 respectively.

Assumption 4.1.2 W_t is assumed to be $\mathcal{F}_t$-measurable hence can capture only the information that becomes available between times $t - \Delta t$ and t.

Assumption 4.1.3 The exogenous information process W_t is independent of the physical state R_t. In other words, the amount of energy in the battery at time t does not influence the process that generates the exogenous information.

Assumption 4.1.4 The grid is assumed to be an infinite source of power hence we do not place any constraint on the amount of energy that can be imported from or exported to the grid.

4.2 Sample Mathematical Model

The mathematical model described below is from Halman et al. (2018) and is similar to that used in Jiang et al. (2014) and Natarajan et al. (2014).

However, the buying and selling price on the spot market is assumed to be the same in Jiang et al. (2014), whereas the selling price is assumed to be always less than or equal to the buying price in Halman et al. (2018). Also, Jiang et al. (2014) did not consider the storage rent in their contribution function, but like Scott and Powell (2012) they included an additional revenue in every time period for satisfying the demand which according to Halman et al. (2018) could make the problem easy to solve as the value function becomes non-negative.

Representing the grid by letter G, battery by letter R, demand by D and the renewable energy source by E; the decision variables at each time t is denoted by the non-negative vector $x_t = (x_t^{GR}, x_t^{GD}, x_t^{EG}, x_t^{ER}, x_t^{ED}, x_t^{RG}, x_t^{RD})$, where x_t^{ij} indicates energy transferred from device i to device j at time t.

Figure 1 below shows a sketch of the main components of this system as well as the possible energy transfers that can occur in this system.

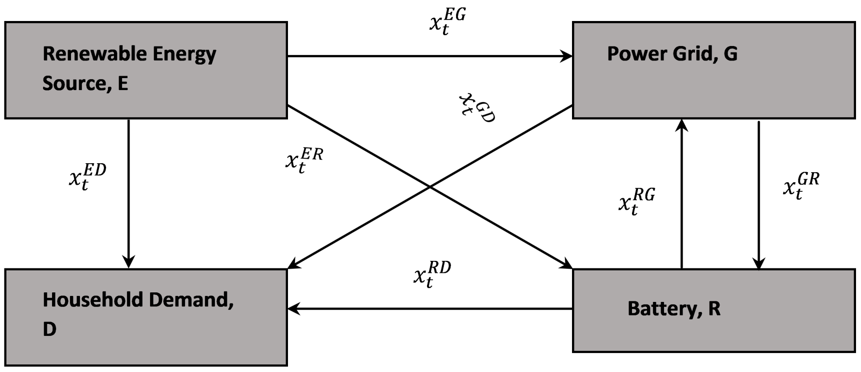

Figure 1: Main components of the Single-Node Energy Storage problem as illustrated by Halman et al. (2018).

The amount of energy in the battery at time t is denoted by R_t and the transition function used to determine the energy in the battery storage at the next time period R_{t+1} is given by:

$$R_{t+1} = R_t - x_t^{RD} - x_t^{RG} + (1-\eta^I)(x_t^{GR} + x_t^{ER}). \tag{4.2.1}$$

The decision vector x_t should satisfy the following constraints:

$$\begin{aligned} x_t^{ER} + x_t^{GR} &\leq min\{\gamma^I, R_{max} - R_t\} & (4.2.2)\\ x_t^{RD} + x_t^{RG} &\leq min\{\gamma^W, R_t\} & (4.2.3)\\ x_t^{EG} + x_t^{ER} + x_t^{ED} &\leq E_t & (4.2.4)\\ x_t^{ED} + (1-\eta^W)x_t^{RD} + x_t^{GD} &= D_t & (4.2.5)\\ x_t &\geq 0 & (4.2.6) \end{aligned}$$

Constraints (4.2.2) and (4.2.3) ensure that the injection or withdrawal amounts into or from the battery do not exceed the maximum charging and discharging rates respectively. With constraint (4.2.4), the maximum amount of energy used from the renewable resource is restricted by E_t. Constraint (4.2.5) guarantees that the entire demand is fully met and constraint (4.2.6) shows that the decision variables are non-negative continuous variables.

Assumption 4.2.1 The selling price P_t is always less than or equal to the buying price C_t ($P_t \leq C_t$).

The profit in each time period t given the state S_t, the decision vector x_t and the exogenous information W_t is:

$$g_t(R_t, x_t, W_t) = P_t((1-\eta^W)(x_t^{RG}) + x_t^{EG}) - C_t(x_t^{GR} + x_t^{GD}) - c^h R_{t+1} \tag{4.2.7}$$

The goal of this model is to determine the optimal policy that can attain the maximum profit over all the time periods in the given finite time horizon.

We will describe our proposed novel mathematical model and API approach and highlight the key differences between both mathematical models.

4.3 Proposed Mathematical Model

Contrary to the sample mathematical model given above, our decision variables do not represent the energy flows amongst the main components of the SNES problem. Instead, we are interested in determining a discrete non-negative integer valued vector $x_t : (x_t^s, x_t^b, x_t^r)$, where x_t^s, x_t^b, x_t^r indicates the amount of energy sold to the grid, purchased from the grid and stored in the battery respectively in time period t. We define our decision variables as integer variables since we consider energy as being measured in discrete units even though most papers define their decision variables as continuous variables for the sake of computational convenience (Jiang et al., 2014; Durante et al., 2017; Halman et al., 2018). Having three decision variables instead of seven variables as in the case of Halman et al. (2018)'s model reduces the dimensionality of the decision space and this may facilitate the determination of the optimal decisions within reasonable time.

The decision vector x_t should satisfy the following constraints:

$$x_t^b - x_t^r - x_t^s = D_t - E_t - x_{t-1}^r \tag{4.3.1}$$

$$x_t^r - x_{t-1}^r <= \gamma^I \tag{4.3.2}$$

$$x_{t-1}^r - x_t^r <= \gamma^W \tag{4.3.3}$$

$$x_t^r \leq R_{max} \tag{4.3.4}$$

$$x_t^b, x_t^s, x_t^r \geq 0 \text{ and integer} \tag{4.3.5}$$

Constraint (4.3.1) expresses the total energy balance and ensures that all the demand in each time period is fully satisfied. With constraints (4.3.2) and (4.3.3), the amount of energy injected into the battery or withdrawn from the battery will not exceed the maximum charging (γ^I) and discharging (γ^W) rates of the battery respectively. Also, constraint (4.3.4) guarantees that the amount of energy stored in the battery at any point in time t does not exceed the maximum amount of energy the battery can contain due to its size (R_{max}). Finally constraint (4.3.5) ensures that the decision variables are non-negative integers.

A major contribution of our proposed model based on an idea from Secomandi (2010) is that the charging and discharging losses (η^I, η^W) due to the transmission of energy into and from the battery are included in the contribution function as cost. This is a form of relaxation for our constraints which can help our model achieve very close-to-optimal solutions.

Assumption 4.3.1 The selling price P_t is always less than the buying price C_t ($P_t < C_t$).

The profit in each time period t given the pre-decision state R_t, decision vector x_t, exogenous information (D_t, E_t, C_t, P_t) and the post-decision state R_t^x is:

$$\begin{aligned} g_t(x_t, W_t) &= C_t D_t + P_t x_t^s - C_t x_t^b - c^h x_t^r - [S\eta^I (R_t^x - R_t)^+ + \\ &\quad S\eta^W (R_t^x - R_t)^-] - M(R_{max} - R_t^x)^+, \end{aligned} \tag{4.3.6}$$

where S is the cost incurred due to the loss of energy from transmitting energy into or from the battery, M represents a large number as often used in optimization literature, R_t is the initial battery level at the start of the time period and R_t^x is the remaining battery level at the end of the time period after the decision vector x_t has been applied.

The profit is calculated as the revenue generated from the energy sold minus the costs incurred from: buying energy; storing energy in the battery; and the transmission losses generated from injecting or withdrawing energy into or from the battery.

In any given time t the valuation of the loss S depends on whether the decision-maker decides to buy or sell. Buying and selling at the same time is sub-optimal since the selling price is always less than the buying price ($P_t < C_t$). We now have to consider how to value these transmission losses. One way is to value them at the buying price C_t in the case of buying energy to fill the battery and at the selling price P_t when energy is withdrawn from the battery to be sold. However, we valued all the transmission losses at the buying price C_t in our preliminary experiments and analysis. Like Scott and Powell (2012) and Jiang et al. (2014), an additional revenue is added to the profit for the satisfaction of demand at each time period to enable easier analysis and interpretation of our results and metrics from the experiments performed on the model. In our case, the additional revenue is valued at the buying price C_t.

To simplify the notation, we introduce two variables $N_t = D_t - E_t$ and $a = x_t^r - x_{t-1}^r$. Using equation Equation 4.3.1 and this notation,we get the equation $x_t^b - x_t^s = N_t + a$ which makes it easy to solve for the value of variables x_t^b and x_t^s since it is not optimal to buy and sell at the same time. The following definition will be useful.

$$p(x, W) = g(x, W) + c^h x \tag{4.3.7}$$

Note that with this notation the decision variable at any given time is the single-dimensional variable a which when positive means injection of energy into the battery and when negative means withdrawal of energy from the battery. One could therefore argue that the dimensionality of the decision space is further reduced to one as a result of this.

Following Secomandi (2010), at any decision time, the sets of feasible withdrawal and injection decisions, respectively, with current battery level x are defined as A^{FW} and $A^{FI} = [0, R_{max} - x]$. We denote the set of all feasible actions by $A^F(x)$. Since the charging and discharging capacities are inherently captured within these actions sets we drop the last term of the profit function. The profit function $p_t(a, W)$ can be rewritten as:

$$\begin{aligned}
p_t(a, W_t) &= -(C_tN_t + aC_t(1+\eta^I)) \quad \text{buy and inject} && (4.3.8)\\
&= -(C_tN_t + aC_t(1+\eta^W)) \quad \text{buy and withdraw} && (4.3.9)\\
&= -(P_tN_t + aP_t(1+\eta^I)) \quad \text{sell and inject} && (4.3.10)\\
&= -(P_tN_t + aP_t(1+\eta^W)) \quad \text{sell and withdraw} && (4.3.11)\\
&= -(P_tN_t) \quad \text{sell} && (4.3.12)\\
&= -(C_tN_t) \quad \text{buy} && (4.3.13)\\
&= -(aP_t(1+\eta^I)) \quad \text{inject} && (4.3.14)\\
&= -(aC_t(1+\eta^W)) \quad \text{withdraw} && (4.3.15)
\end{aligned}$$

An energy management policy can be obtained by solving this finite horizon MDP using the following dynamic programming recursion.

$$
\begin{aligned}
V_T(x^r_{T-1}, W_T) &= \max\nolimits_{a \in A^{FW}(x)} \min\{-(C_T N_T + a C_T(1+\eta^W)), \\
&\quad -(P_T N_T + a P_T(1+\eta^W))\} \\
V_t(x^r_{t-1}, W_t) &= \max\nolimits_{a \in A^F(x)} v_t(a, x^r_{t-1}, W_t), t \in \mathcal{T}, (x^r_{t-1}, W_t) \in \mathcal{X} \times \mathcal{W}_t, \\
v_t(a, x^r_{t-1}, W_t) &= p_t(a, W_t) - c^h x^r_t + \delta_t \mathbb{E}_t[V_{t+1}(x^r_{t-1} + a, \tilde{W}_{t+1})] \qquad (4.3.16)
\end{aligned}
$$

The formulation should be interpreted as: in the last stage ($t = T$) we can buy or sell depending on the net surplus energy but injection into the battery cannot take place whereas we have eight actions resulting from the Cartesian product of {sell, buy, neither sell nor buy} and {inject, withdraw, neither inject nor withdraw} in the remaining stages ($t < T$).

4.4 Analysis of the feasible decision space structure

This section analyses the feasible decision space structure of the policy that can be generated from the implementation of our proposed model. Although there are a number of studies in literature which focused on solving or approximately solving the DP, we are not aware of any study that does the aforementioned analysis except Secomandi (2010). Secomandi (2010) focused on the determination of the inventory-trading policy that aids the merchant in deciding whether to buy and inject into the warehouse or withdraw and sell to the wholesale market given constraints such as injection and withdrawal capacity limits, the current spot price and the inventory level (space of the warehouse). In our setting, the inventory levels can be likened to the minimum and maximum capacity of the battery (R_{min}, R_{max}) and the withdrawal and injection capacity limits are depicted by the maximum charging (γ^W) and discharging (γ^I) rates of the battery. We provide an extension to Secomandi (2010) in two ways.

1. We consider the demand and available energy production from the renewable energy source in each time period.
2. Buying and selling happen at different prices.

Secomandi (2010) showed that when γ^I and γ^W are (much) less than the maximum battery capacity, the optimal decision at each time period depends not only on the spot price but also on the initial battery level. This is a very important insight as it results in the feasible decision space having a specific structure which can be split into three phases (inject, do-nothing, withdraw) depending on the initial battery level. Departing from this structure and employing sub-optimal decisions can result in very low payoff. We therefore extend this same idea from Secomandi (2010) to describe the feasible decision space structure in our case of different prices and improved model formulation.

Observation 4.4.1 When buying and selling prices are equal, the optimal decision in a period only depends on the initial battery level, prices, injection and withdrawal rates but not on the demand and renewable energy source (solar or wind) profiles.

Proof. This statement means that changing the renewable energy source and demand levels in each period when the prices, battery level and injection and withdrawal rates are fixed will not shift the optimal decision from injection to withdrawal and vice-versa. To see this note that if injection is said to be the optimal decision in the current period for a given demand and solar or wind profile, the injection may be done for two reasons: to satisfy demand in the future or to sell in the future. Storing in the current period is equivalent to buying as both decisions do not earn revenue in that period since the buying and selling prices are equal. Therefore, irrespective of demand it is optimal to store as long as prices remain the same. □

This aligns with the results in (Halman et al., 2009, 2018) which showed that having the same buying and selling price makes the SNES problem much easier to solve compared to the use of different buying and selling prices. On the other hand, different demand patterns can result in different optimal strategies when buying and selling prices are not equal.

Observation 4.4.2 When buying and selling happen at different prices optimal decisions also depend on the renewable energy and demand profiles.

Proof. Consider for example the price profile given in Figure 2 where the blue dotted lines indicate the selling price (P_t) and black thick lines represent the buying price (C_t). Clearly, it is not optimal to buy in period one ($t = 1$) to sell in period three ($t = 3$) or in period two ($t = 2$) since the buying price in period one is larger than the selling price in period two and period three ($C_1 > P_2$ and P_3). Therefore, injecting is optimal at any initial battery level only if the net demand ($D_t - E_t$) in period three is positive (i.e. there is still some remaining demand to be satisfied). In this case, it may be optimal to buy and store in period one to satisfy the demand in period three depending on the injection and withdrawal rates. On the other hand, it may not be optimal to buy and inject in period one if the net demand ($D_t - E_t$) is non-positive (all the demand has been fully satisfied via the renewable energy source) in period three. This illustrates the complexity of our problem compared with Secomandi (2010). □

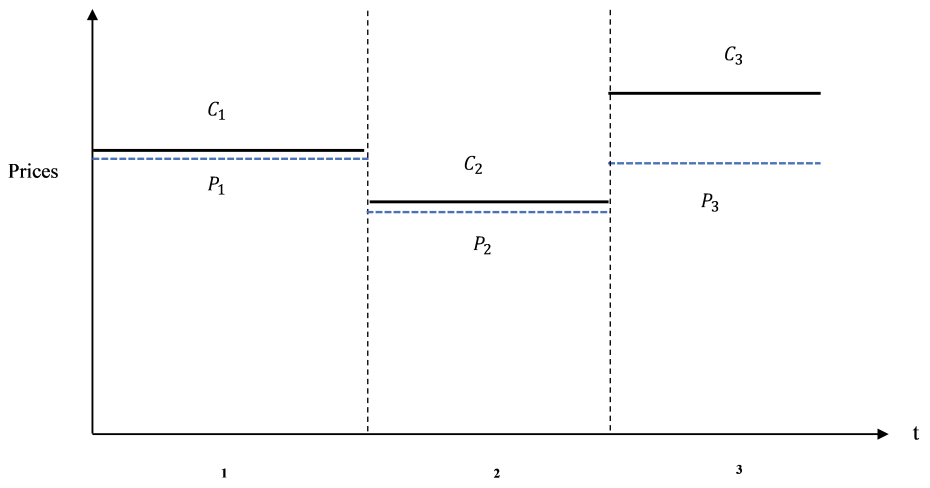

Figure 2: Example price profile for three periods

4.4.3 Negative surplus

In the case of negative surplus, that is when solar or wind energy plus energy in battery is less than demand ($D_t - E_t - R^x_{t-1} > 0$), the feasible decision space structure is similar to that in Secomandi (2010) with three phases: buy-and-inject, buy, buy-and-withdraw. As indicated in Figure 3, depending on the initial battery level and buying price the optimal decision varies. We highlight here that the injection and withdrawal rates also play a key role in determining the optimal decision out of the feasible decision space (Secomandi, 2010).

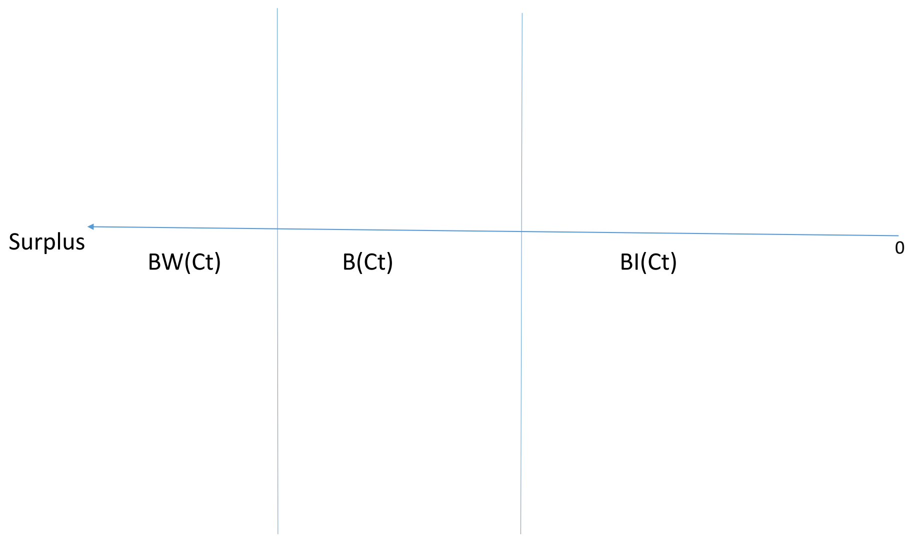

Figure 3: Illustration of the feasible decision space structure for a given stage and prices in the case of negative surplus

We discuss some scenarios under which each of these three phases of the feasible decision space structure is selected as the optimal decision.

Buy-and-inject

Energy is bought to satisfy the remaining demand and also top up the battery if it is not at its maximum capacity and the buying price is low. Note that the injection amount must be less than or equal to the maximum injection rate (γ^I) and also the maximum capacity of the battery (R_{max}) should not be exceeded.

Buy

This refers to the scenario where it is optimal to satisfy the remaining net demand without using the battery. This could happen if the battery is empty ($R_t = 0$) and the buying price is so high that buying more than what is needed to satisfy the remaining net demand as well as replenish the battery when $t < T$ will cause a lower payoff.

Buy-and-withdraw

This could happen when the amount of energy that must be withdrawn from a battery with enough stored energy to satisfy the remaining net demand exceeds the maximum withdrawal rate ($D_t - E_t > \gamma^W$) or when there is simply not enough storage in the battery to satisfy the remaining net demand ($D_t - E_t > R_t$).

4.4.4 Positive Surplus

In contrast to the negative surplus case, the structure of the feasible decision space in any given period with positive surplus is much more complicated and depends at a given time and given buying and selling prices, on both the initial battery level and the total net surplus ($D_t - E_t - R^x_{t-1}$) as shown in Figure 4.

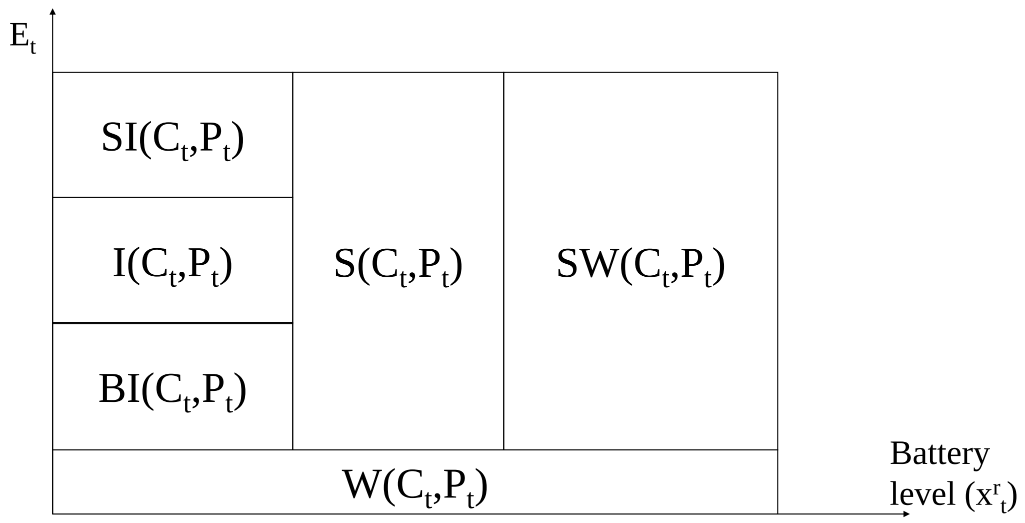

Figure 4: Illustration of the optimal policy structure for a given stage and prices in the case of positive surplus

We also discuss some scenarios under which each of these six phases of the feasible decision space structure is selected as the optimal decision.

Buy-and-inject

This can happen when the buying price is low and the total demand has been satisfied and there is still some storage capacity left in the battery which can be replenished ($R_t < R_{max}$). The injection amount however must be $\leq \gamma^I$.

Inject

This could happen when there is a high quantity of the available energy from the renewable energy source (E_t) such that it can satisfy the demand on its own with some surplus even remaining ($E_t > D_t$). In this case however, it is more optimal to store all this surplus energy from the renewable energy source due to the low selling price being offered on the spot market and also the surplus energy amount from the renewable energy source is $\leq \gamma^I$.

Sell-and-inject

In this case, a portion of the surplus energy from the renewable energy source E_t after all the demand has been satisfied (described above) is stored in the battery and the other portion sold to the grid. This could be as a result of the surplus energy amount being $\geq \gamma^I$ and since the injection amount must be $\leq \gamma^I$, not all of it can be added to the battery.

Sell

Over here, either the selling price on the spot market yields a better payoff compared to storing the excess energy from the renewable energy source E_t or the battery is at its maximum capacity ($R_t = R_{max}$).

Sell-and-withdraw

When the total demand has been satisfied and there is still some available storage in the battery, we can withdraw and sell to the grid provided the withdrawal amount $\leq \gamma^W$ and the selling price at the time results in a higher payoff or it is the last time period ($t = T$) and we do not want to have energy in the battery since this will be lost and valued at an extra cost.

Withdraw

In this case, we may require an amount of energy from the less than or equal to γ^W in addition to the renewable energy to satisfy the entire demand.

4.4.5 Zero Surplus

For a zero surplus, solar or wind energy plus energy in the battery equals demand ($D_t - E_t - R^x_{t-1} = 0$). One scenario to consider is when the battery is empty and the amount of energy in the renewable resource is enough to handle the total demand. It may therefore be optimal to **do-nothing** if the buying price on the market is quite high and will result in a low payoff or it is the last time period ($t = T$) and there is no need to replenish the battery.

Another optimal decision that can be considered is to **buy-and-inject** for all other time periods apart from the last time period provided the injection limit γ^I is not exceeded and the buying price is quite low on the market.

4.5 Approximate policy iteration with neural networks (APINN)

Policy Iteration (PI) is a well-known algorithmic technique used to solve stochastic dynamic programs (DP). There are several results from academic literature that exist on the convergence of PI for DPs (Bertsekas and Tsitsiklis, 1996; Bertsekas, 2018). A policy is a mapping of the state space to the action space and it is said to be feasible if it satisfies all the constraints for the given problem.

PI has two main steps, *evaluation* and *improvement*. The idea is to start with a feasible policy and iteratively improve it after evaluating at each iteration. It is shown that the policy converges to the optimal policy under certain mild assumptions. Exact policy iteration requires evaluating the entire state space multiple, if not many, times which is almost an impossible task to achieve even for moderately large state spaces. This made many researchers focus on approximate ways of implementing policy iteration while still maintaining convergence. An extensive survey about Approximate Policy Iteration (API) is provided in Bertsekas (2011).

One of the ways to avoid a full state space evaluation is to use simulation. We took this approach in our study and used monte-carlo simulation to generate a fixed number of sample states after which we applied a machine learning model to assist in the learning of the value function. The machine learning model takes as input the current state and actions and predicts the future profit.

Some studies refer to the machine learning framework used to learn an approximate value function as approximation architectures (Jiang et al., 2014). Many approximation schemes have been proposed in literature and have been discussed in Section 2.3.1. Most of these works use (some sort of) linear approximation architectures such as the expression of the value function as a linear function of known set of features which are also called the basis functions. The policy evaluation data from simulation is then used as input to fit coefficients for these features (Jiang et al., 2014; Jiang and Powell, 2015) .

Bertsekas (2018) proposed the idea of using learning algorithms such as neural networks to learn features in cases where they are not already known. He also argued that the value function may be approximated more accurately by non-linear functions of the features. Therefore in this study, instead of using linear architectures, we employed neural networks for the value function approximation due its ability to capture many kinds of dynamic or non-linear relationships and patterns in the underlying data. Aside this, neural networks are known to have superior predictive properties, better performance, and a higher robustness to over-fitting.

We give a formal description of our algorithm in Algorithm 1, where $\mathcal{NN}$ in Step 5 stands for neural networks. The basic idea is that for each improvement iteration n, the given policy is evaluated for the number of samples Y over the entire finite horizon ($t \in T$) and the result is used to train the neural network model. After the evaluation process, the policy is then improved at each time step t for a set of randomly generated exogenous information, each possible value of the initial battery level in the set $[R_{min}, R_{max}]$ and a set of feasible storage actions by using Algorithm 2 and the trained neural network model to select the optimal storage action (x_t^r) and its corresponding decisions (x_t^b, x_t^s) that give the maximum total profit.

Using machine learning within API is not new and has already been explored before, owing to the fact that learning high dimensional functions is central to machine learning theory. In Jiang et al. (2014) several machine learning techniques like support vector machines are used for approximating the value function shown in Step 5. Original ideas of using neural networks within approximate policy evaluation was proposed within reinforcement literature many years back in the work of Tesauro (Tesauro, 2002).

More recently, Bertsekas (2018) proposed API with neural nets to approximate cost function of policy evaluation and learning linearly linked features using feature based approximation. Our work is based on exactly similar ideas but we do not use any feature based combination to approximate the value function. Instead, we use deep neural networks as black box models to predict function values. Bertsekas (2018) pointed out that using neural networks which does not assume a linear combination of features within API requires the development of models that enable dimensionality reduction. This was achieved in our model as our battery injection and withdrawal losses have been valued in monetary terms and moved to the objective function ad this has enabled us to express the decision vector in just one dimension as the amount of energy stored in the battery in each time period (x_t^r). The policy improvement stage is much simpler since we now have a single dimensional optimization problem rather than a non-linear multi-dimensional problem which is much harder to handle. We can therefore easily derive the values for buying and selling decisions (x_t^b, x_t^s) for a fixed value of storage x_t^r in a time period.

Observation 4.5.1 For any t either $x_t^b = 0$ or $x_t^s = 0$.

Proof. The statement has to be true due to the assumption $P_t < C_t$ which makes it sub-optimal to buy and sell at the same time. $\square$

Lemma 4.5.2 Given x_t^r Algorithm 2 computes the optimal values of x_t^b and x_t^s.

Algorithm 1 Approximate Policy Iteration with Neural Networks (APINN)
(Inputs: sample size Y, improvement iterations N, policy π, neural network architecture $\mathcal{NN}$)

```
Step 0: Set initial policy $\pi^0$, set $n = 1$
Step 1: Set $y = 1$
Step 2: Select initial battery level $x^r_{t-1}$
for $i = 1$ to $T - 1$ do
    Step 3a: sample $W_t$
    Step 3b: evaluation Apply policy:
        $(x^r_t, x^b_t, x^s_t) = \pi_t(x^r_{t-1}, W_t), C^y_t = C(x^r_{t-1}, W_t),$    (4.5.1)
end for
Step 4: if $y < Y$, $y = Y + 1$ and return to Step 1.
Step 5: Approximate value function:
    $\tilde{V}^{x,n-1} = \mathcal{NN}^n(x^r_{t-1}, x^r_t, x^b_t, x^s_t, C^y_t)$    (4.5.2)
Step 6: Improvement:
    $\pi^n_t(x^r_{t-1}, W_t) = \underset{x^r_t}{argmax}[C(x^r_{t-1}, W_t) + \tilde{V}^{x,n-1}]$    (4.5.3)
Step 7: if $n < N$, $n = n + 1$ goto Step 1
```

Algorithm 2 Compute-Actions

```
Input: $E_t$, $D_t$, $x_t^r$, $x_{t-1}^r$
if $t < T$ then
  if $x_t^r > x_{t-1}^r$ then
    if $E_t - D_t \geq x_t^r - x_{t-1}^r$ then
      $x_t^s = E_t - D_t - x_t^r + x_{t-1}^r$; $x_t^b = 0$
    end if
    if $E_t - D_t < x_t^r - x_{t-1}^r$ then
      $x_t^b = D_t - E_t + x_t^r - x_{t-1}^r$; $x_t^s = 0$
    end if
  end if
  if $x_t^r < x_{t-1}^r$ then
    if $E_t - D_t \geq x_t^r - x_{t-1}^r$ then
      $x_t^s = E_t - D_t + x_t^r - x_{t-1}^r$; $x_t^b = 0$
    end if
    if $E_t - D_t < x_t^r - x_{t-1}^r$ then
      $x_t^b = D_t - E_t + x_t^r - x_{t-1}^r$; $x_t^s = 0$
    end if
  end if
  if $x_t^r = x_{t-1}^r$ then
    if $E_t - D_t \geq 0$ then
      $x_t^s = E_t - D_t$; $x_t^b = 0$
    end if
    if $E_t - D_t < 0$ then
      $x_t^b = D_t - E_t$; $x_t^s = 0$
    end if
  end if
else
  if $E_t - D_t \geq x_t^r - x_{t-1}^r$ then
    $x_t^s = E_t - D_t - x_t^r + x_{t-1}^r$; $x_t^b = 0$
  end if
  if $E_t - D_t < x_t^r - x_{t-1}^r$ then
    $x_t^b = D_t - E_t + x_t^r - x_{t-1}^r$; $x_t^s = 0$
  end if
end if
```

CHAPTER FIVE

EXPERIMENTS, RESULTS AND DISCUSSIONS

This section is organized as follows: we first introduce the Integer Program (IP) model that provides the optimal solution for the deterministic benchmark problems used to test the performance of the proposed APINN algorithm. The parameters selected for the proposed mathematical model, APINN algorithm and neural network are then discussed and the numerical results and findings are presented and analysed to conclude this chapter.

5.1 Integer Program (IP) Formulation

Studies such as Salas and Powell (2013) and Jiang et al. (2014) assessed the performance of their proposed API algorithm by comparing the policies generated from their algorithm to the optimal policies obtained from the exact solution of some designed benchmark problems (either deterministic or stochastic) via linear programming (Salas and Powell, 2013) or backward dynamic programming (Jiang et al., 2014). In our case, we focused on deterministic benchmark problems (i.e. demand, wind and prices in each period are known) which can be seen as a special case of the lot sizing problem with losses and bounded inventory. Lot sizing problems are very well studied in operations research literature including deterministic and stochastic variants. We developed an integer programming (IP) model to obtain the optimal solution for these deterministic benchmark problem instances based on ideas gathered from these lot sizing research papers. The IP formulation is given below:

Problem Parameters

Parameter Notation	Parameter Description
T	number of time periods
c^h	storage rent, in \$ per MWh per time step
γ^I	maximum charging rate of the battery
γ^W	maximum discharging rate of the battery
C_t	buying price of electricity at time t
P_t	sale price of electricity at time t
η^I	% of transmitted energy lost due to battery charging
η^W	% of transmitted energy lost due to battery discharging

Let: $\mathcal{A}$ ={buy-inject, sell-inject, buy-withdraw, sell-withdraw, buy, sell, inject, withdraw, do-nothing}; $\mathcal{II}$ = {buy-inject, sell-inject, inject}; $\mathcal{WW}$ = {buy-withdraw, sell-withdraw, withdraw}.

We have the following variables for each time period t:

$$
\begin{aligned}
z_t^j &= 1 \text{ if } j \text{ is true, where } j \in \mathcal{A}, \\
&= 0 \text{ otherwise.} \\
w_t^j &= \text{total injection into battery when } j \text{ is one of } \mathcal{II}, \\
&= \text{total withdrawal from battery when } j \text{ is one of } \mathcal{WW} \\
x_t^j &= \text{total units stored } (j = r) \text{ in period } t \\
&= \text{total units sold } (j = s) \text{ in period } t \\
&= \text{total units bought } (j = b) \text{ in period } t \\
\mathrm{M} &= \text{large positive number (often used in most integer programming formulations)}
\end{aligned}
$$

LS-L-BI

$$\sum_{t=1}^{T} C_t D_t + \max \sum_{t=1}^{T} \left(P_t x_t^s - C_t x_t^b - c^h x_t^r - \eta^I \sum_{i \in \mathcal{II}} C_t w^i - \eta^W \sum_{i \in \mathcal{WW}} C_t w^i \right)$$

subject to:

$$\sum_{i \in \mathcal{A}} z_t^i = 1 \;\; \forall t \tag{5.1.1}$$

$$\gamma^I \sum_{i \in \mathcal{II}} z_t^i \geq x_t^r - x_{t-1}^r \;\; \forall t \tag{5.1.2}$$

$$\gamma^W \sum_{i \in \mathcal{WW}} z_t^i \geq -(x_t^r - x_{t-1}^r) \;\; \forall t \tag{5.1.3}$$

$$w_t^j \leq M z_t^j \;\; \forall j \in \mathcal{WW} \cup \mathcal{II} \text{ and } \forall t \tag{5.1.4}$$

$$\sum_{i \in \mathcal{II}} w_t^i \geq x_t^r - x_{t-1}^r \;\; \forall t \tag{5.1.5}$$

$$\sum_{i \in \mathcal{WW}} w_t^i \geq -(x_t^r - x_{t-1}^r) \;\; \forall t \tag{5.1.6}$$

$$x_t^b - x_t^r - x_t^s = D_t - E_t - x_{t-1}^r \;\; \forall t \tag{5.1.7}$$

$$Z \in \{0,1\}, X, W \geq 0 \text{ and integer} \tag{5.1.8}$$

Lemma 5.1.1 LS-L-BI is a valid IP formulation for the deterministic variant of the SNES problem.

Proof. The additional revenue from satisfying the total demand ($\sum_{t=1}^{T} C_t D_t$) is added to the objective function to simplify the analysis of our performance metrics. The objective function includes the revenue generated from the energy sold minus the costs incurred from: buying energy; storing energy in the battery; and the transmission losses generated from injecting and withdrawing energy into/from the battery. The aim is to find the the optimal amount of energy bought, sold and stored in each time period that gives the maximum contribution.

Constraints (5.1.1) and (5.1.4) ensure that exactly one type of decision is selected from the decision space $\mathcal{A}$ at each time t. Withdrawal and injection limits are respected in each time period as a result of constraints (5.1.2)-(5.1.3) and (5.1.5)-(5.1.6). The entire demand for each time period t is satisfied and the energy flow is balanced due to constraint (5.1.7). As shown in (5.1.8), the decision variables are non-negative integers whereas the z-variables are binary.

□

5.2 Neural Network Parameters

We used the keras deep learning library in python to implement the neural network model. The data used for training the neural network consists of six columns, with the first five columns (time (t), previous battery level (x^r_{t-1}), energy bought (x^b_t), energy sold (x^s_t) and energy stored (x^r_t) as the inputs and the last column (future contribution ($\overline{V}_t$)) as the output which must be predicted. The data was then split with 70% of it as the training samples and 30% as the test samples. We employed a feed forward neural network with 10 nodes in each of the input layers, 10 nodes in the second layer and a single node in the output layer for the predictions. The nodes mentioned above are of a dense layer type, i.e, all the nodes in the previous layer are connected to the nodes in the current layer. We also included two dropout layers set to 0.2 that randomly select neurons to be ignored during training. This along with the L2 regularization with weight penalties of 0.01 helped in the reduction of over-fitting and generalization errors (Srivastava et al., 2014).

Each input was multiplied with its own assigned relative weight and summed together. A bias was then added to the summation and the result was transformed via the hyperbolic tangent ($tanh$) activation function to produce the predictions. We used the mean squared logarithmic error as the loss function to determine the errors between the predicted and actual output value. For accurate predictions, the mean squared logarithmic error must be minimized via back propagation which ensures that the current error is propagated to the previous layer and is used to modify the network's weights and bias through an optimization function. This cycle is repeated until the minima of the loss function is reached and the error for the output node and the entire neural network is minimized to its lowest possible value.

In our study, the Adam optimization algorithm was chosen as our optimizer because of its easy implementation, computational efficiency, little memory requirement and intuitive hyper-parameters that require little tuning (Kingma and Ba, 2017). Empirical results also demonstrate that Adam works well in practice and compares favourably to other stochastic optimization methods. We used the default settings for the Adam optimizer suggested in the keras documentation for training our model.

We also implemented early stopping as another mechanism to prevent over-fitting. The training sample was further split, with 80% as the training set, and 20% as the validation set. The validation samples are predicted and the validation loss is computed after each epoch (one pass across the samples in the training set). This process is continued until the validation loss does not decrease for the patience period. The training is then stopped and the weights of the model with the lowest validation loss is selected (Yuan et al., 2007). We chose a batch size of 100, 15 epochs and an early stopping patience of 5.

5.3 Experimental Design

5.3.1 Battery Storage Parameters

Parameter Name	Parameter Notation	Value
Minimum battery capacity	R_{min}	0
Maximum battery capacity	R_{max}	30
Maximum charging rate of the battery	γ^I	6
Maximum discharging rate of the battery	γ^W	3
% of transmitted energy lost due to battery charging	η^I	5 %
% of transmitted energy lost due to battery discharging	η^W	5 %
Storage rent	c^h	0.0005

5.3.2 Exogenous Information Parameters

Parameter Name	Parameter Notation	Value
Minimum renewable energy source	E_{min}	1
Maximum renewable energy source	E_{max}	7
Minimum demand	D_{min}	1
Maximum demand	D_{max}	15
Minimum buying price	C_{min}	3
Maximum buying price	C_{max}	13
Minimum selling price	P_{min}	2
Maximum selling price	P_{max}	12

As mentioned in Section 4.1.1, various stochastic and deterministic models can be used to describe the exogenous information process. Before we show the models used in our case for the four exogenous information processes, the discrete uniform distribution and the discrete pseudonormal distribution described in Jiang et al. (2014); Salas and Powell (2013) are presented since they are the main probability distributions used in our sampling process.

Discrete Uniform Distribution

The probability mass function for the Discrete Uniform Distribution is:

$$u_X(x) = \frac{\Delta X}{b - a + \Delta X} \forall x \in \mathcal{X} \tag{5.3.1}$$

where each element in $\mathcal{X} = \{\ a, a + \Delta X, a + 2\Delta X, ..., b - \Delta X, b\}$ has the same probability of occurring.

Discrete Pseudonormal Distribution

Even though it is standard practice to denote the pseudonormal distributions as $\bar{X} \sim \mathcal{PN}(\mu, \sigma^2, a, b, \Delta)$ since it is characterized by these five parameters (Jiang et al., 2014), we provide the same simplified notation as Salas and Powell (2013) which is $\bar{X} \sim \mathcal{N}(\mu_X, \sigma_X^2)$. The support of X is defined as $\mathcal{X} = \{\ a, a + \Delta X, a + 2\Delta X, ..., b - \Delta X, b\}$. For $x_i \in \mathcal{X}$, the probability mass function is:

$$\mathbf{P}(X = x_i) = \frac{f_X(x_i; \mu_X, \sigma_X^2)}{\sum_{x_j \in \mathcal{X}} f_X(x_j; \mu_X, \sigma_X^2)}$$

where $f_X(x_i; \mu_X, \sigma_X^2)$ is the normal probability density function with mean μ_X and variance σ_X^2.

Demand process

Similar to Salas and Powell (2013) and Jiang et al. (2014), the demand is assumed to be deterministic and is described by a sinusoidal distribution to depict the seasonality structure of energy demand.

$$\hat{D}_t = \lfloor 3 - 4sin(\frac{2\pi(t)}{T}) \rfloor + \epsilon_t^D \tag{5.3.2}$$

where ϵ_t^D is pseudonormally distributed $\mathcal{N}(0, 2^2)$ and discretized over the set {0, ±1, ±2} and the demand sampling support is [1, 15]. The demand at time t is then selected using:

$$D_t = \min\{\max\{\hat{D}_t, D_{min}\}, D_{max}\} \tag{5.3.3}$$

Renewable energy source process

First-order Markov processes such as an order-1 autoregressive process (Löhndorf and Minner, 2010; Zhou et al., 2018) or a first-order Markov

Chain (Cheng et al., 2018; Jiang et al., 2014; Salas and Powell, 2013) can be used for modelling the renewable energy profile. The latter is employed in this study. Therefore, the renewable energy for the next time period is given by,

$$E_{t+1} = \min\{\max\{E_t + \epsilon^E_{t+1}, E_{min}\}, E_{max}\} \tag{5.3.4}$$

where ϵ^E_t are independent and identical random variables that can be either uniformly distributed over the set $\{0, \pm 1\}$ or pseudonormally distributed and discretized over the set $\{0, \pm 1, \pm 2, ..., \pm 5\}$ with the sampling support as [1,7].

Price process

The same models are considered for the buying price C_t and selling prices P_t. The main difference is the values of the sampling support used. In this thesis, first-order markov chain with jumps which enable the simulation of price spikes are used to model the prices. This model has also been used in Salas and Powell (2013) and Jiang et al. (2014). Other models which are not considered in this study are first-order markov chain (Tseng and Barz, 2002; Mokrain and Stephen, 2006; Jiang et al., 2014; Salas and Powell, 2013) and an inter-temporal independence model in conjunction with a lognormal distribution (Xiaomin et al., 2014).

The sampling support for the buying price C_t is [3, 13] and that of the selling price P_t is [2, 12]. Therefore, the prices at the next time period is given by:

$$C_{t+1} = \min\{\max\{C_t + \epsilon^C_{t+1} + \mathbf{1}_{\{}u_{t+1} \leq p_{\}}\epsilon^{CJ}_{t+1}, C_{min\}, C_{max}\}} \tag{5.3.5}$$

$$P_{t+1} = \min\{\max\{P_t + \epsilon^P_{t+1} + \mathbf{1}_{\{}u_{t+1} \leq p_{\}}\epsilon^{PJ}_{t+1}, P_{min\}, P_{max}\}} \tag{5.3.6}$$

ϵ_t^C and ϵ_t^P are pseudonormally distributed and discretized over the set {0, ±1, ±2, ..., ±8 }; ϵ_t^{CJ} and ϵ_t^{PJ} are pseudonormally distributed and discretized over the set {0, ±1, ±2, ..., ±40 }; u_t is uniformly distributed $\mathcal{U}(0,1)$ with $p = 0.031$.

We designed five different uniform and pseudonormal distributions to generate the samples for our exogenous information during the experimentation of our APINN algorithm. The demand is sampled from the deterministic sinusoidal distribution described above. The sampling process used for renewable energy source, buying price and selling price for these five experimental data classes are summarized in Table 5 .1 below:

Data Class	ϵ_t^E	C_t/P_t Processes	$\epsilon_t^C/\epsilon_t^P$
S1	$\mathcal{U}(-1,1)$	Markov Chain with jumps	$\mathcal{PN}(0,0.5^2)$
S2	$\mathcal{U}(-1,1)$	Markov Chain with jumps	$\mathcal{PN}(0,1.0^2)$
S3	$\mathcal{U}(-1,1)$	Markov Chain with jumps	$\mathcal{PN}(0,2.5^2)$
S4	$\mathcal{U}(-1,1)$	Markov Chain with jumps	$\mathcal{PN}(0,5.0^2)$
S5	$\mathcal{PN}(0,0.5^2)$	Markov Chain with jumps	$\mathcal{PN}(0,5.0^2)$

Table 5 .1: Data Classes

In order to define the state space, the state variables are discretized using step size $\Delta = 1$ and their minimum and maximum values. Therefore the state space $\mathcal{S} := [R_{min}, R_{max}] \times \Omega_t \times T$, where $\Omega_t = [D_{min}, D_{max}] \times [E_{min}, E_{max}] \times [C_{min}, C_{max}] \times [P_{min}, P_{max}]$, denotes the set of possible realizations of W_t. Hence $\Omega_t = 15488$ and $\mathcal{S} = 480128 \times T$.

5.3.3 Initial policy

Policy iteration starts with an initial policy and improves it until it converges to the optimal policy. In our initial policy (also known as the *naive policy*), energy is bought to satisfy the demand; the renewable energy is sold and the previous battery storage amount is also stored in the current time period. However in the last time period, the entire amount of energy in the battery is withdrawn and sold in addition to the renewable energy source if the previous battery level is less than the maximum discharging rate (γ^W) whereas the energy sold equals the renewable energy source plus γ^W when the previous battery level is greater than γ^W. The naive policy is outlined in Algorithm 3.

Algorithm 3 Naive Policy

```
if $t < T$ then
    $x_t^s = E_t$; $x_t^b = D_t$; $x_t^r = x_{t-1}^r$
else
    if $x_{t-1}^r <= \gamma^W$ then
        $x_t^b = D_t$; $x_t^s = E_t + x_{t-1}^r$; $x_t^r = 0$
    else
        $x_t^b = D_t$; $x_t^s = E_t + \gamma^W$; $x_t^r = x_{t-1}^r - \gamma^W$
    end if
end if
Output: $x_t^b, x_t^s, x_t^r$
```

The reason for this choice of naive policy is to see if our APINN algorithm can start learning even with a very bad policy. We acknowledge that starting with a better policy can result in a better performance, but chose to test our proposed algorithm with this initial bad naive policy since this is the first time neural network is being used within our APINN algorithm.

5.3.4 APINN Algorithm Parameters

Parameter Name	Parameter Notation	Value
Sample size	Y	3000
Number of improvement iterations	N	50
Number of time periods	T	10 and 25
Time step	Δt	1

The sample size and number of improvement iterations influence the computation times and solution quality of the proposed API algorithm. Even though approximate policy iteration is known to require a few number of improvement iterations to reach convergence, we chose to run the proposed APINN algorithm for 50 iteration steps to investigate the margin of improvement from one iteration to another, in steps of 10. The algorithm was ran for each of the 5 data classes described in Table 5 .1 and for two sets of time periods (T = 10 and T = 25) and the policy generated after each improvement step was stored for further analysis.

Recall that neural networks typically require large amount of data to perform well and provide more accurate predictions. We realised that setting the sample size Y to smaller values such as 1000 did not allow any injection decisions to be taken in the policies generated by the proposed algorithm even in situations where it was optimal to replenish the battery. As such we selected Y as 3000 for all the experiments. Note that the sample size of the data used to train the neural network model during the policy evaluation stage at each time t is given by $Y \times T$. At each time t, the sample size of the data used to train the NN in the case of $T = 10$ is 30000 and when $T = 25$ is 75000. This data was then split into the training, test and validation sets based on the percentages given in Section 5.2. Also, we randomly selected 0.1% of this sample size as the exogenous states to be visited during the improvement stage. During the improvement step, at each time step and for each of the states in the randomly selected exogenous information and each battery level R_t in $[R_{min}, R_{max}]$, we looped through a set of feasible storage amounts and selected the one that satisfied all the constraints of the problem and gave the maximum contribution as the amount of energy we decide to store (x_t^r) in that time t. Recall that with our proposed mathematical model in Section 4.3, knowing the amount to store (x_t^r) in each time period can help to easily determine the other decision variables (x_t^b, x_t^s; see Equation (4.3.7)).

In the case of $T = 10$, at each time period t, the size of the state space used for the improvement process is 930 ($0.1\% \times 30000 \times 31$) whereas for $T = 25$ is 2325 ($0.1\% \times 75000 \times 31$). Note that the value 31 used here is the state space of the battery level $[R_{min}, R_{max}]$ with Δ R = 1. One may argue that 0.1% of the sample size may not be a suitable representation of the entire sample space. However, this value was chosen so as to further speed up the improvement process even though we acknowledge that visiting more states may provide better estimates to determine the optimal decisions.

5.3.5 Deterministic benchmark problems

For our deterministic benchmark problems, 300 instances were generated for each of the 5 experimental data classes described in Table 5 .1. An instance includes a sample path for the exogenous information process (D_t, E_t, C_t, P_t) for each time period in the finite-time horizon ($\forall t \in T$).

Assumption 5.3.6 For the deterministic benchmark problems, the battery is empty at the beginning of the first time period in the finite horizon.

The deterministic optimal policy value for each of these 300 instances was computed by implementing the IP model given in Section 5.1 with the IBM CPLEX optimization solver. During the experimentation of the proposed algorithm, the policy generated after each improvement step was stored and used to determine the algorithm's payoff for these 300 instances. If the sample path for an instance can be found in the generated algorithm policy, the corresponding actions were retrieved and the profit was computed using the contribution function described in **Algorithm 1**. Otherwise, the naive policy was applied to determine its contribution. The sum of the profit for all the sample paths in an instance is the optimal policy value for that instance.

5.3.7 Algorithmic performance metrics

The two metrics used to evaluate the performance of the policies generated from the proposed algorithm against the IP optimal solution for the deterministic benchmark instances are relative regret and % optimality. These metrics were also employed in Salas and Powell (2013) and Jiang et al. (2014) for the analysis of their results.

$$RelativeRegret = \frac{\text{Optimal Profit} - \text{APINN Profit}}{\text{Optimal Profit}} \times 100, \tag{5.3.7}$$

$$\%Optimality = \frac{\text{APINN Profit}}{\text{Optimal Profit}} \times 100. \tag{5.3.8}$$

Note that relative regret plus % optimality equal 100%.

The % optimality (or relative regret) value of the policy under consideration is the average % optimality (or relative regret) over the 300 deterministic generated for that policy's data class.

5.4 Numerical Findings

We first present the computation times of our APINN algorithm in Table 5 .2.

	N = 10	50
25	0.75	5
10	0.55	3

Table 5 .2: Computation times in hours

Computation times are dominated by the neural network prediction times in the improvement stage. This is part of the reason for only using a sample of the total state space at each improvement stage. However, the improvement step can be sped up considerably by:

- efficiently using the structure of the objective function (contribution function + future value) such that the future value predicted using the keras model does not depend on the prices.
- computing the objective function for all actions simultaneously rather than through a for loop. This is referred to as *parallelizing* and it depends on the configuration of the GPU used to run the experiments.

Training the neural network model can also be time intensive, especially, when scaling up to hundreds of time periods. The mean-square-log-error is approximately 1.5 in all rounds. Note that since we did not explore the full state space when applying our policy, the naive policy is employed when we encounter a state which has not yet been visited in the improvement stage.

We present the summary of the results for 10 and 25 time periods in Table 5 .3 and Table 5 .4 respectively. In both cases, number of improvement iterations $N > 10$ seem to be sufficient with the performance only improving marginally from $N = 20$ onwards. The performance of the algorithm is identical over all instances, with the algorithm always obtaining a solution which is at least better than 60% optimality. Even though an average performance slightly better that 80% optimality may seem discouraging compared to the results from previous work, we must note that our problem is more general and difficult to solve because of the discreteness of our decision variables and the use of different buying and selling prices. In addition to this, we start with a very bad naive policy and our neural network parameters need more fine-tuning to provide better predictions. Compared to the T = 10 case, there is only a small dip in the performance when T = 25 and this gives an indication of promise that our approach holds and could provide better results with a more exhaustive tuning of the neural network parameters. In Tables 5 .3 and 5 .4, we can observe less variance across the instances in each problem data class which suggests that the algorithm is consistent.

Data Class	Metrics	10	20	30	40	50
S1	Relative Regret	27.27%	17.69%	17.64%	17.65%	17.64%
	Opt Gap	72.73%	82.31%	82.36%	82.35%	82.36%
	Worst case	27.97%	56.67%	56.67%	56.67%	56.67%
	Best case	98.57%	99.24%	99.24%	99.24%	99.24%
	Std. deviation	0.160	0.065	0.064	0.064	0.064
S2	Relative Regret	27.49%	17.11%	17.05%	17.03%	16.97%
	Opt Gap	72.51%	82.89%	82.95%	82.97%	83.03%
	Worst case	19.93%	62.52%	62.52%	62.52%	61.77%
	Best case	98.28%	98.37%	98.37%	98.37%	98.37%
	Std. deviation	0.167	0.067	0.066	0.066	0.067
S3	Relative Regret	26.60%	16.81%	16.82%	16.82%	16.83%
	Opt Gap	73.40%	83.19%	83.18%	83.18%	83.17%
	Worst case	27.70%	60.25%	60.25%	60.25%	60.25%
	Best case	100.00%	100.00%	100.00%	100.00%	100.00%
	Std. deviation	0.155	0.066	0.066	0.066	0.066
S4	Relative Regret	27.84%	17.57%	17.59%	17.59%	17.58%
	Opt Gap	72.16%	82.43%	82.41%	82.41%	82.42%
	Worst case	27.74%	57.42%	57.42%	57.42%	57.42%
	Best case	98.91%	98.91%	98.91%	98.91%	98.91%
	Std. deviation	0.163	0.066	0.066	0.066	0.066
S5	Relative Regret	28.67%	17.88%	17.88%	17.88%	17.88%
	Opt Gap	71.33%	82.12%	82.12%	82.12%	82.12%
	Worst case	31.21%	59.71%	59.71%	59.71%	59.71%
	Best case	97.97%	97.97%	97.97%	97.97%	97.97%
	Std. deviation	0.169	0.066	0.066	0.066	0.066

Table 5 .3: Results for 10 time periods

Data Class	Metrics	10	20	30	40	50
S1	Relative Regret	21.03%	20.63%	20.49%	20.44%	20.39%
	Opt Gap	78.97%	79.37%	79.51%	79.56%	79.61%
	Worst case	55.12%	63.51%	63.51%	66.12%	66.12%
	Best case	89.95%	89.95%	89.95%	89.95%	89.95%
	Std. deviation	0.049	0.043	0.042	0.041	0.040
S2	Relative Regret	21.30%	21.12%	20.96%	20.87%	20.87%
	Opt Gap	78.70%	78.88%	79.04%	79.13%	79.13%
	Worst case	53.50%	53.50%	54.52%	54.52%	54.52%
	Best case	90.52%	90.52%	90.52%	90.52%	90.52%
	Std. deviation	0.054	0.051	0.049	0.048	0.048
S3	Relative Regret	20.95%	20.69%	20.57%	20.48%	20.43%
	Opt Gap	79.05%	79.31%	79.43%	79.52%	79.57%
	Worst case	63.56%	65.77%	65.77%	66.75%	66.75%
	Best case	90.62%	90.62%	90.62%	90.62%	90.62%
	Std. deviation	0.046	0.043	0.042	0.041	0.040
S4	Relative Regret	21.14%	20.97%	20.86%	20.80%	20.76%
	Opt Gap	78.86%	79.03%	79.14%	79.20%	79.24%
	Worst case	55.74%	56.06%	56.06%	56.06%	56.06%
	Best case	89.52%	89.52%	89.52%	89.52%	89.52%
	Std. deviation	0.048	0.046	0.046	0.045	0.044
S5	Relative Regret	20.67%	20.63%	20.62%	20.62%	20.60%
	Opt Gap	79.33%	79.37%	79.38%	79.38%	79.40%
	Worst case	65.94%	65.94%	65.94%	65.94%	65.94%
	Best case	88.35%	88.35%	88.35%	88.35%	88.35%
	Std. deviation	0.042	0.042	0.042	0.042	0.041

Table 5 .4: Results for 25 time periods

In Table 5 .5 we compare the performance of our algorithm with the case when only the naive policy is used to deduce the actions. As can be seen in almost all the data classes, the naive policy gives a relative regret of more than 50% whereas our APINN algorithm gives a relative regret of less than 18% in the 10 period case and less than 20% in 25 period case, thereby closing a significant performance gap over the naive policy. The results below also show that the naive policy is not applied too many times in the improvement stage during the experiments which further proves that our proposed APINN algorithm does indeed learn and improves upon the naive policy.

Data Class	Metrics	T=10		T=25	
		APINN	Naive	APINN	Naive
S1	Relative Regret	17.69%	49.75%	20.63%	50.97%
	% Optimum	82.31%	50.25%	79.37%	49.03%
	Worst case	56.67%	22.21%	63.51%	30.56%
	Best case	99.24%	76.79%	89.95%	65.96%
	Std. deviation	0.065	0.097205915	4.33%	6.31%
S2	Relative Regret	17.11%	49.11%	21.12%	51.10%
	% Optimum	82.89%	50.89%	78.88%	48.90%
	Worst case	62.52%	19.93%	53.50%	34.45%
	Best case	98.37%	77.15%	90.52%	65.07%
	Std. deviation	0.067	0.100335853	5.11%	5.69%
S3	Relative Regret	16.81%	49.71%	20.69%	50.99%
	% Optimum	83.19%	50.29%	79.31%	49.01%
	Worst case	60.25%	18.14%	65.77%	26.00%
	Best case	100.00%	76.62%	90.62%	65.44%
	Std. deviation	0.066	0.099845797	4.33%	6.20%
S4	Relative Regret	17.57%	49.54%	20.97%	51.47%
	% Optimum	82.43%	50.46%	79.03%	48.53%
	Worst case	57.42%	22.87%	56.06%	32.23%
	Best case	98.91%	77.62%	89.52%	65.63%
	Std. deviation	0.066	0.102402335	4.64%	5.60%
S5	Relative Regret	17.88%	49.64%	20.63%	50.52%
	% Optimum	82.12%	50.36%	79.37%	49.48%
	Worst case	59.71%	26.09%	65.94%	34.66%
	Best case	97.97%	77.27%	88.35%	64.69%
	Std. deviation	0.066	0.103584375	4.20%	6.05%

Table 5 .5: Comparison of Naive and APINN policies

CHAPTER SIX

FURTHER WORK

Our preliminary experiments show that approximate policy iteration with neural networks can give good results but the full potential of this approach can only be understood after experimenting with different neural network architectures. It is well-known that larger and more complicated neural networks are useful in improving approximation accuracy. However, a more complicated network can result in over-fitting and more importantly will increase the running time of the algorithm. We realized that changing the keras optimizers did not really result in a better performance of the algorithm. Experimenting with different learning rates and loss functions can also help improve the prediction accuracy.

The sensitivity of the algorithm to problem parameters needs to be studied. Preliminary experiments seem to suggest that there may be other factors other than the battery capacity, injection and withdrawal rates that can really impact the determination of the optimal policy and algorithm performance.

The SNES deterministic benchmark problem described in Section 5.1 is a variant of the lot-sizing problem with losses, bounded inventory and constrained withdrawal and injection rates. To the best of our knowledge this problem is yet to be studied in literature. The complexity of this problem is unknown and algorithms for this problem are very relevant in many contexts.

As shown in Section 4.4, having different buying and selling prices make the problem harder compared to them being equal. In that respect, it is imperative to also investigate and compare the performance of our proposed algorithm when applied to the case of equal buying and selling prices to that of the case of different buying and selling prices.

Scaling the algorithm to more time periods will require speeding the algorithm in the improvement step. One idea towards this is to apply the neural network once to create a look-up matrix which stores the predicted value functions for all possible range of net demand, previous battery levels and feasible storage amounts that satisfy the problem constraints. With this approach, we do not have to apply the neural network continuously for each sampled state and possible storage amount in each improvement step.

Even though most studies capture injection and withdrawal losses in their models, we are not aware if any of these studies give insights into the impact of these losses on the optimal policies. This is a crucial question given that battery technology is improving with time and in the future these losses may be insignificant.

The storage-as-a-service business model is increasingly becoming popular, where the battery is owned and operated by a third-party supplier. In this case, a research idea will be to investigate the amount of storage the customer may want to rent or determine the optimal price charged by the supplier.

REFERENCES

Arnold, M., and Andersson, G. (2011). *Model predictive control of energy storage including uncertain forecasts.* Power Systems Computation Conference (PSCC), Stockholm, Sweden, *23*, 24-29.

Barbour, E., Parra, D., Awwad, Z., and González, M. (2018). *Community energy storage: A smart choice for the smart grid?* Applied Energy, *212*, 489-497.

Bean, J., Birge, J., and Smith, R. (1987). *Aggregation in Dynamic Programming.* Operations Research, *35 (2)*, 215 - 220.

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.

Bertsekas, D., and Castanon, D. (1989). *Adaptive aggregation methods for infinite-horizon dynamic programming.* IEEE Transactions on Automatic Control, *34 (6)*, 589 - 598.

Bertsekas, D., and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.*

Bertsekas, D. P. (2011). *Approximate Policy Iteration: A Survey and Some New Methods.* Journal of Control Theory and Applications, 310-315.

Bertsekas, D. P. (2018). *Feature-Based Aggregation and Deep Reinforcement Learning: A Survey and Some New Implementations.* arXiv preprint arXiv:1804.04577.

Charnes, A., Dréze, J., and Miller, M. (1966). *Decision and horizon rules* *for stochastic planning problems: a linear example.* Econometrica, *34 (2)*, 307–330.

Cheng, B., Asamov, T., and Powell, W. B. (2018). *Low-Rank Value Function Approximation for Co-Optimization of Battery Storage.* IEEE Transactions on Smart Grid, *9 (6)*, 6590-6598.

Dong, L. (2010). *An approximate dynamic programming approach to the scheduling of impatient jobs in a clearing system.* PhD Thesis, Lancaster University.

Durante, J., Nascimento, J., and Powell, W. (2017). *Backward Approximate Dynamic Programming with Hidden Semi-Markov Stochastic Models in Energy Storage Optimization.* arXiv:1710.03914v1 [math.OC].

El-Batawy, S. A., and Morsi, W. G. (2018). *Optimal Design of Community Battery Energy Storage Systems With Prosumers Owning Electric Vehicles.* IEEE Transactions on Industrial Informatics, *14*, 1920-1931.

Evangelopoulos, V., Georgilakis, P., and Hatziargyriou, N. (2016). *Optimal operation of smart distribution networks: A review of models, methods and future research.* Electric Power Systems Research, *140*, 95-106.

Eyer, J., and Corey, G. (2010). *Energy storage for the electricity grid: Benefits and market potential assessment guide.* Technical Report SAND2010-0815, Sandia National Laboratories, 69–73.

Eyer, J., Iannucci, J., and Corey, G. (2004). *Energy storage benefits and market analysis handbook, a study for the DOE energy storage systems program.* Technical Report SAND2004-6177, Sandia National Laboratories.

Farias, D. D., and Roy, B. V. (2000). *On the existence of fixed points for approximate value iteration and temporal-difference learning.* Journal of Optimization Theory and Applications, *105 (3)*, 589-608.

Fischer, T., and C.Krauss. (2018). *Deep learning with long short-term memory networks for financial market predictions.* European Journal of Operational Research, *270 (2)*, 654-669.

Granado, P. C. D., Pang, Z., and Wallace, S. (2016). *Synergy of smart grids and hybrid distributed generation on the value of energy storage.* Applied Energy, *170*, 476-488.

Graves, F., Jenkins, T., and Murphy, D. (1999). *Opportunities for electricity storage in deregulating markets.* The Electricity Journal, *12 (18)*, 45–56.

Halman, N., Klabjan, D., Mostagir, M., Orlin, J., and Simchi-Levi, D. (2009). *A fully polynomial time approximation scheme for single item inventory control with discrete demand.* Mathematics of Operations Research, *34 (3)*, 674–685.

Halman, N., Nannicini, G., and Orlin, J. (2018). *On the complexity of energy storage problems.* Discrete Optimization, *28*, 31-53.

Halman, N., Orlin, J., and Simchi-Levi, D. (2012). *Approximating the nonlinear newsvendor and single-item stochastic lot-sizing problems when data is given by an oracle.* Operations Research, *60 (2)*, 429–446.

Han, J., and Weinan, E. (2016). *Deep Learning Approximation for Stochastic Control Problems.* arXiv preprint arXiv:1611.07422v1.

Hannah, L., and Dunson, D. (2012). *Approximate Dynamic Programming for Storage Problems.* Proceedings of the 29th International Conference on Machine Learning.

Harsha, P., and Dahleh, M. (2015). *Optimal management and sizing of energy storage under dynamic pricing for the efficient integration of renewable energy.* IEEE Transactions On Power Systems, *30 (3)*, 1164–1181.

Huang, L., Walrand, J., and Ramchandran, K. (2012). *Optimal demand response with energy storage management.* . in: Proceedings of the IEEE Third International Conference on Smart Grid Communications (SmartGridComm), 61–66.

Jacobs, J., Freeman, G., Grygier, J., Morton, D., Schultz, G., Staschus, K., et al. (1995). *SOCRATES: A system for scheduling hydroelectric generation under uncertainty.* Annals of Operations Research, *59 (1)*, 99-113.

Jiang, D., Pham, T., Powell, W., Salas, D., and Scott, W. (2014). *A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work?* 2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), *58 (12)*, 1–8.

Jiang, D., and Powell, W. (2015). *Optimal Hour-Ahead Bidding in the Real-Time Electricity Market with Battery Storage using Approximate Dynamic Programming.* INFORMS Journal on Computing, *27 (3)*, 525-543.

Kingma, D., and Ba, J. (2017). *Adam: A Method for Stochastic Optimization.* Computing Research Repository, *abs/1412.6980v9*.

Kraning, M., Wang, Y., Akuiyibo, E., and Boyd, S. (2011). *Operation and Configuration of a Storage Portfolio via Convex Optimization.* IFAC Proceedings Volumes, *44 (1)*, 10487 - 10492.

Kuperman, A., and Aharon, I. (2011). *Battery–ultracapacitor hybrids for pulsed current loads: A review.* Renewable and Sustainable Energy Reviews, *15 (2)*, 981 - 992.

Lai, G., Margot, F., and Secomandi, N. (2010). *An Approximate Dynamic*

Programming Approach to Benchmark Practice-Based Heuristics for Natural Gas Storage Valuation. Operations Research, *58*, 564-582.

Löhndorf, N., and Minner, S. (2010). *Optimal day-ahead trading and storage of renewable energies—an approximate dynamic programming approach.* Energy Systems, *1 (1)*, 61–77.

Liu, D., and Wei, Q. (2014). *Policy Iteration Adaptive Dynamic Programming Algorithm for Discrete-Time Nonlinear Systems.* IEEE Transactions On Neural Networks And Learning Systems, *25 (3)*, 621-634.

Loureiro, A., Miguéis, V., and Silva, L. da. (2018). *Exploring the use of deep neural networks for saless forecasting in fashion retail.* Decision Support Systems, *114*, 81-93.

Lüth, A., Zepter, J., Granado, P. C. D., and Egging, R. (2018). *Local electricity market designs for peer-to-peer trading: The role of battery flexibility.* Applied Energy, *229*, 1233-1243.

Mes, M., and Rivera, A. P. (2016). *Approximate dynamic programming by practical examples.* TU Eindhoven, Research School for Operations Management and Logistics (BETA working papers), *495*, 32.

Moazeni, S., Powell, W., and Hajimiragha, A. (2015). *Mean-conditional value-at-risk optimal energy storage operation in the presence of transaction costs.* IEEE Transactions On Power Systems, *30 (3)*, 1222–1232.

Mokrain, P., and Stephen, M. (2006). *A stochastic programming framework for the valuation of electricity storage.* 26th USAEE/IAEE North American Conference, 24–27.

Nascimento, J. (2008). *Approximate Dynamic Programming for Complex Storage Problems.* PhD Thesis, Princeton University.

Nascimento, J., and Powell, W. (2010). *Dynamic Programming Models and Algorithms for the Mutual Fund Cash Balance Problem.* Management Science, *56 (5)*, 801-815.

Nascimento, J., and Powell, W. (2013). *An optimal approximate dynamic programming algorithm for concave, scalar storage problems with vector-valued controls.* IEEE Transactions On Automatic Control, *58 (12)*, 2995–3010.

Natarajan, G., Xu, Y., and Bradley, J. (2014). *Meeting inelastic demand in systems with storage and renewable sources.* in: Proceedings of the IEEE Fifth International Conference on Smart Grid Communications (SmartGridComm), 97-102.

Porteus, E. (2002). *Foundations of Stochastic Inventory Theory.* Stanford Business Books, Palo Alto, CA.

Powell, W. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality, Second Edition.* John Wiley Sons Incorporated.

Powell, W., George, A., Simão, H., Scott, W., Lamont, A., and Stewart, J. (2012). *SMART: A Stochastic Multiscale Model for the Analysis of Energy Resources, Technology, and Policy.* INFORMS Journal on Computing, *24 (4)*, 665–682.

Powell, W., and Meisel, S. (2016). *Tutorial on Stochastic Optimization in Energy—Part I: Modeling and Policies.* IEEE Transactions on Power Systems, *31 (2)*, 1459-1467.

Powell, W., Simao, H., and Bouzaiene-Ayari, B. (2012). *Approximate dynamic programming in transportation and logistics: a unified framework.* EURO Journal on Transportation and Logistics, *1 (3)*, 237-284.

Rempala, R. (1994). *Optimal strategy in a trading problem with stochastic* *prices.* in: J. Henry, J.-P. Yvon (Eds.), System Modelling and Optimization, in: Lecture Notes in Control and Information Sciences, *197, Springer Berlin Heidelberg*, 560–566.

Salas, D., and Powell, W. (2013). *Benchmarking a scalable approximation dynamic programming algorithm for stochastic control of multidimensional energy storage problems.* Technical report, Princeton University.

Scott, W., and Powell, W. (2012). *Approximate dynamic programming for energy storage with new results on instrumental variables and projected Bellman errors.* Technical report, Princeton University.

Secomandi, N. (2010). *Optimal commodity trading with a capacitated storage asset.* Management Science, *56 (3)*, 449–467.

Senn, M., Link, N., Pollak, J., and Lee, J. (2014). *Reducing the computational effort of optimal process controllers for continuous state spaces by using incremental learning and post-decision state formulations.* Journal of Process Control, *24 (3)*, 133 - 143.

Simão, H., Powell, W., Archer, C., and Kempton, W. (2017). *The challenge of integrating offshore wind power in the U.S. electric grid. Part II: Simulation of electricity market operations.* Renewable Energy, *103*, 418-431.

Singh, S., Jaakkola, T., M.I. Jordan, D. T., In G. Tesauro, and Leen, T. (1995). *Reinforcement learning with soft state aggregation.* Advances in Neural Information Processing Systems 7.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, H., and Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Journal of Machine Learning Research, *15*, 1929-1958.

Staffell, I., and Rustomji, M. (2016). *Maximising the value of energy storage.* Journal of Energy Storage, *82*, 212-225.

State of the energy market 2017 report (Accessed December 2018). `https://www.ofgem.gov.uk/system/files/docs/2017/10/state_of_the_market_report_2017_web_1.pdf`.

Falling lithium-ion battery prices to drive rapid storage uptake (Accessed February 2019). `https://pv-magazine-usa.com/2017/08/03/falling-lithium-ion-battery-prices-to-drive-rapid-storage-uptake/`.

Teleke, S., Baran, M., Bhattacharya, S., and Huang, A. (2010). *Rule-Based Control of Battery Energy Storage for Dispatching Intermittent Renewable Sources.* IEEE Transactions On Sustainable Energy, *1 (3)*, 117–124.

Tesauro, G. J. (2002). *Programming Backgammon Using Self-Teaching Neural Nets.* Artificial Intelligence, *134*, 181-199.

Tseng, C., and Barz, G. (2002). *Short-term generation asset valuation: a real options approach.* Operations Research, *50 (2)*, 297–310.

Vazquez, S., Lukic, S. M., Galvan, E., Franquelo, L. G., and Carrasco, J. M. (2010). *Energy Storage Systems for Transport and Grid Applications.* IEEE Transactions on Industrial Electronics, *57 (12)*, 3881-3895.

Xiaomin, X., Sioshansi, R., and Marano, V. (2014). *A stochastic dynamic programming model for co-optimization of distributed energy storage.* Energy Systems, *5 (3)*, 475–505.

Yuan, Y., Lorenzo, R., and Andrea, C. (2007). *On Early Stopping in Gradient Descent Learning.* Constructive Approximation, *26*, 289-315.

Zhang, C., Wu, J., Zhou, Y., Cheng, M., and Long, C. (2018). *Peer-to-Peer energy trading in a Microgrid.* Applied Energy, *220*, 1-12.

Zhou, Y., Scheller-Wolf, A., Secomandi, N., and Smith, S. (2016). *Electricity* *trading and negative prices: storage vs. disposal.* Management Science, *62 (3)*, 880–898.

Zhou, Y., Scheller-Wolf, A., Secomandi, N., and Smith, S. (2018). *Managing wind-based electricity generation in the presence of storage and transmission capacity.* Technical Report 2011-E36, Tepper School of Business, Carnegie Mellon University, Available at SSRN: https://ssrn.com/abstract=1962414 or http://dx.doi.org/10.2139/ssrn.1962414.

Zilong, H., Jinshan, T., Ziming, W., Kai, Z., Ling, Z., and Qingling, S. (2018). *Deep learning for image-based cancer detection and diagnosis - A survey.* Pattern Recognition, *83*, 134-149.

Zipkin, P. (2000). *Foundations of Inventory Management.* McGraw-Hill, New York, NY.