Optimal energy management for a grid-tied solar PV-battery microgrid: A reinforcement learning approach

A thesis submitted for the award of the degree of

Master of Science in Electrical Engineering

Faculty of Engineering and The Built Environment

University of Cape Town

GRACE MURIITHI

October 2021

Supervised by

A/Prof Sunetra Chowdhury

DECLARATION

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Grace Muriithi

Sign: Signed by candidate

27/10/2021

DEDICATION

To God, thank you! You gave me the strength to finish this research work; Your grace has been sufficient through it all.

To my family, thank you for their unwavering support and kind prayers.

To my friends for their encouragement

ACKNOWLEDGEMENT

I want to express my sincere gratitude to Almighty God for providing me with the strength to complete this study. All glory, praise and honour go back to Him.

I deeply thank my supervisor, A/Prof. Sunetra Chowdhury for her tireless and invaluable guidance and support during this research work.

I am grateful to the MasterCard Foundation scholars' program for welcoming me into their global family and fully funding my master's studies. I am thankful for all the leadership and career development webinars that challenged my way of thinking and interacting with people.

I am grateful to the University of Cape Town for providing the infrastructure and enabling environment that permitted this research to be completed successfully.

Many thanks to the University of Cape Town (UCT's) Properties and Services for providing the data that was used in this research work.

A huge thank you also goes to friends Jeff Waititwa, Bright Tetteh, Erick Arwa, Dorcas Wanjiku, Innocent Safari, Willie Macharia, Alex Mirugwe and George Irungu for their invaluable contribution to my education journey, particularly this study.

Finally, I sincerely thank my parents Roselyn and Samuel Muriithi, brother Alex Njagi and sister Cynthia Wambui for all their prayers, love and support over the years.

ABSTRACT

There has been a shift towards energy sustainability in recent years, and this shift should continue. The steady growth of energy demand as a result of population growth, as well as heightened worries about the number of anthropogenic gases released into the atmosphere and deployment of advanced grid technologies, has spurred the penetration of renewable energy resources (RERs) at different locations and scales in the power grid. As a result, the energy system is moving away from the centralized paradigm of large, controllable power plants and toward a decentralized network based on renewables. Microgrids, either grid-connected or islanded, provide a key solution for integrating RERs, load demand flexibility, and energy storage systems within this framework. Nonetheless, renewable energy resources, such as solar and wind energy, can be extremely stochastic as they are weather dependent. These resources coupled with load demand uncertainties lead to random variations on both the generation and load sides, thus challenging optimal energy management.

This thesis develops an optimal energy management system (EMS) for a grid-tied solar PV-battery microgrid. The goal of the EMS is to obtain the minimum operational costs (cost of power exchange with the utility and battery wear cost) while still considering network constraints, which ensure grid violations are avoided. A reinforcement learning (RL) approach is proposed to minimize the operational cost of the microgrid under this stochastic setting. RL is a reward-motivated optimization technique derived from how animals learn to optimize their behaviour in new environments. Unlike other conventional model-based optimization approaches, RL doesn't need an explicit model of the optimization system to get optimal solutions. The EMS is modelled as a Markov Decision Process (MDP) to achieve optimality considering the state, action, and reward function. The feasibility of two RL algorithms, namely, conventional Q-learning algorithm and deep Q network algorithm, are developed, and their efficacy in performing optimal energy management for the designed system is evaluated in this thesis.

First, the energy management problem is expressed as a sequential decision-making process, after which two algorithms, trading and non-trading algorithm, are developed. In the trading algorithm case, excess microgrid's energy can be sold back to the utility to increase revenue, while in the latter case constraining rules are embedded in the designed EMS to ensure that no excess energy is sold back to the utility. Then a Q-learning algorithm is developed to minimize the operational cost of the microgrid under unknown future information. Finally, to evaluate the performance of the proposed EMS, a comparison study between a trading case EMS model and a non-trading case is performed using a typical commercial load curve and PV generation profile over a 24-hour horizon. Numerical simulation results indicated that the algorithm learned to select an optimized energy schedule that minimizes energy cost (cost of power purchased from the utility based on the time-varying tariff and battery wear cost) in both summer and winter case studies. However, comparing the non-trading EMS to the trading EMS model operational costs, the latter one decreased cost by 4.033% in the summer season and 2.199% in the winter season.

Secondly, a deep Q network (DQN) method that uses recent learning algorithm enhancements, including experience replay and target network, is developed to learn the system uncertainties, including load demand, grid prices and volatile power supply from the renewables solve the optimal energy management problem. Unlike the Q-learning method, which updates the Q-function using a lookup table (which limits its scalability and overall performance in stochastic optimization), the DQN method uses a deep neural network that approximates the Q-function via statistical regression. The performance of the proposed method is evaluated with differently fluctuating load profiles, i.e., slow, medium, and fast. Simulation results substantiated the efficacy of the proposed method as the algorithm was established to learn from experience to raise the battery state of charge and optimally shift loads from a one-time instance, thus supporting the utility grid in reducing aggregate peak load. Furthermore, the performance of the proposed DQN approach was compared to the conventional Q-learning algorithm in terms of achieving a minimum global cost. Simulation results showed that the DQN algorithm outperformed the conventional Q-learning approach, reducing system operational costs by 15%, 24%, and 26% for the slow, medium, and fast fluctuating load profiles in the studied cases.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
A3C	Asynchronous Advantage Actor Critic
ANN	Artificial Neural Network
AMI	Advanced Metering Infrastructure
ADP	Adaptive Dynamic programming
BESS	Battery Energy storage system
BRL	Batch Reinforcement Learning
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
DQN	Deep Q Network
DNN	Deep Neural Network
DOD	Depth of Discharge
DSO	Distribution System Operator
ESS	Energy storage system
EMS	Energy Management System
FBES	Flow Battery Energy Storage
GHG	Green House Gases
GA	Genetic Algorithm
IEC	International electronics Commission
LP	Linear programming
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
ML	Machine learning
MDP	Markov Decision Process
MPPT	Maximum Power Point Tracking
MGCC	Microgrid Central Controller
MG	Microgrid
MC	Monte Carlo
OEM	Optimal Energy Management
PV	Photovoltaic
PSO	Particle swarm optimization
PCC	Point of Common Coupling
RL	Reinforcement Learning
ReLu	Rectified Linear Unit
RDG	Renewable Distributed Generators
SARSA	State Action Reward State Action
SDG	Stochastic Gradient Descent
SOC	State of Charge

ToU	Time of Use
T & D	Transmission and Distribution
TD	Temporal Difference

NOTATIONS

C_g	Grid power cost
C_{deg}	Battery degradation cost
DoD	Depth of discharge
j	Depth of discharge index
t	Time step
G_t	Grid Tariff
$P_{g_{-p}}$	Imported Power
P_{g_s}	Exported Power
ϑ	Grid Tariff Discounting Factor
$L(DoD)$	Battery Life as a function of DoD
C_{bt}	Battery Capital Investment Cost
$P_{g\_s}^{max}$	Grid Import Power Limit
$P_{g\_s}^{max}$	Grid Export Power Limit
s_t	State
β	Guard Ratio
P_t^{PV}	Solar PV Generation
$P_{l,t}$	Instantaneous Load demand
E	Battery Energy Capacity
$P_{l,t}^{Net}$	Net Load
$P_{l,t}^{rem}$	Remainder energy
$\mathcal{A}_{s_t}$	Action Set
a_t	action
SoC_t	State of Charge
SOC^{min}	Minimum state of charge
SOC^{max}	Maximum State of Charge
SOC_{t+1}^{next}	Next State of charge
Δp	Battery Power Unit
$\pi(s,a)$	Policy Function
c_b^t	Pre-charge reward
c_o^t	Negative reward received if battery limits are violated
s_{t+1}	Next state
$r(s_t,a_t)$	Total reward of a specific time slot
$\mathbb{P}$	Probability

V_t^{π}	Value Function
γ	Gamma
s_0	Initial State
$Q(s_t, a_t)$	Q function
$\mathbb{E}$	Expectation operator
α	Learning rate
n	Number of episodes Index
ε	Epsilon
C	Exponential Decay
R	Reward
w	Neural Network Vector Parameter
$L(w_{train}^{t})$	Mean Squared Loss Function
K^t	Mini-batch size
w_{train}^{t}	Vector Parameter for policy network
w_{target}^{t}	Vector Parameter for target network
y_{DQN}^{t}	Target/ training output value for the policy(online) network
w^*	Optimized Vector Parameters (DNN)
T_u	Target network update interval

1 INTRODUCTION

1.1 Problem Statement

The power system, developed roughly a century ago, has been used for decades without any significant changes. Customers who were solely functioning as energy consumers received energy from centralized power plants. If more power were needed, these power generators would increase the power generated to balance power between the supply and demand sides. As a result, the supply side would adjust to meet the load demand requirements. However, things began to change with the advent of renewable energy resources (RER), such as biomass, solar, and wind power. The percentage of renewable energy output was initially low, but it has quickly increased in recent years. There are numerous explanations for this increase: climate change and legislation to reduce carbon emissions, electronic advancements, novel technologies that are low in carbon, unique market models, and government incentives are all factors to consider [1].

Consequently, the energy system is shifting from a centralized system comprised of large, controllable power units to a distributed network composed of numerous uncontrollable renewable energy sources. In addition, countries worldwide are putting in place targets to reduce greenhouse gas (GHG) emissions, improve the power grid energy efficiency, and expand clean energy production in their power generation mix. As specified in the recent Paris Agreement, the primary goal is to keep global temperatures "far below 2°C pre-industrial levels and to pursue measures to restrict the temperature increase to 1.5°C" [2]. Therefore, the construction of small local electrical networks, known as micro/mini grids, is increasingly viewed as a crucial enabler in integrating renewable energy resources and providing sustainable energy to end consumers. Moreover, the massive advancements in power electronics, (which provide valuable tools for controlling and converting electric power from DC to AC and vice versa and allow bidirectional interconnection between microgrids and the utility grid) decreasing investment costs of distributed energy technologies, and advancements in information and communication technologies (ICT) are further accelerating the diffusion of renewable distributed energy resources (DERs) in the energy markets and reshaping energy systems from a control viewpoint [3].

Although it is recommended that renewable DERs be deployed to reduce over-reliance on convectional fossil-based sources and improve energy efficiency, renewable sources are highly uncontrollable and stochastic. Most renewable energy technologies cannot guarantee a constant and consistent electricity generation because their output varies depending on the weather and time of day. Furthermore, electricity consumption may be partially unpredictable with microgrids due to economic and social activities, new technologies on the end-users side such as electric vehicles, sudden changes of weather and different types of electric consumers [4], adding another layer of complexity to optimal power control. As variable power sources (such as wind and solar) reach a high level of grid penetration, optimal energy management becomes very difficult to describe with a proper mathematical formulation due to the uncertainty in the load demand, renewable resources, and bidirectional flow of power between the microgrid and the utility grid. Deploying a energy storage system(ESS) [5] can significantly buffer the impacts of these uncertainties however, to guarantee a reliable supply of power and ensure efficient utilization of the battery storage, it’s vital that an efficient energy management system be developed.

In power system, an EMS is needed to ensure system requirements are met by maximizing the net benefit of renewable energy resources while minimizing the entire system's energy losses and operating costs. It also ensures optimal decision making during the microgrid operations, thus ensuring the load demand is met at minimal or reasonable costs while satisfying all technical and operational constraints [6]. In microgrid context however, the EMS faces significant challenges because of the microgrid’s small energy capacity, intermittency of renewable DERs, energy consumption unpredictability and dynamic electricity tariffs. Thus, to optimize energy dispatch and overcome microgrid’s uncertainties (stemming from intermittent renewable electricity sources, uncertain load demand and varying electricity market prices), further enhancements on the microgrid’s architecture and advanced control techniques that can take optimal real-time actions instantly and adaptively are required to guarantee continuous balance between supply and demand, thus ensuring system stability. Furthermore, additional flexibility on the demand side can be explored at the design level to balance the high volatility of renewable DERs.

On the control level, several control approaches have been proposed to maximise energy usage or reduce operational cost by intelligently managing the different types of energy resources and controllable loads in grid-tied microgrids. For years, conventional techniques such as mixed-integer linear programming, linear programming, and dynamic programming have been proposed to optimally manage energy in microgrids [7]–[9]. These methods, however, are reported to suffer from the famous curse of dimensionality and are highly susceptible to getting sub-optimal results in environments that are highly stochastic, i.e., they contain volatile variables such as load demand, grid tariffs and renewable energy. Such techniques, therefore, have limited flexibility and scalability. Further, metaheuristic techniques, including particle swarm optimization (PSO), genetic algorithm (GA), and their hybrids, have also been used in literature to tackle the issue of energy management in microgrids [10]–[13]. However, these techniques involve extensive computational time and are unable to handle dynamic online operations. The online function allows computing resources to be used more economically as it doesn’t require one to have another committed computer for performing the optimization process offline. The algorithms mentioned above also don’t have a learning component, i.e. they are incapable of storing the optimization knowledge and reusing it for a new optimization task [14]. Given that the load demand varies on an hourly basis, it is required to calculate the schedule for every new generation and demand profile, which is not computationally efficient. In addition, the performance of this techniques may deteriorate if accurate models or appropriate state variables forecasting are unavailable. Often, metaheuristic methods are hybridized with other linear methods for an advantage complementation [6], [15].

In the last decade, data-driven techniques based on machine learning (ML) have shown great potential in achieving near-optimal microgrid control using operational data. They have also proved ideal in overcoming the limitations above. The reinforcement learning (RL) paradigm, in which a microgrid's dynamics are learned by an agent interacting with its components, is amongst the most promising learning-based techniques for energy management systems. Reinforcement learning is a solution method that is motivated by rewards. Reinforcement learning methods are more capable of dealing with stochastic systems than other optimization approaches due to their learning element and capacity to generalize solutions [14], [16]. As a result, this work proposes reinforcement learning to perform the energy management task.

1.1.1 Proposed Approach

The reinforcement learning (RL) method, one of the machine learning algorithms, is well known because of its ability to solve problems in stochastic environments. It aims at making optimal time-sequential decisions in an uncertain environment. Reinforcement learning involves a decision-maker (agent) that learns how to act (action) in a particular situation (state) through continuous interaction with the environment to maximize cumulative rewards [17], [18]. In the learning process, the agent can learn about the system and take action that affects the environment to achieve its objective. In RL, the agent considers the long-term reward instead of simply getting the immediate maximum reward. This concept is essential for resource optimization problems in renewable-powered microgrids, where supply and demand change rapidly. Q-learning, one of the RL methods, is commonly used to solve sequential decision-making problems, as authors in [18] explain. Q-learning is an off-policy algorithm that doesn’t require any prior knowledge of a system's rewards or state transition probabilities, thus making it applicable to systems that manage real-time data. Many scholars, focusing on microgrid EMS [14], [19]–[21], specifically have used Q-learning to control energy. The key benefit of RL techniques is their adaptability to stochastic systems and ability to transfer knowledge, i.e. the information gained when learning policies for specific load demands can be retrieved to learn an optimal schedule for other load profiles [14]. Furthermore, RL is capable of relaxing the idea of an explicit system model to ensure optimal control. This is of great benefit since the problem of energy management is usually a partly observable problem, i.e., hidden or unknown information always exists.

Several scholars have used this approach to solve the microgrid energy management problem to take advantage of these characteristics. For instance, Brida et al. [22] used batch reinforcement learning to implement a microgrid EMS optimizes battery schedules. The charge and discharge efficiency of the battery and the microgrid nonlinearity caused by inverter efficiency were considered. Elham et al. [23] presented a multi-agent RL method for adaptive control of energy flow in a microgrid. The results indicate that the grid-tied microgrid learned to reduce its dependency on the utility grid significantly. Authors in [24] presented an optimal battery scheduling scheme for microgrid energy management. A Q-learning technique is implemented to reduce the overall power consumption from the utility in [22], and simulation results show that the algorithm reduces dependency on the main grid. However, this work fails to consider battery trading with the utility and the impact of the battery life cycle from those actions. In recent advances reported on the implementation of RL in microgrid energy management [22]–[25], [26]–[32], [33]–[35], modelling of microgrid operational cost with consideration of battery degradation cost is not yet thoroughly studied. Most studies only consider the generation cost and power exchange cost. Therefore, this research focuses on extending the emerging studies in the application of RL in microgrid energy management in the context of a solar PV battery microgrid connected to the utility grid. In addition, a battery degradation model is incorporated to reduce strain on the battery during the (dis)charge operation.

1.2 Research Aim and Objectives

This thesis aims to design an EMS that can optimally manage energy flows from and to the main grid and optimize battery charging and discharging such that the overall operational costs (including the cost of power purchased from the utility and battery wear cost) are reduced and utilization of solar PV is maximized. To minimize the operational costs, RL based algorithm is implemented to learn the control actions for energy storage systems (ESS) under a very complex environment (e.g., battery degradation, intermittent renewable energy supply and grid tariff uncertainty). To access the performance of the developed algorithms, several case studies are considered. First, a trading algorithm is studied where excess microgrid's energy can be sold back to the utility to increase revenue. Second, a non-trading algorithm where constraining rules are embedded into the learning process to curtail excess energy from been sold back to the utility is also developed. Both are tested using both summer and winter solar PV generation data. On the other hand, a deep reinforcement learning approach, specifically the deep Q network (DQN), is also developed and tested with different load profiles, i.e., slow, medium, and fast fluctuating load demands to evaluate its efficacy in coping with system uncertainties. Based on this explanation, the mentioned research aim can be divided into several sub-objectives as follows:

1. To develop a mathematical optimization model for the energy management problem in a grid-tied solar PV-battery microgrid system considering network technical constraints and the uncertainty of solar PV generation, load consumption, and grid tariff.
2. To formulate the energy management problem as a Markov decision problem (MDP) considering the state, action, and reward function.

3. To evaluate the efficacy of the proposed Q learning algorithm in reducing system operational costs, i.e., cost of power purchased from the utility and battery wear cost under both summer and winter case studies.
4. To develop a deep reinforcement learning algorithm, notably the deep Q network algorithm for optimizing energy management in the designed microgrid system. The algorithm's performance is tested and evaluated with different types of load profiles (with slow, medium, and fast fluctuations) to assess its efficacy in coping with system uncertainties.

1.3 Research Questions

Several research questions are established to meet the objectives stated above, and they will serve as a guideline throughout this dissertation, as mentioned above.

1. How can the EMS design be improved such that energy scheduling in grid-tied microgrids is optimized?
2. What is the impact of including and excluding grid constraints at the interconnection point on the proposed energy management system for the identified case studies, i.e., summer and winter scenarios?
3. How can resources allocation in grid-tied microgrids be optimized using state-of-the-art machine learning algorithms by extracting its operational data and controlling the system based on it?

1.4 Limitation and Scope

This thesis presents a microgrid energy management system capable of exchanging energy with the main grid at rates set by the utility company. Two algorithms were considered under the Q-learning section, a trading algorithm and a non-trading algorithm while considering a Time of Use (ToU) grid tariff.

The DQN approach develops an algorithm that can optimally schedule the battery and perform optimal power flows from and to the utility grid. The algorithm is tested with different types of load profiles (with slow, medium, and fast fluctuations) to evaluate its efficacy in coping with system uncertainties.

In the research, voltage and current levels required for grid-tied microgrids are not studied, i.e., voltage stability and effects of power flow to system voltages on different power network nodes were ignored. It was assumed that the microgrid's operations were within the voltage and current levels set by the distribution system operator.

For simplicity's sake, power electronics efficiency was assumed to be 100%, meaning power loss at the power electronics interface was assumed to be zero.

Concerning the battery size, the developed algorithm can work with any battery size; hence the battery size considered for this study may not be optimal as optimal battery sizing lies outside the scope of this work.

1.5 Thesis Structure

The rest of the thesis is structured as follows:

Chapter 2: gives a brief background about microgrids, renewable energy resources and energy storage systems. Later, a detailed review of previously existing literature on optimization techniques used in microgrid energy management systems (EMS) is presented.

Chapter 3: discusses reinforcement learning as a reward-based control algorithm. Each component of a general MDP is explained using a simple illustration of an RL environment. This is followed by elaborating the various reinforcement learning-based control techniques used to solve Markov Decision Processes (MDPs). It also introduces deep reinforcement learning techniques, particularly batch reinforcement and deep Q network.

Chapter 4: presents the application of Q-learning in optimizing energy management. It begins by first explaining the problem background in the introduction part. Then, a mathematical formulation including the battery degradation model, the utility grid model, and the network constraint is explained to solve the optimization problem. After that, a detailed design of the proposed EMS as an MDP is discussed, and the proposed Q-learning algorithm is presented. Finally, the results obtained for all considered case studies and the performance of the algorithm are analysed. (This work reported in this chapter has been published in the Energies Journal as a research paper.)

Chapter 5: presents the application of deep Q network energy scheduling for different load profiles. The chapter begins by providing a brief introduction. Then, an overview of the proposed method, i.e., the DQN algorithm, is discussed. The mathematical formulation begins optimized present, followed by a detailed EMS design as an MDP. Several input parameters are defined. How these input data are acquired is extensively described, and several case studies are analysed in this chapter to assess the performances of the proposed algorithm. The chapter also includes simulation results and analysis for the DQN algorithm.

Additionally, a comprehensive discussion of the results achieved is done. Finally, the results obtained by the DQN algorithm are benchmarked with the convectional Q learning results, and a comparison cost study between the two algorithms is presented. (The work reported in this chapter is currently under review as a research paper in Applied Energy Journal)

Chapter 6: gives the conclusions drawn from the results and the discussions and several recommendations of future works for the improvement of the presented work.

1.6 Research Output

Some of the research's outputs had been published in credible journals while working on this research.

Journal Publications:

1. **Muriithi, Grace**, and Sunetra Chowdhury. 2021. "Optimal Energy Management of a Grid-Tied Solar PV-Battery Microgrid: A Reinforcement Learning Approach" *Energies* 14, no. 9: 2700. https://doi.org/10.3390/en14092700
2. **Muriithi Grace** and Sunetra Chowdhury 'Deep Q-Network Application for Optimal Energy Management in A Grid-Tied Solar PV-Battery Microgrid, The Journal of Engineering. https://doi.org/10.1049/tje2.12128

2 LITERATURE REVIEW

Due to increased population growth in the urban and rural areas, the total energy demand is expected to radically increase, resulting in renewable energy resources (RERs), distributed generation (DGs) and distributed storage (DSs) being deployed to the power grid. Consequently, leading to a significant transformation in how distribution systems are operated. Fortunately, the microgrids idea surfaced to cope with the needs of this new paradigm in power systems. A brief background about microgrids and their optimization techniques, with a major focus on energy management systems, will be discussed in this chapter.

2.1 Microgrids

Energy is a scarce resource that faces new challenges due to the current global expectations for sustainability and carbon emission reduction. The actual transition to a reliable, efficient, sustainable, and flexible power system calls for more sophisticated system planning and operation methods. However, this continuing transition results in increased complexity and uncertainty in operating the existing grid. Therefore, the idea of the new advanced electricity grid, known as the 'smart grid,' has arisen to resolve many of the limitations of the conventional grid. The 'smart grid' is a complex network consisting of different system components at different integration levels that enables a bi-directional flow of energy accompanied by a bi-directional flow of information between all participants (such as power generators, end-consumers, transmission and distribution network operators and demand response or DR aggregator) and enables a near-instantaneous balance of supply and demand [36].

Furthermore, a reliable communication network will be the leading enabler of the smart grid that will differentiate it all from the conventional power system. Smart grid is expected to acquire solutions that enable smooth integration of distributed energy sources at all voltage levels [37]. In this regard, microgrids will play a vital role in the modernisation cycle of the existing grid by offering a scalable way of incorporating renewable and low-carbon distributed energy resources (DERs). Many countries are promoting the deployment of DER to reduce environmental effects, operation costs and improve system reliability, resilience, and efficiency. This, coupled with the rapid advancement in information and communication technologies (ICT), would make microgrid a perfect candidate for distributed power systems [12]. A microgrid is defined as a group of distributed energy resources (DERs), including renewable energy sources (RES) and energy storage systems (ESS), plus loads, with clear electrical boundaries which operate locally as a single controllable entity with respect to the utility grid [38]. Microgrids can be found in low and medium voltage operating ranges, usually between 400 V and 69 kV[38]. Microgrids can be large and complex networks, up to tens of megawatts, with different generation sources and storage systems serving multiple loads. On the other hand, they can also be small and simple systems, with a range of hundreds of kW, serving only a few end-users [39]. A microgrid can work in parallel with the utility grid to make full use of the DERs or operate in islanded mode to provide a reliable guarantee for local service if there is a breakdown in the main power system. Grid-connected microgrids have a point of common coupling (PCC) with the main power grid and should be capable of a seamless transition to standalone mode. The combination of several autonomous microgrids operating with each other is expected to become a predominant mode of operating microgrids in the future smart grid [33]. The default configuration of the microgrid is shown in Fig. 2.1 where the network is interconnected with the grid system using a PCC breaker and consists of specific units such as loads, different DERs, and ESS. In grid-connected mode, voltage and frequency control is done by the extensive network; however, if the microgrid is a stand-alone system, the DERs must be controlled to ensure standard voltage and frequency ranges for stability in the microgrid [40].

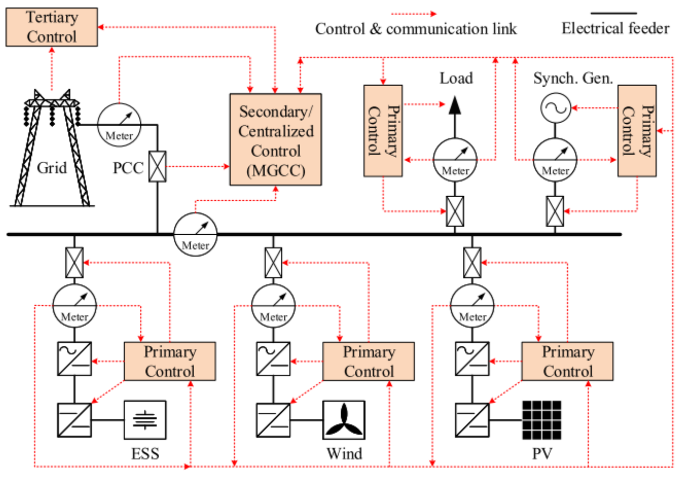


Figure 2.1: A typical microgrid configuration [37]

2.2 Distributed Energy Resource Used in Microgrids

Distributed energy resources are intended to be significant providers of microgrid power. DERs are small-scale energy resources that could be connected at the distribution network or at consumers' premises to provide local electricity supply [40]. Two common and broadly used DERs are renewable distributed generators (RDGs) and energy storage systems (ESSs). In recent years, increasing emphasis has been placed on the use of RDGs, such as solar and wind power. The key drivers of this trend are the clean and sustainable nature of these resources compared to the polluting and limited fossil fuels which have conventionally been used to produce energy. RDGs are also dependent on weather conditions, rendering these resources highly volatile and hence generating immense uncertainties in their power generation. Intermittency of renewable energy sources is one of the drawbacks that must be addressed in order to permit the vast integration of renewable energy sources to the main grid. Some of the RDGs are solar panels, wind turbines, biomass energy, geothermal energy, hydroelectric energy, etc. They deliver a range of advantages, including high efficiency, sustainability, emissions-free and an almost ubiquitous primary energy resource[40].

The application of RDGs in microgrids is one of the most comprehensively researched topics in literature. However, a digression to applications of RDGs will not be done in this section, only a brief description of the two commonly deployed renewable sources, solar PV, and wind.

2.2.1 Solar PV Energy

Solar photovoltaic power is a common term used for electrical energy produced by sunlight. The solar photovoltaic system converts sunlight to electricity. The typical photovoltaic system employs solar panels, each consisting of several solar cells. The operating principle of solar cells is based on the photovoltaic effect, which describes the generation of a potential difference at the junction of two different materials in response to electromagnetic radiation. First, electrons from the semiconductor material are excited by photons, and therefore electron-hole pairs are created. Then, the charge carries are collected at either end of the semi-conductor material by electrical contacts connected to an external circuit. In this way, a current can flow through the circuit and power is produced [41], [42].

Typically, PV is integrated into the grid system with specific PV inverters to deliver the maximum possible power. Maximum power point tracking (MPPT) is essential because it takes full advantage of the available solar energy [43]. However, maximum power output from the PV panel is not simple as it seems, as the I-V characteristics of the PV panels are non-linear. The inverter must establish the appropriate voltage setpoint at the terminals of the panel, allowing the PV device to operate at its optimum level; this is referred to as the *maximum power point.* Since the PV panel has a non-linear relationship between P-V and I-V characteristics, there is an optimal point in the voltage curve where the panel produces maximum power. Therefore, advanced PV systems are integrated with MPPT to ensure optimal working conditions.

2.2.2 Wind Energy

Wind turbines are aerodynamic devices capable of converting the kinetic energy stored in the wind into electrical energy. They consist of blades, a hub and a rotor-nacelle assembly supported by a tower and a base. The elements included in the nacelle differ with each technology. Still, they mainly have a hub, low-and high-speed shafts, rotor bearings, a gearbox, a brake, a generator and a converter. The main working principle of a wind turbine is presented. From Figure 2.2 below, it can be seen how the wind arrives and spins the rotor blades. This spinning power is passed to an electric generator, which generates an AC voltage, oscillating at the frequency of the gearbox. Subsequently, the induced voltage at variable frequency is rectified and then inverted to produce a voltage at the desired frequency before feeding it into the grid [44].

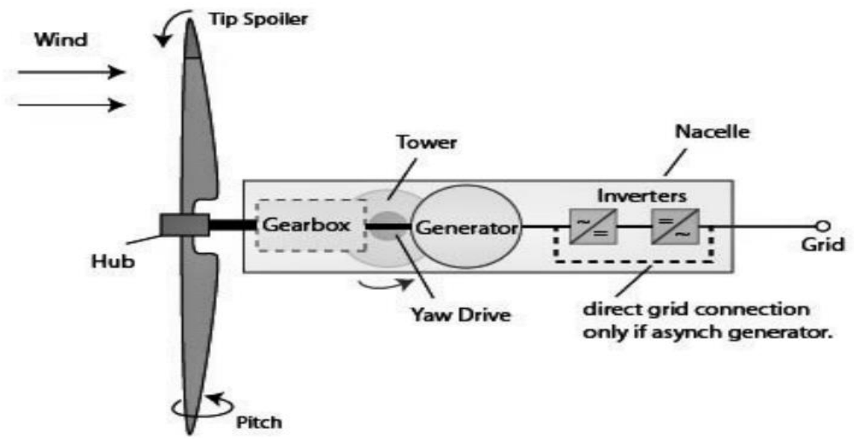


Figure 2.2: Simple schematic layout of a wind turbine [44]

The intermittent and stochastic nature of RDGs necessitates incorporating buffer capacity, i.e. diesel generators and energy storage sources, to compensate for their fluctuations [45]. Due to the adverse environmental effects that come with the use of diesel generators, ESS is most preferred to provide buffer capacity. ESS's technology enables variable generation resources to continue their generation of power even in the absence of wind and sunlight by storing excess power and dispatching it when required, thus ensuring continuous and reliable power in the microgrid. ESS provides microgrids with several benefits, which include[44]:

- Frequency control: Help to maintain the frequency of the grid within set limits.
- Reactive power support by power factor tuning.
- Voltage support: maintaining the power system voltage within nominal values.
- Storing excess RDGs's output to be utilized later when needed.
- Level RDGs's output by buffering lower energy supply.
- Grid support through peak shaving: power can be dispatched from the battery to level the load curve during peak demand.

- ❖ Pre-planned/unplanned islanding: This allows the microgrid to operate with stability during the islanded mode.
- ❖ Grid flexibility: power balancing between production and consumption.

2.3 Energy storage technologies used in microgrids.

Energy Storage is the process of converting electrical energy from a power source or grid to another form through an energy conversion module, stored and returned to the system when required [46]. Over the last decade, engineers and academic researchers have demonstrated significant advances in energy storage technology through several collaborative research. Such advances in ESS technology have led to the transition from small-scale to large-scale energy storage systems and increased penetration of renewable energy at all levels of the spectrum of the power systems [5]. Depending on the ESS used, the energy storage is roughly classified into four types; mechanical energy storage, thermal energy storage, chemical energy storage and electrical energy storage [47]. Energy technologies classifications are shown in Figure 2.3 below.

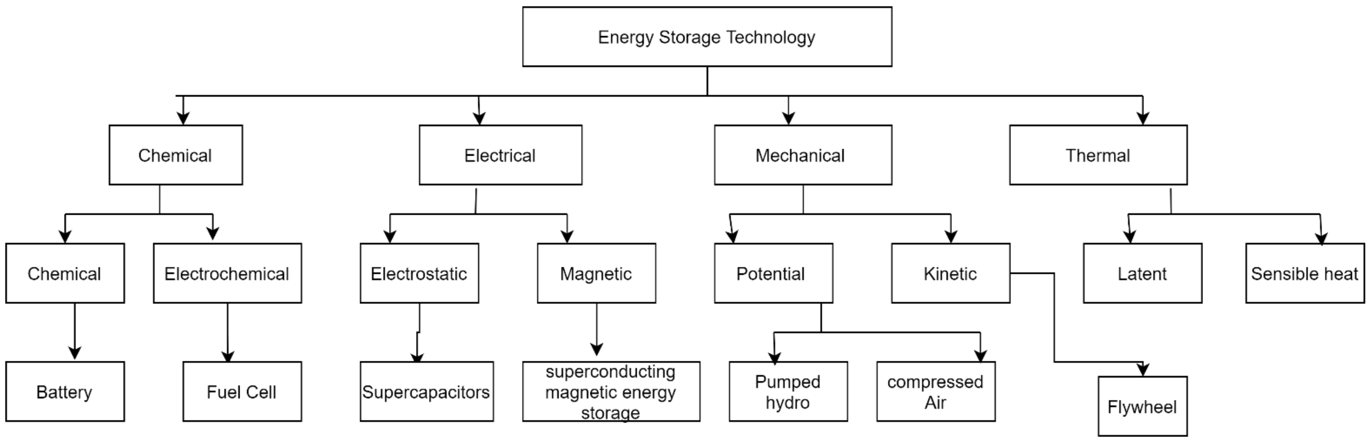


Figure 2.3: Classification of energy storage technologies [44]

A detailed review of the existing technologies relating to microgrids will be presented in this chapter including batteries, supercapacitors, and flow battery energy storage.

2.3.1 Battery Energy Storage Systems

Battery stores electrical energy in the form of chemical energy. During discharging process, the chemical energy inside the battery is converted into electrical energy to supply the external circuit, while as when charging electrical energy is converted to chemical energy within the battery. The different parts of a BESS are usually the batteries, a control and power conditioning system and some protective devices. Examples of batteries technologies are Lithium-Ion batteries, Lead Acid batteries, Nickel batteries and Sodium Sulphur batteries. The batteries consist of several cells containing an electrolyte (solid/liquid) and a positive and negative electrode. Although it is one of the oldest technologies for electrical power storage, considerations such as ease of use and reliability ensure that they are still used. Their power storage capabilities range from 100W-20MW. The modular design of the batteries enables multiple units to be connected to increase the power capacity. Lead-acid is known to be the oldest and most popular battery energy storage system in existence since 1859. Despite this, lead-acid is still a commonly used BESS technology due to its benefits, including low cost, minimal maintenance requirements and low level of self-discharge. Nonetheless, considerations such as high toxicity, increased weight due to lead content, and low power density prevent the use of lead-acid batteries for some applications [46].

Lithium-ion (Li-ion) is another energy storage technology that has become the fastest-growing technology in recent years. Li-ion storage systems can store energy on a megawatt-scale. The major advancement in energy storage capacity of this technology is attributed to its high efficiency (90% and above), high energy density, the rapid response time (in milliseconds) and attractive self-discharge rate (5 per cent per mount). Its primary drawbacks are deep depth of discharge (DoD) and high cost. However, with large-scale production, the cost of the Li-ion cell is expected to decrease soon. Li-ion batteries are the best-suited storage technology for microgrid operations because they have high reliability, a long lifetime of 25 years or more, high safety standards and are cost-effective. In [48]–[50], the lithium-ion battery has been examined for microgrid operation both in grid-connected and islanded mode. The authors considered different scenarios of operation, i.e., black start, peak shaving, rejection capabilities and low voltage faults, among others. From [48]–[50], it's clear that lithium has good efficiency, high energy density, a long life cycle, and a low discharge rate. For this work, a Li-ion battery is considered in the design section because of its favourable performance. More benefits of deciding in favour of the Li-ion technology are [50],

- They give a compact, redundant, and extensible solution when connected to the grid.
- They have high reliability.
- They have a long calendar lifetime of 25 years plus.
- Exhibit high standards of safety.
- Less expensive.
- They have a high energy density.

Sodium-sulphur is another battery technology that is considered promising since it has characteristics such as high energy efficiency, good performance and deep discharge tolerance [46], [51]. Sodium-Sulphur batteries could be utilized to support RDGs that are connected to the electric grid by stabilizing the power output, provide voltage support, and also provide value through energy arbitrage, among others.

2.3.2 Supercapacitors

Supercapacitors/ultracapacitors (also known as Electro-Chemical Double Layer Capacitors (EDLCs)) contains two conduction electrodes, a porous membrane separator and an electrolyte. A porous carbon is used as the current collector; thus, a greater surface area is obtained, consequently storing an increased amount of energy on the collector [46]. There is no chemical reaction that occurs in this technology. This has become an alternative to the traditional capacitor used in several electronic applications and, in general, batteries. Supercapacitors have the advantages of high-power density and high peak power output. It also has a long life span since it can be charged and discharged up to millions of times without degrading [52]. Supercapacitor's energy density has been improved due to the use of high-surface material such as activated carbon. This has attracted its applications in power systems, communication, and spacecraft technology, especially in pulse load (i.e., modern radars). This form of load may cause severe power and thermal disturbances in the microgrid, and supercapacitors have a rapid response in regard to power levelling and power balancing applications with proper control systems hence resolving these issues [53]–[55]. The capacitance of supercapacitors is not constant; somewhat, it varies with the voltage shift, which is dependent on the current demand and supply of SC. Consequently, the magnitude of the charge often varies [51].

The operating voltage of the current supercapacitors is low (below 3.5V); hence for high voltage applications (200-400V), they must be connected in series [56]. As a result, supercapacitors are very expensive ($6000/kWh) and have a high self-discharge rate (up to 40% per day). To address these challenges, ongoing research focuses on cost-effective multi-layer supercapacitors composed of materials such as carbon, graphene or paper [47], [57]. The application of the supercapacitor in microgrid operation for both grid-connected and isolated modes under stable and unstable conditions was demonstrated in [58].

2.3.3 Flow Battery Energy Storage (FBES)

As a traditional battery, Flow Battery Energy Storage (FBES) operates by converting chemical energy to electrical energy. Like conventional batteries, FBES uses two aqueous electrolytic solutions in different tanks. Every aqueous solution is pumped through an electrochemical cell during regular operation, where a reversible electrochemical redox reaction occurs, and power is produced. FBES technology is relatively new, and there are three major types of commercially produced flow batteries: Vanadium Redox Battery (VRB), Polysulfide Bromide Battery (PSB) and Zinc Bromine Battery (ZnBr). Combined with low self-discharge and the ability to withstand maximum discharge without damage (electrolyte is contained in separate tanks), FBES technologies have both a long calendar life cycle and a low maintenance requirement, thus, making them suitable for long-term energy storage [46], [59], [60].

Despite their benefits, flow batteries have always been more expensive than batteries and have a shorter service life. According to research published in [61], a means to substantially reduce the cost of redox flow batteries to \$25/kWh or less has been developed (using inexpensive ingredients such as manganese and sulphur, which are abundant in nature). Redox flow batteries are expected to provide a cost advantage over Li-ion batteries, which currently cost \$135/kWh [62]**.**

2.4 Microgrid operation, control, and islanding

A microgrid control system refers to a series of software and hardware components that ensures that the microgrid operates in an optimal, reliable, and stable way. A hierarchical control system has been proposed and generally adopted as a structured solution for efficient microgrid management [63]. As can be seen in Figure 2.4, the microgrid control system can be divided into three hierarchies: primary, secondary, and tertiary. Primary control is engineered to manage distributed generation (DG) components by adding virtual inertia or regulating their output impedances. It controls local power, voltage and current. It reacts first in case of any set parameter change by stabilizing frequency and voltage using droop controllers. Secondary control is a slower mechanism that rectifies steady-state errors in the frequency and voltage magnitudes of the primary control system. It also handles issues related to power quality control, including voltage imbalance and harmonic compensation. The tertiary control tier introduces intelligence to the entire system and focuses on energy management and optimization of the system by factoring in microgrid stability, environmental problems, and economic issues. This tertiary control layer is configured at an entity called the microgrid central controller (MGCC). The MGCC decides the control action for energy management based on the active power capacity of distributed energy resources (DERs), microgrid consumption and storage specifications. The communication network enables the MGCC to disseminate real-time power set points to the DERs and loads. In addition, each decentralized controller assures that there is no violation in the power reference from the main control level [41], [63], [64]. Mostly, this research will concentrate on the tertiary control layer as it is the first natural step towards optimal energy management in microgrids. This layer is responsible for increasing the sustainability of the supply and demand balance by minimizing the economic costs.

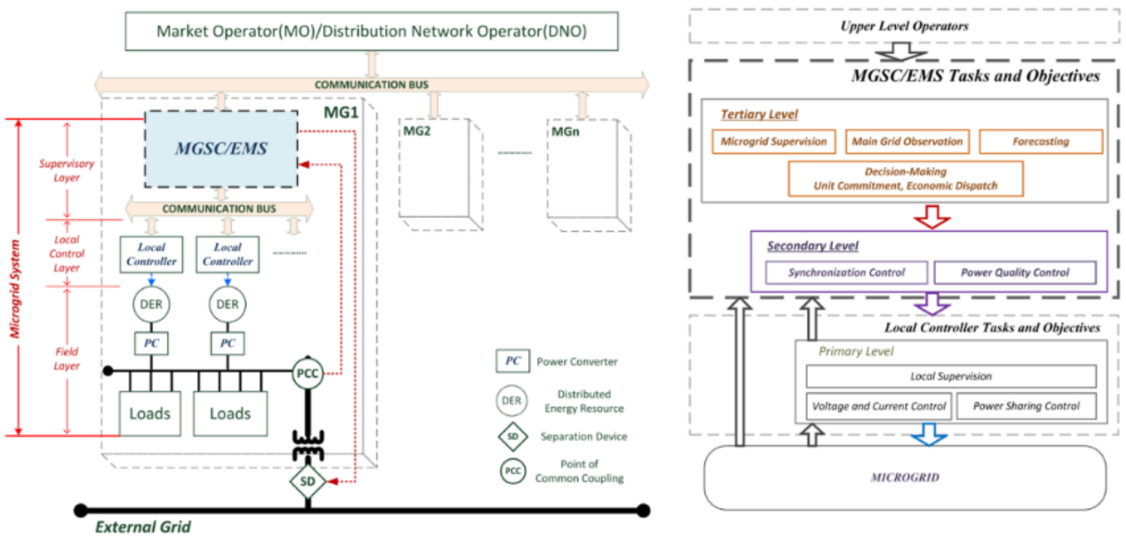


Figure 2.4: General structure of microgrid control systems [59]

Besides the control structure, microgrid control methods are equally important. Authors in [40] point out that the main microgrid control methods are: master-slave control and peer-to-peer control. Master-slave control is associated with voltage-frequency (V-f) control and /or P-Q control to the appropriate levels of both the active and reactive power. Peer-to-peer control involves the management of frequency-active power (f-P) and voltage-reactive power(V-Q). The two main proposed control methods have their own benefits and drawbacks.

Two main architectures for microgrid control are centralized and decentralized. The centralized approach is easy to implement and has a standardized procedure where control is done centrally. In a distributed architecture, each DG, load and inverter has its own controller, which pursues specific goals. In decentralized architecture, the number of signals transmitted between the various units and the microgrid controller increases as the microgrid size enlarges, thus requiring a higher communication bandwidth. A decentralized control scheme is implemented using two approaches, the multi-agent system and the cooperative system. In the theory of multi-agents, each of the controllable components in the microgrid, including power inverters, distributed generators, and loads, have agents associated with them, where the theory multi-agents guide their communication and coordination. Whereas in cooperative control, all participants (primary and secondary) co-operate with one other and operate as a single entity to achieve common objectives, which are to stabilize voltage and frequency back to its nominal range[40]. More information about control architectures can be found in [40].

The main advantage of microgrids is their capability to separate from the main power network through upstream switches. Islanding may be done for both economic and reliability purposes. The microgrid is islanded from the main grid during power grid disruptions, and local DERs provides a stable and uninterrupted supply of user loads. The microgrid master controller will give efficient operation by controlling the frequency and voltage to be within the normal levels. The isolated microgrid is resynchronized back with the main grid once the fault is alleviated. Resynchronization refers to the reconnection of the isolated microgrid to the main utility network while ensuring that the microgrid voltage and frequency are synchronized with that of the main power grid. Significant losses owing to current surges may occur to the microgrid elements during the switching phase if synchronization is not ensured [40], [65].

2.4.1 Microgrid clusters

Microgrid clusters are a combination of multiple autonomous microgrids that work together. Currently, studies are being done to see the possibility of having microgrid clusters in the future smart grid. Microgrid clusters can be studied from various perspectives. Microgrid clusters' economic benefits are analysed in [64] and [65]; their studies reveal that operating clustered microgrids reduces the carbon footprint and consumers energy cost. They also come in handy in addressing load growth problems in remote areas. They also enable efficient energy trading by encouraging collaboration.

Authors of [66] proposed a game-theoretical approach of modelling and analysing strategic occasions emerging from the interactions of various decision-makers in a decentralized micro-grid environment, such as intelligent agents, decentralized computing, intelligent meters, intelligent sensors, and a robust and fast communication network. Microgrid clusters is a fertile area of research that will significantly benefit remote regions.

2.4.2 Microgrid economics

Although the establishment of microgrids may require high initial costs, well-planning its operation will lead to significant economic and social benefits. Microgrids can potentially generate cheap power stemming from local renewable sources. They can also produce energy in high market price hours and even during power grid congestion, thereby reducing power purchases from the power grid. Moreover, microgrids can sell back their surplus power to the primary grid and earn revenue, and this will directly translate to less expensive energy for the local users. Furthermore, local microgrid generation may also support the entire power grid by reducing congestion in the transmission and distribution (T&D) networks and allowing an optimal economic dispatch of the available energy resources in the power system. The economic benefits of microgrids are debated extensively in [40] and [67]. Microgrid advantages, however, need to be analysed and contrasted to their investment costs to ensure a total return on investment and further justify the deployment of microgrids [68]. Therefore, efficient planning models are needed to ensure the economic viability of microgrid deployment and further explain the investment.

2.5 Power Electronics Applications in Microgrids

Power electronics interfaces are necessary for integrating most of the DERs into the microgrid network [69]. For instance, solar PV and ESS (outputs DC power), wind turbines (require power quality and frequency improvement) and microturbines (high-frequency AC power) require electrical interfaces such as AC/DC and DC/AC converters to communicate with the electric system. While power electronic devices improve the integration and controllability of DERs, they also present new challenges in terms of control and security. In isolated microgrids, synchronous generators can act as a source of voltage to manage the grid's frequency; however, if they are absent, electronic power converters need to act as a source of voltage. In grid-tied mode, converters work as current sources that feed the microgrid [63].

Besides efficient connection of DERs to microgrids, other benefits of power electronic devices include power quality improvement (improving harmonics), providing extremely swift switching capabilities for sensitive loads, providing reactive power control and voltage regulation and reducing or eliminating fault current [40].

Since most DERs provide DC power and, most end loads, such as power electronics, LED lights and variable speed drives for heating, ventilation and air conditioning, operate on DC power, all-DC bus microgrids have been suggested to avoid power losses from switching between DC and AC and often again back to DC power. In addition, protection in DC systems is much simpler since faults can be isolated by blocking diodes, and problems of synchronization, harmonic distortion, and problematical flowing reactive currents can be mitigated. Finally, a grid-connected DC-based, non-synchronous system simplifies the integration of microgrids to the main AC grid, and it further allows for an easy plug-and-play functionality whereby components can be introduced into the grid without significant re-designing of the system [39], [70]–[72].

2.6 Benefits of Microgrids in the Energy Sector.

Studies have been done on the benefits of microgrids with socio-economic benefits like promoting rural electrification, increased urbanization, improved healthcare, increased security, and so much more. However, microgrids can benefit the main grid in many ways. Authors in [73] identified some of the advantages of microgrids, including enhanced reliability and resiliency, reduced emissions, reduced costs of periodic system upgrades, increased energy efficiency and quality of power, and lowered cost of energy. Nonetheless, the most critical microgrid benefits and appealing to grid operators to boost grid efficiency are reliability, resiliency, and power quality [74].

2.6.1 Reliability

One of the most critical advantages of microgrids is increasing the reliability of the consumer power supply. In general, customer reliability is measured in terms of network and consumer average interruption frequency and length of the interruption. Outage factors, such as floods, generator/transmission line loss, infrastructure failure, etc., severely affect system reliability levels. Still, when a local microgrid is implemented, these metrics can be improved dramatically. This is attributable to the intrinsic ability of microgrids to island themselves from the main power network seamlessly. Also, generators in microgrids are near customer loads, making their generation less prone to transmission and distribution grid disturbances and infrastructure issues. Additionally, most microgrid master controllers are integrated with demand response actions which can vary/adjust loads to increase system stability and reliability. Thus, improved reliability will increase economic benefits for both the consumer and energy provider.

2.6.2 Resiliency

Resiliency refers to the ability of power systems to endure low-probability high-impact incidents by reducing potential power outages and returning rapidly to normal working conditions [75]. Such events generally involve extreme weather incidents and natural disasters, such as hurricanes, tornadoes, earthquakes, snowstorms, flooding, cyber-security threats, malware threats, etc. If these incidents compromise the main grid and crucial elements, i.e., T&D infrastructure and generation components, are seriously damaged, service providers may be interrupted for days or even weeks. The effects of these events on customers can be minimized by deploying microgrids that ensure local loads are supplied with power even when the supply is not available from the main grid.

2.6.3 Power quality

Over the last decades, consumer demand for premium / higher power quality has increased dramatically due to the increasing deployment of voltage-sensitive loads, e.g., electronic loads and LEDs. As a result, power utility companies are indeed searching for effective and efficient ways to boost power quality problems by resolving the prevalent concerns arising from harmonics and voltage. Microgrids are best suited for this because they provide a fast and efficient response to power quality needs by allowing local frequency control, voltage, load and fast response from energy storage systems [40].

2.7 Microgrid Energy Management Systems

In the standard IEC 61970, which concerns the interface of energy management system (EMS) applications in power management systems, the International Electronics Commission (IEC) defines an EMS as “a computer system comprising a software platform providing basic support services and a set of applications providing the functionality needed for the effective operation of electrical generation and transmission facilities to assure adequate security of energy supply at minimum cost" [15]. A microgrid EMS, which has these characteristics, typically includes decision-making strategy, load prediction modules, Human Machine Interfaces (HMI), and supervisory, control, and data acquisition (SCADA) modules, among others, ensure that EMS decision-making techniques are implemented efficiently by transmitting optimal decisions to every generation, energy storage, and load unit [15].

Microgrid EMS has two types of supervisory control architecture: centralized and decentralized as has been mentioned earlier in Section 2.4. The central controller in a centralized case collects all the data, including renewable DER power production, cost-functions, meteorological data, and each consumer's power consumption pattern, among other things. The centralized EMS then finds the best energy schedule for the microgrid and communicates this information to all local controls (LCs). On the other hand, in a decentralized EMS design, the MGCC transmits all information to the LCs in real-time. Each LC submits to the MGCC a load demand or power generation request for the present and future. The MGCC chooses the best schedule and informs the LC. The latter may oppose the current information and continue to negotiate until global and local goals are met. Microgrid EMS solutions have expanded beyond economic dispatch and unit commitment with integrating renewable energy, energy storage, electric vehicles and demand response. Other solutions include DERs and load scheduling, system loss and outage avoidance, renewable energy intermittency and volatility control, and microgrid operation that is cost-effective, reliable and sustainable.

Many scholars have used various solutions to tackle these energy management strategies to accomplish the microgrid's optimal and efficient operation. The following subsections provide an in-depth analysis of these tactics and solution options.

2.7.1 EMS based on classical methods

The literature has offered various techniques for microgrid energy management. Most studies on optimization strategies focus on energy scheduling between DERs, battery energy storage systems, and connected loads for lowering electricity costs, increasing generation efficiency, conserving energy and ensuring power system stability [15], [76]. Because of the unpredictable nature of microgrid optimization and the significant financial benefits that improved solutions potentially provide, considerable effort is being put into developing better optimization algorithms and modelling frameworks. Examples of classical methods applied in literature are priority list and Langrangian relaxation techniques. The priority list technique entails listing all the possible solutions that satisfy the optimization problem's predetermined system objective. To determine the best solution in the developed merit list, intuition is applied so that the solution that minimizes system cost is selected based on some simple pre-set rules and constraints. This technique is the most basic optimization technique that is applied to power systems for unit commitment [77]. Delarue et al. suggested an improved priority list strategy in conjunction with mixed-integer linear programming techniques to aid with the search of power units operating with minimum cost implemented under low system load characteristics [77]. Yang Tingfang [78] also developed a priority list technique by combining several highly efficient methodological procedures based on the PL method to solve the UC problem. In the work the lambda iteration method is introduced to solve the economic dispatch function [78]. Priority list methods are simple to implement and converge fast; however, they return erroneous solutions if they are not optimally initialized.

2.7.2 EMS based on linear and mixed-integer linear programming methods

Linear programming methods are typically used to search for optimal solutions in linear, continuous, and differentiable environments. Linear programming (LP) technique is a computationally efficient method for mathematical optimization; however, it only works with linear objective functions. For scenarios where the mathematical function is not linear, it must be linearized. This linearization comes with some disadvantages as assumptions that may render the solution suboptimal are introduced to the mathematical function. Nevertheless, if at least one of the control variables is conditioned to be an integer value, mixed-integer linear programming can be leveraged to solve energy systems optimization problems successfully. Combined with the Model Predictive Control (MPC) paradigm, these methods provide a robust real-time controller optimization technique. MPC uses the system's model to forecast its future state, then tackles the online optimization problem to find the best control action to achieve the optimal trajectory.

Zhai et al. [79] suggested a robust predictive control that could be extended to an islanded microgrid to deal with a complex system. The management model used mixed deterministic integer programming (MILP) intending to minimize system costs. The microgrid consisted of wind and solar PV generators, an energy storage system, and controllable loads. The model significantly improved the system reliability and reduced the operational cost. Zhang et al. in [80] proposed an MPC technique to manage a microgrid incorporating distributed renewable energy resources effectively. The main objective of the model was to minimize the operational cost considering both the generation and energy demand constraints. The method provided accurate schedules for the microgrid, but its computational time was high. Model-based techniques are highly dependent on an explicit forecast of future uncertainty, and their effectiveness gets compromised when inaccurate system models, imprecise forecasts or prediction horizon choices are used.

Classical methods based on linear programming laid the foundation of the other optimization techniques used today. These techniques are simple to implement, but they quickly get trapped in a local optimum for non-convex optimization functions. As a result, they get slower as the systems get larger, leading to high computational time. Furthermore, when handling systems with randomness like renewable-based microgrids, these methods fail because of their simplistic model. Thus, heuristic techniques, which have been reported to outperform classical methods, are generally adopted for stochastic environments.

2.7.3 EMS based on meta-heuristics techniques (genetic algorithm and swarm optimization)

Genetic algorithm (GA) is one of the evolutionary-based methods. Evolutionary methods are motivated by Charles Darwin’s theory of natural selection. The theory states that all species of organisms arise and develop through the natural selection of minor, inherited variations that increase the individual's ability to compete, survive, and reproduce [81]. An initial population of random solutions (population) is created in a GA. After evaluating each member of the population for fitness (using the objective function), random processes such as crossing over, mutation, and fertilization are used to make changes to the population [81]. In [11], a multi-objective optimization model is introduced to minimize the cost of power production and optimize the lead-acid batteries' useful life through a non-dominated GA. The results show that GA can optimize system operations under various conditions and allow users to achieve optimal operating systems. In [82], and enhanced fast genetic algorithm is used to evaluate an economic load sharing scenario in a standard microgrid by reducing the operational cost, maintenance and pollution. GA technique has been used extensively in the literature to perform power scheduling; however, it has been reported that swarm intelligence based algorithms outperform GA to achieve optimized results [15], [40].

Swarm intelligence algorithms are motivated by the social behaviour of organisms, i.e., swarm of birds or school of fish, as they maneuver their environment to search for food. We have several algorithms under this category; however, the most famous of them all is particle swarm optimization (PSO). In PSO, particles move in the constrained multi-dimensional search space in search of the global optimal point. Each particle updates its position during motion based on its best experience and the global best (best position achieved by the neighboring particles) [83]. As a result, PSO is said to be more efficient and faster to converge than GA.

An ant colony optimization-based two-layer EMS model is presented in [84] to reduce the operational costs of an islanded microgrid. It covers renewable energy resources, conventional generators, battery bidding costs, load shedding penalty costs, and demand response incentives in day-ahead and 5-minute interval real-time scheduling layers. With experimental validation, three scenarios of regular operation, sudden high load demand, and plug and play capability are investigated. The proposed solution lowers microgrid's operational costs by nearly 23% and 5%, respectively, when compared to modified traditional EMS and PSO-based EMS. The authors of [84] employed a PSO method to build an optimal EMS for grid-tied microgrid that considers renewable energy uncertainty, load demand, and electricity tariffs.

In comparison to GA, the results obtained by the PSO method are demonstrated to be better. Li et al. [85] presented a new PSO-based optimal energy scheduling technique for industrial microgrids in both islanded and grid-tied modes. The objective function in the islanded mode was to reduce the microgrid's operation and maintenance costs. For the grid-connected scenario, on the other hand, the aim was to maximize energy trading profit with the main utility grid. Compared to GA, the suggested PSO approach outperformed GA in terms of global optimum solution and computing time.

Heuristic-based techniques are often slow and incapable of running online, despite their capacity to handle stochasticity in optimization. As a result, they must be hybridized with fast algorithms such as dynamic programming (DP). Authors in [10] presented a hybridized PSO_DP optimization approach for a grid-tied PV battery system. First, the PSO algorithm was used to find the optimal battery SoC for the system. Then, the DP algorithm was used to do the actual battery control in 10 minutes for the whole optimization horizon. It is also noteworthy that these algorithms are designed for static optimization settings; thus, they have limited capacity to get optimal results in dynamic stochastic environments.

2.7.4 EMS based on (Deep) Reinforcement learning methods

Reinforcement learning (RL) is a formal computational framework for decision-making in sequential problems to account for stochasticity within the system. RL is formalized by programming a software agent through a careful reward scheme. The agent acts as the decision-maker that learns how best to operate by iterative trial and error. RL is a goal-oriented algorithm, such that the designed agent should learn how to maximize its cumulative reward in its stipulated environment. The simulation environment must be formalized as a Markov Decision Process (MDP) for the learning to occur. An MDP is a mathematical framework for formulating situations that are partly random and partly deterministic. In the standard RL framework, the learning agent repeatedly observes a state in the simulated environment and acts as the set of actions provided. When the agent acts, a state transition occurs. Also, the agent receives a reward based on the state and action performed. Thus, the agent's goal is to learn to maximize the long-term total reward it can receive [83], [86].

In an energy management context, an agent can be trained to make near-optimal real-time decisions on power flow in grid-tied microgrid scenarios utilizing multiple RL methods. The state can include the current load profile, the current grid tariff, current generation from the different sources of power and the battery state of charge, among others. On the other hand, the action space may be constrained depending on the objective of the environment. Most researchers tend to model the action space as a vector of three decisions for battery scheduling problems, namely charge, discharge, and idle [24], [87]. The reward must be appropriately designed to ensure that system objectives are achieved during the optimization process [88], [89].

Developed by Christopher Watkins in 1989 [55], Q-learning is one of the most popularly applied RL algorithms in the microgrid power scheduling sphere. An off-policy method imposes a more relaxed computational burden than on-policy algorithms such as DP and Monte Carlo sampling [55]. It also comes with the advantage of simplicity and versatility, i.e., the Q-learning construction may be framed for a large variety of sequential decision-making problems. Several scholars have used this approach to solve the microgrid energy management problem to take advantage of these characteristics. For instance, Brida et al. [22] used batch reinforcement learning to implement a microgrid EMS optimizes battery schedules. The charge and discharge efficiency of the battery and the microgrid nonlinearity caused by inverter efficiency were considered. Elham et al. [23] presented a multi-agent RL method for adaptive energy control in a microgrid. The results indicate that the grid-tied microgrid learned to reduce its dependency on the utility grid significantly. Authors in [24] presented an optimal battery scheduling scheme for microgrid energy management. A Q-learning technique is implemented to reduce the overall power consumption from the utility in [22], and simulation results show that the algorithm reduces dependency on the main grid. However, this work fails to consider battery trading with the utility and the impact of the battery life cycle from those actions. In [25], Zeng et al. suggested an Approximate Dynamic Programming (ADP) method to tackle microgrid energy management, considering demand volatility. Authors in [67] explored the feasibility of applying RL to schedule energy in a grid-connected PV-battery electric vehicle (EV) charging station. From the results, the algorithm managed to successfully obtain a day-to-day energy schedule that decreases the transactive cost between the microgrid and the utility grid. Authors of [26] and [27] proposed a battery management strategy in microgrids using the RL technique. However, the incorporation of the battery wear cost in the EMS model was absent. The work reported in [28] used RL to develop a real-time incentive-based demand response program; the RL algorithm focussed at aiding the service provider to buy power from its subscribed customers to balance load demand and power supply and improve grid reliability. R. Lu et al. [29] leveraged RL to design a dynamic pricing demand response (DR) algorithm in a hierarchical electricity market. From the results, the algorithm is seen to successfully balance energy supply and demand and reduce energy cost for consumers. A RL method combined with Monte-Carlo Tree Search and knowledge rules is used to optimize the system. Although, the simulation results show the efficacy of the proposed algorithm, a detailed model of battery degradation is not considered in [32]. Y. Shang, et al.[90] proposed an EMS model aimed at minimizing the microgrid's operation cost, considering the nonconvex battery degradation cost. E. Samadi, et al.[32] proposed a multi-agent based decentralized energy management approach in a grid-connected microgrid. The different microgrid components were designed as autonomous agents who adopted a model-free RL approach to optimize their behaviour.

The most basic and widely used RL approaches, namely Q-learning [91], suffer from several challenges, including inefficient data utilization, inability to handle continuous/large state-space, and curse of dimensionality, which cause the method to fail for large-scale tasks. The study in [87] designed a Q-learning technique to manage energy in a grid-tied solar PV battery microgrid optimally. A comparative case study using both winter and summer data profiles was conducted on the algorithm. Although the employed Q-learning technique was simple and the proposed algorithm was reported to learn an optimized battery schedule while minimizing system costs, the approach is unscalable. As the state spaces increases, the Q table grows infeasibly large, and most state-actions pairs are not visited during training.

Furthermore, sequentially updating the algorithm leads to low usage of data and high state transition correlations, which leads to poor learning capabilities. Authors in [22] presented a batch RL technique to solve inefficient data utilization by employing a memory reply and training the algorithm with a batch of previous experiences. Deep reinforcement learning (DRL) approaches, on the other hand, use artificial neural networks as function approximators, allowing them to learn continuous state-action transitions in the face of uncertainty. Deep neural networks will also enable the utilization of continuous and large-dimensional state spaces and the extraction of hidden features. As a result, the DRL agent is capable of overcoming the environment's uncertainties and partial observability [92], [93].

Several studies have demonstrated interest in DRL applications to solve microgrid control challenges, owing to DRL's recent achievements in tackling complicated tasks, as evidenced by Alpha Zero's superhuman performance in several complex computer games [94]. The authors in [95] presented a multi-agent DRL to manage energy in interconnected EV charging stations. The results obtained indicated that adding a communication model improves the cooperative working of the agents in multi-agent systems. In [2], a novel microgrid architecture that included a wind generator, battery, price-responsive and thermostatically controlled loads, and a utility grid-tie is presented. The proposed EMS was modelled to coordinate the various energy sources, and DRL algorithms were used to analyse multiple scenarios. Compared to the other strategies studied, the authors [2], suggested an improved asynchronous advantage actor-critic algorithm (A3C++) demonstrated improved convergence and superior control strategies. Authors in [96] investigated a dynamic pricing and energy consumption scheduling program in the microgrid. The service provider (i.e. the microgrid's owner) operates as a broker between the utility company and the customers, buying electric energy from the utility company and selling it to the customers. The RL algorithm was designed to overcome the difficulties of developing an adaptive dynamic pricing scheme in the face of diverse sources of uncertainty, i.e., consumers' load demand levels and the wholesale electricity cost. In [97], Wang and Huang looked at interconnected autonomous microgrids and devised a cooperative energy trading and scheduling technique. Work presented in [98] shows that high-rise buildings' on-site wind power generation can sustain all the city's electric vehicles. The coordination of electric car charging with locally generated wind power in a microgrid of buildings using the Markov decision process was examined because the charging demand of electric vehicles does not always correspond with the unreliable wind output. A Markov perfect equilibrium policy investigated the users' long-term load scheduling problem. Bahrami et al. [99] investigated the users' long-term load scheduling problem and developed an online load scheduling learning technique based on the actor-critic strategy. A deep Q-learning algorithm is used to learn the optimal decision-making policies. Simulation results on actual data confirmed that the approach was effective, outperforming the rule-based heuristics methods. Ji et al. [33] showed how to use the deep Q network (DQN) approach to schedule microgrid's power generation and consumption while taking demand response and power generation prediction into account. The researchers discovered that DQN reduced energy costs by 20.75 per cent, compared to 13.12 per cent leveraging a fitted Q-iteration method. François et al. [34] employed a DQN-based approach to obtain a battery and hydrogen storage schedule for a microgrid. They utilized convolutional neural networks to learn the best battery schedule under unpredictably high load demand and power production conditions. Lu et al. [100] employed a DQN technique to trade energy between a grid-tied microgrid and saw a 22.3 per cent increase in self-consumption of power produced within the microgrid.

2.8 Summary

Due to increased complexity, uncertainty, and data dimensionality, traditional approaches often encounter obstacles when handling decision and control problems in microgrid energy management systems. As a result, data-driven approaches are being thoroughly investigated to overcome these issues. One of these data-driven strategies is reinforcement learning which has been used to handle various complicated sequential decision-making problems in the industry, including those involving power systems. This chapter particularly examined the different techniques used in the literature to optimize energy management in grid-connected microgrids. It was noted that linear algorithms like linear programming (LP) and mixed-integer linear programming (MILP) had been utilized to find solutions quickly in less intricate domains. Still, they have limitations when it comes to dealing with stochasticity. Heuristic techniques such as genetic algorithm (GA) and swarm intelligence algorithms, including particle swarm optimization (PSO), were also extensively applied in the literature. These techniques employ many non-learning agents to get optimal solutions for the optimization model. Owing to their capacity to accommodate stochastic system variables, these approaches outperform linear optimization algorithms. However, these techniques are slow and unable to handle dynamic online operations.

The reinforcement learning (RL) paradigm, in which a microgrid's dynamics are learned by an agent interacting with its components, has gained a lot of interest in the application of energy management. It was discovered that this method is more capable of dealing with dynamic stochastic situations than other optimization approaches due to its learning element and capacity to generalize solutions. Thus, this thesis proposed reinforcement learning for optimizing power flow in grid-tied microgrids.

It was established that recent advances on the implementation of RL in microgrid energy management barely incorporated battery degradation cost in the mathematical formulation. Most studies only consider the generation cost and power exchange cost. Estimating the degradation process is very difficult and finding a precise and straightforward mathematical degradation model that can be used in the energy management algorithm is not easy. As the charging and discharging behaviours of a BESS directly impact its life span, lifecycle degradation costs should be factored into the complex dispatch model of BESS. It is important to note that Lithium-ion batteries are expensive and incorporating a battery degradation model while computing the overall system cost is critical as a realistic system cost estimate is established. Thus, this thesis also reports on the development of an EMS for a grid-tied solar PV-battery microgrid considering battery degradation in the energy trading process, focusing on reducing the strain on the battery.

3 REVIEW ON REINFORCEMENT LEARNING

A general overview of machine learning algorithms will be presented, as well as current state-of-the-art RL techniques. This chapter focuses on the potential application and value generation of RL in power systems. The review begins with an overview of artificial intelligence, reinforcement learning, the Markov decision process (MDP), and the various methods for solving the MDP. Also, the use of deep learning in conjunction with reinforcement learning will be briefly discussed.

3.1 Introduction

Due to rapid developments in computing hardware and the declining cost of data storing systems, artificial intelligence (AI) has lately prompted a paradigm change in various sectors throughout the continent. Machine learning is currently the most prominent issue in AI. The scientific discipline of machine learning (ML) is defined as the study and development of algorithms and statistical models that enable machines to learn tasks explicitly without being taught to do so [101]. Supervised, unsupervised, semi-supervised, and reinforcement learning are the four types of machine learning categories.

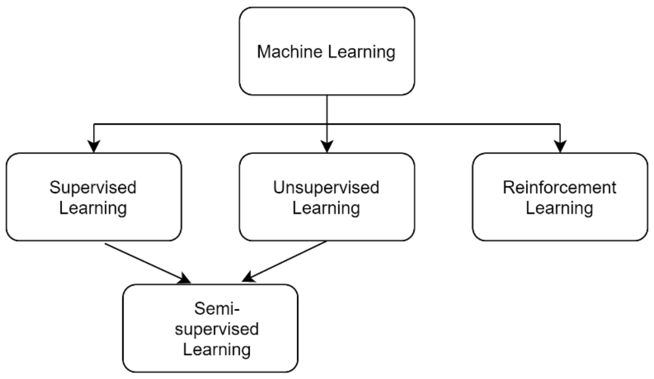


Figure 3.1: Outline of machine learning field [101]

The software agent learns the input-output mapping using a training set labelled by subject matter expert(s) in supervised learning [17], [101]. Usually, in supervised learning, the algorithm generalizes across various training samples and then utilizes that information to correctly predict output (labels) for data that hasn't been seen yet. Because the supervised learning agent essentially duplicates the expert's labelling behaviour, the agent's performance cannot beat that of the subject matter expert or supervisor. On the other hand, unsupervised learning is most commonly employed to uncover hidden patterns in unlabelled data sets. Dimensionality reduction, feature extraction, and clustering are three of the key aims of unsupervised learning. By merging ideas from supervised and unsupervised learning, semi-supervised learning is created. Because manually labelling data is time-consuming, semi-supervised learning, on the other hand, can be used to train a small data set of the labelled dataset while extracting more relevant information from the unlabelled dataset [101].

Reinforcement learning is a method of learning that employs trial and error. The system is made up of a software agent that observes a state that represents the environment and reacts to it by taking action. Simply put, the agent will get a positive reward when it takes good actions and negative rewards for bad actions. When the agent takes a wrong action, it will be less likely to choose that action later. Similarly, it is more likely to choose a similar action given the same observed state when it gets a positive reward. Allowing the agent to experience various states and actions will enable it to learn a behaviour that leads to many positive rewards. RL is derived from how animals and humans obtain knowledge of living optimally in an environment that is initially unknown to them [86]. One way that animals develop complex behaviours is by learning to acquire more rewards and avoiding penalties. For instance, a few examples of an animal learning to trap its food and a baby learning how to walk. It was established that if animals can learn to behave optimally in their environment by trial and error, and motivated by reward signals, then such a learning methodology could be transmitted to cognitive science and computing. A reinforcement learning agent learns from its own experiences of the environment. An experience is defined as a set consisting of a state, an action, the next state, and a reward.

Figure 3.2 illustrates the interaction between an agent and its environment. By seeking to maximize the total reward in a set of several experiences, the agent learns a policy that maps every state to the best action. Thus, every learning problem portrays similar modules: a learner (agent), a teacher (reward function), a performance measurement of how well the learning agent is behaving (usually tests represented numerically by rewards), and a couple more variables to be considered; several properties comprising a Reinforcement Learning problem are explained in the next section.

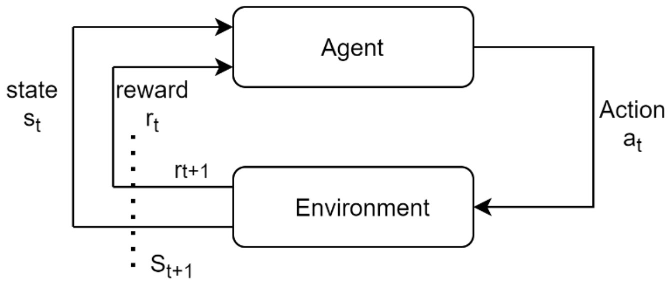


Figure 3.2: The general Markov decision process framework

3.2 Markov Decision Process

Figure 3.2 depicts the reinforcement learning paradigm, which is made up of two parts: a learning agent and a system (environment). The agent is a decision-maker who is always learning (i.e., RL algorithm). Through meaningful interactions with the environment, the agent learns about the unknown environment to overcome it. Everything that the agent cannot modify at will is included in the environment. The Markov decision process (MDP) is used to formalize the decision-making process of reinforcement learning, ensuring that the agent's experiences converge to an optimal policy as it interacts with the environment. MDP is a way of modelling discrete stochastic control problems and formulating sequential decision making in environments where the outcome is partly random and/or partly deterministic [17]. When agents rationalize about how to plan and act in the face of uncertainty, MDPs provide a formalization for that. MDPs are generally formulated as four elements tuple (S, A, P, R), where S is the state space, A is the defined action space, P is the state transition probability, and R is the reinforcement/ reinforcement/reward function [102]. Suppose the state space is such that the future state can be calculated using the current state variables and the current action without remembering the events that led to the present state. In that case, the process is said to be "Markovian." This condition makes it possible to define an optimality criterion in the learning process [103].

3.2.1 State and state-space

State-space in the RL problem is the set of possible states that the agent can occupy at different instances of time [101]. At every timestep, the agent must be at a state that is part of the entire state space. The agent’s state at time t is denoted as s_t which represents all information about the environment at that position that the agent may require to make the correct decision. The entire state space is then taken as S, which is the union of the states. This can be expressed as $S = s_0 \cup s_1 \cup, \dots \cup s_{T-1}$ for all time slots, $t = 0$ to $t = T - 1$. To get to the final state s_{T-1} from the starting state s_0, the agent must take a sequence of actions $a_0, a_1, \dots, a_{T-1}$

3.2.2 Action and action space

An action is a choice that the agent makes in each state. Every state has a specific action space from which an action may be selected. At any instance t, the agent can take an action a_t from the set of allowable actions in the action space [83]. The union of all action sets in the entire action space is enumerated as equation (3.1) [83] below.

$$\mathcal{A} = \mathcal{A}_0 \cup \mathcal{A}_1 \cup, \dots \cup \mathcal{A}_{T-1} \tag{3.1}$$

3.2.3 State Transition

State transition probabilities describe the probability of transitioning between state s_t and s_{t+1} and receiving a reward r after performing an action a_t. It is mathematically expressed, as shown below in equation (3.2) [88].

$$P_{ss'}^{a} = \Pr\left(s_{t+1} = s' | s_t = s, a_t = a\right) \tag{3.2}$$

Where p defines the system's dynamics and Pr denotes the probability of transiting to the next state. Depending on the system's environment, the state transition can be probabilistic or deterministic. In a deterministic environment, pr is one where the same action, if selected in the same state, will make the system transition to the same next state[88], although in a probabilistic environment, like in partially observable MDPs p is a probability distribution satisfying the equation (3.3) below [101]:

$$\sum_{s' \in S} \sum_{r \in R} p(s', r | s, a) = 1, \forall s \in S, a \in A \tag{3.3}$$

3.2.4 Reward/ Reinforcement Function

Modelling a reward function is a critical issue in RL. A reward is a real-valued scalar quantity that relays the aim of the learning activity to the learning agent. The designer of the MDP must formulate an intelligent reward function to achieve the purpose of the learning process [89]. In RL, the agent learns by reinforcement (as in psychology). A negative reward leads to undesirable behaviour. Conversely, a sequence of positive rewards leads towards an appreciable policy. The agent aims to maximize the accumulated sum of rewards (see equation (3.4)) [83],

$$R_t = r_{t+1} + r_{t+2} + r_{t+3}, \dots, r_{T-1} = \sum_{k=0}^{T} r_{t+k+1} \tag{3.4}$$

where R_t represents the return, t a particular time step up to $T - 1$. In general, for stochastic environments, the goal is to maximize the expected return value within a task. The reward does not have to represent the exact effect of the chosen action accurately. However, it shows the progress of the learning agent in achieving the objective function in the system being controlled. An action is not only as good as the immediate reward it returns but also the state into which it leads [83]. Thus, a good action will produce high immediate rewards and lead to a state where there are potentially high rewards to be received [6].

3.2.5 Policy Function

A behaviour translates into a control policy; the reinforcement agent selects an action in each state through its policy π. The agent's policy decides what action to take in a given state by mapping the state space to the action space, $\pi: S \rightarrow A$. The policy can either be probabilistic or deterministic. A deterministic policy maps a given state to the same action every time, while a stochastic policy maps the state to a probability distribution over the action space [17]. For the deterministic case, the policy function π is given by equation (3.5) [17] ,

$$\pi(s) = a \tag{3.5}$$

where a is the action chosen by the policy. For the stochastic case, it gives the probability of selecting action a in state s as in equation (3.6) [17]

$$\pi(a|s) = \mathbb{P}(a|s) \qquad (3.6)$$

Theoretically, the policy gathering most rewards in a particular environment is considered an optimal policy, denoted by π^*. Moreover, after applying a learning algorithm coupled with a proper exploration strategy until convergence, given sufficient episodes, the policy obtained can be considered "optimal" or "sub-optimal".

3.2.6 Value Function

A Value function computes the goodness of a policy, given a state, $s_t \in S$ and following the same policy π after that. The usefulness of a policy comprises a gathered discounted sum of reward as shown in equation (3.7) [104].

$$V^{\pi}(s) = \mathbb{E}_{\pi}\{R_t|s_t = s\} = \mathbb{E}_{\pi}\{\textstyle\sum_{t=0}^{\infty} \gamma^t r_{T+t+1|s_t=s}\} \qquad (3.7)$$

The value function V^{π} is estimated by "trial-and-error" since the expected value unfolds from experience samples within the optimization task. The value function is computed recursively, as seen in dynamic programming. Unfolding equation (3.7), a value function given a specific state is equal to the value of the next state plus the reward [17]. This is expressed in equation (3.8) as below [17].

$$V^{\pi}(s) = \mathbb{E}_{\pi}\{R_t|s_t = s\} = \mathbb{E}_{\pi}\left\{\textstyle\sum_{t=0}^{\infty} \gamma^t r_{T+t+1|s_t=s}\right\} = \mathbb{E}_{\pi}\{r_{t+1} + \textstyle\sum_{t=0}^{\infty} \gamma^t r_{T+t+2|s_t=s}\} \qquad (3.8)$$

If a stochastic policy, π then equation 3.8 unravels into the Bellman Equation of V^{π} as given in equation (3.9) [17].

$$V^{\pi}(s) = \sum\nolimits_a \pi(a|s) \sum_{s_{t+1}} p(s_{t+1}|s_t, a)[r(s, a, s_{t+1}) + \gamma[\sum_{t=0}^{\infty} \gamma^t r_{T+t+2|s_t=s}]]$$

$$= \textstyle\sum_a \pi(a|s) \sum_{s_{t+1}} p(s_{t+1}|s_t, a)[r(s, a, s_{t+1}) + \gamma V^{\pi}(s_{t+1})] \qquad (3.9)$$

There are several methods to obtain a near-optimal policy empirically. One of these methods uses the Q-function [91] (interchangeably Q-value) to evaluate the quality of selecting a particular action in a specific state. This will be further discussed in the next section. Q-function is defined similarly as the value function; however, it considers the state-action value. Thus, it comprises the long-term rewards of applying an action to a state and following the considered policy π after that [91]:

$$Q : S \times A \rightarrow R \qquad Q(s,a)^{\pi} = \mathbb{E}_{\pi}\{R_t|s_t = s, a_t\} = \mathbb{E}_{\pi}\left\{\textstyle\sum_{t=0}^{\infty} \gamma^t r_{T+t+1|s_t=s, a_t=a}\right\} \qquad (3.10)$$

If an optimal policy π^* is considered, the value of a state $V^{\pi^*}(s_t)$ is equal to the Q-value $Q^{\pi^*}(s_t, a_t)$ when taking the optimal action [91]:

$$s_t \in S \quad a_t \in A \quad V^{\pi^*}(s_t = Q^{\pi^*}(s_t, a_t) = argmax_a Q^{\pi}(s_t, a_t) \qquad (3.11)$$

The optimal policy to the RL problem is the policy that produces the maximum reward over a path. Generally, the optimal solution must satisfy the Bellman principle of optimality that says π^* is optimal only if $V^{\pi^*}(s) \geq V^{\pi \neq \pi^*}$[104]. It should be noted that there may exist multiple optimal solutions, such that $V^{\pi_1^*} = V^{\pi_2^*} =, \ldots, V^{\pi_n^*}$. Mathematically, the optimal value function is expressed as:

$$V^*(s) = max_\pi v^\pi(s), \quad \forall s \in S \tag{3.12}$$

Similarly, the optimal action-value function is shown as follows:

$$Q^*(s,a) = max_\pi v^\pi(s,a), \quad \forall s,a \in S,A \tag{3.13}$$

The optimal value function and action-value function expressed in terms of equations (3.14) and (3.15) are provided in a more explicit form [86] as,

$$V^* = \underset{a}{max} \mathbb{E}_\pi\{r_{t+1} + \gamma V^*(s_{t+1}|s_t = s, a_t = a)\} \tag{3.14}$$

$$Q^*(s,a) = \mathbb{E}_\pi\{r_{t+1} + \gamma \underset{a_t}{max} Q^*(s_{t+1}|s_t = s, a_t = a)\} \tag{3.15}$$

where max means that the best action will be chosen in the next timestep.

3.3 Solving the Markov decision process

Dynamic programming (DP), Monte Carlo (MC) Methods, and Temporal Difference (TD) Methods are the three groups of algorithms utilized to solve the MDP in the RL problem. DP approaches can yield exact answers to optimal policies; however, they demand an accurate system model and have computational needs that are infeasible for simple problems. Both MC and TD approaches, on the other hand, use less computer effort to approximate dynamic programming solutions and do not require an accurate system model. The value function is averaged over several sampled experiences of states, actions, and rewards in Monte Carlo approaches to identify the optimal policy; but the results of the sampled trajectories may exhibit a large variance. Temporal difference approaches blend the concepts of DP and MC into a single algorithm. These methods, however, have a significant level of bias because they estimate values using previously estimated values. This section will go over the elements of each approach. Reference [86] provides a detailed explanation of each algorithm.

3.3.1 Dynamic programming

Dynamic programming (DP) is a collection of methods that can identify optimal policies in the presence of an accurate mathematical model. Because of the high computing cost of DP algorithms for non-trivial situations, they are rarely extensively used. Policy iteration and value iteration are the two most used strategies in DP. Policy iteration seeks out the best solution by iterating over a large number of policies $\pi \in \boldsymbol{\Pi}$, and selecting the one with the best cumulative returns. When the cumulative returns cannot be improved any further, the best policy is discovered. Value iteration computes the optimum policy by determining the optimal value functions rather than analysing a huge number of policies. It evaluates the policy after every timestep by traversing through the entire state space and determining the actions that correlate to the greatest values, and the optimum policy can be quickly extracted. It's worth noting that V(s) can only be used to extract the optimal policy if the system has a correct model. The state transition probabilities may be found using the model, and the actions having the highest probabilities of traversing to high-value states can then be identified [86], [101]. DP method has some drawbacks when it comes to solving the MDP problem: value iteration requires fewer computing resources; however, it takes longer to converge to the optimal policy. Also, like the linear programming techniques, DP is not scalable. When dealing with problems that have high dimensionality, this method fails to achieve the optimal policy.

3.3.2 Monte Carlo Methods

Monte Carlo (MC) approaches, unlike dynamic programming, do not strictly require an accurate model of the system. MC approaches to determine the best policy by sampling multiple sequences of states, actions, and rewards under the policy in order to estimate the average returns for various policies. When many samples are produced, $V_n(s) \rightarrow V^{\pi}(s)$ for all $s \in S$, the estimated return is updated for every trajectory. Because of the characteristics of MC updates, finite tasks with a clear terminal state, also known as an episodic task, is the most appropriate system. Given a policy $\pi(s)$, the prediction problem is to find V^{π}(s). This is achieved by running an episode, say n times. Assuming the return obtained from the state s in an episode is $R^i(t)$. For every state s, the averaged return from all visited states is used to compute the value function $V^{\pi}(s)$ as shown below [86];

$$V^{\pi}(s) = [G(t)] = \frac{1}{n}\sum_{i=0}^{n} R^i(t) \tag{3.16}$$

where $R^i(t)$ is the average of the rewards accumulated while following the i^{th} visit s, and $N(s)$ is the total number of visits to state s. MC computation changes the current state value with respect to the return on every time slot. This return, however, is contingent on the action and state transition collected in each succeeding state, which might be very accurate. Generally, MC estimations of the true value function V have low bias with a significantly high variance. In this study, the interest is performing optimal control using policy iteration, which can be done using the Q-function. More information on the Monte Carlo approach can be found in [86].

3.3.3 Temporal Difference methods

Because of their simplicity and low processing cost, temporal difference (TD) approaches are the most extensively utilized RL algorithms. TD methods combine the art of learning from experience (like Monte Carlo methods) and bootstrapping (updating estimates from previous estimates) as seen in dynamic programming methods, thus making them ideal in performing optimal control. The two most prominent TD approaches are SARSA and Q-learning, which have slightly different update procedures. SARSA is an on-policy algorithm, which means its behaviour and target policies are similar. The agent's target policy is the one he or she wishes to find in the end. This is, in most cases, the best course of action. Off-policy agents, like Q-learning, may, on the other hand, have a small probability of selecting random policy during training in order to undertake a deep exploration of other policies and late on change to the best policy [101].

3.3.4 Q-Learning

Q-learning is the most widely used model-free RL algorithm, i.e. it can implicitly learn an optimal policy (e.g., a sequence of battery action selection strategy) by interacting with the environment without any prior knowledge of the environment (as opposed to model-based methods where the agent has to learn the entire dynamics of the system then plan to obtain the optimal policy) [17]. Q-learning involves the finding of the so-called Q-values where Q-values are defined for all state-action pairs (s, a). The Q-value gives the measure of goodness of selecting an action a in state s.

Let $Q(s, a)$ represent the State-Action value function that computes the estimated total discounted rewards as calculated in equations (3.18) and (3.19) [86]; if an action a_t is executed at the state s_t when a policy π is followed, it will be described as,

$$Q(s_t, a_t) = \mathbb{E}\{V_t^{\pi}(s)\} \tag{3.17}$$

$$Q(s_t, a_t) = \mathbb{E}\{r(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i r(s_{t+1}, a_{t+1})\} \tag{3.18}$$

where $\mathbb{E}$ indicates the expected action value for each state-action pair.

The Q-value that reflects the optimal policy is denoted as $Q^*(s,a) = Q^{\pi^*}(s,a), \forall s \in S, \forall a \in A_{s_t}$. If all possible actions in each state s are selected and executed multiple times in the environment, and their Q-values are updated a sufficient number of times, then Q-values eventually converge [91], and the optimal action in that state can be found by taking the action that maximizes the Q-values. The optimal Q-value is given by,

$$Q^*(s,a) = \underset{a}{max} Q^{\pi}(s,a), \forall s \in S, \forall a \in A_{s_t} \tag{3.19}$$

and the optimal policy is acquired as (3.20) [17] for each state s is given as,

$$\pi^*(s) = argmax_{a \in A} Q^*(s,a) \tag{3.20}$$

The implies that an optimal action-value in any state s is described as $Q^*(s,a^*) > Q(s,a_i), \forall a_i \neq a^*$, where $a^*$is the optimal action for state s, commonly known as the greedy action a_g. During the learning process, the agent interacts directly with the dynamic environment by performing actions. Generally, the agent observes a state s_t as it occurs, with the possible action set A_{s_t} And by use of an action selection technique, it selects an action a_t and consequently, moves to the next state s_{t+1}, and receives an immediate reward, $r(s_t, a_t, s_{t+1})$. Then updating of the Q-values is done based on the Bellman equation as shown in (3.21) [103],

$$Q^{n+1}(s,a) = Q^n(s,a) + \alpha[\, r(s_t, a_t, s_{t+1}) + \gamma max_{a_{t+1}} Q^n(s_{t+1}, a_{t+1}) - Q^n(s,a)] \tag{3.21}$$

where $\alpha \in [0,1]$ denotes the learning rate which determines the extent by which the new Q-value is modified, $Q^n(s,a)$is the current estimate of Q-value, $Q^{n+1}(s,a)$ represents the next estimated Q-value in the next iteration, whereas $\gamma \in [0,1]$ denotes the discounting factor and n is the specific iteration number. When α is sufficiently small, and all possible state-action pairs are visited enough times Q^n eventually converges to the optimal value Q^* so that the best action will be selected at each state in the successive iterations [91]. When the agent reaches the terminal states_{T-1}, since there are no future rewards, the Q-value is update as [14];

$$Q^{n+1}(s,a) = Q^n(s,a) + \alpha[\, r(s_t, a_t, s_{t+1}) - Q^n(s,a)] \tag{3.22}$$

As an agent chooses actions from the action set, it is always necessary to cleverly deal with the exploitation versus exploration dilemma. This concept will be further explored in Section 3.4.

3.4 The Exploration Vs Exploitation Dilemma

The exploration versus exploitation dilemma is a recurring theme in RL and AI in general. The question here is that should we exploit acquired knowledge, i.e., should we follow a known high-reward path or explore an unknown path in search of a better new policy [14], [105]. The balance between both highly improves the agent's learning performance. The plausible answer comes to the surface: first, as the training process begins, the agent is required to explore as many state-action pairs as possible; then, after gathering enough information, the agent should exploit the acquired knowledge to gain more rewards. It is very difficult for environments with high uncertainties to know whether sufficient exploration has been employed to avoid exploiting without the proper know-how. Nevertheless, a couple of methods to select actions through a Q-function have been recommended in literature which work well.

3.4.1 Epsilon-Greedy strategy

A simple yet effective method to select an action at each time step is presented as the epsilon-greedy (ε−greedy) selection strategy. However, more sophisticated strategies are also possible. Greedy action is defined as an action that has the greatest Q-value in every episode while the rest actions in the action space are not considered as best actions. However, there is a chance that one of the remaining actions which is not considered as optimal will be as excellent as or way better than the greedy action. Thus, given the Q-function $Q(s,a)$, the greedy action is chosen with $(1-\varepsilon)$ probability while the random action (an action that is not optimal) is selected with probability ε where ε value range within [0,1] as it represents a probability. The value of ε is usually chosen as near to 1 as possible or 1 in the beginning and then gradually reduced as the training progresses to a value close to zero for instance 0.001 [14], [106]. As it progresses through the stages of learning, greedy action will become the optimal action, and ε will be lowered to a very low value.

3.4.2 Boltzmann Exploration (Soft-max method)

One major intrinsic flaw of ε-greedy selection is that it relies on considering one action as the best (highest Q-value) and considering all other bad actions as equiprobable. While drawing a bad action from a uniform distribution, information about the relative quality of each action is discarded (i.e. their Q-values not considered), losing potential exploitation gains. Furthermore, the problem highly intensifies if the gap between Q-values of the best and second-best action is large as the second-best actions are severely penalized. Using the Boltzmann distribution, also referred to as the soft-max selection method, knowledge about the Q-value of every state-action pair is computed as shown in the equation (3.23) [107]:

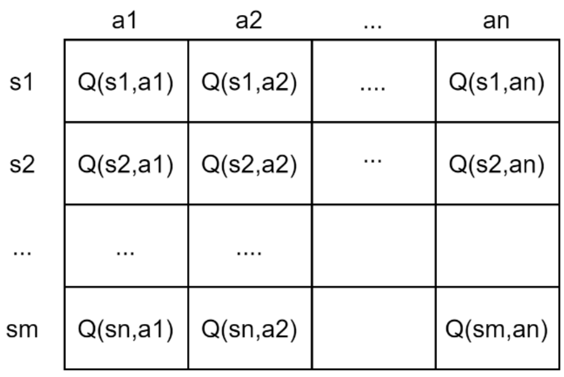
$$p(a_t|s_t,Q_t) = \frac{\exp Q(s_{t,}a_t)/\tau}{\sum_{a_g\in A}\exp Q(s_{t,}a_g)/\tau} \tag{3.23}$$

where τ is referred to as temperature from thermodynamics. If τ presents high values, each numerator is pushed to a value leading to equiprobable actions, thus rewarding exploration. Decreasing τ with the number of episodes leads to exploitation, assigning a high probability to the most promising actions [6].

3.5 Fitted Q-iteration: Q-Learning with Function Approximation

In industrial reinforcement learning applications, function approximators are commonly utilized. In Q-learning methods, the Q values for the discrete tasks are updated and recorded in the Q-table. Figure 3.3 shows a typical Q-table with m states and n actions. A typical power system control application, on the other hand, has multi-dimensional and continuous states and actions. In such sophisticated activities, the storing of value functions could be infinite and intractable. Artificial neural networks

(ANNs) are commonly used to estimate the Q-function based on statistical regression methods to circumvent the curse of dimensionality.



	a1	a2	...	an
s1	Q(s1,a1)	Q(s1,a2)	....	Q(s1,an)
s2	Q(s2,a1)	Q(s2,a2)	···	Q(s2,an)
...	...	....		
sm	Q(sn,a1)	Q(sn,a2)		Q(sm,an)

Figure 3.3: Lookup table example (Q-table).

Various approaches in the literature can be used to approximate the Q function; however, ANNs are preferred to be applied in RL as:

i) They are capable of dealing with target functions that change over time.
ii) They can successfully learn by accumulating data through sequential experiences [17].

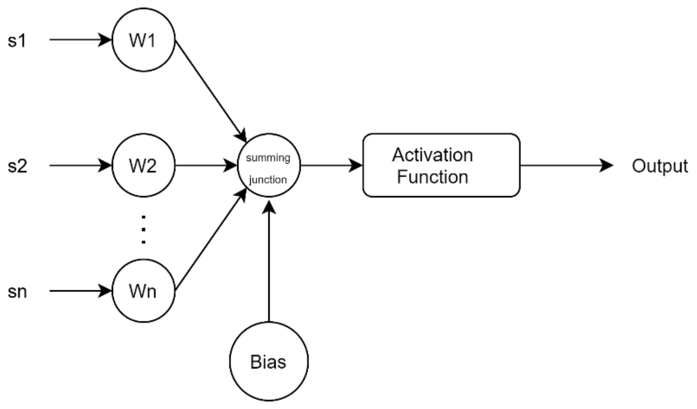


Figure 3.4: Anatomy of a single neuron in an artificial neural network.

In convectional Q-learning, learning is detached in the tabular form, i.e., updating one Q-value has no effect on any other Q functions in the proceeding iterations; thus, the optimal value functions are determined for all states. However, in the approximation scenario, this is no longer possible. Instead, when the weight vectors $w, w_i$are updated, all Q values estimates utilizing w_i will be changed, which makes it extremely difficult for all Q values to be perfectly right [17]. As a result, RL approaches based on function approximation can come close to achieving complete optimal control but never quite do so. The purpose of approximation value function approaches is to identify the optimal weight vector w^* in the end. Figure 3.4 shows a simplified illustration of the structure of a neuron in ANN. The weights are typically initialized arbitrarily and by optimized leveraging stochastic gradient descent (SGD). Because of the simplicity and benefits of handling big data, SGD is a popular machine learning method. It's a subset of the gradient descent algorithm and is utilized to compute the gradient by randomly sampling a portion of data. SGD is discussed in greater depth in [101]. At each step, SGD adjusts the weights w based on predicted versus the actual value and then multiplied by a learning rate $\propto$ [108].

$$w_{k+1} = w_k - \frac{1}{2} \propto \nabla[Q^{\pi}(s_k - \hat{Q}(s_t, w_k))]^2 \tag{3.25}$$

$$w_{k+1} = w_k + \propto [Q^{\pi}(s_k - \hat{Q}(s_k, w_k))] \nabla \hat{Q}(s_{k,} w_k) \tag{3.26}$$

where w_{k+1} is the vector that follows the $(k+1)^{th}$ update, $\propto$ and ∇ are the learning rate and the gradient operator. Larger update steps are produced by higher alpha values, which are usually applied at the start of training. As $k \to \infty$, $\propto \to 0$ to make sure that the weight does not miss the optimum point. The update is done iteratively until the vectors w converge to w^*.

As authors in [60] pointed out, this strategy does not always result in correct convergence and can even diverge in some circumstances. This is due to the fact that the best policy is probabilistic rather than deterministic. As a result, even if the action is the best for that state, a little change in the Q function for that action can make that action not be selected. When the Q-function is approximated with neural networks (NNs), instabilities do arise. Two factors contribute to the instabilities:

1. Within the state-action transitions, there are significant positive correlations. This is due to the fact that the state transitions take place in a specific order. As a result, every state has a relationship with its predecessors and successors.
2. True gradient descent is not used in Fitted Q-iteration. Instead, the method adjusts the weights of an NNs based on a loss relative to a target that is likewise dependent on the weights [65], [66]. This issue was further addressed by introducing another neural net, termed the target network, to predict the target values as seen in the deep Q network model.

The first issue may be addressed using batch reinforcement learning as discussed in Section 3.6.

3.6 Batch Reinforcement Learning

In batch reinforcement learning (BRL), instead of updating Q-values each time an action is taken, the approach accumulates the agent's experiences of the environment in a replay memory and uses a random batch of them to train an ANN (or other models) to estimate the Q-function [65]. As a result, using a method known as batch gradient descent, the agent approximates the best policy utilizing a randomly selected batch of its previous experiences. As a result, batch RL is reported to converge faster than fitted Q-iteration and conventional Q-learning algorithms. Storing transitions in a replay buffer and using random mini-batches of the transitions to update the policy is called experience replay. Apart from breaking correlations within successive transitions, experience replay enables the retrieval of experiences that are more beneficial to the agent instead of discarding the transitions after each update as in traditional Q-learning. However, the application of experience replay alone does not completely solve the instability problem [6].

3.7 Deep Q Network

Deep Q network (DQN) was developed by Google's DeepMind Technologies to address both issues that cause instability in the fitted Q-iteration [109]. The DQN algorithm not only employs the experience replay technique but also uses two different neural networks. One neural network (policy or prediction network) approximates the current Q-value while the other estimates the target Q-value. The separation of the target network from the policy network effectively reduces the instabilities generated by the neural network training process. The target network is just but a copy of the prediction network, which is updated with the weights of the prediction network after every set number of steps. To achieve the optimal policy, both prediction and target networks are initialized by random weights. The replay buffer capacity is also set to a fixed value. In every episode, the algorithm transitions through the entire state space. The values of the current state are computed by forwarding them to the prediction network. The prediction network returns the estimated Q-values for all the possible actions. In each state, an action is selected, normally using the ϵ-greedy method. The action is executed in the simulation environment, and the new state is computed. The reward is then calculated, after which the transition tuple is stored in the experience replay memory as a single unit: (state, action, next state, reward). After a given number of transitions, a random batch of the transitions is taken. A vector of current states in the minibatch is forwarded to the prediction network to get the predicted Q-values. The vector of all corresponding next states is forwarded to the target network to obtain the target Q-values. The squared loss vector between the target and the predicted Q-values is then computed and used to update the prediction network according to the stochastic gradient descent. After training the online network(prediction network), the weights of the target network are updated with those of the prediction network [93], [110], [111]. The process continues until the objective is met. Figure 3.5 illustrates the flow of data in a DQN algorithm.

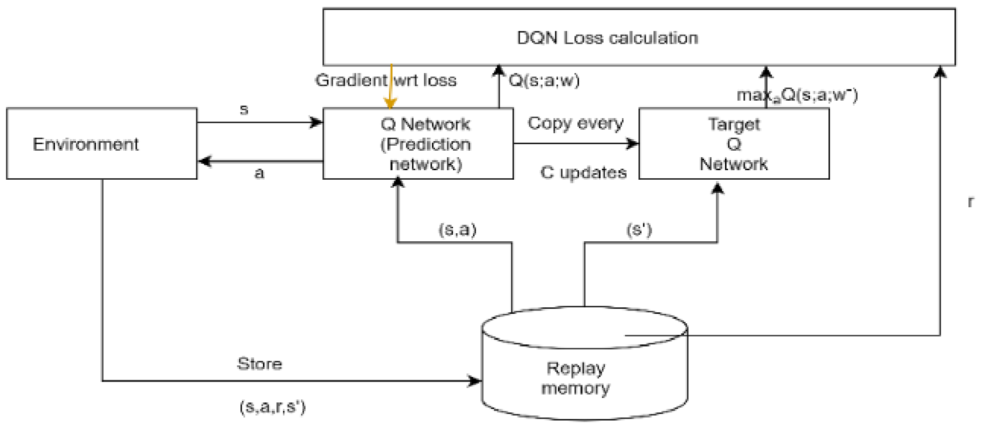


Figure 3.5: Data flow in a deep Q network algorithm with target network and replay memory [110]

4 OPTIMAL ENERGY MANAGEMENT FOR A GRID-TIED MICROGRID SOLAR PV-BATTERY MICROGRID USING Q LEARNING APPROACH

In the near future, microgrids will become more prevalent as they play a critical role in integrating distributed renewable energy resources into the main grid. Nevertheless, renewable energy sources, such as solar and wind energy, can be extremely volatile as they are weather dependent. These resources coupled with demand can lead to random variations on both the generation and load sides, thus complicating optimal energy management. In this chapter, a Q learning method has been proposed to deal with this non-stationary scenario, in which the energy management system (EMS) is modelled as a Markov decision process (MDP). A novel modification of the control problem has been presented that improves the use of energy stored in the battery such that the dynamic demand is not subjected to future high grid tariffs. A comprehensive reward function has also been developed, which decreases infeasible action explorations, thus improving the performance of the data-driven technique. A Q-learning algorithm is then proposed to minimize the operational cost of the microgrid under unknown future information. To assess the performance of the proposed EMS, a comparison study between a trading EMS model and a non-trading case is performed using a typical commercial load curve and PV profile over a 24-hour horizon. Numerical simulation results indicate that the agent learns to select optimized energy schedules that minimize system operational costs in all considered case studies.

4.1 Introduction

Increasing interest in renewable energy sources has led to the massive deployment of microgrids as they offer a scalable way of integrating renewable sources into the main grid while allowing maximum usage of battery energy storage systems. In the long run, the installation of microgrids is expected to reduce the cost of power, dependency on the utility grid and increase rural electrification [37]. Nonetheless, increased integration of renewable DERs raises significant challenges in the stability and economic functioning of the microgrid as they are extremely volatile and random. These multiple stochastic resources combined with the load demand make the preparation of accurate generation schedules very challenging. Deploying a battery energy storage system (BESS) [5] can significantly buffer the impacts of these uncertainties as it provides various auxiliary services to the power system, i.e. load shifting, frequency regulation, voltage support and grid stabilization [47]. However, for a microgrid to guarantee a reliable supply of power and efficient utilization of the battery storage, an energy management system (EMS) needs to be developed to optimally dispatch and distribute these energy resources based on their availability and associated costs.

Optimal energy management (OEM) involves the management/scheduling of various power system variables in a day ahead context in order to satisfy the load demand at minimal or acceptable costs while satisfying all technical and operational constraints. The main goal of developing an effective EMS is to achieve different objectives such as levelling peak loads, balancing energy fluctuations, maximizing renewable energy usage, reducing power losses, and increasing system load factor, among others [10].

Aimed at maximizing energy usage or reducing operational cost by intelligently managing the different types of energy resources and controllable loads in a grid-tied microgrid, several control approaches have been proposed. For years, conventional techniques such as mixed integer linear programming (MILP), linear programming (LP), and dynamic programming (DP) have been leveraged to optimally manage energy in microgrids [7]–[9]. These methods, however, are reported to suffer from the famous curse of dimensionality and are highly susceptible to getting sub-optimal results in environments that are highly stochastic, i.e., they contain volatile variables such as load demand, grid tariffs and renewable energy. Such techniques, therefore, have limited flexibility and scalability. Further, metaheuristic techniques, including particle swarm optimization (PSO), genetic algorithm (GA), and their hybrids, have also been used in literature to tackle the issue of energy management in microgrids [10]–[13]. However, these techniques involve extensive computational time, and hence, they cannot be executed online. The online operation allows computing resources to be used more economically as it doesn't require one to have another committed computer for performing the optimization process offline. The aforementioned algorithms also don't have a learning component, i.e., they are incapable of storing the optimisation knowledge and reusing it for a new optimisation task [14]. Given that the load demand varies on an hourly basis, it is required to calculate the schedule for every new generation and demand profile, and this is not computationally efficient. In addition, the performance of this techniques may deteriorate if accurate models or appropriate state variables forecasting are unavailable. Often, metaheuristic methods are hybridized with other linear methods for an advantage complementation. A comprehensive review of these decision making strategies and their methods of solution has been presented in [6] and [15].

In the last decade, intelligent learning-based techniques have made major progress in decision-making problems and have also proved ideal in overcoming these limitations, as they can automatically extract, monitor, and optimize generation and demand patterns. Additionally, they are capable of relaxing the idea of an explicit system model to ensure optimal control. This is of great benefit since the problem of energy management is normally a partly observable problem, i.e. hidden or unknown information always exists.

The reinforcement learning (RL) method, one of the machine learning algorithms, is well known because of its ability to solve problems in stochastic environments. It aims at making optimal time-sequential decisions in an uncertain environment. Reinforcement learning involves a decision-maker (agent) that learns how to act (action) in a particular situation (state) through continuous interaction with the environment so as to maximize cumulative rewards [17], [18]. In the learning process, the agent is in a position to learn about the system and to take action that affects the environment so as to achieve its objective. In RL, the agent considers the long-term reward instead of simply getting the immediate maximum reward. This is very important for resource optimization problems in renewable-powered microgrids, where supply and demand are changing rapidly. Q-learning, one of the RL methods, is commonly used to solve sequential decision-making problems, as explained by authors in [91]. Q-learning is an off-policy algorithm that does not require any prior knowledge of rewards or state transition probabilities of a system, thus making it applicable to systems that manage real-time data. Many scholars, focusing on microgrid EMS [14], [19]–[21], specifically have used Q-learning to control energy. The key benefit of RL techniques is their adaptability to stochastic systems and ability to transfer knowledge, i.e., the information gained when learning policies for a specific load demand can be retrieved to learn an optimal schedule for other load profiles [14].

In recent advances reported on the implementation of RL in microgrid energy management [22]–[25], [26]–[32], [33]–[35], modelling of microgrid operational cost with consideration of battery degradation cost is not yet thoroughly studied. Most studies only consider the generation cost and power exchange cost. The estimation of the degradation process is very difficult and finding a simple and precise mathematical degradation model that can be used in the energy management algorithm is not easy. As the charging and discharging behaviours of a BESS have a direct impact on its life span, lifecycle degradation costs should be factored into the complex dispatch model of BESSs [112]. This paper reports on the development of a novel EMS design for a grid-tied solar PV-battery microgrid which also embeds a battery degradation model to reduce strain on the battery during the energy trading process. The aim of the designed EMS is to manage energy flows from and to the main grid by scheduling the battery such that the overall system cost (including the cost of power purchased from the utility and battery wear cost) is reduced, and utilization of solar PV is maximized. The EMS problem is modelled as a Markov Decision Process

(MDP) that fully explains the state set, action set and reward function formulation. In addition, two case studies have been considered where, in the first case, energy trading with the utility grid is permitted (trading case), whereas in the second case, it's not (non-trading case). To minimize the operational costs, a Q-learning based algorithm is implemented to learn the control actions for battery energy storage systems (BESS) under a very complex environment (e.g., battery degradation, intermittent renewable energy supply and grid tariff uncertainty). Simulation results show that the agent learns to improve battery actions at every time step by experiencing the environment modelled as an MDP.

The development of a control system architecture with an intelligent decision-making module using a reinforcement learning algorithm is the main contribution of this work. The key accomplishments of this research are summarised below:

I. Considering the technical constraints of the BESS, and the uncertainty of solar PV generation, load consumption, and grid tariff.
II. Designing a novel energy storage decision problem that focuses on energy consumption optimization by maximizing the use of available PV energy and energy stored in the battery instead of focusing solely on direct storage control. In this architecture, excess microgrid energy can be sold back to the utility to increase revenue; however, a non-trading algorithm scheme has also been studied where constraining rules are embedded into the learning process to curtail excess energy from been sold back to the utility. In addition, a battery degradation model is incorporated to reduce strain on the battery during the (dis)charge operation.
III. Using the RL algorithm to learn the electrical resources and demand patterns such that system costs are reduced, and an optimized battery schedule is achieved. Simulations results verify that the proposed algorithms substantially reduce daily operating costs under typical load demand and PV (summer and winter) generation data sets.

The rest of this chapter is structured as: Section 4.2 presents the EMS problem formulation and introduces the two costs models considered, i.e., grid transaction cost and battery degradation costs. Section 4.3 presents the MDP framework for the EMS problem formulation. Section 4.4 explains the proposed Q-learning algorithm. Section 4.5 presents the simulation setup while in section 4.6 simulation results are presented, and the performance of the algorithms are evaluated.

4.2 Energy Management System Problem Formulation

This section presents a brief description of the EMS and then presents the MDP framework. This work considers a microgrid that consists of a PV system, a group of batteries, and some local loads. The schematic diagram of the microgrid is given in Figure 4.1. The microgrid is capable of exchanging energy with the main grid at rates set by the utility company. Time-of-Use (ToU) grid tariffs adopted from Eskom (which is a South African utility) is utilized [113]. The energy produced by the solar PV is used to meet the load demand at the beginning of every time step and is denoted by P_t^{PV}(kW). Excess energy produced by the PV during low energy demand can charge the battery. The battery has a maximum capacity denoted as E (kWh). It is also presumed that there are no charge and discharge losses. The microgrid system has an EMS for scheduling power flows to and from the main grid and manage battery charge and discharge.

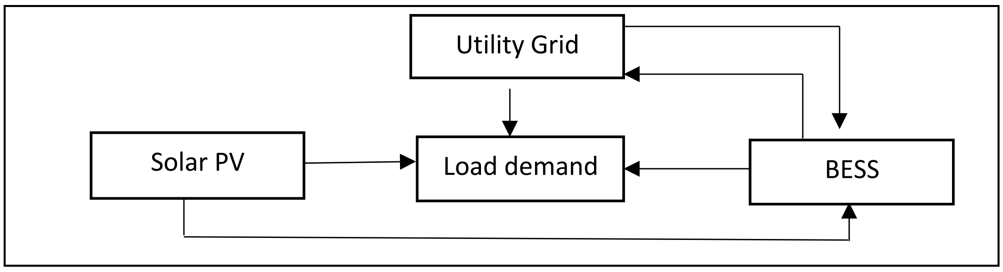


Figure 4.1 : Illustration of grid-tied solar PV-battery microgrid

4.2.1 Objective Function

A real-world microgrid system seeks to supply its total load demand using minimal energy cost. On this basis, the objective function of the designed grid-tied microgrid is computed as equation (4.1).

$$min\{\sum_{t=1}^{t=24}[(C_g(t) + C_{deg}(t))] \tag{4.1}$$

Equation (4.1) defines the need to minimize the daily energy cost (i.e. over a 24-hour horizon); $C_g(t)$ is the cost associated with grid and $C_{deg}(t)$ is the battery wear cost (the two cost components are expressed in R/kWh). The two cost models that the EMS tries to optimize will be illustrated below. Then the mathematical model of the EMS and all system constraints will be explained in section 3 that presents the MDP framework.

4.2.1.1 Battery Degradation Cost Model

To formulate the battery's wear cost, stress factors that affect battery life are considered. These include the temperature at which the battery charges, depth of discharge (DoD), the average state of charge (SoC) and current ripple [114]. These stress factors are highly interdependent and independent, thus complicating degradation cost modelling. Since DoD related stress represents a proper estimation of battery degradation, this research will only consider the effect of depth of discharge on the battery. DoD is described as a function of the battery's SoC and is depicted as [115] $DoD(t) = 1 - SoC(t)$. Authors in [116] researched on the relationship between lithium ion battery DoD and its life cycle data and established that the battery's life cycles increase exponentially with a reduction in the DoD as,

$$L(DoD) = \alpha DOD^{-\beta} \tag{4.2}$$

In equation (4.2) $\alpha \; and \; \beta$ are the curve fitting constants, and the authors in [116] discovered that they are 694 and 0.795, respectively. The battery wear cost (C_{deg}) resulting from related dynamics of the battery life-cycle depicted by equation (4.2) considering a battery that operates from $DOD_1 \; to \; DOD_2$ with $DOD_2 > DOD_1$, could be approximated with equation (4.3) as shown below [115],

$$C_{DoD} = C_{bt}|\left(\frac{1}{L(DOD_2)} - \frac{1}{L(DOD1)}\right)| \tag{4.3}$$

$L(DOD_j)$ denotes the battery's life cycle at DOD_j computed by (4.3) and C_{bt} is the initial capital investment of the battery per kWh. The cost of degradation is assumed to be independent of the direction of power flow in the battery; hence absolute values are considered by the solver. Finally, the battery degradation costs of the control action are calculated as,

$$C_{deg}^t = -(C_{DoD}(t)\Delta p(t).t) \tag{4.4}$$

4.2.1.2 Utility Grid Model

The main grid can have two states: ON (available) and OFF (unavailable) and can supply the unmet load demand or/and charge the battery adequately for the microgrid whenever it is in the ON state. At a given time step t, the microgrid must either be supplying power to the grid network through the battery system or buying power from the grid system (but not both actions at the same time) through the point of common coupling (PCC). Let $G_t(t)$ denote the instantaneous grid tariff given in (R/kWh). In most cases, the selling price is usually lower than the purchasing price in order to encourage local use of solar PV power and minimize the negative effects of microgrid uncertainty on the utility grid [117]. The microgrid selling rates are modelled as a discounted factor ϑ of the ToU tariff. Thus, the cost of exchanging energy in the microgrid is enumerated as,

$$C_g(t) = -(G_t(t)P_{g\_p}(t).t + \vartheta\, G_t(t)P_{g\_s}(t).t) \tag{4.5}$$

where $0 < \vartheta < 1$, $P_{g\_p}(t)$ represents the imported power from the utility while $P_{g\_s}(t)$ represents the exported power at each time step t. This power is further discussed in detail in section 4.3. When $C_g(t)$ is negative the microgrid incurs a cost as power is being imported from utility grid, and when it is positive, the microgrid earns money as power is exported to the utility. The instantaneous grid power limits are set as $0 \le P_{g\_p}(t) \le P_{g\_s}^{max}, P_{g\_s}(t) \le P_{g\_s}^{max}$, $\forall t \in T$ and $P_{g\_s}(t) \cdot P_{g\_p}(t)$. A contract exists between the microgrid owner and the distribution system operator (DSO) that specifies the maximum amount of power exchanged between the microgrid and the utility at the point of common connection (PCC) limits can be changed.

4.3 Markov Decision Framework as Applied to EMS Formulation

An MDP is a mathematical framework used to model decision-making in situations where results are partly random and partly controllable and has been broadly adopted to map optimization problems solved through RL [86]. An MDP is defined as a four-tuple (S, A, T, R), where S and A are the state and action space, T and R denote the state transition probability, and the reward function respectively. Since, for this case the state transitions are deterministic, state transition modelling are not necessary [118], and only the state space, action space, and reward function are considered.

4.3.1 State and State Space Formulation

The information provided by the state is essential for energy management as it contains the information that the agent uses in the decision-making process at each time step t. The state-space of the EMS at any given time is defined by the utility tariff (R/kWh), the BESS state of charge, the load demand (kW) and the PV generation (kW).

Let the state of charge of the battery at time step t be denoted as $SOC = \{SoC_t\}$.

So as not to exceed the battery constraints, a guard ratio β is considered as, $\beta \cdot \frac{E}{E} \le SoC_t \le (1-\beta) \cdot \frac{E}{E}$, where $\beta \in [0,0.5]$ [21] and E denotes the energy capacity of the battery (kWh).

At each time the state of charge of the battery is constrained by, $SOC^{min} \le SoC_t \le SOC^{max}$, where SOC^{min} and SOC^{max} represent the lower and the upper bounds of the battery.

Considering the above battery safety limits, the state s_t at each time step t is,

$$s_t = \{t, P_t^{PV}, SoC_t, G_t, P_{l,t}\} \tag{4.6}$$

where t is the time component denoting the hour of the day, P_t^{PV} is the generation from solar PV at time t, G_t denotes the current electricity tariff at time t notified by the utility company, $P_{l,t}$ is the instantaneous load demand. The state space is enumerated by the union of all sets of states within the optimization horizon as, $S = \text{s}_0 \cup \text{s}_1 \cup, \ldots \cup \text{s}_{T-1}$. The intraday microgrid operation has been divided into T timesteps, indexed as {0,1, 2…, T-1}, where T represents the optimization horizon under consideration.

4.3.2 Action and Action Space Formulation

In order to meet the load demand in every time step *t*, the EMS of the microgrid first uses the available energy from the solar PV and the BESS, and then the remaining energy is purchased from the utility. Net load $P_{l,t}^{Net}$ of the microgrid at each time step *t* is described as the total demand ($P_{l,t}$) minus the energy generated by the solar PV (P_t^{PV}) as shown below:

$$P_{l,t}^{Net} = \max\left((P_{l,t} - P_t^{PV}), 0\right) \tag{4.7}$$

Here, "max" ensures that the complier takes the maximum value always. For instance, if the PV is larger than the load, that equation will output a negative value, which is not the case as the net load is not negative. To prevent that, a zero is put (it will be the max value at that time step), meaning the load has fully been covered by the solar PV.

Since the total load demand $P_{l,t}$ and PV generation P_t^{PV} fluctuate stochastically in a real microgrid, the net demand of the microgrid, $P_{l,t}^{Net}$ is an unknown variable. First, the EMS tries to satisfy the net demand $P_{l,t}^{Net}$ through the energy stored in the BESS. Then, the remaining load demand that cannot be covered by the BESS is provided by the utility. It is described as the reminder energy $P_{l,t}^{rem}$ which can be enumerated as:

$$P_{l,t}^{rem} = \max\left(P_{l,t}^{Net} - (SoC_t - SOC^{min}) * E, 0\right) \tag{4.8}$$

The amount of energy that needs to be purchased at each time step is denoted as $P_{l,t}^{rem}$. At each time step, after covering the load demand, the quantity of energy contained in the BESS is denoted as SOC_t^n is calculated as shown in equation (4.9).

$$SOC_t^n = \min\left(SOC^{max}, (\max\left(P_t^{PV} - P_{l,t}, 0\right) + \max\left((SoC_t - SOC^{min}) * E - P_{l,t}^{Net}, SOC^{min}\right))/E\right) \tag{4.9}$$

Equation (4.9) is generally computing the amount of energy remaining in the battery. The first section checks if there is any remaining solar power after supply the load; if yes, the solver will charge the battery; if there isn't, zero will be taken. Since the EMS is designed to first check if there is any energy in the battery before purchasing from the utility as shown in equation (4.8), the second part of the equation calculates the remaining energy in the battery after supplying the load so that there is an accurate state of charge for the next time step.

Since the agent can only dispatch the battery, i.e., manage charge and discharge, to simplify this problem, the actions are discretized here into discharging/charging action category. The power unit Δp depicts the amount of power that is used to discharge/charge the battery in each discrete instant. The discrete action space is defined as,

$$\mathcal{A}_{s_t} = \{-k\Delta p, \ldots, -\Delta p, 0, \Delta p, .., k\Delta p\}, \tag{4.10a}$$

where $k\Delta p$ and -$k\Delta p$ are the maximum amount of charge and discharge power from the BESS in each time step, respectively, while 0 indicates that the battery is idle. $a_t \in \mathcal{A}_{s_t}$ is defined as the action selected at time step t by agent, where $\mathcal{A}_{s_t}$ represents all the possible actions in the action space $\mathcal{A}$ under state S_t.

$$\mathcal{A}_{s_t} = \{Charging, discharging\} \tag{4.10b}$$

Given the action set $\mathcal{A}_{s_t}$ in $(4.10b)$, at every time step *t*, the agent chooses one possible a_t, from $\mathcal{A}_{s_t}$ by following a policy π, that describes a decision-making strategy for the selection of actions. More details on π can be found in the next section.

Let the function of the amount of power supplied to the battery when an action a_t, is taken by the agent be denoted as BESS (a_t) and computed as

$$BESS(a_t) = \begin{Bmatrix} -k(a_t)/E, & if\ a_t = discharging \\ k(a_t)/E, & if\ a_t = charging \end{Bmatrix} \tag{4.11}$$

where the negative values indicate a discharge from the battery and positive values indicate the charging of the battery. The result of the agent action $BESS(a_t)$ to the battery is based on the status of the BESS SOC_t^n. After Equation 4.11 is executed the SOC_{t+1}^{next} is generated.

It is presumed that if the action taken $a_t(charging)$ increase the $SoC_t + k(a_t)/E$ past the maximum guard capacity E^{max}, only the energy chargeable $SOC^{max} - SoC_t$ is used to charge the battery, and the extra energy is discarded. Similarly, for the discharging action, only $SoC_t - SOC^{min}$ is discharged, and the extra discharge energy is discarded; hence the battery constraints are never violated.

4.3.3 Reward Function Formulation

A reward is a scalar value used to express to the agent the goal of the learning process. Once the agent performs an action and moves to the next state, a reward is presented. Intelligent "reward engineering" is key as it links the agent actions to the objective of the algorithm [119]. The objective of the optimization process is to minimize the transaction cost of power purchased from the utility and reduce battery wear costs.

Reward $\mathrm{r(s_t, a_t)}$ of the proposed EMS is structured to evaluate three of the system management, and one is the objective function and the other two aspects suggested by [95] are adopted to improve the agent's performance. The objective function factors in the amount of money incurred by purchasing energy from the main grid $\mathrm{C_g}$ and battery degradation costs $\mathrm{C_{deg}}$. To improve algorithm performance, $\mathrm{C_b}$ and $\mathrm{C_o}$ have been incorporated. $\mathrm{C_b}$ represents gains from pre-charged energy and $\mathrm{C_o}$ is a penalty payment charged to the agent when it chooses an action that exceeds the limits of the battery.

The pay reward C_g^t represents the cost incurred by trading power with the utility at each time step. The agent receives a negative reward if the amount of energy purchased from the grid is greater than the amount of energy sold. Otherwise, the agent will receive a positive reward of C_g^t it was calculated as given below. It should be noted that this reward depends on the action selected by the agent (a_t). The agent gets a positive reward with G_t discounted by ϑ if the amount of energy exported to the utility is greater than the purchased energy otherwise the agent gets a negative reward as shown below.

$$C_g^t = -\left(P_{l,t}^{rem} + BESS(a_t) * E\right) * G_t \tag{4.12}$$

In equation (4.12) $P_{l,t}^{rem}$ represents the total unmet load in the microgrid at each time step (kWh) while G_t denotes the instantaneous grid tariff (R/kWh). The sum total of $P_{l,t}^{rem} + BESS(a_t) * E$ indicates the power being exchanged with the utility grid at each time step t.

In the non-trading mode of operation, the energy supplied to the load (when a discharge action is selected) at any time slot cannot be higher than the load demand. Equation (4.13) ensures energy cannot be sold back to the utility. During training, if the learning agent tries to select actions that cause power to be scheduled back to the utility, a small negative penalty C_p^t, calculated using equation (4.14), will be charged.

$$P_{l,t}^{rem} + BESS\left(a_{t.discharging}\right) * E \geq 0 \tag{4.13}$$

$$C_p^t = P_{l,t}^{rem} + BESS\left(a_{t.discharging}\right) * E * G_t * \vartheta \tag{4.14}$$

Next, c_b^t reward is computed as the amount of available energy in the battery to cover the net load demand $P_{l,t}^{Net}$from the energy stored in the BESS SoC_t. This reward mainly encourages the agent to always ensure that the SoC of the battery can satisfy the net load at any time. When the current grid tariff G_t increase this benefit reward increases as well. In simple terms, the reward reflects reduced payment that results from using the battery instead of purchasing power from the grid [95].

$$c_b^t = \begin{cases} P_{l,t}^{Net} \cdot G_t & if\ P_{l,t}^{Net} \leq (SoC_t - SOC^{min}) * E \\ (SoC_t - SOC^{min}) * G_t * E, & else \end{cases} \tag{4.15}$$

Then, c_o^t as shown in equation (4.16) below, represents a penalty received by the agent at each time step for any extra energy supplied but is not used in the charging/discharging of the battery due to enforced constraints. As the grid tariffs $G_t$increases, the over-charged penalty becomes high.

$$c_o^t = \begin{cases} -\left((SoC_t + k(a_t) - SOC^{max}\right) * G_t * E & if(SoC_t + k(a_t) > SOC^{max} \\ -\left(|k(a_t)| - (SoC_t + SOC^{max})\right) * G_t * E & elif(SoC_t + k(a_t) < SOC^{min} \\ 0 & else \end{cases} \tag{4.16}$$

Finally, the cost of battery degradation C_{deg}^t is considered as a negative reward received by the agent, and it is calculated as shown in equation (4.4) in Section 4.2.1.1.

Let $r(s_t, a_t s_{t+1})$ denote the cumulative reward that the agent receives when it takes an action a_t at state s_t. The total reward that the agent gets at each time step is given by equation (4.17), however in the non-trading mode of operation C_p^t is incorporated in equation (4.17)

$$r(s_t, a_t) = C_g^t + C_{deg}^t + C_b^t + C_o^t \tag{4.17}$$

As an RL agent traverses the state space, it observes a state s_t takes an action a_t and moves to the next state, s_{t+1}. In order to compute the impact of an action taken by the agent on future rewards while following a certain policy π, $V_t^\pi(s)$ has to be computed. It is defined as the cumulative discounted rewards at time slot t and calculated as.

$$V_t^\pi(s) = r(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i r(s_{t+1}, a_{t+1}) \tag{4.18}$$

The first term in equation (4.18) is the immediate reward at time step t and the second term is the discounted rewards from the next state s_{t+1}. Here, $\gamma \in [0,1]$ is the discount factor, which determines the weight given to future rewards by the agent, where a high value makes the agent more forward-thinking. π is used to represent a stochastic policy that maps states to actions: $\pi(s_t, a_t) \rightarrow S \times A$. The agent's goal is to find a policy π (here, battery schedules) that maximizes the long-term discounted rewards. An optimal policy $\pi^*$is the MDP's solution, i.e. a policy that constantly selects actions that maximize the cumulative rewards for the (T) hours horizon starting from the initial state s_0 [17]. To solve the MDP, several RL techniques can be applied. Model-based methods, such as Dynamic programming (DP), assume that the dynamics of the MDP are known (i.e., all state transition probabilities). On the other hand, model-free techniques such as Q-learning learn directly from experience and do not assume any knowledge of the environment's dynamics. To get the solution of the MDP designed above, Q-learning has been adopted, and it is explained in detail below.

4.4 Q-learning Algorithm for Energy Management Problem

Q-learning is the most widely used model-free RL algorithm, i.e., it can implicitly learn an optimal policy (a sequence of battery action selection strategy) by interacting with the environment without any prior knowledge of the environment (as opposed to model-based methods where the agent has to learn the entire dynamics of the system then plan to obtain the optimal policy) [17]. Q-learning involves the finding of the so-called Q-values where Q-values are defined for all state-action pairs (s, a). The Q-value gives the measure of goodness of selecting an action a in state s.

Let $Q(s, a)$ represent the state-action value function that computes the estimated total discounted rewards as calculated in equation (4.20) if an action a_t is executed at the state s_t when a policy π is followed. It will be described as shown in equation (4.20) while equation (4.19) [86] expresses the state value function,

$$Q(s_t, a_t) = \mathbb{E}\{V_t^{\pi}(s)\} \tag{4.19}$$

$$Q(s_t, a_t) = \mathbb{E}\{r(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i r(s_{t+1}, a_{t+1})\} \tag{4.20}$$

where $\mathbb{E}$ indicates the expected action value for each state-action pair.

The Q-value that reflects the optimal policy is denoted as $Q^*(s, a) = Q^{\pi^*}(s, a), \forall s \in S, \forall a \in A_{s_t}$ [17]. If all possible actions in each state s are selected and executed multiple times in the environment, and their Q-values are updated a sufficient number of times, then Q-values eventually converge [91], and the optimal action in that state can be found by taking the action that maximizes the Q-values. The optimal Q-value is given by,

$$Q^*(s, a) = \underset{a}{max} Q^{\pi}(s, a), \forall s \in S, \forall a \in A_{s_t} \tag{4.21}$$

and the optimal policy is acquired as (4.22) [103]for each state s,

$$\pi^*(s) = argmax_{a \in A} Q^*(s, a) \tag{4.22}$$

Equation (4.22) implies that an optimal action-value in any state s is described as $Q^*(s, a^*) > Q^*(s, a_i), \forall a_i \neq a^*$, where $a^*$is the optimal action for state s, commonly known as the greedy action a_g. During the learning process, the agent interacts directly with the dynamic environment by performing actions. Generally, the agent observes a state s_t as it occurs, with the possible action set A_{s_t}. By use of an action selection technique, it selects an action a_t and consequently, moves to the next state s_{t+1}, and receives an immediate reward, $r(s_t, a_t, s_{t+1})$. Then updating of the Q-values is done based on the Bellman equation as shown in equation (4.23) [103],

$$\mathrm{Q^{n+1}(s, a) = Q^n(s, a) + \alpha[\, r(s_t, a_t, s_{t+1}) + \gamma max_{a_{t+1}} Q^n(s_{t+1}, a_{t+1}) - Q^n(s, a)]} \tag{4.23}$$

where $\alpha \in [0,1]$ denotes the learning rate which determines the extent by which the new Q-value is modified, $Q^n(s,a)$ is the current estimate of Q-value, $Q^{n+1}(s,a)$ represents the next estimated Q-value in the next iteration, whereas $\gamma \in [0,1]$ denotes the discounting factor and n is the specific iteration number. When α is sufficiently small, and all possible state-action pairs are visited enough times Q^n eventually converges to the optimal value Q^* so that the best action will be selected at each state in the successive iterations [91]. When the agent reaches the terminal states_{T-1}, since there are no future rewards, the Q-value is update as shown in (4.24) [106] below:

$$Q^{n+1}(s,a) = Q^n(s,a) + \alpha[\, r(s_t, a_t, s_{t+1}) - Q^n(s,a)] \tag{4.24}$$

As an agent chooses actions from the action set, it is always necessary to cleverly deal with the exploitation versus exploration dilemma[14], [105]. Exploration helps the agent to avoid getting stuck in a local optimum, while exploitation allows the agent to select the best actions in the later episodes. Epsilon greedy (ε_{greedy}) method is adopted here because of its simplicity. Epsilon greedy is a method of selecting actions with uniform distribution from an action space. Using this strategy, it is possible to select a random action (exploration) from the action space $\mathcal{A}_{s_t}$ with probability ε. It is also possible to choose a greedy action (exploitation) with probability $1-\varepsilon$, for $\varepsilon \in [0,1]$, from the Q-values at the given state in each episode. An exponential decay function is also leveraged, so in each iteration, the value of ε is modified as follows [14].

$$\varepsilon = \varepsilon_{min} + (\varepsilon_{max} - \varepsilon_{min})\exp\{-C \times n\} \tag{4.25}$$

where ε_{min} and ε_{max} represents the minimum and maximum values of ε respectively, C is the exponential decay rate and n denotes the total number of iterations.

It is to be noted that epsilon ε varies from case to case depending on system design. But the idea is to allow the agent to explore all the actions in the initial episodes so as to learn. As learning proceeds ε epsilon should gradually be decreased to enable the agent to choose greedy actions. But we should still leave a very small percentage for taking a random action as there is a probability that the current estimate may be wrong and there is another better action. For practical problems during training, start with a very large number of epsilons, i.e., ε=1 and keep lowering that value to 0.001 or 0.01 so that the agent can exploit the best action in the final iterations.

4.4.1 Algorithm for Learning Energy Management

To tackle the MDP, a Q-table is first created and initialized with zeros. At the beginning of the learning, initialization of hyperparameters $\gamma, \propto, and\ \varepsilon$ is done in lines 2-3 of the algorithm shown below. Lines 6 to11 shows the loop for every time step t. In line 5, the microgrid environment is initialized, while in line 6, the algorithm reads the current state. In line 7, action $a_t$is selected depending on the action selection policy π. In line 8, the selected action is executed in the environment, and the environment produces a reward $r(s_{t,}a_t)$ and the next state s_{t+1}.Based on the return of the environment, $Q(s_{t,}a_t)$is updated according to equation (4.19), and if it's a terminal state, an update is done by Equation 4.20 in line 9; In line10, the time step t is incremented by one, $t+1$ and the system move to the next state. After the terminal state T-1, the next episode proceeds with an updated value of ε. Then, the learning process continues, as seen in the algorithm presented in Section 4.5.2.

4.4.2 EMS algorithm using Q learning

1: **Create** a Q-table and **initialize** $Q(s,a)\forall s \in S, \forall a \in A$, with zeros,

2: **Initialize** learning rate and gamma ($\propto$ and γ)

3**:** **Initialize** epsilon (ε)

```
4:  For episode(n) =1, max Episode do
5:    Initialize Microgrid Environment
6:    For time step (t) =0, T-1 do
7:        Read the current state
8:        Select an action using $a_t$ from $A_{s_t}$ using the $\varepsilon$ greedy policy $\pi^{\varepsilon}(s)$ (4.25)
9:        Execute the selected action $a_t$ in the Simulation Environment and observe the reward $r_t$ and the next state $s_{t+1}$
10:       Update Q-values according to (4.19)
11:         $t = t + 1$
12:       End
13:   Update $\varepsilon$
14:   $n = n + 1$
15: End
```

4.5 Simulation Setup

To evaluate the performance of the proposed energy management algorithm using Q-learning, this work considers a commercial load grid-tied microgrid environment with solar PV and BESS. Numerical simulations are performed based on commercial building load profile data adopted from [120]. Summer and winter solar PV output data for (November(summer) and June (winter)) in a 250kWp solar PV system located in Cape Town, South Africa, adopted from [121], are used in the simulation. To facilitate the assessment of optimized control strategy, the work considers an hourly time of use (ToU) tariff obtained from Eskom, a utility company operating in South Africa, which specifies three price levels applied based on time of day during summer and winter seasons. Peak prices are equivalent to R130.69/kWh, mid-peak prices equal to R90.19/kWh and off-peak prices equal to R57.49/kWh during summer while during winter peak tariff is R399.17/kWh, the mid peak is R121.46/kWh, and the off-peak price is R66.27/kWh [113]. The forecasted time series inputs to the algorithm, which include the commercial load demand and solar PV generation, are shown in Figures 4.2 and 4.3 for the summer and winter seasons, respectively. The peak load of the commercial consumption profile is noted to occur between 09:00 and 16:00, when most HVAC and loads are switched on. For the BESS, two Lithium-ion batteries are used, where each battery has a capacity of 200kWh. The initial SoC of the BESS is set to 0.25, and the guard ratio $\beta = 0.05$ is considered since any value in this range [0,0.5] can be selected. Thus, the maximum and minimum limits of the BESS are set up to $E^{max} = 380$kWh and $E^{min} = 20kWh$ respectively. The initial battery cost is determined based on the current market price of Li-ion battery which is \$135/kWh (R2025/kWh) [62]. The charge and discharge energy units Δp are set to 25kWh, where the charge energy of BESS is uniformly discretized to k is 6. Thus, the discretized charging and discharging the energy of the battery is, $\mathcal{A}$ ={-150,…,-50,-25,0,25,50,…150} in kWh where 150 and -150 represent the maximum charge and discharge energy; 0 indicates the battery is idle, while the rest are values within the limit's interval. The maximum charge and discharge energy are limited to 150 and -150 to ensure a safe battery operation limit, while the charge and discharge energy unit is set to 25kWh to give the agent more variables in the action space. For simplicity purposes, power inverter efficiencies for Solar PV and battery are assumed to be

1. The algorithm is implemented in Python programming (version 3.7.6) and executed by a computer with a 1.60GHz processor and 8GB RAM.

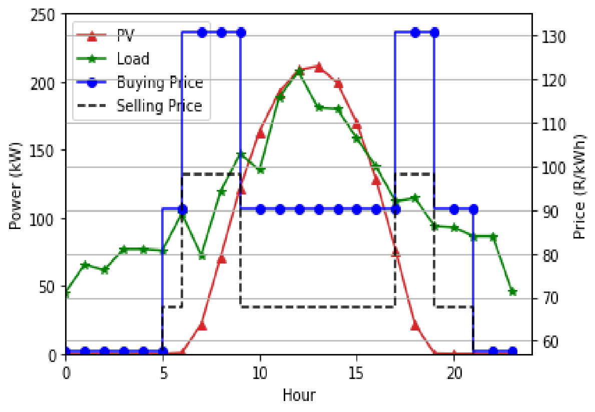


Figure 4.2: Input data – summer solar PV power, load profile, and summer grid prices

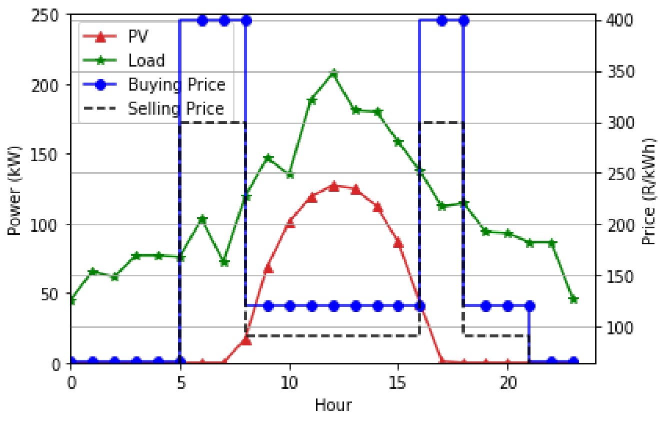


Figure 4.3: Input data – winter solar PV power, load profile, and winter grid prices

It is critical to properly select parameters, especially those to which the algorithm is highly sensitive, such as the learning rate and the discount factor, in order to achieve a suitable convergence speed and quality policies. If a large step-size rate is selected, $Q(s,a)$ values can oscillate significantly, and if it is too small, Q-values might take long before they converge. The choice of α was by trial and error, and a value of 0.01 gave the best convergence. The ε-greedy parameter ε was initialized to 1 to ensure the entire search space is explored as much as possible, and a discount factor γ of 0.85(for winter case) and 1 (for summer) is taken as the future rewards are significantly important as the immediate rewards. The simulation input parameters for the EMS algorithm can be seen in Table 4.1.

Table 4.1: Simulation Parameters for the Q learning algorithm

Hyperparameters	**Selected Values**	***Values***
Epsilon	ε	1.0

Learning rate	α	0.01
Discount factor	γ	1 summer data / 0.85 winter data
Timestep	Δt	1 hour
Battery initial cost	C_{bt}	R2025/kWh
Battery capacity	E_b	400kWh
Initial SoC of the ESS	SOC_0	0.25
Battery guard ratio	β	0.05
Energy unit	Δp	25kWh
Selling price discount factor	ϑ	0.75

In order to evaluate the performance of the proposed grid-tied microgrid energy management system, two case studies are simulated on the basis of the data characteristics mentioned above. First, two different seasons are examined to assess the impact of PV penetration. Second, the comparison between including and excluding grid constraints at the interconnection point is then performed with the aim of studying the impact on total operating costs. In the case of grid constraints (non-trading algorithm) equations (4.13) and (4.14) are included in the optimization model to ensure that the microgrid does not sell its surplus energy back to the utility grid while for no-grid constraints (trading algorithm) they are removed.

4.6 Results and Discussions

4.6.1 Summer Solar PV and Grid Tariff Profile

The performance of the proposed energy management system in a one-day summer operation will be assessed in the current section. The summer PV profile and the summer grid tariffs are considered. Summer solar PV is considered to be the best-case study in the trading algorithm as it is more profitable to increase operating revenues by selling any excess energy back to the utility grid. The total produced energy by PV during summer is 1587kWh.

4.6.2 Reward Convergence During Summer

The primary assessment explores how the system performance is improved by the EMS algorithm as the learning process progresses. Figures 4.4 and 4.5 display the training curves for the trading and non-trading case studies, respectively, which show the average Q-learning algorithm's cumulative reward profile for 20,000 training episodes. Between episodes 0 and 5000, the agent is still in the initial stages of learning, and the reward curve starts at a lower average value of -R140,000 for the trading algorithm and -R175,000 for the non-trading, as can be observed in Figure 4.4 and Figure 4.5 below (here negative values for the reward indicates a cost to the microgrid as power is being purchased from the main grid). This is because initially, for both cases, the value of ε is set to 1.0, i.e., every action has an equal probability of being selected as the action space is still being explored on a trial-and-error basis by the learning agent. Later, as the exploration rate decays and the learning agent starts to exploit the best actions, it is seen that the training curves begin to rise, and then they converge at a higher value at about episode 7500 for the trading algorithm and 8000 for the non-trading one. Convergence is achieved because the agent begins to select better actions learned through the process of experiencing more state-action pairs. It can be observed that the non-trading reward curve reaches a high value of about 0 compared to the trading algorithm, which only achieves -R50000. The reason for this is non-trading algorithm has an additional negative penalty on the reward formulation if the grid constraint is violated, as shown in equation (4.14) which is not present in the trading algorithm. It can be concluded that both proposed energy management schemes are able to achieve optimized policies, and Figure 4.6 and Figure 4.8 show the selected battery actions of the optimal policy for both the trading and non-trading algorithms, respectively.

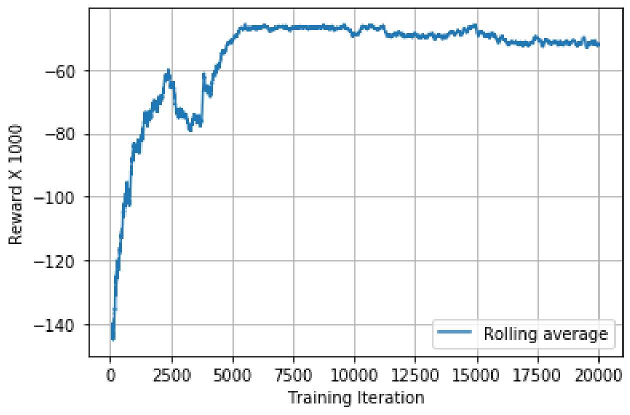


Figure 4.4: Training curve showing reward convergences for episode number 0 to 20000 (trading algorithm)

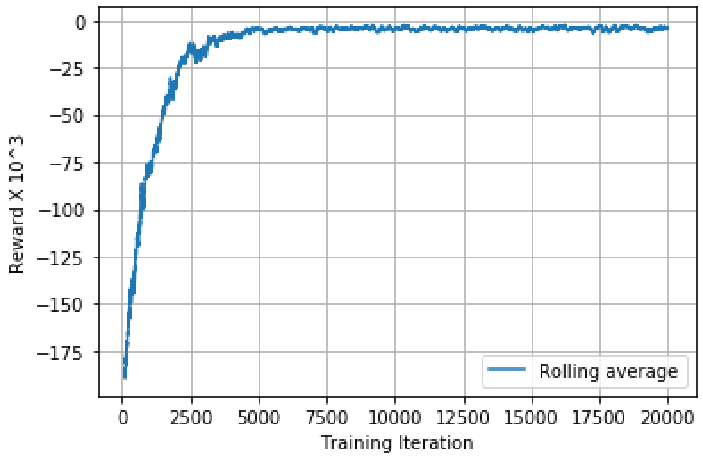


Figure 4.5: Training curve showing reward convergences for episode number 0 to 20000 (non-trading algorithm)

4.6.3 Energy Management

4.6.3.1 Results for Case Study 1 (Trading Algorithm)

This section presents the results for the trading algorithm, which is executed through equation (4.12), i.e., the agent obtains revenue by discharging the battery if the remainder power $P_{l,t}^{rem}$ is zero at any time step t. Analysis of how energy stored in the BESS is used as the EMS seeks to meet net demand is also carried out. When it comes to the system running cost, charging the battery when tariffs are low and discharging the battery when tariffs are high is important so as to rip some revenue. Since the energy demand varies randomly, an efficient charging management algorithm should manage to effectively cope with any unanticipated event and still reduce system operational costs. Between 00:00 and 05:00, PV power is zero. Hence, in Figure 4.7, a decrease in the SoC is seen since the battery is supplying the net load. Also, the load that is not met by the battery, however small, is met by the grid at low prices. Between 11:00 and 17:00, the SoC of the BESS is seen to gradually increase to 0.85 as the battery is being charged by the utility grid. The utility peak load occurs two times a day, i.e. between 07:00 to 09:00 and 18:00 to 19:00 as seen in Figure 4.2. During the first peak load, it is seen in Figure 4.6 that the algorithm learns to lower power intake from the utility to 25kW. In the second utility peak which occurs between 18:00 and 19:00, we see the algorithm learns to raise the battery SoC to 0.85 at time 16:00. From 17:00 to 20:00, battery SoC decreases because the battery is fully supplying the microgrid's net-load, and zero grid power has been scheduled at that time as the prices are very high. From 21:00 to 23:00, a low SoC is seen as only 25kW is being charged to the battery. A final SoC of 1.25 is recorded at 21:00, as seen in Figure 4.7. Given the stochasticity of the load demand, grid tariff and solar PV, it is crucial that the battery energy can deal with unforeseen circumstances, and it is seen that the agent learns policies to increase the SoC to meet its load demand fully during peak tariff hours.

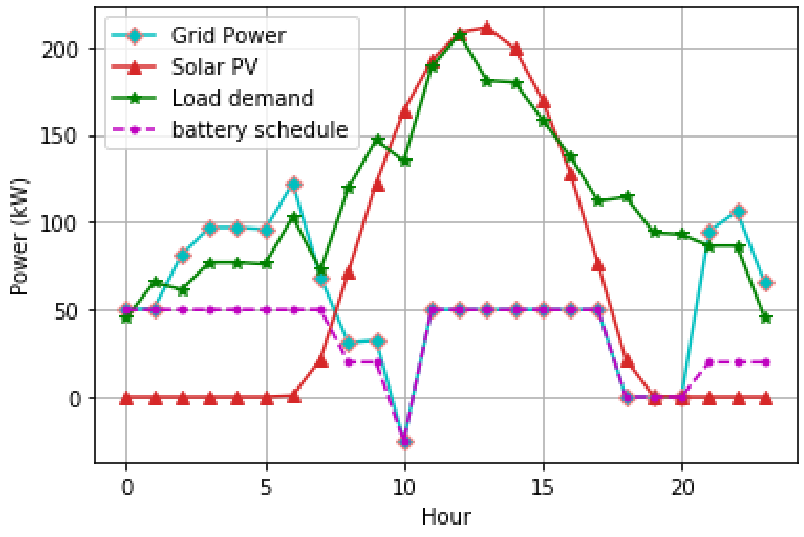


Figure 4.6: Optimized grid and battery schedules for 24 hours horizon (trading algorithm)

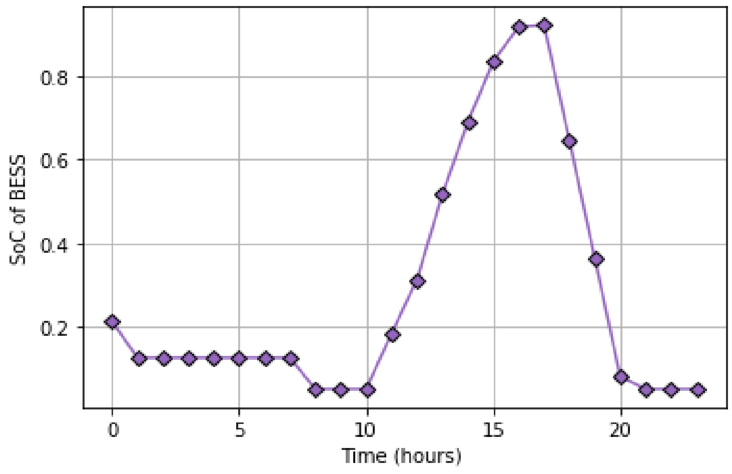


Figure 4.7: Tendency of the SoC of the BESS (trading algorithm)

Figure 4.6 displays the energy schedules of the grid and the battery plotted beside the solar PV and microgrid's load curve. Between 00:00 and 07:00, it is clearly seen that the algorithm opts to charge the battery with 50kW throughout that period. From 02:00 to 06:00, a gradual increase in power absorbed from the utility is seen because the battery cannot fully meet the net load; hence the unmet load is being covered by the grid. Furthermore, the tariff is very low (please refer to Figure 4.2), and it would be optimal to utilize the cheap grid power to supply the net load and charge the battery. At 07:00, there is a sharp increase in grid tariff (R40 increase is noticed), and the algorithm lowers the amount of power drawn from the utility by (25kWh) for two consecutive hours. At 10:00, the agent sells 25kW back to the utility during mid-peak tariff, thus maximizing its revenue. This would be evident by looking at Figure 4.6 and Figure 4.20 simultaneously. In Figure 4.6, at 10:00, grid power is -25kW and also in the same hour in Figure 4.20, it is seen that R2000 was deducted from total cost as power was sold to the utility at that hour. From 10:00 to 15:00, solar PV power is sufficient to fully cater for the load; however, the algorithm opts to constantly charge the battery with 50kW from 11:00 to 17:00. At 18:00, grid tariff shoots to its peak prices, and it is clearly seen that the algorithm schedules zero grid power from 18:00 to 20:00 as the battery can fully cater for load even when solar PV is scarce. This shows that the algorithm manages to foresee the utility peak load and takes proactive decisions of buying power from the utility at the mid-peak price and thus shifts its load from 18:00 to 20:00. At 20:00, the grid prices decrease by R40, and the agent beings to gradually increase the utilization of the grid's power. Full utilization of the main grid is observed from 22:00 to 23:00 as the grid tariffs are at their lowest value and solar PV is not available at that time.

4.6.3.2 Results for Case Study 2 (Non-trading Algorithm)

Figure 4.8 displays the energy schedules of the grid power and the battery for the non-trading algorithm, which constraints the microgrid's power exchange with the utility such that at each time step, no energy can be sold back to the grid as equation (4.13) has been incorporated in the optimization model. In comparison to Figure 4.6, it is seen that at 07:00, when the grid tariff is increased by R40 (peak tariff), the algorithm doesn't reduce its battery power intake, unlike in the trading case study. However, at 09:00, while the grid tariff is still peak, it is seen that the algorithm schedules zero grid power, thus managing to support the grid by lowering its power intake for one hour. From 10:00, the algorithm raises the batter SoC by consecutively charging the battery with 50kw and 25kW. At 19:00 and 20:00, when solar PV is zero and grid tariff is high, it can be observed the agent shifts its load by scheduling zero power from the utility at that time. During off-peak prices at 21:00 and 23:00, we observe maximum usage of utility power as also power from solar PV isn't available. In Figure 4.6, the energy trading algorithm (case 1) is seen to sell 25kW back to the main grid at 10:00; however, in case 2 in Figure 4.8 and Figure 4.21, where grid constraints are enforced, no trading of power was observed. It can be concluded that both algorithms learn to reduce the power absorbed from the main grid at the utility's peak load demand during which buying prices are very high; however, with the trading algorithm, better policies are achieved as the operational cost is lower. Also, it learns to delay drawing power from the utility for 3 hours (from 18:00 to 20:00) until the energy prices lower, as seen in Figure 4.6 in contrast to the non-trading algorithm shown in Figure 4.8 that delays for 2 hours (from 19:00 to 20:00).

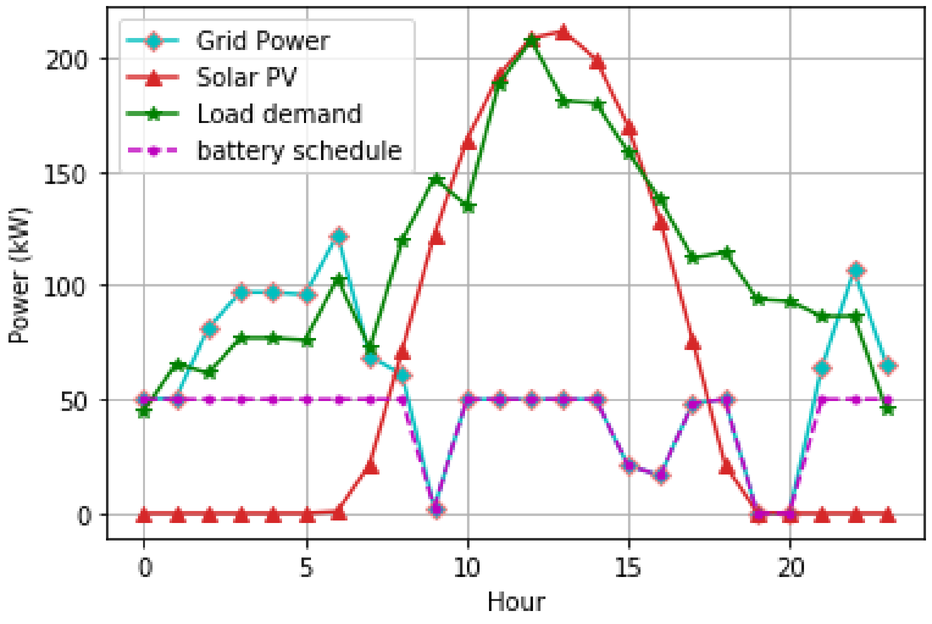


Figure 4.8: Optimized grid and battery schedules for 24 hours horizon (non-trading algorithm)

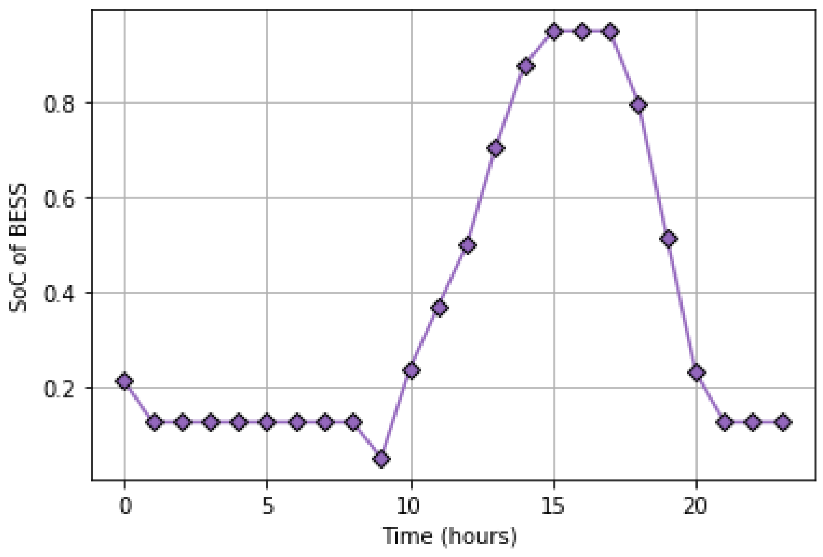


Figure 4.9: Tendency of the SoC of the BESS (non-trading algorithm)

Figure 4.9 shows the battery SoC trajectories as the non-trading algorithm is being executed. Similar to Figure 4.7 from 01:00 to 09:00, the battery SoC decreases slightly and then remains constant since the battery is partly supplying the net load. Between 10:00 and 15:00, the peak load is catered fully by the PV, and the surplus solar PV can charge the battery. The second utility peak demand occurs between 18:00 and 19:00 when PV power is scarce; it can be observed that the SoC of the BESS gradually increases to 0.87 at around 15:00 to support the main grid during its peak demand. The way the BESS is supporting the main grid is by not drawing any power during peak tariff which occurs at 19:00 and 18:00, hence supporting the main grid through shifting the microgrid load. It can be seen from Figure 4.8 that at 19:00 and 20:00 zero power is drawn from the main grid. From 17:00 to 21:00, the SoC decreases, as can be seen in Figure 4.9; zero power is scheduled from the utility for two consecutive hours, and PV power is decreasing; thus, the battery is fully supplying the microgrid's net load. A final SoC of above 0.125 is recorded at 21:00. The plot shows that the learning agent learns to increase SoC to cope with any unanticipated uncertainties, maintains reasonable SoC trajectories throughout the 24-hour horizon and ensure no battery's constraints are violated.

4.6.4 Operational Cost During Summer

Figures 4.10 and 4.11 represent the total daily operation cost plotted versus the training episode number, for trading and non-trading cases respectively. The moving average values are computed for every 100 episodes window. A decreasing trend can be noticed as the learning episodes increase. The daily operating cost at any time step is the grid trading cost and cost of battery degradation, as shown by equations (4.4) and (4.5). As can be seen in the graphs presented in Figures 4.10 and 4.11, the agent explores different possible energy dispatches during the initial stages of learning, and very high costs are registered during the initial stages of learning. For the trading algorithm, an average value of about R120,000 is registered, and for the non-trading algorithm, a value of is R 140000 recorded. As the agent learns better policies, it begins to constantly exploit control actions that reduce energy costs in the final iterations. In the final episodes, it is seen that the algorithm settles at an average global cost of about R105,000 for the trading algorithm and R 110,000 for the non-trading algorithm.

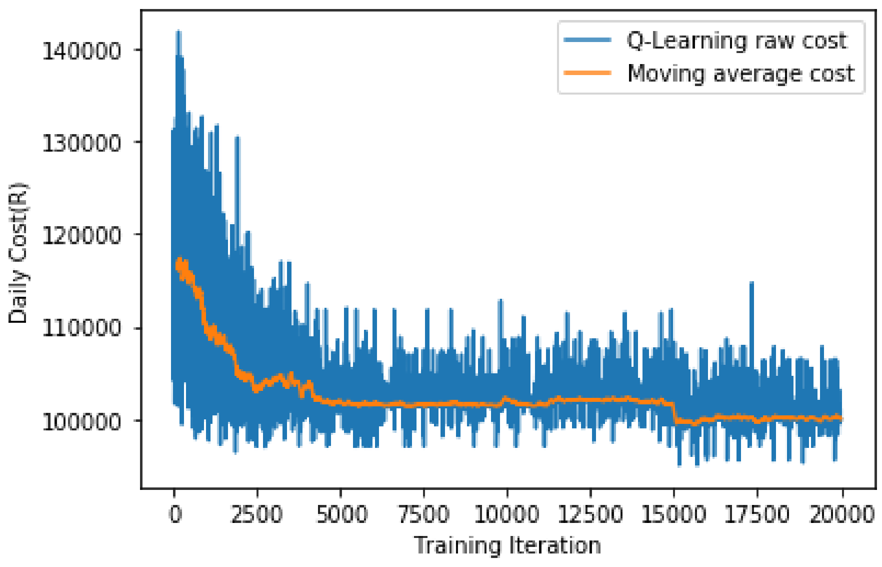


Figure 4.10: Daily operational cost against the number of episodes (trading algorithm)

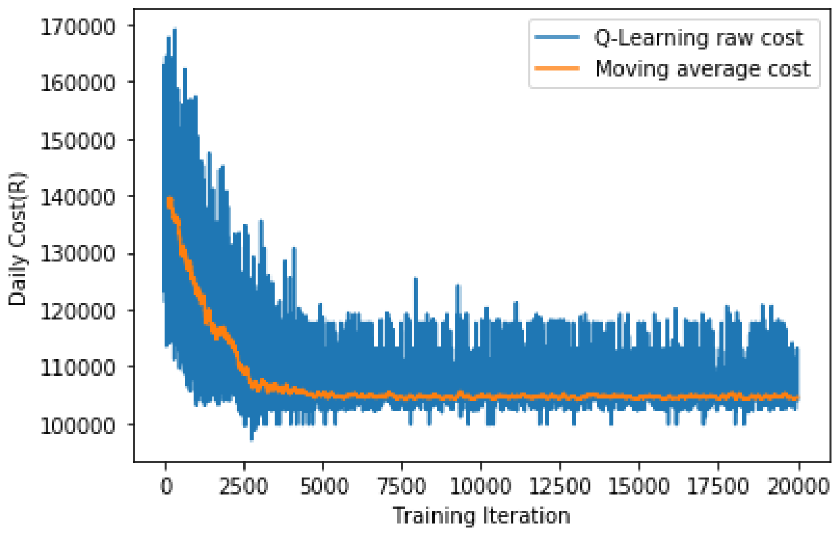


Figure 4.11: Daily operational cost against the number of episodes (non-trading algorithm)

4.6.5 Winter Solar PV and Grid Tariff Profile

This section evaluates the behaviour of the proposed EMS during a day operation in the winter season. The winter PV profile is considered to be the worst-case study as the PV energy output is expected to be lower than summer output as a result of shorter daylight hours, change in the angle of the sun, which reduces the sun's rays hitting solar panels, and extreme atmospheric conditions such as cloud covers and wet weather. The total energy output of PV production during the considered day amounts to a sum total of 801kWh. Also, it can be noted that the winter tariff is rather high compared to the summer tariff as cold and dark weather cause people to stay indoors more, to turn on the lights for longer hours, and to switch on heating equipment, thereby increasing energy demand. In addition, extreme weather conditions could also damage the power system, resulting in high repair costs.

4.6.6 Reward Convergence During Winter

In Figure 4.12 and Figure 4.13, it can be observed that both the trading algorithm and non-trading algorithm are capable of increasing the average reward over 20,000 training episodes. Between episodes 0 and 5000, the agent is still in the initial stages of learning, and the reward curve starts at a lower average value of -R500,000 for the trading algorithm and -R540,000 for the non-trading algorithm. This is because initially, the learning agent is still exploring the stochastic environment on trial and error. Later, as the exploration rate decays, the learning agent starts to exploit the best actions; it is seen that the training curves begin to rise and then converge to higher values ats episode 12500. It can be observed that the trading algorithm converges to a lower average value (-R380,000) in comparison to the non-trading algorithm, which converges at an average value of about -R300,000. The reason for this is non-trading algorithm has an additional negative penalty on the reward formulation if the grid constraint is violated, as shown in equation (4.14) which is not present in the trading algorithm. The retrieved optimal winter battery schedule is shown in Figure 4.14 for the trading case and Figure 4.16 for the non-trading case. In comparison to the summer PV profile and tariff, it can be seen that rewards converge to very low values for the winter case. This is mainly attributed to the low PV profile and high winter grid tariffs for any energy purchased from the utility.

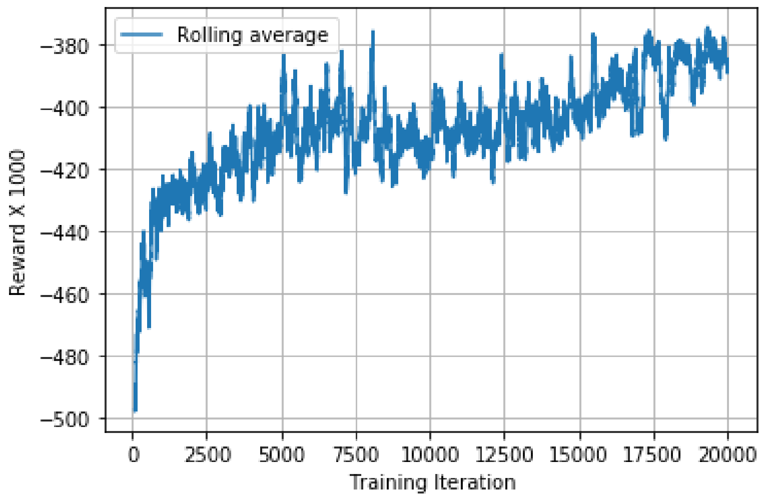


Figure 4.12: Training curve showing reward convergences for episode number 0 to 20000(trading algorithm)

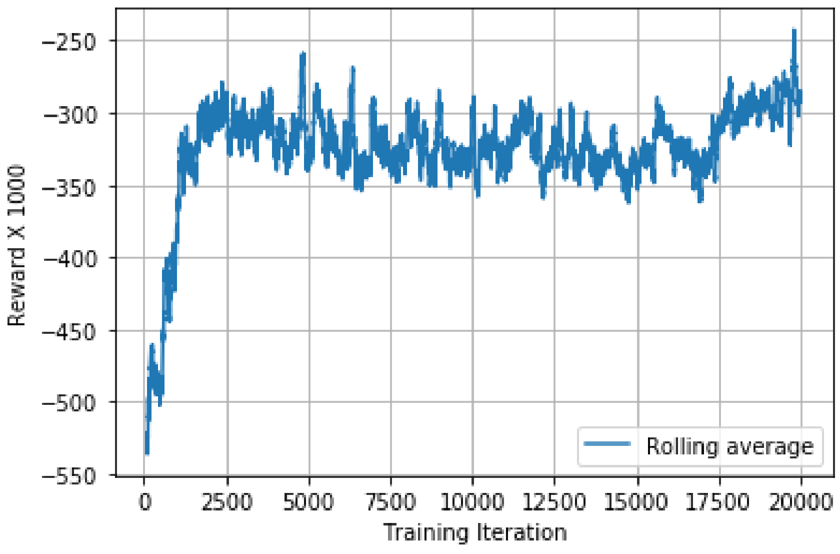


Figure 4.13: Training curve showing reward convergences for episode number 0 to 20000(non-trading algorithm)

4.6.7 Energy Management

4.6.7.1 Results for Case Study 1 (Trading Algorithm)

Figure 4.14 displays the energy schedules of the grid power and the battery plotted beside the winter solar PV output and grid tariff. As can be seen in Figure 4.3, the PV system produces a small amount of energy between 08:00 and 16:00. Consequently, the trading operation will be limited as the total available PV generation will partly cover the microgrid's demand. Also, the winter peak prices occur from 06:00 to 08:00, unlike the summer-time case where utility peak load starts at 07:00 [113]. In the trading algorithm, the EMS begins by scheduling zero grid power as the initial battery energy can fully meet the net load and later on, a gradual increase in grid power utilization is seen. Between 00:00 and 05:00, the tariff is at its lowest, thus for about four hours, very high-power absorption from the utility is recorded. When grid tariff increases from 06:00 to 09:00, the algorithm is seen to drastically lower the amount of power purchased from the main grid. At 11:00, when prices have reduced to mid-peak, the algorithm decides to increase power intake from the distribution grid network. From 13:00 to 23:00, the agent takes control actions of constantly charging the battery with 25kW and supplying the remaining net load with power from the utility. In Figure 4.15, it can be observed that the algorithm gradually increases the SoC of the battery up to 0.45 in the morning hours (04:00 to 06:00) to meet its net load as it anticipates the utility peak tariff, which occurs from 06:00. As a result of raising the SoC, the algorithm is able to shift a large percentage of its net load until grid prices are reduced. Unlike during summertime, it can be seen that battery utilization is rather low. As the PV is insufficient throughout the optimization horizon, the high deficit load computed by equation (4.8) must be supplied by the utility grid. Thus from 13:00, the algorithm opts to keep the charge power as low as possible so as not to incur the high cost of importing utility power to cover its deficit load and charge the battery. Similarly, the fact that the winter tariff is more expensive makes the algorithm schedule lower charge energy so that the amount of power drawn from the grid is minimized. Finally, it can be observed that the SoC is maintained at its lowest level, and the battery constraints are not violated.

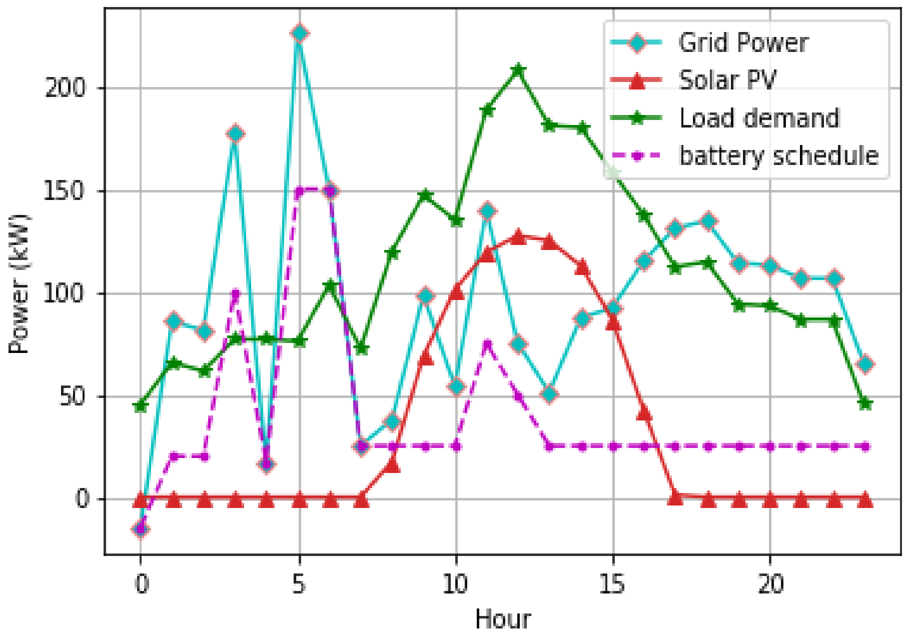


Figure 4.14: Optimized energy schedule (trading algorithm)

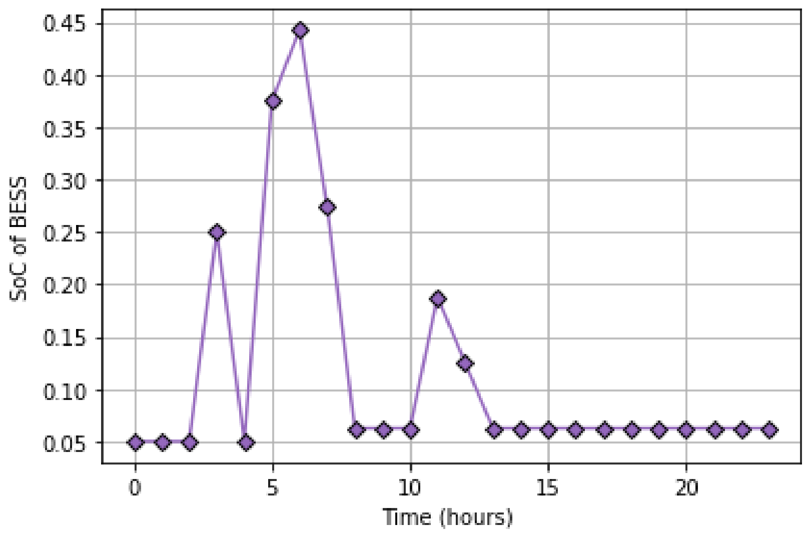


Figure 4.15: Tendency of the SoC of the BESS

4.6.7.2 Results for Case Study 2 (Non-trading Algorithm)

Figure 4.16 presents the results of case study 2, which, as mentioned earlier, ensures that no energy is sold back to the utility grid. Between 00:00 and 05:00, the grid tariff is very low, and PV power output is zero. A gradual increase in grid power utilization is observed as the algorithm chooses to charge up the battery with the cheap grid power so as to supply its net load as it would be optimal to do so. Between 06:00 to 09:00, the grid tariff shoots to its peak (R278 increase in grid tariff is noticed in Figure 4.3), the algorithm gradually lowers power intake from the utility from 04:00 to 06:00. A constant charge power of 25kW is recorded from 06:00 to 08:00 which has the potential to lower the operation cost as not much energy is draw by the battery during the high tariff times. This is noted because the agent is trying to avoid charging the battery at high tariff which is an expense to the microgrid. This, however, leads to a low SoC value which cannot cover the microgrid's load for long hours. Thus, from 07:00 to 08:00 around 100kW is draw from the utility as PV production very minimal (almost zero) and the microgrid's load demand is steadily increasing. From 09:00 to 17:00, when the tariff changes to mid-peak, it can be observed that the algorithm slightly increases the battery charging power to 50kW. A constant power intake by the battery is seen until the next peak tariff, which occurs at 17:00, where the algorithm reduces the charging power to 25kW. Observing Figure 4.14, the trading algorithm learns to raise the SoC value to 0.45, unlike the non-trading, which only reaches about 0.35 at 04:00; this causes the latter algorithm to only lower the grid power for one hour and later on rely heavily on the utility as the energy stored in the battery cannot support the microgrid's net load. However, in Figure 4.14, the power drawn from the utility is lowered for two consecutive hours during peak prices. In both cases, solar PV is very low, and peak grid prices are also very high; however, the algorithm learns to lower costs in these extremities.

It is noted that for the winter case, the battery is not utilized well. As can be seen in Figure 4.17 the battery is charged between 02:00 and 04:00 to a SoC of about 0.35, after which it is discharged between 05:00 and 06:00 to a SoC of around 0.05 where the power draw from the grid is lowered. Then, from 10:00 to 17:00 the battery is charged with 50kW consistently, however, since the microgrid peak load occurs at that time and the solar PV production is still low, part of the energy stored in the battery is used to cover the load net load and thus the battery SoC is at very low levels.

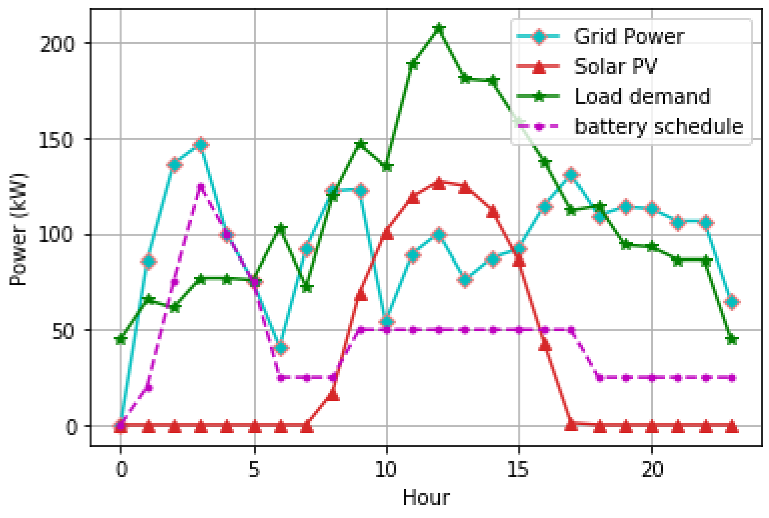


Figure 4.16: Optimized energy schedule (non-trading algorithm)

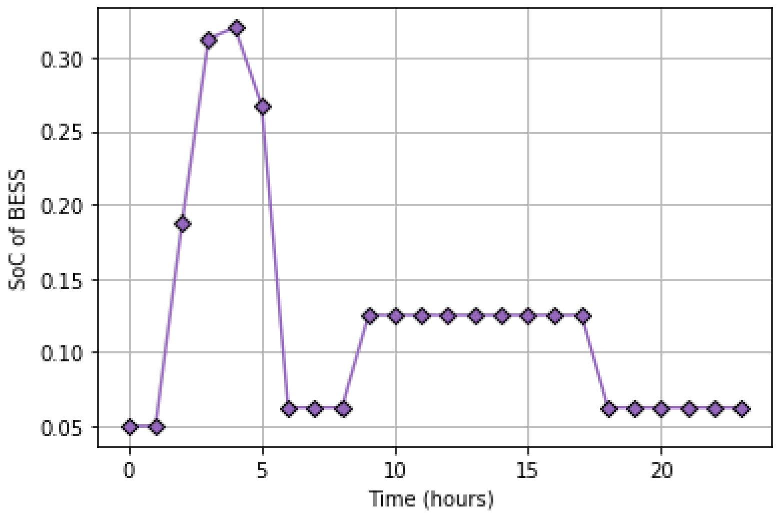


Figure 4.17: Tendency of the SoC of the BESS (non-trading algorithm)

4.6.8 Operational Cost During Winter

Figures 4.18 and Figure 4.19 represent the average running cost variations during training. These curves tend to have almost similar characteristics, although the non-trading plot is more erratic compared to the trading one. This is most likely attributed to the low PV generation and high and dynamic grid tariff seen in winter. At the beginning of the training, the learning agent explores the action space and learns to avoid actions that result in high costs. In the final episodes, actions that minimize cost are exploited for both cases. A final average global cost of about R395,000 is recorded for both scenarios.

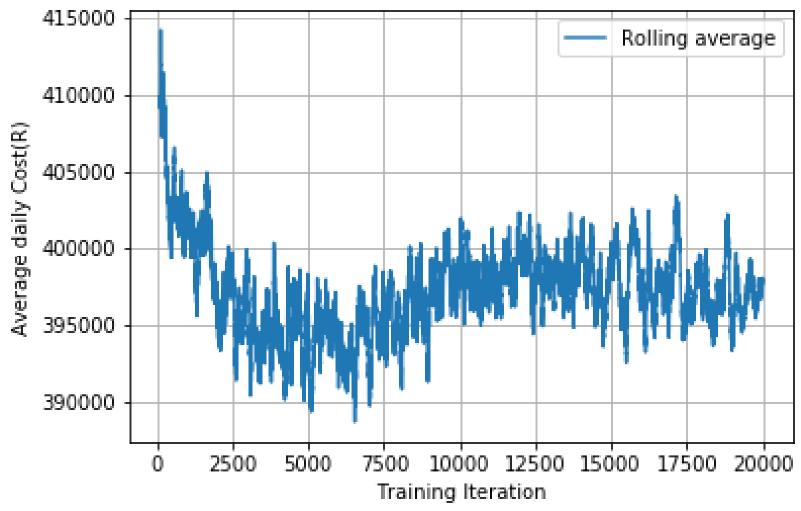


Figure 4.18: Daily operational cost against the number of episode (trading algorithm)

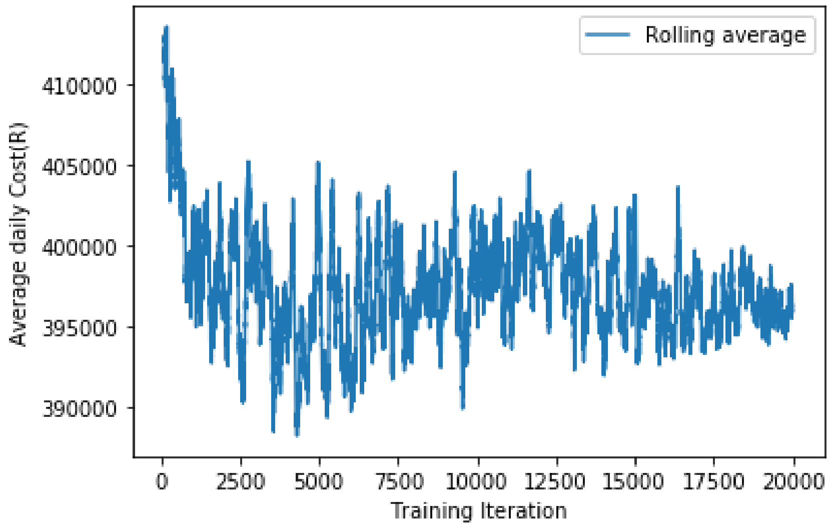


Figure 4.19: Daily operational cost against the number of episodes (non-trading algorithm)

4.7 Comparative Cost Study for Case 1 and Case 2

This section presents the energy cost comparison assessment for the optimized energy schedules. The comparison is based on the two case studies investigated, i.e., trading and non-trading cases, using both summer and winter PV and grid tariff data. Table 4.2 below shows the retrieved schedule energy cost for the two case studies in different season profiles. In this problem, energy cost is the product of the power imported from the grid to cover the microgrid's deficit power or/and charge the battery and the grid tariff. In the case of the trading algorithm, the cost of exported energy is deducted.

$$\text{Energy cost} = \sum_{t=1}^{t=24} G_t(t)P_{g\_p}(t) - \vartheta\, G_t(t)P_{g\_s}(t) \tag{4.26}$$

where $P_{g\_p}(t)$ denotes the power imported from the main grid, $P_{g\_s}(t)$ is the power exported to the utility grid, $G_t(t)\ is\ the\ instateneous\ grid\ tariff\ and\ \vartheta$ represents the selling price discounting factor. From Table 4.2, it's apparent that in summer, the total operating costs are the lowest in both cases compared to the winter season. It can be reported that increasing PV generation would result in a much more profitable EMS operation in both the summer and winter seasons.

Table 4.2: Overview of energy cost for the optimal episode in the case studies considered

Energy cost over a 24-hour window	**Summer data (**PV & grid tariff**)**	**Winter data (**PV & grid tariff**)**
Trading algorithm	R103,708.71	R367,322.73
Non-trading algorithm	R107,891.05	R375,403.00

To calculate the increase in the percentage of total operating costs between the trading and non-trading case studies, equation (4.27) is used.

$$I_{TC} = \frac{TC_{non-trading} - TC_{trading}}{TC_{trading}} \times 100 \tag{4.27}$$

I_{TC} denotes the increase of the total operational cost (in percentage) $TC_{trading}$ and $TC_{non-trading}$ are the total operational cost of the trading and non-trading studied algorithms, respectively. The implementation of the proposed EMS for commercial load profile considering the absence of grid constraints (i.e., excess energy can be sold back to the utility), the total operating costs can reduce by 4.033% for summer data and 2.199% for winter data when compared to the non-trading algorithm. This phenomenon happens because, with the trading case, there is more flexibility to feed power to the utility and earn some revenue, whereas, for the non-trading algorithm, less flexibility is experienced by the agent when learning the environment as grid constraints cannot be violated. However, taking into account grid constraints is also technically beneficial, particularly from the perspective of the local utility grid operators as the non-trading EMS avoids feeding any power back to the utility, and this could lead to both technical and economic benefits to the microgrid owner and utility system operator.

Figures 4.20 to 4.23 below display the dispatching cost for both case studies for the optimized retrieved schedule.

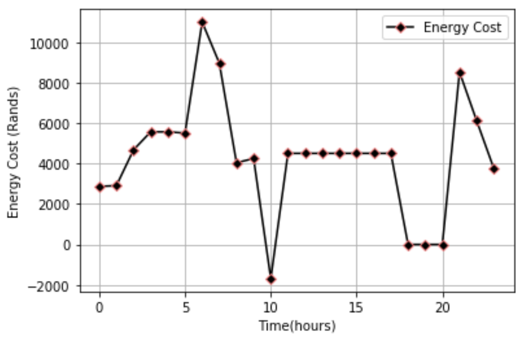


Figure 4.20: Summer dispatching cost profile for the optimized schedule (trading algorithm)

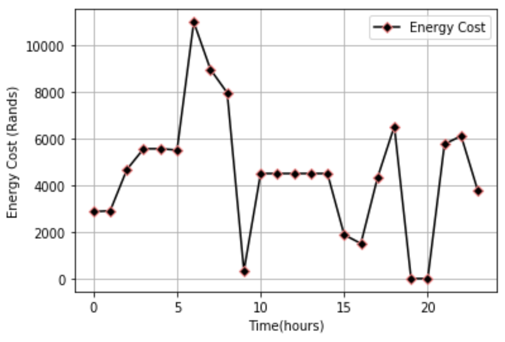


Figure 4.21: Summer dispatching cost profile for the optimized schedule (non-trading algorithm)

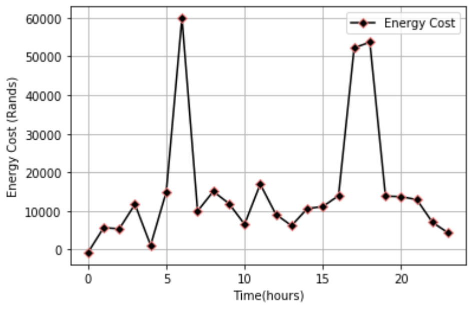


Figure 4.22: Winter dispatching cost profile for the optimized schedule (trading algorithm)

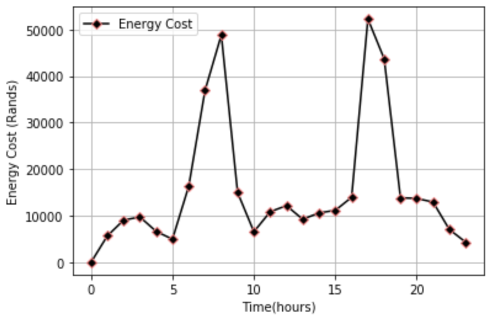


Figure 4.23: Winter dispatching cost profile for the optimized schedule (non-trading algorithm)

Summary

Two case studies have been considered trading and non-trading settings. The objective is to reduce the total daily operation cost under the uncertainty of PV power, load demand, and grid tariff in both the summer and winter seasons. Using numerical simulations and proper hyperparameter tuning, we confirmed that the proposed energy management schemes can efficiently minimize system operational costs (battery wear cost and cost of power purchased from the grid) under widely used south African time of use (ToU) grid tariff and achieves desirable control actions which maximize solar PV usage while minimizing strain on the local utility during peak hours. The proposed energy management algorithm is intended to be applied in a number of intelligent grid environments, including residential microgrids and smart energy facilities under different tariff structures to optimally schedule for energy consumption by efficiently managing the total energy produced and trading the surplus energy into the utility grid to make some profits [87].

5 OPTIMAL ENERGY MANAGEMENT IN A GRID-TIED SOLAR PV-BATTERY MICROGRID USING DEEP Q NETWORK APPROACH

The increasing electricity demand leading to increased integration of renewable energy resources into the power grid necessitate intelligent energy management systems that may improve energy efficiency and maximize the dispatch of renewable distributed energy resources. This chapter presents a deep Q network (DQN) technique to optimally manage energy resources in a microgrid in which the algorithm learns tasks in the same way as humans do. Thus, every move the agent makes in the environment generates feedback. These feedbacks motivate the agent to learn more about the environment and perform far more intelligent steps later in its learning stages. Specifically, the chapter proposes an energy management system based on DQN to learn system uncertainties, including load demand, grid prices and volatile power supply from the renewables and ensure that energy is optimally dispatched in such a setting. The method uses recent learning algorithm enhancement techniques, including experience replay and target network, shown to increase learning speed and improve stability in previous research. The performance of the proposed method has been evaluated with different types of load fluctuations, i.e., slow, medium, and fast. Another essential contribution is designing an algorithm that minimizes both power purchase cost and battery degradation simultaneously. Simulation results substantiate the efficacy of the proposed method as the algorithm learns from experience to raise the battery state of charge and optimally shift loads from a one-time instance, thus supporting the utility grid in reducing aggregate peak load. Furthermore, the proposed DQN approach outperformed the conventional Q-learning approach in reducing system operational costs by 15%, 24%, and 26% for the slow, medium, and fast fluctuating load profiles in the studied cases.

5.1 Introduction

The transition from traditional centralized energy resources to distributed energy resources (DERs) with minimal carbon impacts on the environment is at the core of the power system's continuing transformation. This transformation needs innovative solutions to address the issues posed by renewable resources intermittent nature. Intelligent grid technologies, including advanced metering infrastructure (AMI) to monitor consumer energy consumption in real-time, renewable distributed generators (DGs), bidirectional power inverters, and intelligent energy storage technologies, are increasingly being deployed globally to enable this transition [30]. A microgrid is defined as a group of Distributed Energy Resources (DERs), including Renewable Energy Sources (RES) and Energy Storage Systems (ESS), plus loads, with clear electrical boundaries which operate locally as a single controllable entity with respect to the grid. Microgrids are typically low-voltage power networks that have small energy capacity in comparison with the main grid. They operate either in parallel with the grid, purchasing and selling energy through the electricity market, or autonomously, using local generation and storage [38]. As a result, they provide technical and economic advantages, such as increased system resilience, reliability of on-site energy supply and additional investment opportunities for renewable DGs [30].

Two tiers of control are required to ensure the microgrid's reliability. The lower-level control entails managing the electricity voltage and current and the power grids frequency, which is commonly accomplished at the power electronics interface. On the other hand, the energy management system (EMS) maximizes the overall system efficiency by optimizing energy dispatch of local resources at the higher-level control and maintaining energy reserves. However, because of the microgrid's nature, i.e., its small energy capacity, intermittency of DERs, as well as energy consumption unpredictability and dynamic electricity tariffs, its EMS faces significant challenges. Thus, integrating microgrids into the power grid requires new control architecture and intelligent energy management algorithms that can take optimal real-time actions instantly and adaptively, consequently overcoming these challenges and ensuring that supply and demand of energy are balanced in real-time operation. Specifically, a well-designed energy management algorithm should optimally decide which of the system's energy resources should produce energy, how much energy they should be producing, and when they should produce it to meet the system's load at the lowest possible cost while ensuring that no technical constraints are violated [10].

Several algorithms for managing energy in grid-connected solar PV battery systems have been developed in the past. Linear algorithms like linear programming (LP) and mixed-integer linear programming (MILP) have been utilized to find solutions quickly in less intricate domains, but they have limitations when it comes to dealing with stochasticity [7]–[9]. In addition, for microgrid energy management, heuristic techniques such as genetic algorithm (GA) and swarm intelligence algorithms, including particle swarm optimization (PSO), and others have been applied in the literature [10]–[13]. Owing to their capacity to accommodate stochastic system variables, these approaches outperform linear optimization algorithms. They are, however, often slow and unable to handle dynamic online operations.

Data-driven techniques based on machine learning (ML) have recently shown great potential in achieving near-optimal microgrid control using operational data. The reinforcement learning (RL) paradigm, in which a microgrid's dynamics are learned by an agent interacting with its components, is amongst the most promising learning-based techniques for energy management systems. Reinforcement learning is a solution method that is motivated by rewards. Reinforcement learning methods are more capable of dealing with dynamic stochastic situations than other optimization approaches due to their learning element and capacity to generalize solutions [14], [16]. RL generally involves a learning agent that interacts with its environment through a sequence of discrete-time steps. At each time slot representing a particular scenario, the learning agent selects an action executed in the environment. As a result, the agent gets a reward, and its environment changes to the next state. RL aims to establish an optimal policy that maps states to actions that maximize the overall rewards, depending on the agent's knowledge gained from unsupervised direct interaction with the environment [17]. RL has now become a powerful method for optimizing the control of power networks that always deal with continuous variations such as intermittent renewable resources, dynamic energy rates, and uncertainty in load demand, owing to its distinct feature of “model-free” and “no need for prior domain knowledge” concepts. Google DeepMind is a practical pioneer of this technique, having successfully implemented an RL-based model to lower electricity expense related to their data centre cooling by 40% [122], which is a major motivator for applying RL technology to energy systems.

Several studies have shown that RL-based EMSs can be successfully implemented in various microgrid topologies, either as a single agent [112], [123], [124] or as a multiagent scheme [95], [125]. Nonetheless, the most basic and widely used RL approaches, namely Q-learning [91], suffer from several challenges, including inefficient data utilization, inability to handle continuous/large state-space, and curse of dimensionality, which cause the method to fail for large-scale tasks. The study in [87] designed a Q-learning technique to optimally manage energy in a grid-tied solar PV battery microgrid. A comparative case study using both winter and summer data profiles was conducted on the algorithm. Although the employed Q-learning technique was simple and the proposed algorithm was reported to learn an optimized battery schedule while minimizing system costs, the approach is unscalable. As the state spaces increases, the Q-table grows infeasibly large, and most state-action pairs are not visited during training. Furthermore, sequentially updating the algorithm leads to low data usage and high state transition correlations, leading to poor learning capabilities. Authors in [22] presented a batch RL technique to solve inefficient data utilization by employing a memory reply and training the algorithm with a batch of previous experiences. Deep reinforcement learning (DRL) approaches, on the other hand, use artificial neural networks as function approximators, allowing them to learn continuous state-action transitions in the face of uncertainty. Deep neural networks also will enable the utilization of continuous and large-dimensional state spaces and the extraction of hidden features. As a result, the DRL agent can overcome the environment's uncertainties and partial observability [93].

Several studies have demonstrated interest in DRL applications to solve microgrid control challenges, owing to DRL's recent achievements in tackling complicated tasks, as evidenced by Alpha Zero's superhuman performance in solving several complex computer games [94]. The authors in [95] presented a multiagent DRL to manage energy in interconnected EV charging stations. The results obtained indicated that adding a communication model improves the cooperative working of the agents in multiagent systems. In [2], a novel microgrid architecture that included a wind generator, battery, price-responsive and thermostatically controlled loads, and a utility grid-tie is presented. The proposed EMS was modelled, and DRL algorithms were used to analyse various scenarios to coordinate the different energy resources. Compared to the other strategies studied, the authors suggested an improved asynchronous advantage actor-critic algorithm (A3C++) demonstrated improved convergence and superior control strategies. Authors in [96] investigated a dynamic pricing and energy consumption scheduling program in a microgrid system, where the service provider (i.e. the microgrid's owner) operates as a broker between the utility company and the customers, buying electric energy from the utility company and selling it to the customers. The RL algorithm was designed to overcome the difficulties of developing an adaptive dynamic pricing scheme in the face of different sources of uncertainties, i.e., consumers' load demand level and wholesale electricity costs. In [97], Wang and Huang looked at the interconnections between interconnected autonomous microgrids and devised a cooperative energy trading and scheduling technique. Work presented in [98] shows that high-rise buildings' on-site wind power generation can sustain all the city's electric vehicles. The coordination of electric car charging with locally generated wind power in a microgrid of buildings using the Markov decision process was examined because the charging demand of electric vehicles does not always correspond with the unreliable wind output. The users' long-term load scheduling problem was investigated by a Markov perfect equilibrium policy and optimized using a DRL algorithm. Bahrami et al. [99] investigated the users' long-term load scheduling problem and developed an online load scheduling learning technique based on the actor-critic technique.

In the recent decade, RL has been widely used in the wholesale electricity market [96], [98] as well as the retail energy sector [87]. RL or DRL-based control approaches are being used to investigate a growing number of energy-related challenges, including microgrid operation [124], [9], online cyber-attack detection in smart grids, and building energy management [125], to mention a few. However, research on the performance of DRL in a grid-tied microgrid where different fluctuating load curves are examined is scarce in the literature. Hence, this chapter seeks to develop an EMS for a grid-tied microgrid based on the DQN approach and then analyse its performance with load demands that have different levels of fluctuation, i.e., slow, medium, and fast fluctuating load profiles. The system is designed to schedule power flows from and to the grid and learn to optimize battery charge and discharge under a complex environment (e.g., battery degradation, price uncertainty, and load uncertainty). More specifically, the key contributions of this chapter are summarized below as.

I. The problem is mathematically formulated, outlining the cost elements considered, such as the utility grid cost model and battery degradation costs. After that, the EMS problem is formulated as a Markov Decision Process (MDP).
II. The DQN technique is applied as the control method to learn the system's energy resource patterns to optimize the energy consumption while minimizing operational costs (trading cost and battery degradation cost) of the microgrid. In addition, the algorithm is tested with different types of load profiles (with slow, medium, and fast fluctuations) to evaluate its efficacy in coping with system uncertainties.

III. A comprehensive performance evaluation of the proposed method is provided through various case studies and simulations to verify its feasibility in dealing with system uncertainties. A cost comparison with traditional Q-learning is also shown to highlight the superiority of the proposed DQN approach.

The rest of the chapter is organized as follows: Section 5.2 elucidates the DQN algorithm for energy management; Section 5.3 presents the utility cost and battery degradation cost model. Section 5.4 demonstrates the detailed design of the proposed EMS as a Markov Decision Process (MDP). Section 5.5 explains the training process. Section 5.6 presents the simulation setup, while Section 5.7 provides the simulation results of the proposed algorithm as well as observations and analysis.

5.2 Proposed Deep Q-Network Methodology

5.2.1 Q-learning Overview

The agent, in the reinforcement learning paradigm, learns to accomplish its task by interacting with its surrounding. It observes the present status of the environment at each time step and takes an action. After the action is executed in the environment, a reward is received, and the state changes. In general, state transitions and reward values are unknown, and the learning agent is unaware of their expected values or probability distribution. The goal of the agent is to find a policy for decision making (i.e., states to optimal actions mapping) that maximizes the total future rewards, starting from the initial state. The most intriguing and challenging instance is learning from delayed rewards [88], where the agent is to optimize its immediate rewards and take into consideration the long-term implications of its actions. This is usually expressed as maximizing the expected future discounted rewards the agent receives throughout the entire episodic task, within each of various independent episodes. This is mathematically shown in equation (5.1) [88]:

$$E[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{5.1}$$

where r_t denotes the total rewards accumulated at a time step t and $\gamma \in [0,1]$ is the discounting factor determining the proportion of future rewards valued at present. RL is an excellent method for dealing with stochastic control problems. When its *trial-and-error* and *learning-as-you-do* method is applicable, it benefits from requiring little or no domain knowledge [31].

Q-learning is a well-known RL technique in which the policy is modified by the value function known as the Q-function. This is discussed in this section.

Let S represent a set of possible states and A a set of discretized actions, where a policy $\pi(s, a)$ represents the likelihood of choosing action $a^t \in A$ in state $s \in S$. At each timestep t, the agent observes a state $s^t \in S$ and selects an action $a^t \in A$, which is then executed in the environment. The agent then gets a reward r^t (which signify the goodness of selecting the specific action at that state) and moves to the next state s^{t+1}. Without prior knowledge of the reward function and transitions probabilities, the Q-learning algorithm seeks to find the optimal policy π that maps every state to the best action over the entire episodic task. At time t, the reward is computed as the future cumulative discounted rewards as shown by equation (5.2) [103]:

$$R^t = \sum_{T=0}^{\infty} \gamma^T r^{t+T+1} \tag{5.2}$$

Here $\gamma \in [0,1]$ represents a discounting factor that balances the importance of future rewards versus immediate rewards. Here $\gamma = 0$ indicates an exclusive interest in immediate rewards. A higher value for γ suggests that future rewards play a more significant role in the optimization task. Typically, as the RL agent traverses through the state space and takes action in state s while following a policy π, and thus transitioning to the next state s' the value of the actions selected are evaluated based on the Q-function as shown in equation (5.3), which is an action-value function that satisfies Bellman's equation [17]:

$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in S} p_{ss'}^{a} (\sum_{a' \in A} \pi(s'a') Q^{\pi}(s'a'))$ (5.3)

where $R(s,a) = \mathbb{E}[r^{t+1} | s^t = s, a^t = a]$ represents the expected reward after taking action a in state s while following a policy π thus moving to the next state whose value is $Q^{\pi}(s'a')$ with a state transition probability $p_{ss'}^{a} = \Pr(s^{t+1} = s' | s^t = s, a^t = a)$. From equation (5.3) [17] , the Q-value of (s,a) is updated by taking the maximum Q-values of $(s'a') \in S \times A$ (next state) multiplied by the state transition probabilities and the discounting factor and then added to the immediate reward the agent receives after performing the action. Literature shows that after updating the action-value function enough times, the iterative algorithm eventually converges to the optimal value [17]. The optimal value function, which denotes the maximum value associated with the best policy, is thus stated in equation (5.4) as [17]:

$Q^{*}(s,a) = R(s,a) + \gamma \sum_{s' \in S} P_{ss'}^{a} \max_{a'} Q^{*}(s'a')$ (5.4)

and the best policy in state s is represented as equation (5.5):

$\pi^{*}(s) = argmax_{a \in A} Q^{*}(s,a)$ (5.5)

As a substitute for the optimum Q-function, the traditional Q-learning algorithm creates a lookup table, also referred to as the Q-table, constituting Q-values for every state-action pairs. After initializing the lookup table randomly, there is a need to balance exploration with exploitation as the Q-values are still unknown to the agent as the learning phase starts. At every time step, there is an action whose current estimate Q-value is best. The action is referred to as the greedy action; however, that current estimate might be erroneous as a better action may exist in the action space. As a result, the solution plan should make use of the goodness gained from the greedy actions while also exploring other actions in the action space. The ε-greedy or epsilon greedy approach will be adopted to select an action from the action space with uniform distribution. This strategy ε is a variable that can any value up to [0,1] [10] where a small value increases the exploitation of the best action in the action space, and values closer to 1 increase the exploration of other actions at every time step. Initially, an immensely large value of ε is leveraged to ensure that all actions are executed in the simulation environment. During the progression of the algorithm, ε is decreased as the greedy action turns to be the optimal action. Where, a^{*} is the optimal action, ε- greedy strategy states that the learning agent should select action a^{*} with probability $1 - \varepsilon$ from the lookup table, i.e., the action with the highest Q-value is selected from the Q-table. In contrast, a random action, i.e. any action in the action set, is established with probability ε [14], [105]. Therefore, Q-learning modifies the corresponding Q-values for each state-action pair in the lookup table according to Bellman's Equation after acquiring the new experience because of actioning a^t, as shown below using equation (5.6) [17],

$Q(s^t, a^t) = Q(s^t, a^t) + \alpha[\, r^{t+1} + \gamma \max_{a_{t+1}} Q(s^{t+1}, a^{t+1}) - Q(s^t, a^t)]$ (5.6)

This update adjusts the Q-value for state s_t and action a_t towards a new estimate value, computed as an immediate reward r_t plus the discounted maximum Q-value for the next state s_{t+1} which the selected action leads to. This value is controlled by the learning rate $\alpha \in [0,1]$.

When the state and action spaces are extremely large, as in large-scale power control problems, the traditional Q-learning method fails for two fundamental causes:

1) Most states in the state space are seldom visited.
2) Lookup table storage becomes intractable [126].

As explained in the section above, a lookup table is generally used to save the updates in the Q-function. However, with increased state-action pairs, the size of the Q table increases as well. Thus, the approach suffers from curse of dimensionality. To overcome this problem, neural networks (NN) are used to approximate the Q-function based on statistical methods. NNs are best selected for RL because of their capability to deal with time-varying target values and their ability to effectively learn from data accumulated through iterative experiences [17].

5.2.2 Deep Q Network (DQN) Overview

A deep Q network (DQN) is a deep reinforcement learning algorithm, that can tackle the problems outlined in section 5.2.1 by combining supervised learning and RL [11]. DQN incorporates deep learning techniques into Q-learning while also utilizing the experience replay method borrowed from the batch reinforcement learning technique [127]. In place of a lookup table, a deep neural network termed the deep Q-network or DQN is utilized to estimate the Q-function [93]. It is formalized as $Q(s, a, w)$, where w is a real-valued vector that represents the DQN's parameters. The general architecture of DQN is depicted in Figure 5.1. The basis of DQN is that w completely determines the function $Q(\cdot,\cdot, w)$. As a result, the challenge of finding the optimal Q-value in an uncountably large functional space is reduced to finding the optimal w value of limited dimensions. However, the agent gathers experiences through its interactions with the environment, just like in traditional Q-learning. The agent creates an experience replay memory by gathering experiences at time t in the form of s^t, a^t, r^t, s^{t+1}. DQN trains with a sampled mini batch (K^t) selected randomly from the experience replay memory rather than just the current experience. According to quasi-static target network strategy [93], two neural networks are employed: the target network with parameters w^t_{target} and the trained/prediction network with parameters w^t_{train} are defined. Once every T_u time steps, w^t_{target} is modified to be equal to w^t_{train}. Target network helps in network performance stabilization. For a sampled mini batch K^t , the mean squared loss of training DQN with experience replay can be defined as equation (5.7) [93] :

$$L(w^t_{train}) = \sum_{s^t a^t r^t s^{t+1} \in K^t} (y^t_{DQN}(r^t, s^{t+1}) \quad - Q(s^t, a^t; w^t_{train}))^2 \tag{5.7}$$

where $Q(s^t, a^t; w^t_{train})$ is the Q-value that the model predicts, $L(w^t_{train})$ is the loss function, $y^t_{DQN}(r^t, s^{t+1})$ is the target value.

The loss is calculated by taking the squared difference between the predicted value and the target value, then its mean is calculated based on the size of the mini-batch K^t.

Here the target output $y^t_{DQN}(r^t, s^{t+1})$ which is used to train the Q network is expressed as equation (5.8) [93].

$$y^t_{DQN}(r^t, s^{t+1}) = r^t + \gamma \cdot max_{a'} Q(s^{t+1}, a', w^t_{target}) \tag{5.8}$$

The right-hand side of equation (5.8) is the immediate reward plus the maximum Q-value of the next state approximated by the target network.

Experience replay is a technique that stores the agent's experiences collected from the simulation environment (experience consisting of state, action, reward, and next state). It adds the advantage of more efficient use of previous experiences by learning from them multiple times. As the experiences acquired from the replay memory are arbitrarily sampled, the sequential correlations in the data during learning are interrupted, thus increasing the algorithm's stability and ensuring the best policy does not get trapped in a local optimum. To reduce the training error defined by equation (5.7), that is, the error over the selected mini batch K^t in each training step, the stochastic gradient descent approach is leveraged. The method returns the new training network parameters based on gradients enumerated from the batch of data sampled of the replay memory and has been demonstrated to swiftly converge to a set of suitable parameters [128].

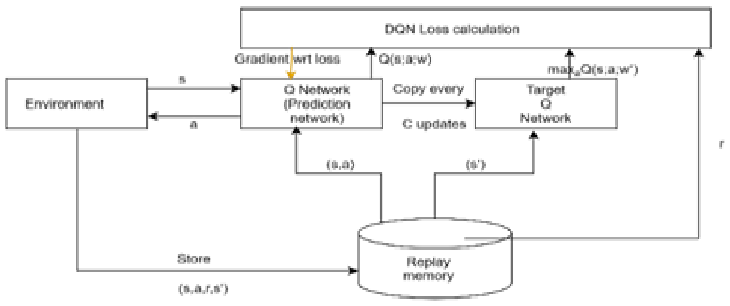


Figure 5.1: Data flow for DQN algorithm with memory replay and target network [110]

5.3 The mathematical formulation for optimization with Deep Q Network

5.3.1 Objective Function

A real-world microgrid aims to meet its actual load demand at the lowest possible cost of energy. Therefore, to compute the optimization algorithm's intended objective function for a grid-tied microgrid equation (5.9) is used.

$$min\{\sum_{t=0}^{t=23}[(C_g(t) + C_{deg}(t))] \tag{5.9}$$

The aim of reducing system running cost (i.e., over a 24-hour horizon) is defined by equation (5.9), where $C_g(t)$ is the grid cost and $C_{deg}(t)$ is the battery degradation cost, with both cost components stated in South African Rand ®. These cost components which the algorithm intends to minimize are elaborated in the next section. In section 5.4, which offers the MDP framework, the mathematical model of the EMS and its system limitations are explained in detail.

5.3.2 Battery Degradation Cost Model

Stress elements that impact battery life are considered while formulating the battery degradation model. Battery ageing can be divided into two types: calendar and cyclic ageing [112]. Calendar ageing reflects the battery's inherent degeneration over time, which is influenced by temperature and an extremely high or low state of charge. Cyclic ageing refers to the capacity loss that occurs each time the battery is charged and drained, and it is affected by the depth of (dis)charge, discharge rate, ambient temperature, and other factors. Excessively high or low states of charge (SoC) will significantly reduce battery charging and discharging efficiency. However, the battery's SoC can be maintained within an acceptable range by constraining the maximum and minimum SoC, as shown in Section 5.4, to avoid overcharging or over-discharging. Temperature can also shorten battery life because the ageing process is exacerbated at high temperatures. The temperature controller, on the other hand, is frequently employed in battery management systems in practice. As a result, battery degradation due to thermal heating effects is assumed to be negligible [114]. This research will exclusively address the impact of depth of discharge (DoD) on the battery because DoD-related stress reflects a proper estimation of battery degradation. DoD is expressed as a function of the SoC of the battery and is represented as [115] $DoD(t) = 1 - SoC(t)$. Interrelation between lithium ion battery DoD and its life cycle data was studied by authors of [116], and they discovered that as the DoD is reduced, the battery's life cycles increase exponentially as

$$L(DoD) = \alpha DoD^{-\beta} \tag{5.10}$$

Equation (5.10) α and β illustrate curve fitting constants proposed by the authors in [116] as 694 and 0.795, respectively [36]. Battery degradation cost (C_{deg}) stemming from battery life-cycle related dynamics is given by equation (5.10) for a battery that runs from $DoD_1\ to\ DoD_2$ with $DoD_2 > DoD_1$, can be calculated with equation (5.11) as illustrated below [115],

$$C_{DoD} = C_{bt}\left|\left(\frac{1}{L(DoD_2)} - \frac{1}{L(DoD1)}\right)\right| \tag{5.11}$$

$L(DoD_j)$ denotes the life cycle of the battery at DoD_j enumerated by equation (5.10) and C_{bt} denotes the battery's initial capital investment per kWh in South African Rand ®. The solver (which can be implemented in Matlab or Python, or any other suitable software) considers absolute values because the cost of degradation is independent of the direction of power flow in the battery. Finally, the costs of battery degradation associated with the control operation are computed as follows using equation (5.12):

$$C_{deg}^{t} = -(C_{DoD}(t)\Delta p(t) \cdot t) \tag{5.12}$$

The cost of drawing or storing power from the BESS in (R) is denoted by C_{deg}^{t} while C_{DoD} is the cyclic degradation cost of the battery per kWh.

5.3.3 Utility grid cost model

When the microgrid is grid-tied, the grid can charge the microgrid's battery sufficiently and meet any load demand that would be otherwise unmet for the microgrid on its own. At every time step t, the microgrid is either exporting power to the utility through the energy storage system or importing power from the grid (but not both actions simultaneously) at the PCC. For this study, the battery has been modelled as the only component that can inject power to the utility, and at each time step, the agent can either charge or discharge the battery with the main grid, as explained in section 5.4. The instantaneous grid tariff is given by $G_t(t)$ in (R/kWh). To stimulate local usage of PV power and reduce the negative impact of microgrid’s uncertainties to the power system, including low power quality and voltage instability resulting from volatility of RES, selling prices are usually lower than the buying prices in most circumstances [117]. The time of use (ToU) purchasing tariff is discounted by a factor ϑ [26] to model the selling rates. As a result, the cost of exchanging power with the utility at PCC is:

$$C_g(t) = -(G_t(t)P_{g\_p}(t).t + \vartheta\, G_t(t)P_{g\_s}(t).t) \tag{5.13}$$

where $0 < \vartheta < 1$, $P_{g\_p}(t)$ represents the imported power from the utility while $P_{g\_s}(t)$ represents the exported power at each time step t. In section 5.4 equation (5.20), power is further discussed in detail. When $C_g(t)$ is negative the microgrid incurs a cost as power is being imported from utility grid, and when it is positive, the microgrid earns money as power is exported to the utility. The instantaneous grid power limits are set as $0 \le P_{g\_p}(t) \le P_{g\_s}^{max}$, $P_{g\_s}(t) \le P_{g\_s}^{max}$, $\forall t \in T$ and $P_{g\_s}(t) \cdot P_{g\_p}(t)$. A contract exists between the microgrid owner and the distribution system operator (DSO) that specifies the maximum amount of power exchanged between the microgrid and the utility at the point of common connection (PCC) limits can be changed.

5.4 EMS Formulation Using the Markov Decision Process (MDP) Framework

A study presented in [87] explores the applications of Q-learning algorithm to manage energy flows in a grid-tied microgrid. The work is modelled as an MDP where a Q-learning method is employed to tackle that MDP. A similar work employing the DQN method incorporates recent learning algorithm enhancements such as experience replay, and target network is studied in this article. It is worth noting the MDP modelling presented in this section is a continuation of [87]. As already mentioned, a Markov decision process (MDP) is a mathematical framework used to model decision-making when the outcomes are partially random and partially controllable. MDP is widely used to map optimization issues tackled using the reinforcement learning paradigm [86]. It is formalized as a four-tuple (S, A, T, R), with S and A denoting state and action spaces, T and R denoting state transition probability, and reward function, respectively. If MDPs are solved through RL, state transition probabilities are not necessarily needed [18], [39]. Thus, the state, action and reward function will be modelled in this section.

5.4.1 State and State Space

The state's data is critical for energy management since it provides the agent's information in making decisions at each time step *t*. For example, the BESS state of charge, load demand (kW), the PV production (kW), and grid tariff (R/kWh) all define the EMS's state at every time step. Let the battery's state of charge at time step t be represented by $SoC = \{SoC_t\}$. To ensure that the battery SoC limits are not violated by preventing overcharging or over-discharging a guard ratio β is factored in as, $\beta \cdot \frac{E}{E} \leq SoC_t \leq (1-\beta) \cdot \frac{E}{E}$ where $\beta \in [0{,}0.5]$ [21] and E represents the battery's energy capacity in (kWh). It is to be noted that β is a range used to determine the battery's upper and lower bounds depending on the battery type and manufacturers' specifications. The term $\beta \cdot \frac{E}{E} = SoC_{min}$ while the term $\beta \cdot \frac{E}{E} \leq SoC_t \leq (1-\beta) \cdot \frac{E}{E} = SoC_{max}$. Thus, at every timestep, the battery is constrained by, $SoC^{min} \leq SoC_t \leq SoC^{max}$, where SoC^{min} and SoC^{max} depicts the battery's minimum and maximum state of charge boundaries.

Given the specified battery operating bounds above, the state s_t at each time slot, *t* is shown as,

$$s_t = \{t, P_t^{PV}, SoC_t, G_t, P_{l,t}\} \tag{5.14}$$

where t represents the time component, P_t^{PV}(kW) represents solar PV generation (kW) , G_t is the instantaneous grid tariff (R/kWh), $P_{l,t}$ is the microgrid's load demand at time t(kW). The state space is defined as, $S = s_0 \cup s_1 \cup, \dots \cup s_{T-1}$, where S is a union of all individual state sets in the considered optimization horizon. T time steps have been used to partition the microgrid's intraday operation as $t = \{0,1,2,3, \dots, T-1\}$, where T is the optimization horizon under consideration. For an intraday operation, T varies from 0-23 according to the hours of the day.

5.4.2 Action and Action Space

The energy algorithm seeks first to use the available solar PV energy and the stored battery energy to meet the load demand each time step *t*, then imports the remaining energy from the utility to meet its load. The microgrid's net load $P_{l,t}^{Net}$(kW) at every time step, *t* is defined as the entire load demand ($P_{l,t}$) subtracted by the solar PV (P_t^{PV}) generation as illustrated below in equation (5.15).

$$P_{l,t}^{Net} = \max\left((P_{l,t} - P_t^{PV}), 0\right) \tag{5.15}$$

In a practical microgrid, the total load demand $P_{l,t}$ and solar PV production P_t^{PV} fluctuates randomly; thus the microgrid's net demand $P_{l,t}^{Net}$ is an unknown variable. The EMS first uses the energy stored in the BESS to try and satisfy the net demand $P_{l,t}^{Net}$. The utility then provides the remaining load demand that the BESS is unable to meet. It is termed as the remainder power $P_{l,t}^{rem}$ (kW) which is computed using equation (5.16):

$$P_{l,t}^{rem} = \max\left(P_{l,t}^{Net} - (SoC_t - SoC^{min}) * E, 0\right) \tag{5.16}$$

The quantity of power that is required to be imported from the grid at every time step is represented as $P_{l,t}^{rem}$. After meeting the load demand, the amount of energy stored in the BESS indicated as SoC_t^nIs calculated by equation (5.17) as shown here.

$$SoC_t^n = \min\left(SoC^{max}, (\max(P_t^{PV} - P_{l,t}, 0) + \max((SoC_t - SoC^{min}) \cdot E - P_{l,t}^{Net}, 0))/E\right) \tag{5.17}$$

It is to be noted that the agent can only dispatch the battery, that is, charge or discharge it with the main grid. To make this problem easier to understand, the actions are separated into two categories: discharging and charging. The power unit Δp

represents the quantity of power utilized to charge or discharge the battery at any given time. The discrete action space is described as;

$$\mathcal{A}_{s_t} = \{-k\Delta p, \dots, -\Delta p, 0, \Delta p, .., k\Delta p\}, \tag{5.18}$$

$k\Delta p$ and -$k\Delta p$ represent the maximum charge and discharge power to the BESS in every timeslot respectively, where 0 is an action value indicating the battery is idle. Here positive power indicates charging the battery (power delivered to the battery from the grid), and negative power values indicate discharging. $a_t \in \mathcal{A}_{s_t}$ is the action chosen by the agent at time step t, whereas $\mathcal{A}_{s_t}$ denotes all the possible actions in the action space $\mathcal{A}$ under state S_t. Considering the action space $\mathcal{A}_{s_t}$ in equation (5.18), once the agent observes a state s_t , it selects an action a_t, from $\mathcal{A}_{s_t}$ through a policy π, which outlines a decision-making approach for agent action selection. More information on π is presented in section (5.2).

Let BESS (a_t) be the function that computes the amount of power delivered to the BESS when the agent performs an action a_t be represented as;

$$BESS(a_t) = \begin{Bmatrix} -k(a_t)/E, & if\ a_t = discharging \\ k(a_t)/E, & if\ a_t = charging \end{Bmatrix} \tag{5.19}$$

where negative power values indicate a discharge from the BESS and positive power value indicates battery charging. As a result of the agent action $BESS(a_t)$ to the battery, SoC_t^n is moved to the next SoC value SOC_{t+1}^n. If the action executed $a_t(charging)$ increases the $SOC_t^n + k(a_t)/E$ above the upper guard limit E^{max}, only the energy chargeable $SoC^{max} - SOC_t^n$ is used to charge the battery, and the rest is not used up as charging has stopped to prevent overcharging. Likewise, only $SoC_t^n - SoC^{min}$ is discharged, and any excess energy is not discharged to avoid over-discharging the battery.

5.4.3 Reward Function

A reward is a scalar quantity used to link the goal of the learning process to the agent. A reward is given to the agent once it selects an action and transitions to the next state. Proper reward shaping is crucial because it connects the agent's actions to the algorithm's goal [89], [119]. The goal of optimization in this work is to reduce battery degradation costs while lowering the transaction cost of the power exchanged with the utility grid.

The proposed EMS' reward $\mathrm{r}(\mathrm{s_t}, \mathrm{a_t})$ is constructed to assess three components of system management, one of which is the objective function, while the other two given by [95] are used to enhance the agent's performance. The cost of acquiring energy from the main grid C_g (R) and the cost of battery degeneration C_{deg} (R) are both factored into the objective function given by equation (5.9). C_b (R) and C_o(R) have been included in the algorithm to boost performance. C_o is a negative reward (penalty) charged to the agent if it selects actions exceeding the battery's constraints whereas C_b denoted as pre-charge energy rewards the agent for choosing actions that charge the battery, ensuring its load is not exposed to any future high prices.

At each time step, the pay reward C_g^t covers the total incurred costs of exchanging power with the utility. This reward depends on the action selected by the agent (a_t). The agent gets a positive reward with G_t discounted by ϑ if the amount of energy exported to the utility is greater than the purchased energy. Else, the agent receives a negative reward of C_g^t computed as equation (5.20).

$$C_g^t = -\left(P_{l,t}^{rem} + BESS(a_t) \cdot E\right) \cdot G_t \tag{5.20}$$

In equation (5.20) $P_{l,t}^{rem}$ is the microgrid's unserved load demand at each time step (kW) while G_t is the instantaneous grid tariff (R/kWh). The sum total of power exchanged with the main grid at each time slot t is illustrated as $P_{l,t}^{rem} + BESS(a_t) \cdot E$.

To always ensure that the available stored energy in the BESS SoC_t can satisfy the net load demand $P_{l,t}^{Net}$a reward termed as c_b^t is added as a positive reward to the total reward computation $r(s_t, a_t)$. This reward primarily motivates the agent to keep the battery's SoC high such that at each time step, the net load can independently be covered by the battery. It is modelled with the grid tariff G_t such that as prices increase, the benefit increases as well. In basic terms, the reward represents the lower cost of using the battery rather than buying power from the grid [95].

$$c_b^t = \begin{cases} P_{l,t}^{Net} \cdot G_t & if\ P_{l,t}^{Net} \leq SOC_t^n - SOC^{min}) \cdot E \\ (SOC_t^n - SOC^{min}) \cdot G_t \cdot E, & else \end{cases} \tag{5.21}$$

Finally, the penalty c_o^t is computed as shown in equation (5.22) for any excess energy delivered to the battery by the agent but is not used in the charging/discharging of the battery due to enforced battery limits.

$$c_o^t = \begin{cases} -(2(SOC_t^n + k(a_t) - SOC^{max}) \cdot G_t \cdot E & if(SOC_t^n + k(a_t) > SOC^{max} \\ -2\big(|k(a_t)| - (SOC_t^n + SOC^{max})\big) \cdot G_t \cdot E & elif(SOC_t^n + k(a_t) < SOC^{min} \\ 0 & else \end{cases} \tag{5.22}$$

To compute the total reward, battery degradation cost C_{deg}^t calculated as shown in equation (5.12) is also included as a negative reward in equation (5.23) which computes the total reward that the agent receives [87]. Let $r(s_t, a_t s_{t+1})$ represent cumulative reward received by the agent for taking an action a_t at state s_t which is determined as the total of all rewards stated above as [87].

$$r(s_t, a_t) = C_g^t + C_{deg}^t + C_b^t + C_o^t \tag{5.23}$$

5.5 DQN algorithm for grid-tied PV battery microgrid energy management

```
1.**Inputs**: $N$=20000, $K$=8, $\varepsilon_{min} = 0.001$, $\varepsilon = 1$, $\gamma = 0.95$, $\alpha = 0.001$, $C$=100
2.Initialize replay buffer with capacity $N$ and the mini-batch $K$
3.Initialize the network ($Q(s, a; w)$) with random weights $w_o$
4.Initialize the target $\hat{Q}(s, a; w)$ with random weights $w_o^-$
5. **For** *episode* =0 to max *episode*, **do**
6.      Initialize Simulation Environment
7.      **For** t=0, T-1, **do**
8.          Observe the current state $s_t$.
9.          Choose an action $a_t$ from $A_{s_t}$ using the $\varepsilon$ greedy.
10.         Execute the selected action $a_t$ in the microgrid's simulated environment.
```

```
11.            Get the next state $s_{t+1}$, and reward $r_{t+1}$.
12.            Store the transition set ($s_t, a_t, r_{t+1}, s_{t+1}$) to replay buffer.
13.            Sample mini-batch of transitions ($s_j, a_j, r_{j+1}, s_{j+1}$) from replay memory ($N$) randomly
14.             If $s_{j+1}$ terminal state, then
15.                 $target_j = r_j$
16.             else
17.                 $target_j = r_j + \gamma \max \hat{Q}(s_{j+1}, a_{j+1}; w^-)$
18.      Do a gradient descent with loss ($[target_j - Q(s_t, a_t, w)]^2$) and step size $\alpha$
19.      Every $C$ episodes update target weights $w^-$ with current network weights $w$
20.         Set $s_t \leftarrow s_{t+1}$
21.          Update epsilon $\varepsilon$
22.              End for
23.         End for
24. End for
```

5.5.1 DQN Learning Algorithm

This chapter extends on the works presented in [93] and [109] that used deep Q network with experience replay and the target network. This study however employs DQN algorithm to learn optimal energy flow in the designed EMS environment presented in section 5.4. Q-learning has been proven to converge to the best policy with a probability of one [87]. To approximate the action-value or function in deep Q-learning, a deep Q network (DQN) is used. The learning agent keeps a dedicated DQN that accepts the current state as an input and outputs the value functions for each action in the given state. Multiple episodes are required to train the DQN. The learning agent uses the epsilon greedy or ε -greedy strategy to explore the state-action space in each episode, choosing the action with the maximum Q-value with probability ε and a random action with probability $1-\varepsilon$. The ε −greedy strategy seeks to balance the exploration-exploitation conundrum, i.e., exploiting the currently available best Q-value value and exploration of a better alternative. At each timestep t, the initial state is read, and the value of battery energy is initialized (initial SoC value of 0.25 is selected, which is in the range of maximum and minimum SoC values). After a state is read for every episode, the epsilon greedy strategy is used to pick an action from the list of possible actions. The list consists of battery charge and discharge power values in kW. The action selected is then executed to the simulation environment modelled as MDP and the next state, which is$s_{t+1} = \{t+1, P_{t+1}^{PV}, SOC_{t+1}^{n}, G_{t+1}, P_{l,t+1}\}$and reward are obtained as the output. The agent then collects and stores the state, action, reward, and next state tuple, ($s_t, a_t, r_{t+1}, s_{t+1}$) in the replay memory buffer. Once the experience replay memory has enough sample equivalate to the capacity of the mini-batch, a sample mini-batch K is randomly taken to training DQN. The process continues until the agent reaches the terminal state $(T-1)$, and the algorithm runs up to the last episode [109].

The process of training is shown in the algorithm. After the learning is complete, a trained DQN model with optimized w^* such that $Q\ (s, a, w^*)$ best estimates the optimal Q-value for each state is returned. To retrieve the optimized policy, state values are fed to the trained DQN model, and the model returns the control actions that maximize the Q-value, i.e., optimal actions. Thus, from this, the optimal schedule of the battery and optimal power flows from and to the grid from $t = 0$ to $t = T - 1$ are acquired.

5.6 Simulations Set Up

To test the proposed approach effectiveness and performance, several test cases are studied i.e.

- Slow fluctuating load profile
- Medium fluctuating load profile
- Fast fluctuating load profile

5.6.1 Experimental Data

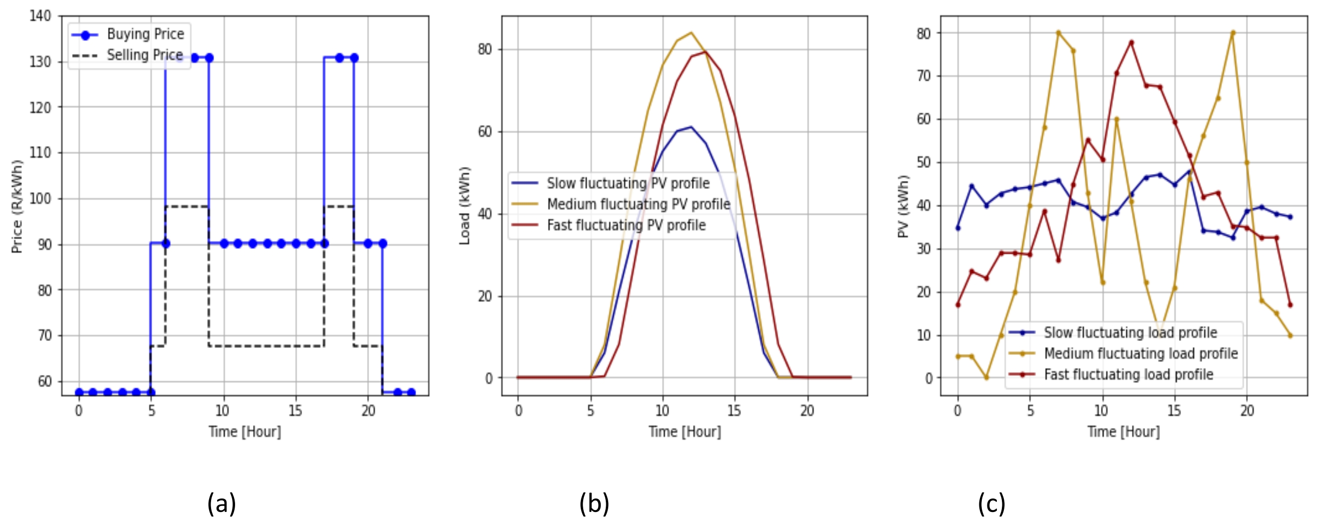


Figure 5.2: (a) ToU price profile (b) Solar PV generation profiles (c) Energy demand profiles

The algorithm is tested using a slow, medium, and fast fluctuating demand profiles to demonstrate the ability to adapt to changes in the environment and deal with uncertainties effectively. Load profile data for the slow fluctuating load curve is obtained from a building in a university in South Africa, for medium fluctuating load, a modified commercial load profile data adopted from [120] is used, whereas, for the case of fast switching load curve, electric vehicle charging station (EVCS) data adopted from [10] is leveraged. Figure 5.2 displays the different sets of load curves considered in this study. The simulation is based on a solar PV system located in Cape Town, South Africa [121]. For the energy prices, a Time of Use (ToU) grid tariff provided by the local grid operator in South Africa (ESKOM) is used. Specifically, summer rate is considered, which are composed of on-peak R139/kWh, mid-peak R90/kWh, and off-peak R57kWh for the buying prices [113]. To stimulate local PV adoption and mitigate the negative effects of microgrid uncertainty on the main grid, selling prices are discounted by a factor of $\vartheta = 0.75$ [26]. The value of 0.75 for this factor is adopted from [46], and the same is also used extensively in similar research. A Lithium-ion battery of 150kWh capacity is used, and the (dis)charging power is discretized to $\mathcal{A} = [-50\text{kW}, -25\text{kW}, 0, 25\text{kW}, 50\text{kW}]$. The initial, lower, and upper SoC values were set to 0.25, 0.05, and 0.95, respectively for all the case studies. The initial battery investment cost is calculated using the current market price of a Li-ion battery of $135/kWh (R2025/kWh) [62]. The efficiencies of solar PV and battery power inverters are assumed to be 100% for simplicity's sake, meaning no power is lost at the power electronics interface. Grid export power is constrained to 50kW, while the import power limit is set to 80kW. Here the grid export power is assumed to be constrained by the battery discharge limit while the import power is not strictly constrained as the agent should learn to minimize the importing cost. However, for all the load curves, it is ensured that the 80kW limit is not exceeded.

5.6.2 DQN Parameters Selection

The Q-function in the tests is estimated using a simple deep neural network (DNN) with three fully connected hidden layers, while each layer consists of 16, 16, and 16 neurons, respectively. As the activation function, the rectified linear unit is used because it learns fast, is simple and seems to work well empirically [93]. Adam's algorithm is used for loss minimization between the predicted value and the target value, as shown in equation (5.7) [129]. The experience replay buffer memory size is set to 20000 based on the RAM capacity. The batch size for this project was selected based on trial and error as a value of 8 gave the best results. The epsilon ε which controls action exploration and exploitation was decreased linearly from a value of 1 to 0.001, and at around episode 4000, exploration probability gets to 0.001. Update of the target network weights with policy network weight occurred after every 100 iterations. The simulation study was carried out using Python 3.7.0 with PyTorch 1.1.0, a machine learning package. The model is trained using the algorithm presented in section 5.5, and the hyper-parameters are shown in table 5.1.

Table 5.1: DQN hyperparameters

Hyperparameters	**Value**
Experience replay memory size N	20000
Experience replay mini-batch size K	8
Learning rate α	0.001
Discounting factor γ	0.95
Epsilon ε in the ε greedy policy	From 1 linearly decreases to 0.001
Number of training episodes	15000
Target network update interval C	After every 100 episodes
Timestep Δt	1 hr

5.7 Simulation Results and Discussions

5.7.1 Reward Convergence Analysis

The controller's performance is initially evaluated in terms of the cumulative rewards earned during the training phase. This thesis looks at how the reward varies over time and how quickly the agent learns the best feasible policy because the agent's goal is to have a policy that maximizes rewards across the entire optimization horizon. Figure 5.3 displays the training curve, which shows the average reward convergence tendency for three different load curves in the proposed energy management scheme over the training phase of 15000 episodes. It should be noted that the results in Figure 5.3 were produced using the same simulation settings as in Table 5.1. This is because the epsilon greedy mechanism incorporates an exploration activity into the learning phase. The initial reward is low because the learning agent is engaging in trial-and-error. The epsilon (ε) value starts at one and decays linearly over time. In this way, the agent begins to explore aggressively early on. At about episode 4000, when epsilon decreases to a value of 0.001, the agent becomes greedier, i.e., control actions that yield higher rewards are selected with higher probability. Finally, the algorithm converges at about episode 5000, producing the optimized policies as seen in the section below. Since the DQN keeps on choosing random actions with a small probability of epsilon 0.001, the episodic rewards fluctuate from episode 5000 to episode 15000. The result demonstrates that the proposed approach succeeded in learning to increase the cumulative rewards. The online DQN neural network is trained using the Adam's algorithm, which backpropagates the gradient of the loss value for every component in the neural network's weights vector. As the training process begins, the weights are initialized arbitrary, and the backpropagation algorithm minimizes the loss function shown in equation (5.7) starting from the output layer and progressing backwards. After training, the optimal weight parameters of DQN are used to acquire the optimized control actions.

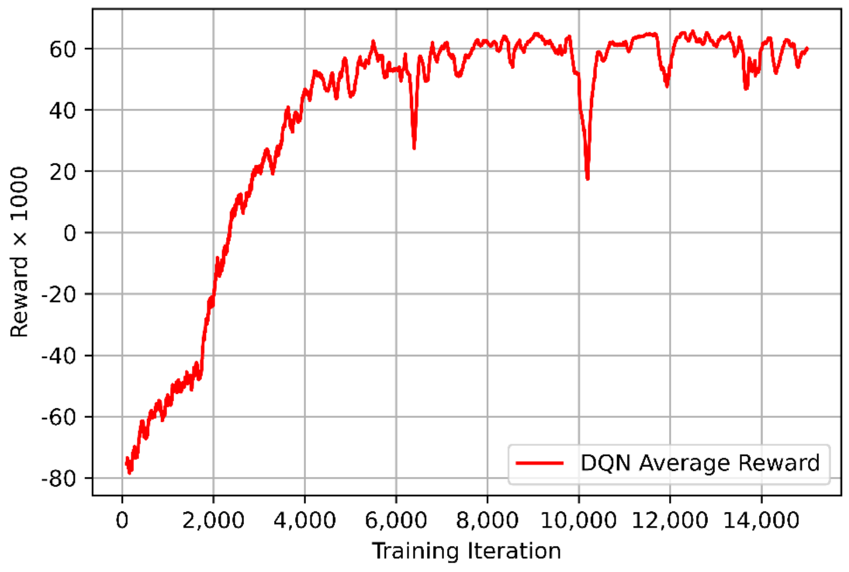
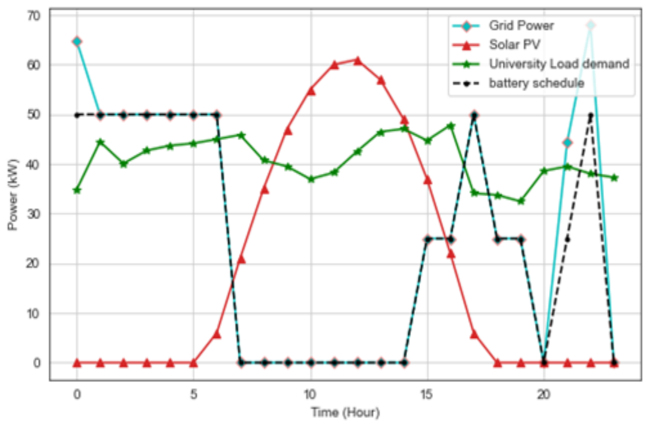


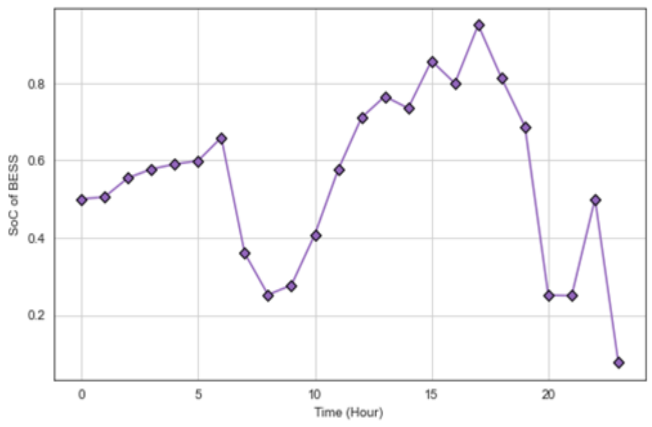
Figure 5.3: Learning characteristics for the DQN algorithm

5.7.2 Energy Management Analysis for the trading case

Case study 1 (Slow Fluctuating Load Profile): Figure 5.4a displays the microgrid's optimized power schedules for Case Study 1 for slow fluctuating load profile. Figure 5.4b shows the SoC trajectories for the battery. Since, for this case, the load curve is smooth, i.e., the power fluctuation rate is very minimal. It is seen that the algorithm learns to shift the microgrid's load from times when grid tariff is peak to off-peak times, thus minimizing the aggregate peak load and reducing system running cost. Between 00:00 to 06:00, when grid prices are very low, power draw from the utility is very high, at 00:00 65kW is imported, wherefrom 01:00 to 06:00 50kW is constantly drawn. This occurred because the algorithm initiated a cost-effective charging process at the low ToU period to completely shift the demand if grid prices increase in the future. During this time, the battery's SoC gradually increased to about 0.65 by constantly charging the battery with 50kW for 7 hours with the cheap grid power. From 07:00 to 09:00, when grid prices peak (a 73R/kWh increase is recorded), it is seen that the agent learns to schedule zero grid power at that time as the load is being met by the battery and partly solar PV. From 10:00 to 14:00, as there is sufficient PV power to meet the demand in full, the excess PV, which amounts to an average of 13kW, is used to charge up the battery. Thus, a slight increase in battery SoC is seen. At 15:00, when tariffs are at mid-peak and PV generation is slowly decreasing, the algorithm opts to gradually charge the battery in advance to support the microgrid in case of any unforeseen uncertainties. Thus, from 15:00 to 17:00, the SoC of the battery is increased to 0.85 to meet the demand during the second peak, which occurs at 18:00 and 19:00 when PV generation is zero. At 18:00, when R40 increases the utility tariff, the SoC drastically decreases because grid power intake has been lowered to 25kW and further reduced to zero at 20:00. This is resultant from the fact the microgrid's demand has been catered for by the battery in full.

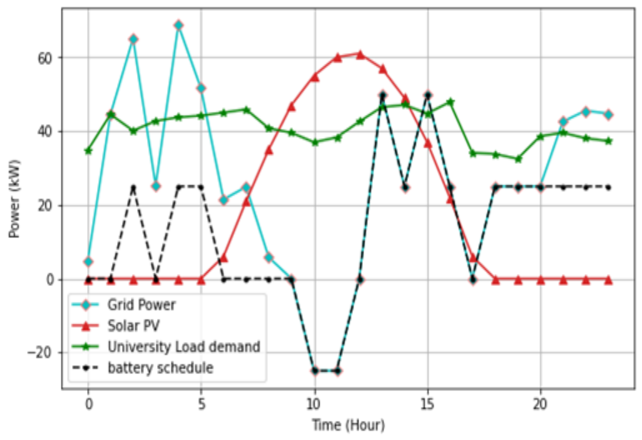


(a)

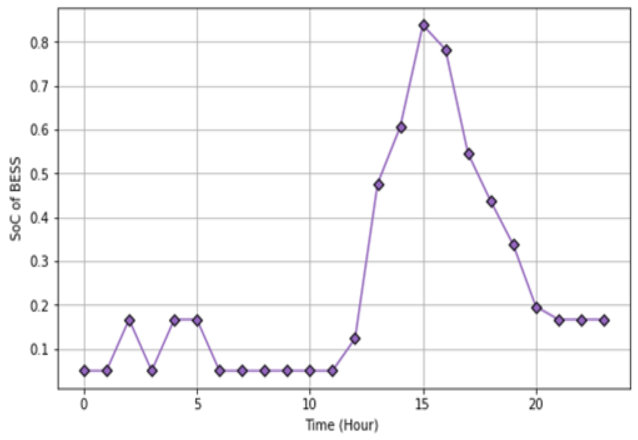


(b)

Figure 5.4: (a) Optimized energy schedules (b) tendency of the SoC of the battery for slow fluctuating load profile (Obtained using DQN algorithm).



(a)



(b)

Figure 5.5: (a) Energy schedules (b) tendency of the SoC of the battery for slow fluctuating load profile (Obtained using conventional Q-learning)

At 22:00, when utility prices change to off-peak, a large uptake of the cheap grid power is seen to charge the battery and partly meet the uncovered load. At 23:00, the policy opts to keep the battery idle; this happens because 23:00 is the last time step, and the agent is rewarded for minimizing power intake from the grid; thus, it opts to idle. From these results, the algorithm is seen to learn grid tariff and solar PV uncertainties instead for the case of a slow fluctuating load profile. From figure 4b, it is seen the algorithm raises SoC by charging the BESS when the utility prices are low and lowering power intake when the prices are high. These charging actions effectively reduce the overall operating cost and increase system efficiency by purchasing energy when prices are low and shifting loads from when the utility is heavily loaded.

Figures 5.5a and 5.5b show the energy schedules and BESS SoC obtained by the traditional Q-learning algorithm. Unlike in the DQN algorithm, it's observed that the Q-learning approach doesn't take advantage of the cheap grid power to raise the battery's SoC; thus, part of the microgrid's load is forced to be covered by the grid at high grid tariffs. From 00:00 to 05:00, it can be observed that for 3 hours, the battery is charged with 25kW, and the utility covers the unmet load demand. From 06:00 to 09:00, when grid tariff's, it is seen that the expensive grid power meets the microgrid's load as the battery cannot cover the load demand in full. At 10:00 and 11:00, the agent is seen to discharge the battery to earn so revenue at mid-peak prices. The net demand, as previously stated, is unknown and stochastic, and therefore, like the DQN agent, the Q-learning agent charges the battery to ensure it meets its net demand in the future. From 13:00 to 16:00, the battery SoC is raised, such that when the grid prices peak for the second time, grid power intake is minimized. In comparison, the DQN algorithm performs better as it takes proactive decisions and can also learn the grid tariff uncertainties and plans by ultimately shifting the microgrid's load.

Case study 2: Figures 5.6a and 5.6b show the optimized energy schedules and the changes in SoC of the battery for the medium fluctuating load profile. For this type of load curve, peak load occurs when PV generation is also at its peak. However, like the slow switching load curve, it is noted that the agent learns the tariff's uncertainties by increasing the battery SoC to minimize utility peak load demand. From 00:00 to 06:00, when grid tariffs are off-peak and lower, the algorithm gradually increases the SoC of the battery to 0.85 to meet its load when tariffs increase. From 07:00 to 09:00, when the first grid tariff peak occurs (utility prices increase by 73R/kWh), the agent prioritizes power discharge into the grid to obtain revenue, and this is done within the contract's limits. At 10:00 to 16:00, the PV is sufficient to supply the load; however, at 10:00, the algorithm

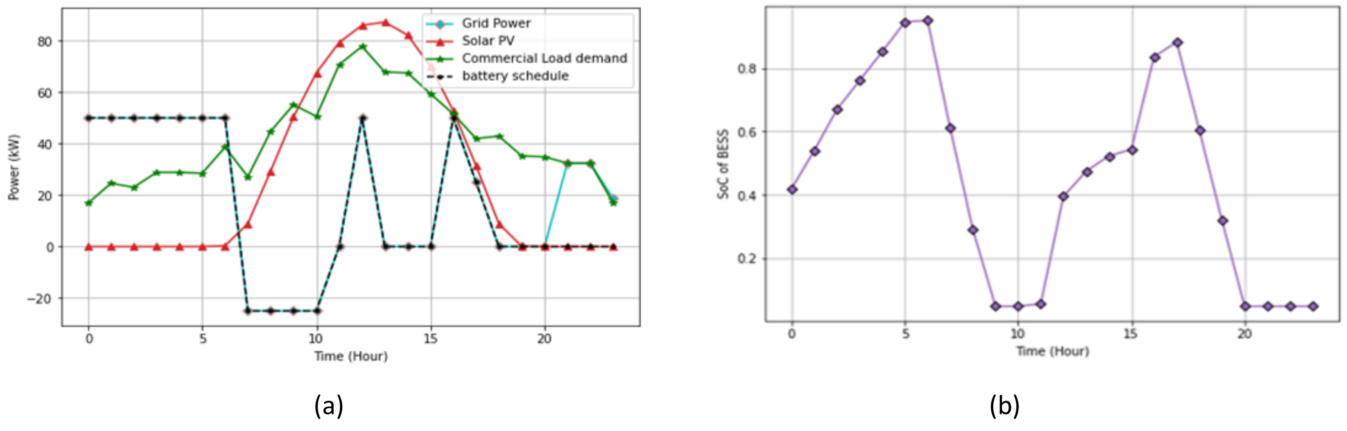
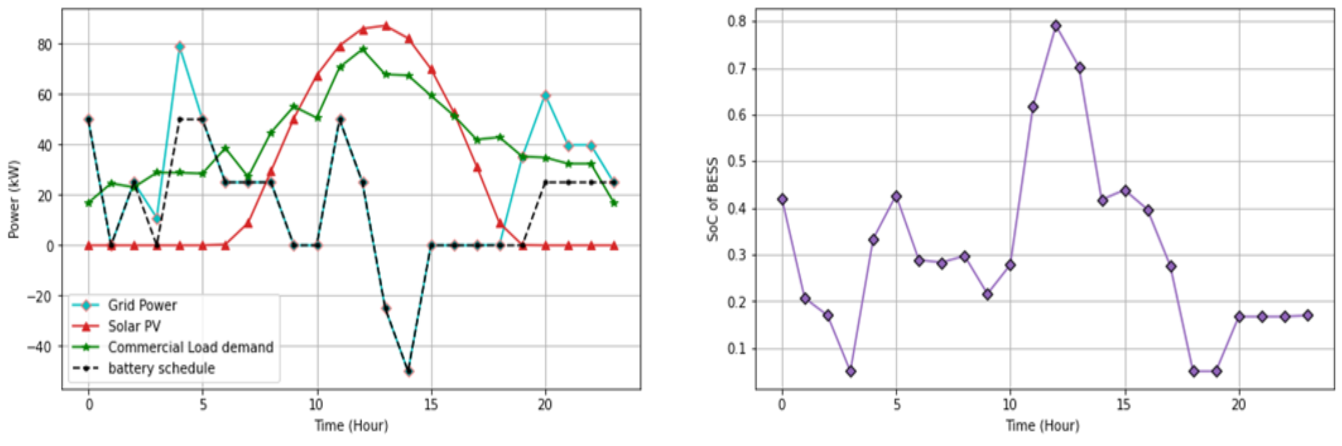


Figure 5.6: (a) optimized energy schedules (b) tendency of the SoC of the battery for medium fluctuating load profile (Obtained using DQN algorithm)

discharges the battery again at mid-peak prices to obtain revenue. This indicates the effectiveness of the method in learning utility tariffs and supporting the utility during peak times. From 13:00 to 15:00, zero grid power is scheduled as PV generation can cater for the total load fully, and any excess PV power is used to charge the battery; thus, a slight gradual increase in SoC for the battery is noted. Owing to the fact that the second peak occurs at18:00 and 19:00 when PV generation is scarce, and the load demand is still high, the BESS is charged with energy in advance at 16:00 and 17:00 for potential future load shifting. It is seen that at 18:00 and 19:00, the algorithm reduces the uptake of power from the utility to zero for 2 hours, thus indicating that the agent can learn policies to boost the ESS's SoC to meet net demand on their own. At 20:00, when grid tariffs are lowered by R40, an increase in power absorbed from the grid is recorded as the policy recommends the battery to remain idle as the SoC is very low. However, the tariffs are cheap; hence, the microgrid can fully utilize the cheap power to supply its load. Regarding microgrid's uncertainties, it is seen that the agent learns to schedule high grid power uptake when tariffs are low and low grid power intake when grid prices are high. This helps in filling grid demand curve valleys.

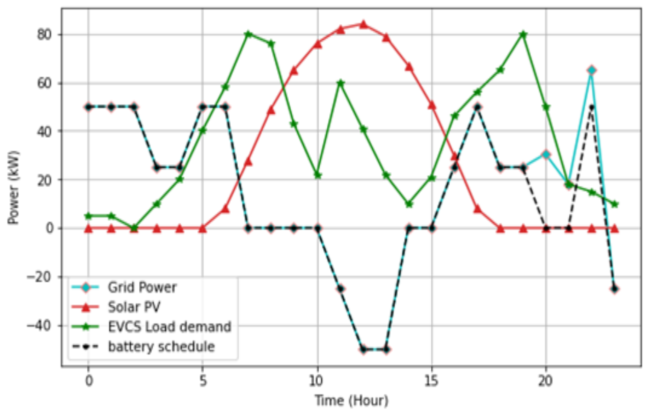


(a) (b)

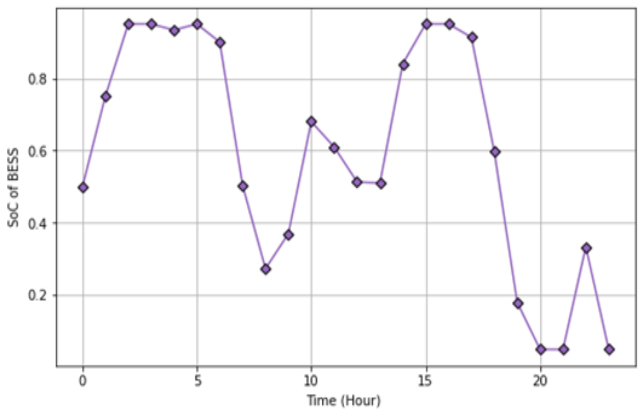
Figure 5.7: (a) Energy schedules (b) tendency of the SoC of the battery for medium fluctuating load profile (Obtained using Q learning algorithm).

The inability to see and plan for the morning grid tariff peak and learn a profitable schedule is evident in the Q learning algorithm. Figures 5.7a and 5.7b display energy schedules and BESS SoC obtained by the traditional Q learning algorithm. Like the slow switching, the Q learning algorithm doesn't quite learn to take full advantage of the cheap grid power between 00:00 and 05:00. As a result, it is observed that at 07:00, when grid tariffs peak, the microgrid can't meet its net demand in full and part of its load is covered by the expensive grid power. In comparison to the DQN algorithm in figure 5.7a, the agent trades power to the grid at that peak tariff time, thus ripping more revenue and supporting the main grid when it experiences its peak demand. At 13:00 and 14:00, the agent discharges the battery with 25kW and 50 kW respectively to earn revenue at mid-peak prices. From 15:00 to 18:00, zero power is scheduled from the grid as the reducing PV power and energy stored in the battery is used to meet the microgrid's load in full. In comparison to the DQN algorithm, which manages to schedule zero grid power for 3 hours at the second peak tariff, which occurs at 18:00 and 19:00, the Q learning algorithm only lowers the load demand to zero for only one hour. From 20:00, when grid tariffs are reduced, the constant charge policy of 25kW is recommended for 4 hours which helps the battery to end with a slightly high energy level compared to the DQN approach. In comparison to the DQN algorithm, it is observed that the DQN algorithm learns to recommend actions that can cope with the unpredictable demand and takes the utility's dynamic tariff to support it in times of peak demand.

Case study 3: The fast-fluctuating load curve considered is an electric vehicle charging station load curve obtained from [10]. Figures 5.8a and 5.8b display the optimized energy schedules and the SoC trajectories during a 24-hour horizon for the fast-fluctuating load demand. The EVCS load demand profile indicates that it peaks in the morning when the grid tariffs are also at their peak and peaks again in mid-day and the evening as well. Since energy consumption, grid tariff, and solar PV are unknown information; the developed algorithm should strategically maximize the utilization of energy stored in the battery and ensure load demand is not exposed to future high grid prices. An efficient EMS should simultaneously meet any unexpected high load demand effectively and still minimize system operational costs. From 00:00 to 02:00, the load demand is very minimal; however, it is seen that the algorithm increases the SoC of the battery to about 0.9 to meet its highest net demand of 80kW that occurs the same time when tariffs are high, i.e., from 06:00 to 08:00. Thus, 25kW is constantly drawn from the utility from 03:00 to 04:00 and 50kW is imported at 00:00 to 02:00 and 05:00 and 06:000 as the prices are low and because the algorithm

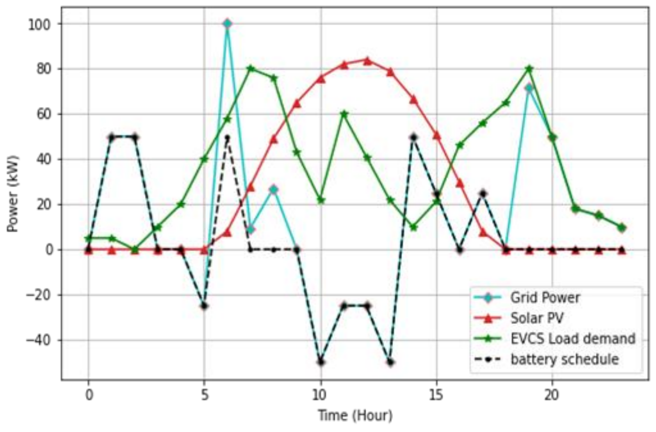


(a)

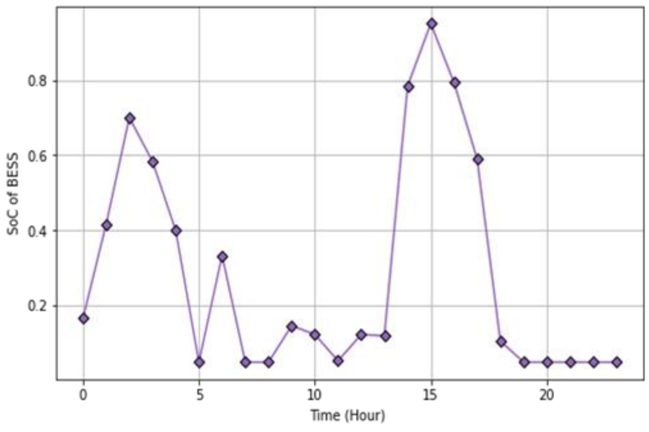


(b)

Figure 5.8: (a) optimized energy schedules (b) tendency of the SoC of the battery for fast fluctuating load profile. (Obtained using DQN)



(a)



(b)

Figure 5.9: (a) Energy schedules (b) tendency of the SoC of the battery for fast fluctuating load profile. (Obtained using conventional Q-learning)

foresees the unpredictable peak demand and prepares for it in advance. From 07:00, when grid tariffs and EVCS load simultaneously change to peak, it is noted that the battery can cope with the unpredictable demand as zero power is scheduled from the grid (grid peak minimization), and all microgrids load is catered for by the battery. The second mid-peak occurs at 11:00 when solar PV is sufficient to cover for it entirely. From 11:00 to 13:00, as solar PV is more than the load demand, the EMS obtained revenue by selling the excess power to the utility grid at mid-peak prices. The third peak demand of the microgrid is also 80kW, and it is seen to occur between 17:00 to 19:00. Note that this peak coincides with the second tariff peak that occurs between 18:00 and 19:00. Selling Power to the utility gives the agent more rewards, but we see the agent was strategic in discharging the battery up to 0.5. This implies that the proposed approach takes proactive decisions by being mindful of its future load demand. At 16:00 and 17:00, the battery is charged up to 0.85, and at 18:00 and 19:00, when the tariffs are high, EVCS peak load is minimized to by 68% as a constant uptake of 25kW grid power is recorded. SoC is observed to decrease as the battery is supporting large amounts of energy demand. From 22:00 to 23:00, when the tariff moves to off-peak, the SoC of the battery is increased by charging the battery with the cheap grid power. At 23:00, we observe a slight instability where the agent performs a suboptimal action of discharging the battery to rip some revenue at off-peak tariff. This may have resulted from the stochasticity in the environment, which causes the suboptimal decision by the agent. However, in all the other timesteps, results demonstrate the efficacy of the proposed DQN approach to optimally dispatch the battery in both load and tariff uncertainties, whereby the agent learns policies to raise the battery's SoC, thus enabling the EMS to cope with a fast-fluctuating load profile.

Figures 5.9a and 5.9b show the energy schedules obtained using the Q-learning algorithm. Like the slow and medium fluctuating load profiles, the algorithm doesn't learn to prepare for the highest net demand, as seen in the plots. The algorithm starts by raising the SoC to around 0.65 between 01:00 and 02:00. However, the consecutive actions are not optimal as it is observed that the battery remains idle and worse still gets discharged when grid tariffs are very low. Consequently, at 07:00, when grid tariffs peak, it is observed that part of the microgrid's load is exposed to the high grid prices. From 10:00 to 13:00, the agent is observed to discharge the battery at the mid-peak prices, when also the PV power is available. Unlike the DQN algorithm, which manages to lower power intake when grid tariffs peak the second time at 18:00 and 19:00, Q-learning doesn't learn to prepare for the second-highest net demand as seen in Figure 5.9a as at 19:00 very high power of about 80kW is being covered by the grid. From 19:00 to 23:00, the load is fully met by the grid as the battery SoC is low, as the algorithm recommends an idle action and PV is zero at that time.

5.7.3 Cost Reduction Analysis

Figure 5.10 shows the comparative analysis of the daily operational costs for the three load demands based on the DQN algorithm and conventional Q-learning. The learning capability of the agent causes a reduction in microgrid overall operation cost, which is computed as shown in equation 5.9. However, as can be seen in figure 5.10, DQN outperforms Q-learning in reducing system operational cost as it obtains a lower global cost over all the tested load curves. For the case studies considered, DQN showed more advantages and better learning capabilities. It significantly reduced daily operational costs by about 15%, 24%, and 26% for the slow, medium, and fast fluctuating load profiles, respectively Q-learning. These results indicate the robustness of DRL techniques in achieving a lower global cost in highly stochastic settings and further serves as a core motivation to deploy learning algorithms in power systems. These improved results are attributed to the fact that the DQN algorithm employs an experience replay memory that allows the agent to learn from past experiences as new data is collected. This strategy ensures that the algorithm is not trapped in a local minimum and handles system data more efficiently. Unlike the vanilla Q-learning method that sequentially updates a Q-table, in DQN, the agent is trained randomly, i.e., training data is sampled from the replay memory randomly and used to update the DQN. This strategy breaks data correlations from consecutive experiences, thus leading to sufficient learning and achievement of profitable energy schedules.

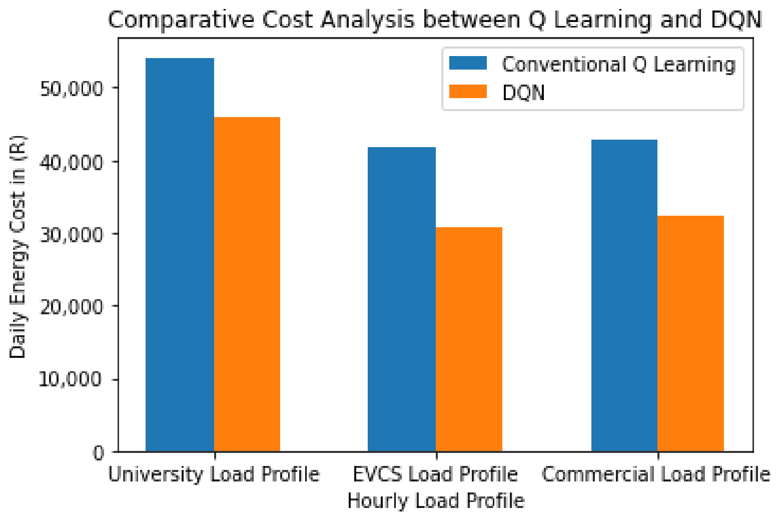


Figure 5.10: Cost comparison between DQN and traditional Q- learning in the case studies analysed.

5.7.4 Summary

The simulation results show that the DQN algorithm does outperform the traditional Q-learning algorithm to achieve stable results with lower operational costs. The fact that reinforcement learning approaches are very adaptable is the most important reason for their superiority. However, designing generic heuristic rules that would operate well in different operating conditions is not easy. In this article, the DQN approach was tested with different fluctuating load profiles. As seen in the section above, the DQN algorithm achieves much better results than the traditional Q-learning method. As the level of stochasticity in the environment increases, it is noteworthy that conventional Q-learning fails to obtain optimal policies. It's worth noting that DQN policies are generated using the same set of hyperparameters, implying that the suggested approach is adaptable to different load fluctuations. DQN is more computationally intensive regarding training time as it takes 584 seconds to train while Q-learning takes 44 seconds. However, the time it takes to retrieve the optimized policies, i.e., the time it takes to observe states and decide the actions that must be taken to operate the microgrid, is 0.0035 seconds for the DQN approach and 0.0039 seconds for the Q-learning method, indicating that proposed method can be effectively leveraged in real-time operations.

6 CONCLUSION AND FUTURE RECOMMENDATIONS

The key results and contributions of this work are presented in this chapter. It also makes recommendations for future works.

6.1 Conclusion

The area of EMS in the last decade has generated tremendous research interest in its microgrid applications. In an energy management system, optimization algorithms are needed to minimize the energy drawn from the utility grid, level peak loads, balance energy fluctuations, maximize renewable energy usage, and reduce power losses. This thesis aimed to explore two reinforcement learning techniques, namely Q learning and deep Q network, for energy scheduling purposes in a grid-tied solar PV battery microgrid.

First, the energy management algorithm based on Q learning was developed to manage energy flows between the battery and the main grid in a microgrid supplying power to a commercial load. The EMS was formulated as a Markov decision process to ensure practical application considering state, action, and reward function. Considering a ToU grid tariff and stationary battery degradation cost, simulation results indicated that cost minimization could be achieved with appropriate hyperparameter tuning and proper restriction to the action space at each state. Moreover, the findings showed that responding appropriately to the dynamic grid tariff is a critical component of cost reduction and system efficiency. It is noteworthy that the Q learning algorithm managed to lower operational costs in the two case studies regardless of the different tariff structures and the seasons considered. However, comparing the non-trading EMS to the trading EMS model, the energy trading algorithm achieved slightly better results as it reduced the energy costs by 4.033% more in the summer season and 2.199% in the winter season. The reinforcement learning approach was observed to avoided high operational prices, efficiently utilized PV generation, and ensured reasonable SoC levels, thus has the potential to be used in grid support applications such as peak load shaving and increasing system efficiency.

The thesis also approached the challenge of designing methods to improve conventional reinforcement learning to deep reinforcement learning by developing a grid-tied microgrid's deep Q network-based energy management scheme. Then, the performance of the deep Q network was evaluated with different types of load fluctuations, i.e., slow, medium, and fast fluctuating load profiles. First, the problem was mathematically formulated, outlining each cost component that must be taken into accounts, such as battery degradation costs and grid energy purchase costs using a ToU tariff. After that, the problem is stated as a Markov Decision Process (MDP), which involves describing the operation of the microgrid as a state, action, and reward function. The MDP was then solved using a deep Q-network method, and the simulation results contrasted to those obtained using the traditional Q-learning method. Experiment results exhibited that the developed DQN approach learns the battery's best control policy considering tariff and load uncertainties and battery degradation. Still, the proposed approach outperformed the conventional Q-learning method in terms of system operation cost reduction and battery utilization for all the different load profiles considered. The proposed unique formulation of the EMS problem and the implementation of recently established learning algorithm improvements, such as experience replay and target network, were the primary reasons for the attained effective performance, i.e., improved convergence and better policies.

6.2 Recommendations for Future works

Due to the high levels of complexity and uncertainty involved in renewable-based microgrids, designing and executing an effective EMS for them is a difficult undertaking. Though reinforcement learning approaches have proven successful in solving complex computer games problems, they are not perfect. When it comes to real-world implementations, RL approaches face significant challenges due to insufficient data, algorithmic hyperparameter tuning, and instability concerns. Presently, academics and researchers worldwide are working together to improve performance and enable the application of these algorithms in solving power system problems.

Further research based on the application of reinforcement learning to power systems can be extended in different directions, as stated below.

- Research to develop better tuning mechanisms rather than trial and error methods should be investigated as hand-tuning of hyperparameters can be problematic.
- More scenarios can be explored using other modern reinforcement learning techniques (such as soft actor critic (SA3C) deep deterministic policy gradient (DDPG) and twin delayed DDPG (T3D) among others) and different grid tariffs. Also, introducing flexibility on the demand side can be an interesting axis for future research.
- The proposed optimal energy management system can be further used in related studies, such as IEEE test feeders.
- Developing an appropriate pricing scheme that considers both the interests of distribution network operators and those of prosumers is also a worthwhile research topic to address.
- Further developments on the objective function aimed at minimizing power losses at the distribution lines and power electronics interface can also be done.
- Further developments of this model, including battery degradation resulting from thermal heating related factors, can also be investigated.
- In the control mechanism, clustered microgrids' exchange of energy and their interoperability is also worth investigating.

7 REFERENCES

[1] T. M. Pippia, “Model-based control for hybrid and uncertain smart energy systems,” Delft Univeristy of Technology, 2020.

[2] “The Paris Agreement | UNFCCC.” https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement (accessed Aug. 25, 2021).

[3] D. Laudiero, “Stochastic Control Strategies for Residential Microgrids,” Delft University of Technology, 2018.

[4] S. Khatoon, Ibraheem, A. K. Singh, and Priti, “Effects of various factors on electric load forecasting: An overview,” *Proc. 6th IEEE Power India Int. Conf. PIICON 2014*, vol. 2014, pp. 1–5, 2014, doi: 10.1109/34084POWERI.2014.7117763.

[5] S. O. Showers, “Benefits and Challenges of Energy Storage Technologies in High Penetration Renewable Energy Power Systems,” *2019 IEEE PES/IAS PowerAfrica*, pp. 209–214, 2019.

[6] E. O. Arwa and K. A. Folly, “Reinforcement Learning Techniques for Optimal Power Control in Grid-Connected Microgrids : A Comprehensive Review,” *IEEE Access*, vol. 8, pp. 208992-209007, 2020, doi: doi: 10.1109/ACCESS.2020.3038735.

[7] P. K. Singhal, “Dynamic Programming Approach for Large Scale Unit Commitment Problem,” in *International Conference on Communication Systems and Network Technologies*, 2011, pp. 714–717, doi: 10.1109/CSNT.2011.152.

[8] V. S. Borra and K. Debnath, “Dynamic Programming for Solving Unit Commitment and Security Problems in Microgrid Systems,” *2018 IEEE Int. Conf. Innov. Res. Dev.*, no. May, pp. 1–6, 2018.

[9] Luu Ngoc An and Tran Quoc-Tuan, “Optimal Energy Management for Grid Connected Microgrid by using Dynamic Programming Method,” *2015 IEEE Power Energy Soc. Gen. Meet.*, no. 1, pp. 1-5, 2015, doi: doi: 10.1109/PESGM.2015.7286094.

[10] M. O. Badawy, Y. Sozer, and S. Member, “Power Flow Management of a Grid Tied PV-Battery System for Electric Vehicles Charging,” *IEEE Trans. Ind. Appl.*, vol. 53, no. 2, pp. 1347–1357, 2017, doi: 10.1109/TIA.2016.2633526.

[11] B. Zhao, X. Zhang, J. Chen, C. Wang, S. Member, and L. Guo, “Operation Optimization of Standalone Microgrids Considering Lifetime Characteristics of Battery Energy Storage System,” *IEEE Trans. Sustain. ENERGY*, vol. 4, no. 4, pp. 934–943, 2013, doi: 10.1109/TSTE.2013.2248400.

[12] S. Asefi, M. Ali, and E. Gryazina, “Optimal Energy Management for Off-Grid Hybrid System using Hybrid Optimization Technique,” *Proc. 2019 IEEE PES Innov. Smart Grid Technol. Eur. ISGT-Europe 2019*, no. September, pp. 1–5, 2019, doi: 10.1109/ISGTEurope.2019.8905550.

[13] F. F. and G. F. M. Hijjo, “PV-Battery-Diesel Microgrid Layout Design Based on Stochastic Optimization,” *2017 6th Int. Conf. Clean Electr. Power (ICCEP), St. Margherita Ligure*, pp. 30–35, 2017, doi: doi: 10.1109/ICCEP.2017.8004787.

[14] E. A. Jasmin, T. P. I. Ahamed, and V. P. Jagathiraj, “A Reinforcement Learning Algorithm to Economic Dispatch Considering Transmission Losses,” in *TENCON 2008 - 2008 IEEE Region 10 Conference, Hyderabad*, 2008, pp. 1–6,

doi: 10.1109/TENCON.2008.4766652.

[15] M. F. Zia, E. Elbouchikhi, and M. Benbouzid, “Microgrids energy management systems: A critical review on methods, solutions, and prospects,” *Appl. Energy*, vol. 222, no. May, pp. 1033–1055, 2018, doi: 10.1016/j.apenergy.2018.04.103.

[16] J. X. Wang *et al.*, “Learning to Reinforcement Learn,” *Mach. Learn.*, vol. 3, p. 17, 2017, [Online]. Available: https://arxiv.org/abs/1611.05763.

[17] R. S. Sutton, A. G. Barto, and F. Bach, *Reinforcement Learning: An Introduction*, 2nd Editio. Cambridge, Massachusetts: The MIT Press, 2018.

[18] K. Mason and S. Grijalva, “A review of reinforcement learning for autonomous building energy management,” *Comput. Electr. Eng.*, vol. 78, pp. 300–312, 2019, doi: 10.1016/j.compeleceng.2019.07.019.

[19] Arwa O. Erick; Komla A. Folly, “Power Flow Management in Electric Vehicles Charging Station Using Reinforcement Learning,” in *2020 IEEE Congress on Evolutionary Computation (CEC), Glasgow, United Kingdom*, 2020, pp. 1–8, doi: 10.1109/CEC48606.2020.9185652.

[20] E. Mocanu *et al.*, “Transactions on Smart Grid On-line Building Energy Optimization using Deep Reinforcement Learning,” *IEEE Trans. Smart Grid*, vol. 3053, no. c, pp. 1–11, 2018, doi: 10.1109/TSG.2018.2834219.

[21] S. Kim and H. Lim, “Reinforcement Learning Based Energy Management Algorithm for Smart Energy Buildings,” *Energies*, vol. 11, no. 8, 2018, doi: 10.3390/en11082010.

[22] B. V Mbuwir, F. Ruelens, F. Spiessens, and G. Deconinck, “Battery Energy Management in a Microgrid Using Batch Reinforcement Learning †,” *MDPI*, pp. 1–19, 2017, doi: 10.3390/en10111846.

[23] E. Foruzan, L. Soh, S. Asgarpoor, and S. Member, “Reinforcement Learning Approach for Optimal Distributed Energy Management in a Microgrid,” *IEEE Trans. Power Syst.*, vol. 8950, no. c, pp. 1–9, 2018, doi: 10.1109/TPWRS.2018.2823641.

[24] R. Leo, R. S. Milton and S. Sibi, “Reinforcement Learning for Optimal Energy Management of a Solar Microgrid,” in *2014 IEEE Global Humanitarian Technology Conference - South Asia Satellite (GHTC-SAS)*, 2014, pp. 183–188, doi: doi: 10.1109/GHTC-SAS.2014.6967580.

[25] P. Zeng, H. Li, H. He, and S. Li, “Dynamic Energy Management of a Microgrid using Approximate Dynamic Programming and Deep,” *IEEE Trans. Smart Grid*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/TSG.2018.2859821.

[26] P. Chen, M. Liu, C. Chen, and X. Shang, “A Battery Management Strategy in Microgrid for Personalized Customer Requirements,” *Energy*, vol. 189, p. 116245, 2019, doi: 10.1016/j.energy.2019.116245.

[27] F. Chang, T. Chen, W. Su, and Q. Alsafasfeh, “Control of battery charging based on reinforcement learning and long short-term memory networks,” *Comput. Electr. Eng.*, vol. 85, 2020, doi: 10.1016/j.compeleceng.2020.106670.

[28] R. Lu and S. H. Hong, “Incentive-based demand response for smart grid with reinforcement learning and deep neural network,” *Appl. Energy*, vol. 236, no. November 2018, pp. 937–949, 2019, doi: 10.1016/j.apenergy.2018.12.061.

[29] R. Lu, S. H. Hong, and X. Zhang, “A Dynamic pricing demand response algorithm for smart grid : Reinforcement learning approach,” *Appl. Energy*, vol. 220, no. November 2017, pp. 220–230, 2018, doi: 10.1016/j.apenergy.2018.03.072.

[30] T. A. Nakabi and P. Toivanen, “Deep Reinforcement Learning for Energy Management in a Microgrid with Flexible Demand,” *Sustain. Energy, Grids Networks*, vol. 25, no. October, p. 100413, 2021, doi: 10.1016/j.segan.2020.100413.

[31] W. Kolodziejczyk, I. Zoltowska, and P. Cichosz, “Control Engineering Practice Real-Time Energy Purchase Optimization for a Storage-Integrated Photovoltaic System by Deep Reinforcement Learning,” *Control Eng. Pract.*, vol. 106, no. August 2020, p. 104598, 2021, doi: 10.1016/j.conengprac.2020.104598.

[32] E. Samadi, A. Badri, and R. Ebrahimpour, “Electrical Power and Energy Systems Decentralized multi-agent based energy management of microgrid using reinforcement learning,” *Electr. Power Energy Syst.*, vol. 122, no. May, p. 106211, 2020, doi: 10.1016/j.ijepes.2020.106211.

[33] Y. Ji, J. Wang, J. Xu, X. Fang, and H. Zhang, “Real-Time Energy Management of a Microgrid using Deep Reinforcement Learning,” *Energies*, vol. 12, no. 12, 2019, doi: 10.3390/en12122291.

[34] R. François-Lavet, V.; Taralla, D.; Ernst, D.; Fonteneau, “Deep Reinforcement Learning Solutions for Energy Microgrids Management,” in *In Proceedings of the 13th European Workshop on Reinforcement Learning (EWRL 2016), Barcelona, Spain*, vol. 3–4, no. 2015, pp. 1–7.

[35] E. Kuznetsova, Y. Li, C. Ruiz, E. Zio, G. Ault, and K. Bell, “Reinforcement learning for microgrid energy management,” *Energy*, vol. 59, pp. 133–146, 2013, doi: 10.1016/j.energy.2013.05.060.

[36] L. S. Communication, “The smart grid: An introduction,” 2011. doi: 10.1002/9781119521129.ch15.

[37] H. E. Brown, S. Suryanarayanan, and G. T. Heydt, “Some characteristics of emerging distribution systems considering the smart grid initiative,” *Electr. J.*, vol. 23, no. 5, pp. 64–75, 2010, doi: 10.1016/j.tej.2010.05.005.

[38] A. L. Dimeas, S. Member, N. D. Hatziargyriou, and S. Member, “Operation of a Multiagent System for Microgrid Control,” *IEEE Trans. Power Electron.*, vol. 20, no. 3, pp. 1447–1455, 2005, doi: 10.1109/TPWRS.2005.852060.

[39] A. Hirsch, Y. Parag, and J. Guerrero, “Microgrids : A review of technologies , key drivers , and outstanding issues,” *Renew. Sustain. Energy Rev.*, vol. 90, no. September 2017, pp. 402–411, 2018, doi: 10.1016/j.rser.2018.03.040.

[40] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad, “State of the Art in Research on Microgrids : A Review,” *IEEE Access*, vol. 3, pp. 890–925, 2015, doi: 10.1109/ACCESS.2015.2443119.

[41] A. V. Souto, “Optimization & Energy Management of a Microgrid Based on Frequency Communications,” TU Delft University of Technology, 2016.

[42] S. X. Chen, S. Member, H. B. Gooi, S. Member, M. Q. Wang, and S. Member, “Sizing of Energy Storage for Microgrids,” *ieee Trans. sma*, vol. 3, no. March 2012, pp. 142–151, 2014, doi: 10.1109/TSG.2011.2160745.

[43] W. Xiao *et al.*, “Real-Time Identification of Optimal Operating Points in Photovoltaic Power Systems,” *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1017–1026, 2006, doi: 10.1109/TIE.2006.878355.

[44] T. Vral, “Frequency Based Cellular Microgrid Control,” TU Delft University of Technology, 2016.

[45] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, “Microgrids,” *IEEE Power Energy Mag.*, vol. 5, no. August, pp. 78–94, 2007.

[46] J. C. Beardsall, “Energy Storage Systems : A Review of the Technology and Its Application in Power Systems,” *2015 50th Int. Univ. Power Eng. Conf.*, pp. 1–6, 2015, doi: 10.1109/UPEC.2015.7339794.

[47] X. Luo, J. Wang, M. Dooner, and J. Clarke, “Overview of current development in electrical energy storage technologies and the application potential in power system operation,” *Appl. Energy*, vol. 137, pp. 511–536, 2014, doi: 10.1016/j.apenergy.2014.09.081.

[48] J. Pegueroles-Queralt, F. D. Bianchi, and O. Gomis-Bellmunt, “Control of a lithium battery storage system for microgrid applications,” *J. Power Sources*, vol. 272, no. 1, pp. 531–540, 2014, doi: 10.1016/j.jpowsour.2014.08.087.

[49] A. Sedic and Z. Guzovic, “Oil drilling rig diesel power-plant fuel efficiency improvement potentials through rule-based generator scheduling and utilization of battery energy storage system,” vol. 121, pp. 194–211, 2016, doi: 10.1016/j.enconman.2016.05.022.

[50] H. Qian, J. Zhang, and W. Yu, “A High-Efficiency Grid-Tie Battery Energy Storage System,” *IEEE Trans. Power Electron.*, vol. 26, no. 3, pp. 886–896, 2011, doi: 10.1109/TPEL.2010.2096562.

[51] M. Faisal, M. A. Hannan, S. Member, and F. Blaabjerg, “Review of Energy Storage System Technologies in Microgrid Applications : Issues and Challenges,” *IEEE Access*, vol. 6, pp. 35143–35164, 2018, doi: 10.1109/ACCESS.2018.2841407.

[52] P. Bajpai and V. Dash, “Hybrid renewable energy systems for power generation in stand-alone applications : A review,” *Renew. Sustain. Energy Rev.*, vol. 16, no. 5, pp. 2926–2939, 2012, doi: 10.1016/j.rser.2012.02.009.

[53] M. Farhadi, S. Member, and O. Mohammed, “Energy Storage Technologies for High Power Applications,” *ieee Trans. ind*, vol. 9994, no. c, pp. 1–9, 2015, doi: 10.1109/TIA.2015.2511096.

[54] J. M. Crider and S. D. Sudhoff, “Reducing Impact of Pulsed Power Loads on Microgrid Power Systems,” *IEEE Trans. Smart Grid*, vol. 1, no. 3, pp. 270–277, 2010, doi: 10.1109/TSG.2010.2080329.

[55] R. Amirante, E. Cassone, E. Distaso, and P. Tamburrano, “Overview on recent developments in energy storage : Mechanical , electrochemical and hydrogen technologies,” *Energy Convers. Manag.*, vol. 132, pp. 372–387, 2017, doi: 10.1016/j.enconman.2016.11.046.

[56] X. Zhou, Z. Fan, Y. Ma, Z. Gao, X. Zhang, and J. Zhang, “Research Review on Energy Storage Technology in Power Grid,” *2018 IEEE Int. Conf. Mechatronics Autom.*, pp. 155–161, 2018.

[57] D. P. Dubal, O. Ayyad, and V. Ruiz, “Hybrid energy storage: the merging of batteyr and supercapacitor chemistries,” *R. Soc. Chem.*, vol. 44, pp. 1777–1790, 2015, doi: 10.1039/C4CS00266K.

[58] H. F. Habib, A. A. S. Mohamed, M. El Hariri, and O. A. Mohammed, “Utilizing supercapacitors for resiliency enhancements and adaptive microgrid protection against communication failures,” *Electr. Power Syst. Res.*, vol. 145, pp. 223–233, 2017, doi: 10.1016/j.epsr.2016.12.027.

[59] F. Díaz-González, A. Sumper, O. Gomis-Bellmunt, and R. Villafáfila-Robles, "A review of energy storage technologies for wind power applications," *Renew. Sustain. Energy Rev.*, vol. 16, no. 4, pp. 2154–2171, 2012, doi: 10.1016/j.rser.2012.01.029.

[60] T. M. I. Mahlia, T. J. Saktisahdan, A. Jannifar, M. H. Hasan, and H. S. C. Matseelar, "A review of available methods and development on energy storage; Technology update," *Renew. Sustain. Energy Rev.*, vol. 33, pp. 532–545, 2014, doi: 10.1016/j.rser.2014.01.068.

[61] B. K. Chakrabarti *et al.*, "Hybrid Redox Flow Cells with Enhanced Electrochemical Performance via Binderless and Electrophoretically Deposited Nitrogen-Doped Graphene on Carbon Paper Electrodes," *ACS Appl. Mater. Interfaces*, vol. 12, no. 48, pp. 53869–53878, Dec. 2020, doi: 10.1021/ACSAMI.0C17616.

[62] "• Worldwide - lithium ion battery pack costs | Statista." https://www.statista.com/statistics/883118/global-lithium-ion-battery-pack-costs/ (accessed Feb. 03, 2021).

[63] L. Meng, E. Riva, A. Luna, T. Dragicevic, J. C. Vasquez, and J. M. Guerrero, "Microgrid supervisory controllers and energy management systems : A literature review," *Renew. Sustain. Energy Rev.*, vol. 60, pp. 1263–1273, 2016, doi: 10.1016/j.rser.2016.03.003.

[64] A. Vargas-mart, L. I. Minchala-avila, and L. E. Garza-casta, "A review of optimal control techniques applied to the energy management and control of microgrids," vol. 52, no. Seit, pp. 780–787, 2015, doi: 10.1016/j.procs.2015.05.133.

[65] J. Hossain, A. Mahmud, and B. Azzopardi, *Renewable Energy Integration*. Springer Berlin Heidelberg, 2014.

[66] P. Aristidou, A. Dimeas, and N. Hatziargyriou, "Microgrid Modelling and Analysis Using Game Theory Methods," in *E Energy , LNICST*, 2011, vol. 54, pp. 12–19.

[67] Arwa O. Erick; Komla A. Folly, "Energy Trading in Grid-connected PV-Battery Electric Vehicle Charging Station," in *Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, 2020, no. February, pp. 1–6, doi: 10.13140/RG.2.2.11795.22568.

[68] E. N. Krapels, "microgrid development," *IEEE Power and Energy Magazine*, no. august, pp. 94–96, 2013.

[69] H. J. Ã, J. Chuanwen, and X. Rong, "A review on distributed energy resources and MicroGrid," vol. 12, pp. 2472–2483, 2008, doi: 10.1016/j.rser.2007.06.004.

[70] J. John, F. Mwasilu, J. Lee, and J. Jung, "AC-microgrids versus DC-microgrids with distributed energy resources : A review," *Renew. Sustain. Energy Rev.*, vol. 24, pp. 387–405, 2013, doi: 10.1016/j.rser.2013.03.067.

[71] L. Che and M. Shahidehpour, "DC Microgrids : Economic Operation and Enhancement of Resilience by Hierarchical Control," no. October, 2014, doi: 10.1109/TSG.2014.2344024.

[72] J. ˜ao A. P. Lopes, A. ´eGuimar ˜aes Madureira, and C. C. L. M. Moreira, "A view of microgrids," *WIREs Energy Env.*, vol. 00, no. August, pp. 1–18, 2012, doi: 10.1002/wene.34.

[73] B. S. Bahramirad, A. Khodaei, and J. Svachula, "Building Resilient Integrated Grids," *IEEE Electrification Magazine*, no. March, pp. 48–55, 2015.

[74] A. D. Paquette and D. M. Divan, “Providing Improved Power Quality in Microgrids,” *IEEE Industry Applications Magazine*, no. July, pp. 34–43, 2014.

[75] A. Khodaei, “Resiliency-Oriented Microgrid Optimal Scheduling,” *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1584–1591, 2014, doi: 10.1109/TSG.2014.2311465.

[76] I. Rendroyoko, “A Literature Survey of Optimization Technique of Unit Commitment Implementation in Microgrid Electricity System With Renewable Energy Sources,” *2019 2nd Int. Conf. High Volt. Eng. Power Syst.*, pp. 344–349, 2019.

[77] E. Delarue, D. Cattrysse, and W. D, “Enhanced priority list unit commitment method for power systems with a high share of renewables,” *Electr. Power Syst. Res.*, vol. 105, pp. 115–123, 2013, doi: 10.1016/j.epsr.2013.07.014.

[78] Y. Tingfang and A. P. Formulation, “Methodological Priority List for Unit Commitment Problem,” in *2008 International Conference on Computer Science and Software Engineering Methodological*, 2008, no. 2, pp. 176–179, doi: 10.1109/CSSE.2008.714.

[79] M. Zhai, Y. Liu, T. Zhang, and Y. Zhang, “Robust Model Predictive Control for Energy Management of Isolated Microgrids,” in *IEEE IEEM*, 2017, pp. 2049–2053.

[80] Y. Zhang, F. Meng, R. Wang, W. Zhu, and X. Zeng, “A stochastic MPC based approach to integrated energy management in microgrids,” *Sustain. Cities Soc.*, vol. 41, no. February, pp. 349–362, 2018, doi: 10.1016/j.scs.2018.05.044.

[81] K. Y. Lee and M. A. El-Sharkawi, *Modern Heuristic Optimization Techniques*, 1st Editio. Wiley Interscience, 2008.

[82] C. S. Edrington and D. A. Cartes, “Online Economic Environmental Optimization of a Microgrid Using an Improved Fast Evolutionary Programming Technique,” in *41st North American Power Symposium*, 2009, pp. 1–6, doi: 10.1109/NAPS.2009.5484060.

[83] R. V. P. Jagathy, “Reinforcement Learning Approaches To Power System Scheduling,” Cochin University of Science and Technology, 2008.

[84] H. Li, D. Yang, W. Su, J. Lu, and X. Yu, “An Overall Distribution Particle Swarm Optimization MPPT Algorithm for Photovoltaic System Under Partial Shading,” *IEEE Trans. Ind. Electron.*, vol. 66, no. 1, pp. 265–275, 2019, doi: 10.1109/TIE.2018.2829668.

[85] H. Li, A. T. Eseye, J. Zhang, and D. Zheng, “Optimal energy management for industrial microgrids with high-penetration renewables,” pp. 1–14, 2017, doi: 10.1186/s41601-017-0040-6.

[86] E. F. Morales and J. H. Zaragoza, “An Introduction to Reinforcement Learning,” *Decis. Theory Model. Appl. Artif. Intell. Concepts Solut.*, pp. 63–80, 2011, doi: 10.4018/978-1-60960-165-2.ch004.

[87] G. Muriithi and S. Chowdhury, “Optimal Energy Management of a Grid-Tied Solar PV-Battery Microgrid : A Reinforcement Learning Approach,” *Energies*, vol. 2021, no. 14, p. 2700, 2021, doi: https://doi.org/10.3390/en14092700.

[88] C.J.C.H. Watkins, “Learning from Delayed Rewards,” University of Cambridge, 1989.

[89] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is Enough," *Artif. Intell.*, vol. 299, p. 103535, 2021, doi: 10.1016/j.artint.2021.103535.

[90] Y. Shang *et al.*, "Stochastic dispatch of energy storage in microgrids : An augmented reinforcement learning approach ☆," *Appl. Energy*, vol. 261, no. December 2019, p. 114423, 2020, doi: 10.1016/j.apenergy.2019.114423.

[91] P. C.J.C.H. Watkins and Dayan, "Technical Note: Q -Learning," *Mach. Learn.*, vol. 292, pp. 279–292, 1992.

[92] H. Song, Y. Liu, J. Zhao, J. Liu, and G. Wu, "Prioritized Replay Dueling DDQN Based Grid-Edge Control of Community Energy Storage System," *IEEE Trans. Smart Grid*, vol. 3053, no. c, 2021, doi: 10.1109/TSG.2021.3099133.

[93] V. Mnih *et al.*, "Human-Level Control through Deep Reinforcement Learning," Macmillian Publisher Limited, 2015. doi: 10.1038/nature14236.

[94] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," vol. 1144, no. December, pp. 1140–1144, 2018.

[95] M. Shin, D. Choi, and J. Kim, "Cooperative Management for PV / ESS-Enabled Electric Vehicle Charging Stations : A Multiagent Deep Reinforcement Learning Approach," *IEEE Trans. Ind. INFORMATICS*, vol. 16, no. 5, pp. 3493–3503, 2020.

[96] B. Kim, Y. Zhang, S. Member, M. Van Der Schaar, J. Lee, and S. Member, "Dynamic Pricing and Energy Consumption Scheduling With Reinforcement Learning," *IEEE Trans. Smart Grid*, vol. 7, no. 5, pp. 2187–2198, 2016, doi: 10.1109/TSG.2015.2495145.

[97] H. Wang and J. Huang, "Incentivizing Energy Trading for Interconnected Microgrids," *IEEE Trans. Smart Grid*, vol. 9, no. 4, pp. 2647–2657, 2018, doi: 10.1109/TSG.2016.2614988.

[98] Y. Yang, S. Member, Q. Jia, S. Member, G. Deconinck, and S. Member, "Distributed Coordination of EV Charging With Renewable Energy in a Microgrid of Buildings," *IEEE Trans. Smart Grid*, vol. 9, no. 6, pp. 6253–6264, 2018, doi: 10.1109/TSG.2017.2707103.

[99] S. Bahrami, S. Member, V. W. S. Wong, and J. Huang, "An Online Learning Algorithm for Demand Response in Smart Grid," *IEEE Trans. Smart Grid*, vol. 9, no. 5, pp. 4712–4725, 2018, doi: 10.1109/TSG.2017.2667599.

[100] X. Lu, X. Xiao, L. Xiao, C. Dai, M. Peng, and H. V. Poor, "Reinforcement Learning-Based Microgrid Energy Trading With a Reduced Power Plant Schedule," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10728–10737, 2019, doi: 10.1109/JIOT.2019.2941498.

[101] R. Nian, J. Liu, and B. Huang, "A review On reinforcement learning : Introduction and applications in industrial process control Artificial intelligence Constrained Markov decision process Dynamic programming," *Comput. Chem. Eng.*, vol. 139, p. 106886, 2020, doi: 10.1016/j.compchemeng.2020.106886.

[102] R. S. Sutton, "On the significance of Markov decision processes," in *Artificial Neural Networks --- ICANN'97*, 1997, pp. 273–282.

[103] R. Bellman and R. Bellman, "A Markovian Decision Process," *J. Math.*, vol. 6, no. 5, pp. 679–684, 1957.

[104] R. Bellman, “The Theory of Dynamic Programming,” in *American Mathematical Society*, 1954, pp. 1–23.

[105] M. Kearns, “Near-Optimal Reinforcement Learning in Polynomial Time,” in *Machine Learning*, vol. 49, Netherlands: 2002 Kluwer Academic Publishers, 2002, pp. 209–232.

[106] T. Remani, E. A. Jasmin, and T. P. I. Ahamed, “Generation in the Smart Grid : A Reinforcement Learning Approach,” *IEEE Syst. J.*, vol. PP, pp. 1–12, 2018, doi: 10.1109/JSYST.2018.2855689.

[107] M. Tokic, “Adaptive Greedy Exploration in Reinforcement Learning Based on Value Differences,” in *KI 2010: Advances in Artificial IntelligenceAdvances in Artificial Intelligence*, 2010, pp. 203–210.

[108] L. Bottou, “Large-Scale Machine Learning with Stochastic Gradient Descent,” in *Proceedings of COMPSTAT’2010*, 2010, pp. 177–186.

[109] “OpenAI Baselines: DQN.” https://openai.com/blog/openai-baselines-dqn/ (accessed Sep. 08, 2021).

[110] A. Nair *et al.*, “Massively Parallel Methods for Deep Reinforcement Learning,” in *International Conference on Machine Learning*, 2015, pp. 1–13.

[111] D. Silver *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play.” [Online]. Available: http://science.sciencemag.org/.

[112] J. Cao, D. Harrold, Z. Fan, T. Morstyn, D. Healey, and K. Li, “Deep Reinforcement Learning-Based Energy Storage Arbitrage with Accurate Lithium-Ion Battery Degradation Model,” *IEEE Trans. Smart Grid*, vol. 11, no. 5, pp. 4513–4521, 2020, doi: 10.1109/TSG.2020.2986333.

[113] “Tariff history.” https://www.eskom.co.za/CustomerCare/TariffsAndCharges/Pages/Tariff_History.aspx (accessed Mar. 01, 2021).

[114] M. O. Badawy and Y. Sozer, “Power Flow Management of a Grid Tied PV-Battery Powered Fast Electric Vehicle Charging Station,” *2015 IEEE Energy Convers. Congr. Expo. ECCE 2015*, no. March, pp. 4959–4966, 2015, doi: 10.1109/ECCE.2015.7310359.

[115] U. Abronzini *et al.*, “Optimal Energy Control for Smart Charging Infrastructures with ESS and REG,” *2016 Int. Conf. Electr. Syst. Aircraft, Railw. Sh. Propuls. Road Veh. Int. Transp. Electrif. Conf. ESARS-ITEC 2016*, pp. 1–6, 2016, doi: 10.1109/ESARS-ITEC.2016.7841427.

[116] C. Zhou, K. Qian, M. Allan, and W. Zhou, “Modeling of the Cost of EV Battery Wear Due to V2G Application in Power Systems,” *IEEE Trans. Energy Convers.*, vol. 26, no. 4, pp. 1041–1050, 2011, doi: 10.1109/TEC.2011.2159977.

[117] Y. Zhang, T. Zhang, R. Wang, Y. Liu, and B. Guo, “Optimal Operation of a Smart Residential Microgrid Based on Model Predictive Control by Considering Uncertainties and Storage Impacts,” *Sol. Energy*, vol. 122, pp. 1052–1065, 2015, doi: 10.1016/j.solener.2015.10.027.

[118] X. Huang, S. H. O. Hong, and S. Member, “Demand Response Management for Industrial Facilities : A Deep Reinforcement Learning Approach,” *IEEE Access*, vol. 7, pp. 82194–82205, 2019, doi: 10.1109/ACCESS.2019.2924030.

[119] D. Dewey, “Reinforcement Learning and the Reward Engineering Principle,” in *2014 AAAI Spring Symposium Series, California, USA*, 2014, vol. pp.1-8, pp. 13–16, [Online]. Available: https://www.aaai.org/ocs/index.php/SSS/SSS14/paper/viewPaper/7704.

[120] H. O. Alwan, H. Sadeghian, and S. Abdelwahed, “Energy Management Optimization and Voltage Evaluation for Residential and Commercial Areas,” *Energies*, vol. 12, p. 1811, 2019, doi: 10.3390/en12091811.

[121] “Global Solar Atlas.” https://globalsolaratlas.info/detail?c=-33.928992,18.417396,11&r=ESP&s=-33.928992,18.417396&m=site&pv=medium,0,29,300 (accessed Feb. 08, 2021).

[122] J. Gao and R. Jamidar, “Machine Learning Applications for Data Center Optimization,” in *Google White Paper*, 2014, pp. 1–13.

[123] E. O. Arwa, S. Member, K. A. Folly, and S. Member, “Improved Q-learning for Energy Management in a Grid-tied PV Microgrid,” *SAIEE Africa Res. J.*, vol. 112, no. 2, no. June 2021, pp. 77–88, 2020, doi: doi: 10.23919/SAIEE.2021.9432896.

[124] P. Lissa, C. Deane, M. Schukat, F. Seri, M. Keane, and E. Barrett, “Energy and AI Deep Reinforcement Learning for Home Energy Management System Control,” *Energy AI*, vol. 3, p. 100043, 2021, doi: 10.1016/j.egyai.2020.100043.

[125] M. Ahrarinouri, M. Rastegar, and A. R. Seifi, “Multiagent Reinforcement Learning for Energy Management in Residential Buildings,” *IEEE Trans. Ind. Informatics*, vol. 17, no. 1, pp. 659–666, 2021, doi: 10.1109/TII.2020.2977104.

[126] Y. S. Nasir, S. Member, D. Guo, and S. Member, “Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, doi: doi: 10.1109/JSAC.2019.2933973.

[127] V. François-Lavet, “Contributions to Deep Reinforcement Learning and its Applications in Smartgrids,” University of Liège, 2017.

[128] T. Zhang and S. Mao, “Smart Power Control for Quality-driven Multi-user Video Transmissions : A Deep Reinforcement Learning Approach,” *IEEE Access*, vol. PP, p. 1, 2020, doi: 10.1109/ACCESS.2019.2961914.

[129] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” in *ICLR*, 2015, pp. 1–15, [Online]. Available: https://arxiv.org/abs/1412.6980.

8 Appendix A: Simulation Data

Table 8.1: Input data used chapter 4

Time	**Commercial load profile**	**Summer PV profile**	**Winter PV profile**	**Summer Grid tariff profile**	**Winter grid tariff profile**
0	45	0.0	0.0	57.49	66.27
1	65.6	0.0	0.0	57.49	66.27
2	61.5	0.0	0.0	57.49	66.27
3	77	0.0	0.0	57.49	66.27
4	77	0.0	0.0	57.49	66.27
5	76	0.0	0.0	57.49	66.27
6	103	0.8	0.0	90.19	399.17
7	72.5	21.6	18	130.69	399.17
8	119.5	71.3	67	130.69	399.17
9	147	122.2	99	130.69	121.46
10	135	163.6	117	90.19	121.46
11	188.5	192.3	126	90.19	121.46
12	207.5	208.5	126	90.19	121.46
13	181	211.5	114	90.19	121.46
14	180	199.3	91	90.19	121.46
15	158.5	169.9	57	90.19	121.46
16	137.5	128.0	1.0	90.19	121.46
17	112	76.2	0.0	90.19	399.17
18	114.5	21.5	0.0	130.69	399.17
19	94	0.3	0.0	130.69	121.46
20	93	0.0	0.0	57.49	121.46
21	86.5	0.0	0.0	57.49	121.46
22	86.5	0.0	0.0	57.49	66.27
23	45.5	0.0	0.0	57.49	66.27

Table 8.2: Input data used in chapter 5

Time	**Slow fluctuating load curve**	**PV**	**Medium**	**PV**	**fast**	**PV**	**Summer Grid tariffs**
0	34.82	0.0	16.87	0. 0	5	0.0	57.49
1	44.49	0.0	24.6	0.0	5	0.0	57.49
2	40.096	0.0	23.06	0.0	0	0.0	57.49
3	42.712	0.0	28.87	0.0	10	0.0	57.49
4	43.708	0.0	28.87	0. 0	20	0.0	57.49
5	44.155	0.0	28.5	0.0	40	0.0	57.49

6	44.991	6.0	38.62	0.3	58	8.0	90.19
7	45.855	21.0	27.18	8.1	80	28.0	130.69
8	40.745	35.0	44.81	26.73	76	49.0	130.69
9	39.525	47.0	55.12	45.82	43	65.0	130.69
10	36.957	55.0	50.62	61.35	22	76.0	90.19
11	38.263	60.0	70.68	72.11	60	82.0	90.19
12	42.516	61.0	77.81	78.18	41	84.0	90.19
13	46.485	57.0	67.87	79.31	22	79.0	90.19
14	47.064	49.0	67.50	74.73	10	67.0	90.19
15	44.737	37.0	59.43	63.71	21	51.0	90.19
16	47.818	22.0	51.56	48. 0	46	30.0	90.19
17	34.109	6.0	42.0	28.5	56	8.0	90.19
18	33.762	0.0	42.93	8.0	65	0.0	130.69
19	32.471	0.0	35.25	0.11	80	0.0	130.69
20	38.601	0.0	34.87	0.0	50	0.0	57.49
21	39.518	0.0	32.43	0.0	18	0.0	57.49
22	38.401	0.0	32.43	0.0	15	0.0	57.49
23	37.298	0.0	17.06	0.0	10	0.0	57.49

9 Appendix B: Simulation Tools

Table 9.1: Python tools for machine learning.

Python Tool	**Function**
Numpy	Arrays manipulation
Pandas	Reading csv/excel files and plotting moving averages
Pytorch	Implementation of neural networks and manipulation of numpy arrays to tensors
Matplotlib	Plotting graphs and visualizing learning behaviours
Random	Generating random numbers
Seaborn	Plotting graphs and visualizing learning behaviours

10 Appendix C: Python Code

Learning Environment

```
class MicrogridEnv ():

  def __init__ (self, agent_action_size):
    self. _battery_power=np.array([[50]])[0]#Charging power
    self. _battery_capacity=np.array([[150]])[0]# battery capacity
    self. _battery_cost=np.array([[2025]])[0]# cost of battery
    self. _action_set=np.linspace(-1,1,num=n_actions, endpoint=True)
    self. _init_SoC=0.25   #initial state of charge

  def read_input_data(self, k):
    d = pd.read_excel('simulation_data')
    return(list(d.loc[k]))



  def step(self,cur_state,_action_ag):
    cur_solar_pv=cur_state[0]
    cur_demand=cur_state[1]
    cur_tariff=cur_state[2]
    cur_SoC=cur_state[3]
    cur_SoC_max=cur_state[4]
    cur_SoC_min=cur_state[5]

    net_load=np.maximum((cur_demad-cur_solar_pv),0)
    residue_load=np.maximum((net_load-(cur_SoC-cur_SoC_min)*s self._battery_capacity _rated),0)
    res_SoC=np.minimum(0.95,((np.maximum((cur_solar_pv-cur_demad),0)+np.maximum((cur_SoC-cur_SoC_min)*    self._battery_capacity    -
net_load,0.05))/ self._battery_capacity))

_delta_SoC=self.action_set[action]* self._battery_power / self._battery_capacity

    if self.action_set[action]<=0: #discharge
      new_SoC=np.maximum(0.05,(res_SoC+_delta_SoC))
      if (res_SoC+delta_SoC)<=0.05:
        penalty=cur_tariff* self._battery_capacity *(np.abs(_delta_SoC)-(res_SoC-cur_SOC_min))*2
      else:
        penalty=0
    else: #charge
      new_SoC=np.minimum(0.95,(res_SoC+_delta_SoC))
      if (res_SoC+delta_SoC)>0.95:
        penalty=((res_SoC+_delta_SoC)-cur_SoC_max)*cur_tariff* self._battery_capacity *2
      else:
        penalty=0
    if net_load<=((cur_SoC-cur_SoC_min)* self._battery_capacity):# load uncertainity
      penalty_2=cur_tariff*net_load*2
    else:
      penalty_2=(cur_SoC-cur_SoC_min)*cur_tariff* self._battery_capacity
    dod_1=1-res_SoC
    dod_2=1-new_SoC
    _dod_1=(694)*dod_1**(-0.795)
    _dod_2=(694)*dod_2**(-0.795)
```

```
        bd_cost= self._battery_cost *(np.abs(1/_dod_2-1/_dod_1))

        grid_power=(residue_load+(new_SoC-res_SoC)* self._battery_capacity)
        if grid_power<0:
            trading_cost=0.75*cur_tariff*(residue_load+(new_SoC-res_SoC)* self._battery_capacity)
        else:
            trading_cost=cur_tariff*(residue_load+(new_SoC-res_SoC)* self._battery_capacity)
        battery_degradation_cost=bd_cost*np.abs((new_SoC-res_SoC)* self._battery_capacity)

        oper_cost= battery_degradation _cost+trading_cost
        reward=(-trading_cost-battery_degradation _cost-penalty+penalty_2)*0.001

        return   new_SoC,reward,grid_power,residue_load,trading_cost,oper_cost, battery_degradation_cost
```

Q Learning Code

```
states=([x[k,:] for k in range(24)])
actions=([i for i in range(5)])
q=([[0.0 for i in actions]for state in states])
print(q)
total_episodes=15000
_alpha_=0.0001
discounting_rate=0.99

max_explore_rate=1
min_explore_rate=0.01
_decay_=0.001

rewards_list= []
operation_cost_lost= []
action_list = []

for ep in range(total_episodes):
    current_episode_rewards=0
    operational_cost=0
    timestep=0
    SoC = np.array([microgrid_env.init_SoC])
    SoC_max=np.array([0.95])
    SoC_min=np.array([0.05])
    Explore_rate=0
    done = False
    k=0

    while not done:
        cur_state = np.concatenate((input_data[timestep, :], SoC,SoC_max,SoC_min), axis = -1)

        _threshold=np.random.uniform(0,1)
        if _threshold>_epsilon_:
            _action_ag=np.argmax(q[k])
        else:
            _action_ag=np.random.choice(action_size)
   new_SoC,reward,grid_power,residue_load,trading_cost,oper_cost, battery_degradation_cost = microgrid_env.step(cur_state, _action_ag)
```

```
        if timestep<23:
            timestep+=1

            new_state = np.concatenate((input_data[timestep, :], new_SoC,SoC_max,SoC_min), axis = -1)
        else:
            break
        state=k
        new_state=k+1
        q[cur_state][_action_ag]=q[cur_state][_action_ag]*(1-_alpha_)+_alpha_*(reward+discount_rate*np.max(q[new_state][action]))
        if k<=23:
            k+=1

        if k==23:
            state=k
            q[cur_state][_action_ag]=q[cur_state][_action_ag]*(1-_alpha_)+_alpha_*(reward-q[state][action])
        SoC=new_SoC
        current_episode_rewards+=reward
        operational_cost+=oper_cost
        done=False

    _epsilon_=min_explore_rate+(max_explore_rate-min_explore_rate)*np.exp(-_decay_*ep)
    Reward_list=.append(current_episode_rewards)
    operation_all_episodes.append(operational_cost)
```

DQN Code

```
class ExperienceReplayMem():

    def __init__(self, max_capacity, input_size, num_actions):
        self.memory_s=max_capacity
        self.memory_counter=0
        self.state_mem=np.zeros((self.memory_s, input_shape))
        self.new_state_mem=np.zeros((self.memory_s, input_shape))
        self.action_mem=np.zeros(self.memory_s)
        self.reward_mem=np.zeros(self.memory_s)
        self.terminal_mem=np.zeros(self.memory_s)

    def store_agent_experiences(self, state, action, reward, next_state, done):
        idx=self.memory_counter %self.memory_s
        self.state_memory[idx]=state
        self.action_memory[idx]=action
        self.reward_memory[idx]=reward
        self.next_state_memory[idx]=next_state
        self.terminal_state_memory[idx]=done
        self.memory_counter+=1

    def sample_agent_experiences(self, min_batch):
        maximum_mem=min(self.memory_counter, self.memory_size)
        batch_s=np.random.choice(maximum_mem, min_batch, replace=False)
        state=self.state_mem[batch_s]
        action=self.action_mem[batch_s]
        reward=self.reward_mem[batch_s]
        next_state=self.next_state_mem[batch_s]
        done=self.terminal_state_mem[batch_s]

        return state, action, reward, next_state,done



class DeepQNetwork(nn.Module):
    def __init__(self,_alpha, num_actions, input_dimention):
        super (DeepQNetwork, self).__init__()
        self.fc1=nn.Linear(input_dimention, out_features=16)
        self.fc2=nn.Linear(in_features=16, out_features=16)
        self.fc3=nn.Linear(in_features=16, out_features=16)
        self.out=nn.Linear(in_features=16, out_features=n_actions)

        self.optimizer=optim.Adam(self.parameters(), _alpha=_alpha)
        self.loss=nn.MSELoss()
    def forward(self,cur_state):
        t=F.relu(self.fc1(state))
        t=F.relu(self.fc2(t))
        t=F.relu(self.fc3(t))
        actions=self.out(t)
        return actions

class DQNAgent():
    def __init__(self, input_dimentions,mem_capacity, batch_size, num_action, _alpha, discount_rate=0.99, epsilon=1.0, _decay_rate=1e-5, epsilon_min=0.1,
replace=1000):
        self.input_dimention=input_dimention
        self.num_action=num_action
        self._alpha=_alpha
        self.discount_rate=discount_rate
        self.epsilon=epsilon
        self._decay_rate=_decay_rate
        self.batch_s=batch_s
        self.epsilon_min=epsilon_min
        self.action_size=[i for i in range(self.num_action)]
        self.replace_target_cnt=replace
        self.learn_step_counter=0
        self.replay_mem=ExperienceReplayMem(mem_capacity, input_dimention, num_action)

        self.Q_value_eval=DeepQNetwork(self._alpha, self.num_action,
                            input_dimention=self.input_dimention)
        self.next_Q_value_eval=DeepQNetwork(self._alpha, self.num_action,
                            input_dimention=self.input_dimention)

    def select_action(self, state):
        if np.random.random()>self.epsilon:

            network_action=self.Q_value_eval.forward(state)
            action=torch.argmax(network_action.item()
        else:
            action=np.random.choice(self.action_size)
        return action
    def update_target_network(self):
        if episode % self.replace_target_cnt==0:
        self.next_Q_value_eval. load_state_dict(self.q_value_eval.state_dict())

    def decrement_exploration_rate(self):
        self.epsilon=self.epsilon-self._decay
        if self.epsilon> self.epsilon_min
        else: self.epsilon_min
    def learn(self):
        if self.replay_.mem_counter<self.batch_s:
            return
                else:
        self.Q_value_eval.optimizer.zero_grad()

        self.update_target_network()
        self. sample_gaent_experiences()=state,action, reward, next_state, done
        index=np.arange(self.batch_s)
        Q_prediction=self.Q_value_eval.forward(states)[index, actions]
        next_Q_value=self.next_Q_value_eval.forward(next_states.max(dim=1)[0]

        next_q_value[dones]=0
        Q_target_value=reward+self.discount_rate*next_Q_value
        network_error=self.Q_value_eval.loss(Q_target_value,Q_prediction)
        self.Q_value_eval.zero_grad()
        network_error.backward()#backprogation
        self.Q_value_eval.optimizer.step()
        self.decrement_exploration_rate()
```