Deep Learning for Power System Restoration

Alexander Danielsson Moses

Thesis of 60 ECTS credits
Master of Science (M.Sc.) in Sustainable Energy Engineering

May 2018
Deep Learning for Power System Restoration

by

Alexander Danielsson Moses

Thesis of 60 ECTS credits submitted to the School of Science and Engineering at Reykjavík University in partial fulfillment of the requirements for the degree of Master of Science (M.Sc.) in Sustainable Energy Engineering

May 2018

Supervisor:

Ragnar Kristjánsson, Supervisor
Assistant Professor, Reykjavík University, Iceland

Samuel Perkin, Co-supervisor
Specialist, Landsnet, Iceland

Examiner:

Ragnar Guðmannsson, Examiner
Head of dispatch center, Landsnet, Iceland
Abstract

Power systems experience outages due to uncontrollable circumstances despite efforts to reduce the frequency of such events. In an attempt to minimize the negative impact of outages, a deep feed forward neural network (FFNN) was trained to perform optimal actions during power system restoration. First, as a prerequisite, a Restoration Model (RM) was developed to simulate system restoration. The RM was designed to handle any topological degradation of a system and enable interactive exploration of the actions required to bring the system back to an ideal state. Moreover, a cost function was developed in order to evaluate the quality of a given sequence of actions in the context of restoration cost. Using the RM as a simulation environment and the cost function as an evaluation measure, a dataset of optimal power system state-to-action pairs was created using a genetic algorithm (GA) by optimizing restoration action sequences on the Icelandic transmission system. The FFNN was trained via supervised learning using the created data, achieving a 75% test accuracy on optimal decisions on the GA and operators of the Icelandic system. The FFNN agent was further tested in a comparison to the human operators on a simple test restoration scenario. Results show that the FFNN is 3 orders of magnitude faster than the GA at developing a restoration plan, and performs comparably to the human operators on a simple test restoration scenario. This thesis demonstrates the feasibility and potential of using deep learning for power system restoration and control.
Gervitauganet notað í uppbyggingu raforkukerfa eftir straumleysi

Alexander Danielsson Moses

mai 2018

Útdráttur

Raforkukerfi verða óhjákvæmilega straumlaus stöku sinnum þó að allt sé gert til að að
minnka tíðni þess. Í verkefni er gerð tilraun til að lágmarka þann tíma sem rafokukerfi
er straumlaust með þjálfun gervitauganets til að finna bestu röð aðgerða við uppbyggingu
kerfisins. Forsenda þess var að þróa gött líkan til að herma uppbyggingu kerfisins. Líkandið
var Hannað til þess að ráða við allar breytingar í rekstri og tengingar kerfisins og gerir það að
verkum að það sé hægt að kanna og herma stöðu og rekstur kerfisins á gagnvirkan hátt til að
koma því aftur í eðlilegt ástand. Sérstakt kostadarfall var notað til að meta geði aðgerðarr-
aðarinnar í samræmi við gæði kerfisrekstrarins og stöðu. Með því að nota líkan af kerfinu og
kostadarfall sem mælikvöða var gagnasafn af hagstæðum aðgerðum búið til með notkun
erðafræðilegaraðir reiknirits sem bestaði aðgerðaraðir í Íslenska raforkukerfinu. Stýrð þjálfun-
araðferð (Supervised learning) var notað með gagnasafninu til að þjálfa gervitauganetíð og
tök það rétrri ákvörðun í 75% tilvika í uppbyggingu á Íslenska raforkukerfinu. Tauganetið
var síðan borið saman við erðafræðilegaraðir reikniritið og „raunveruleg“ viðbrögð kerfisrekstr-
araðila Íslenska kerfisins. Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
ins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Niðurstaðan var sú að tauganetið var um þrem stærðargráðum
fljótari en erðafræðilegaraðir reikniritið í að finna bestu áætlun við uppbyggingu raforkukerfis-
sins og gæði hennar var samhærlígvæg við áætlarir kerfisrekstraraðila Íslenska kerfisins.
Deep Learning for Power System Restoration

Alexander Danielsson Moses

Thesis of 60 ECTS credits submitted to the School of Science and Engineering at Reykjavík University in partial fulfillment of the requirements for the degree of Master of Science (M.Sc.) in Sustainable Energy Engineering

May 2018

Student:

Alexander Danielsson Moses

Supervisor:

Ragnar Kristjánsson

Samuel Perkin

Examiner:

Ragnar Guðmannsson
The undersigned hereby grants permission to the Reykjavík University Library to reproduce single copies of this Thesis entitled Deep Learning for Power System Restoration and to lend or sell such copies for private, scholarly or scientific research purposes only. The author reserves all other publication and other rights in association with the copyright in the Thesis, and except as herein before provided, neither the Thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatsoever without the author’s prior written permission.

Date

alexander danielsson moses
master of science
I dedicate this to Skotta.
Acknowledgements

I would like to thank my advisors Ragnar Kristjánsson and Samuel Perkin for their guidance and support throughout this project. I would also like to thank Samuel Perkin again for setting up the restoration case study and the system operators at Landsnet for their participation in the study.
Contents

Acknowledgements xi

Contents xi

List of Figures xiv

List of Tables xvi

List of Abbreviations xvii

1 Introduction 1
   1.1 Summary ........................................ 1

2 Literature Review 3
   2.1 Power systems ..................................... 3
      2.1.1 Social impact of outages .................. 3
      2.1.2 Power flow .................................... 6
         2.1.2.1 Newton’s method ........................ 7
      2.1.3 Optimal power flow .......................... 9
      2.1.4 Dynamic system phenomena ................. 10
         2.1.4.1 Frequency response .................... 10
         2.1.4.2 Voltage control ........................ 11
         2.1.4.3 Cold load pickup ....................... 12
   2.2 Power system restoration ....................... 13
      2.2.1 Standing phase angle difference constraint 14
         2.2.1.1 SPAD reduction controls .............. 14
         2.2.1.2 SPAD reduction optimization .......... 15
      2.2.2 Traditional restoration techniques ........ 17
         2.2.2.1 Traditional planning .................. 17
         2.2.2.2 Expert system .......................... 19
      2.2.3 Modern restoration techniques ............. 20
         2.2.3.1 Centralized methods .................... 20
         2.2.3.2 Multi-agent methods .................... 23
   2.3 Machine learning models ....................... 24
      2.3.1 Dimensionality reduction .................... 24
         2.3.1.1 Principal component analysis .......... 24
         2.3.1.2 Fisher’s criterion ..................... 25
      2.3.2 Feed forward neural networks .............. 26
         2.3.2.1 Error back-propagation ............... 27
      2.3.3 Genetic algorithm ........................... 28
3 Methods

3.1 Restoration model
   3.1.1 Primary versus secondary actions
   3.1.2 Schedule preserving OPF
     3.1.2.1 OPF constraints
   3.1.3 Model initialization
   3.1.4 State variables
     3.1.4.1 Current and ideal system state
     3.1.4.2 Islands
     3.1.4.3 Action list
   3.1.5 Functions
     3.1.5.1 Branch reconnection
     3.1.5.2 Generator and load reconnection
     3.1.5.3 Reverse and reset
     3.1.5.4 Infeasible actions
   3.1.6 Cost evaluation
   3.1.7 Modeling simplifications

3.2 Optimization methods
   3.2.1 Optimization framework
   3.2.2 Tree representation of restoration
     3.2.2.1 Search methods
   3.2.3 Genetic algorithm
     3.2.3.1 Average position crossover operator
     3.2.3.2 Feasibility preserving evaluation
     3.2.3.3 Parallelization

3.3 IEEE 30 bus system

3.4 Icelandic transmission system
   3.4.1 Modeling simplifications
   3.4.2 Subset system
   3.4.3 Restoration dataset creation

3.5 Deep feed forward neural network
   3.5.1 Data preprocessing
   3.5.2 Configuration

4 Results

4.1 Optimization comparison and validation
   4.1.1 IEEE 30-bus degraded state
   4.1.2 Comparison

4.2 Validation of genetic algorithm

4.3 Deep learning results
   4.3.1 Dimensionality reduction of system state
   4.3.2 FFNN results

4.4 Case study: Operators vs AI

4.5 Computational expense

5 Discussion
5.1 Restoration model improvements ........................................ 67
5.2 Cost function improvements ........................................... 68
5.3 Learning agent improvements ......................................... 69
5.4 Real world application of learning agent .............................. 70
5.5 Conclusion .................................................................. 71

Bibliography ........................................................................ 73

A System parameters .......................................................... 77

B Modeling in Matpower ....................................................... 80
  B.1 Bus ........................................................................ 80
  B.2 Branch ..................................................................... 81
  B.3 Generator .................................................................. 81
  B.4 Dispatchable vs fixed loads ......................................... 82
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Annual cost of outages divided by consumer class, adapted from [3]</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Annual cost of outages divided by interruption type, adapted from [3]</td>
<td>5</td>
</tr>
<tr>
<td>2.3</td>
<td>Simplified block diagram of generator frequency response.</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Transformer overexcitation limit, adapted from [9]</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>Switch transients, adapted from [9]</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Pattern of cold load pickup, adapted from [12]</td>
<td>13</td>
</tr>
<tr>
<td>2.7</td>
<td>Left: Instantaneous voltage at buses $i$ and $j$. Right: Phasor representation of voltages showing the SPAD, $\Theta_{ji}$</td>
<td>14</td>
</tr>
<tr>
<td>2.8</td>
<td>Generalized FFNN, adapted from [38]</td>
<td>27</td>
</tr>
<tr>
<td>2.9</td>
<td>Single point crossover operator, adapted from [41]</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>High level restoration methodology.</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>Generator cost curve for OPF.</td>
<td>34</td>
</tr>
<tr>
<td>3.3</td>
<td>Dispatchable load cost curve for OPF.</td>
<td>35</td>
</tr>
<tr>
<td>3.4</td>
<td>Initialization of the restoration model object.</td>
<td>36</td>
</tr>
<tr>
<td>3.5</td>
<td>Branch reconnection scenarios (a) blackout-energized, (b) energized-energized, (c) island-island</td>
<td>40</td>
</tr>
<tr>
<td>3.6</td>
<td>Logic of the branch connection function.</td>
<td>40</td>
</tr>
<tr>
<td>3.7</td>
<td>Logic of the connect generator, connect fixed load, and connect dispatchable load functions.</td>
<td>41</td>
</tr>
<tr>
<td>3.8</td>
<td>Initialization and logic of the optimization framework.</td>
<td>45</td>
</tr>
<tr>
<td>3.9</td>
<td>Tree structure of restoration.</td>
<td>46</td>
</tr>
<tr>
<td>3.10</td>
<td>Average position crossover operator.</td>
<td>47</td>
</tr>
<tr>
<td>3.11</td>
<td>The IEEE 30 bus transmission network.</td>
<td>49</td>
</tr>
<tr>
<td>3.12</td>
<td>The Icelandic transmission system, from [46]</td>
<td>50</td>
</tr>
<tr>
<td>3.13</td>
<td>Subset of Icelandic system used in element disconnection and restoration optimization, adapted from [46]</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Degraded state used for optimization method benchmarking.</td>
<td>55</td>
</tr>
<tr>
<td>4.2</td>
<td>Comparison of GA and TS methods on optimizing action sequence.</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>Cost and power deviation results from best performing action sequence discovered by GA $\eta = 0.75$</td>
<td>57</td>
</tr>
<tr>
<td>4.4</td>
<td>Data visualization using PCA and Fishers criterion</td>
<td>59</td>
</tr>
<tr>
<td>4.5</td>
<td>Training and testing of FFNN (100,100)</td>
<td>60</td>
</tr>
<tr>
<td>4.6</td>
<td>Case study scenario on the Icelandic system.</td>
<td>61</td>
</tr>
<tr>
<td>4.7</td>
<td>Genetic algorithm performance on the Icelandic case study.</td>
<td>62</td>
</tr>
<tr>
<td>4.8</td>
<td>Restoration performances on test scenario.</td>
<td>64</td>
</tr>
<tr>
<td>4.9</td>
<td>Restoration performance by the FFNN.</td>
<td>64</td>
</tr>
<tr>
<td>4.10</td>
<td>Histogram of action execution time.</td>
<td>66</td>
</tr>
</tbody>
</table>
5.1 Possible infrastructure for a real world application of deep learning agents. . . . 71
B.1 Branch model adapted from [7]. . . . . . . . . . . . . . . . . . . . . . . . . . 81
List of Tables

3.1 Restoration model metadata .................................................. 37
3.2 Ideal state and current state variables ..................................... 38
3.3 Action list ........................................................................... 38
3.4 Objective function parameters ............................................... 43
4.1 Best performing action sequence discovered by GA $\eta = 0.75$ ........... 58
4.2 FFNN results ....................................................................... 60
4.3 Action list ........................................................................... 62
4.4 System operators vs AI ............................................................ 63
4.5 Computational expenses ........................................................ 66
A.1 Base case generator parameters on the IEEE 30 bus system ............ 77
A.2 Base case load parameters on the IEEE 30 bus system ................. 78
A.3 Base case branch parameters on the IEEE 30 bus system ............... 79
List of Abbreviations

AI  Artificial intelligence
TSO  Transmission system operator
SPAD  Standing phase angle difference
GA  Genetic algorithm
OPF  Optimal power flow
ES  Expert system
GRM  Generic restoration milestone
SVM  Support vector machine
FFNN  Feed forward neural network
PCA  Principal component analysis
TS  Tree search
RM  Restoration model
MIPS  Matpower interior point solver
RL  Reinforcement learning
MCTS  Monte Carlo tree search
Chapter 1

Introduction

Power systems experience outages even though careful consideration is taken to reduce the frequency of such events. Outages necessitate system restoration in order to bring systems back to their ideal operating states. Several motivators to improving power system restoration are asserted below.

Increasing the performance of power system restoration works out as an economic benefit to society. Primarily, the benefit comes as a reduction of power outage time which directly impacts economic productivity as nearly all industries rely on electricity. Secondly, power system control automation could provide cost savings in the operation of power systems.

Power systems are also faced with the challenge of integrating intermittent renewable power. The intermittent nature of renewable sources such as wind and solar pose a threat to system stability and reliability because their power injection rates can not be easily controlled. Therefore, power systems with proportionally high amounts of intermittent sources have a hard time balancing consumption and production. Automation and implementation of artificial intelligence (AI) may prove to help secure power systems and enable higher levels of renewable integration.

Lastly, machine learning has not been widely adopted in the power systems industry, presenting an interesting opportunity for research. Feasible improvements via machine learning are relatively low hanging fruit because the theory is being rapidly developed for other applications and the same technologies can be modified and applied to power systems data. This thesis attempts to do just that.

1.1 Summary

A power system is a network of power lines linking substations, enabling the transportation of electrical power. It is useful to think of a power system as a graph as described in graph theory where the nodes are substations and the edges are transmission lines. At each node, there may be power injected to the system by generators, power consumed from the system by loads, or both. The system requires that the power injection be equal to the consumption in the system for both active and reactive power. This equality constraint ensures that the power quality in terms of voltage and frequency is within standard limits, protecting power consuming devices from damage. In this thesis, it is assumed that the Transmission System
Operator (TSO) has control of power generating units and, although to a lesser extent, the loads. These control mechanisms are the tools used by the TSO to ensure system balance and stability.

Power system restoration, as reasoned about in this thesis, is the action sequence taken by the TSO after a catastrophic system failure in an attempt to restore it to its ideal operating state. A catastrophic failure implies that there are de-energized, isolated components in the system, typically including lost loads. The context of restoration in this thesis excludes the process under which the system degrades i.e. the disturbance and any protections designed to guide the system to a stable state. Rather, a stable degraded state is assumed and the restoration back to the initial state is modeled from there.

Optimization and machine learning are two constructs used in this thesis for information gathering and synthesis into actionable decisions. Optimization is a broad term used to describe a set of algorithms that are used to try to find maximums or minimums within a given search space. In this work, optimization is used on individual restoration scenarios to find optimal restoration sequences under a variety of conditions. Machine learning is a broad term for mathematical models that absorb and synthesize data to provide predictions given new information. A specific discipline of machine learning is applied in this thesis: Deep learning. This is a promising field of research for AI, and has proven very effective in predictions within narrow domains.

The problem formulation in this thesis is rather rigid in order to facilitate the training of a deep machine learning model. Therefore it is important that this formulation be understood as it is the keystone of the work. The formulation is outlined in chapter 3, but in short: A power system restoration sequence is defined by a list of required actions that are known at the start and are defined by the known de-energized system components, each of which must be connected in the following restoration. The optimization problem is therefore a combinatorial optimization in which the best performing permutation of the action sequence is sought after. After optimal sequences are found for a variety of different system degradations, each sequence can be split into individual power system states and corresponding optimal actions. This data is formatted into two matrices, one consisting of flattened system states, the other flattened vectors indicating optimal actions corresponding to each state. A machine learning model is then trained via the supervised learning method with the optimized state-action dataset.

This thesis is organized into the following sections: A literature review on power systems theory, restoration, and optimization and machine learning is given in chapter 2. Chapter 3 goes over the methods including, a thorough description of the restoration model, the optimization methods, and finally the creation of a restoration dataset and configuration of the deep learning method. Chapter 4 describes the results of the study. These include an optimization method comparison, accuracy of the deep leaning model, and a case study on the Icelandic transmission system. Finally, chapter 5 discusses the work and how it can be improved upon along with a discussion of a plausible real world application of an AI to power system control.
Chapter 2

Literature Review

Most research in power systems is conducted to minimize the frequency of outages and relatively little attention has been given to restoration after failures. A literature review is performed in this chapter on the latter, outlining research on power system restoration, modeling tools, and machine learning theory. The review starts broad, looking at general power system theory and the impacts of power outages on society, then narrows down to studies specific to restoration, and ends with the theory behind the machine learning methods used in the thesis.

2.1 Power systems

An understanding of power systems is prerequisite to understanding how they should be restored. This section begins by quantifying the impact of system outages on society, then talks about some dynamic issues that occur in restoration scenarios, and ends with power flow theory, which is used to estimate the steady state power flow.

2.1.1 Social impact of outages

Three large scale blackouts occurred in North America and Europe in 2003. Information on the outages is given in a summary of a panel session at the IEEE Power Engineering Society General Meeting [1]. A blackout in Northeastern United States and Canada affected 50 million people, with 63 GW of interrupted load for several hours. A blackout in Southern Sweden and Eastern Denmark occurred affecting 4 million people, with 4.7 GW of interrupted load. And a total system collapse occurred in Italy, interrupting 6.4 GW of load [1].

The effect of the North American 2003 power failure on human mortality is documented in [2]. The study looked at mortality rates in New York, NY from 1987 to 2005. Comparing the 2003 outage to an estimated baseline found that accidental mortality increased by 122% (95% confidence interval of 28% - 287%) and disease related deaths increased 25% (95% confidence interval 12% - 41%), causing approximately 90 excess deaths in New York, NY as a result of the city wide power outage. Increased mortality risk factors due to the outage included people getting trapped in subways and elevators, a lack of potable water, pharmacies closing, power operated home medical equipment failing, slower than usual ambulance
response, cell phone service failing, and four of 75 hospitals in the city losing power.

As a consequence of the 2003 outage in North America, research into the consequences of such outages was conducted. An estimate of the yearly monetary cost of outages in the United States was conducted in [3], using the cost estimation

\[ C_{\text{total}} = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{p} N_{i,j} F_{i,j,k} C_{i,j,k} V_{i,j,k}, \]  

where \( N \) is the number of customers in class \( i \) for region \( j \), \( F \) is the number of outage events of type \( k \) experience annually by customer class \( i \) in region \( j \), \( C \) is the corresponding average cost of each event in \( F \), and \( V \) is the vulnerability or ratio of customers in each class in each region that get exposed to each type of event. Using data gathered for this cost estimation, they calculated a total annual cost of 79 billion USD shown in Fig. 2.1. It should be noted that the cost values \( C \) are estimated via surveys which are inherently biased.

![Figure 2.1: Annual cost of outages divided by consumer class, adapted from [3]](image-url)

The study also breaks up the outages into "momentary" and "sustained" interruptions where momentary is defined as an outage lasting less than 5 minutes. Meta analysis of outage events concluded that an average sustained duration per region is 106 minutes occurring 1.2 times annually, and the momentary outages occur 4.3 times annually.
Some recommendations were made at the panel session regarding how to prevent and lower the impact of blackouts. Below are some key recommendations that would positively impact the restoration of a system after a failure or outage [1].

- Operators who initiate load shedding pursuant to approved guidelines should be shielded against liability or retaliation
- Improve the near-term and long-term training and certification requirements for operators, reliability coordinators, and operator support staff
- Employ automatic load shedding
- The lessons learned from past mistakes must be incorporated into new procedures as well as using such lessons learned to help develop new and improved technologies for system control and monitoring
- Rapid system restoration is extremely important in order to minimize the impact of blackout on society. Thus, means should be put into place to measure and reduce restoration times. System operators should be given regular refresher training and live drills on system restoration to ensure that they remain familiar with restoration procedures and best practices. This issue will be explored in greater detail by a task force recently established by the Power System Dynamic Performance Committee.

The last point is particularly relevant to this thesis as it directly addresses the need for more resources in advancing restoration technologies. With 79 billion USD and being lost to power interruptions annually in the United States alone, and hundreds of lives being lost extrapolating from [2], there is a huge potential for cost savings and preservation of life in power system restoration.
2.1.2 Power flow

The power flow formulation below is based on a combination of [4]–[6]. Power flow analysis computes the voltage angle and magnitude at each system bus, the real and reactive power flows through each connecting element and system losses. The input to the power flow computation generally consists of the parameters required to model each system component including the transmission lines, transformers, generators and loads. The load information is given in power consumption and the generators are modeled as power sources. The system is then modeled as a system of non-linear equations to which a stable solution is found typically through error minimization via gradient descent. In power flow modeling, balanced three-phase steady state conditions are assumed for simplicity. Circuit theory cannot be directly applied to the power flow problem because loads and generators are defined by their combined real and reactive power rather than impedance and voltage or current.

Each bus $k$ in a power flow network is characterized by its voltage magnitude $V_k$, phase angle $\delta_k$, net real power $P_k$, and net reactive power $Q_k$. The real and reactive powers in Eqs. 2.5 and 2.6 are the difference between generation and demand at a particular bus, where a positive value indicates more generation than demand. There are three types of buses:

- **PQ**: Load (PQ) buses accept $P_k$ and $Q_k$ as input and $V_k$ and $\delta_k$ are computed. PQ buses are the most common, occurring at load buses wherever no voltage control devices are connected.

- **PV**: Voltage controlled (PV) bus where $P_k$ and $V_k$ are inputs and $Q_k$ and $\delta_k$ are computed. Buses where generators or tap-changing transformers are connected are modeled this way. Typically in the case of PV buses, reactive power limits are defined and if those limits are reached, the reactive power $Q_k$ is fixed and the bus is converted to PQ.

- **Slack**: Slack bus of which only one is typically defined in a network. The swing bus serves as a voltage reference $V_k/\delta_k$ of value $1/0^\circ$ per unit, which all other buses are relative to. That leaves $P_k$ and $Q_k$ to be computed, which gives the power flow framework the degrees of freedom it needs to compensate for any mismatch in the given loads and generation.

The links between buses are branches. A branch may contain a transmission line, transformer or both. The parameters of each branch component are placed in the branch matrix.

With the goal of finding the complex power flow at each bus, the network equations are taken and a single bus $k$ is considered:

$$I_k = \sum_{n=1}^{N} Y_{kn}V_n. \quad (2.2)$$

The complex power is defined by the product of complex voltage and the conjugate of complex current at bus $k$:

$$S_k = V_k I_k^* = V_k \left( \sum_{n=1}^{N} Y_{kn}V_n \right)^*. \quad (2.3)$$
The power is broken up into real and reactive components and the voltages $V_n$ and admittances $Y_{kn}$ are put into exponential form

$$P_k + jQ_k = V_k \sum_{n=1}^{N} Y_{kn} V_n e^{j(\delta_k - \delta_n - \theta_{kn})}$$

(2.4)

where $V_k$, $Y_{kn}$, and $V_n$ are no longer complex, but represent the magnitudes of each respective phasor, $\delta_k$ and $\delta_n$ are the voltage phasor angles at bus $k$ and $n$ respectively, and $\theta_{kn}$ is the angle of the admittance $Y_{kn}$. The power equation (Eq. 2.3) is broken into its real and imaginary components for simplicity:

$$P_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \cos(\delta_k - \delta_n - \theta_{kn})$$

(2.5)

$$Q_k = V_k \sum_{n=1}^{N} Y_{kn} V_n \sin(\delta_k - \delta_n - \theta_{kn})$$

(2.6)

Solving the power flow can then be formulated as a minimization problem of power mismatches between the expected power injection at the PQ buses due to generation and loads and the calculated power injection given by 2.5 and 2.6. This results in the power balance equation:

$$S_{bus} + S_d - C_g S_g = 0$$

(2.7)

where $S_{bus}$ is the voltage dependent power injection to each bus given by 2.5 and 2.6, $C_g$ is a generator connection matrix of size $n_b \times n_g$ where each element $(i,j)$ is one if generator $j$ is located at bus $i$ and otherwise zero, and $S_d$ and $S_g$ are the known loads and generator outputs at each bus.

### 2.1.2.1 Newton’s method

Newton’s method, also known as the Newton-Raphson method, is an iterative root finding algorithm that can be used to solve the power flow problem. The formulation given here is based on [5], [6]. Although many power flow solving methods exist, Newton’s method is the most common technique [5] along with the fast decoupled power flow method. A initial point $x_i$ is chosen (usually a flat start in the context of power flow), the local derivative of function $f(x_i)$ is estimated using the finite difference method, the root of the tangent line $x_{i+1}$ is found, and finally this root is set as the new estimation of the minimum of $f(x)$. This process is iterated until a stopping criteria is met. One of several criteria can be set, including:

$$\max |f(x_i)| < \epsilon,$$

(2.8)

$$\max |x_i - x_{i+1}| < \epsilon,$$

(2.9)

where $\epsilon$ is a preset tolerance. A maximum iteration criteria is usually also included to stop the algorithm for cases where there is no solution.

In solving power flow, variables $x$ and $f(x)$ are defined as vectors:
The slack bus is assigned voltage $V_1$ and voltage angle $\delta_1$, and is left out of the system because these are assumed to be known in the problem formulation. Notice that the function $f(x)$ is the difference between the power flow solution and the known power at each load (PQ) bus, sometimes referred to as power mismatches. This formulates the problem in a way that minimizes power mismatches whilst solving for the roots of the system of equations. Newton’s method manipulates the voltage magnitudes and angles at each bus to fit the power flow to the known values within the selected tolerance. Since the method involves finding the derivative of $f(x)$, the Jacobian is computed at each iteration using finite differences. The Jacobian is denoted as $J$.

The following steps illustrate an iteration $i$ of Newton’s method:

1. Evaluate $f(x_i)$ from Eq. 2.10
2. Calculate $J$ at $x_i$
3. Solve for the change in $x_i$:
   \[ \Delta x_i = -J^{-1}f(x_i) \]  
   \[ (2.11) \]
4. Update $x$:
   \[ x_{i+1} = x_i + \Delta x_i \]  
   \[ (2.12) \]

The convergence of Newton’s method depends on the initial values of $x$. Regions of attraction for each bus exist and the power flow solution may gravitate to one of many regions depending on the initial conditions and the complexity of the system [5]. Typically in power systems the voltage of any bus $V_n$ is close to 1 p.u. and the voltage angle $\delta_n$ is close to zero, therefore these are generally dominant regions of attraction and initializing $V$ to ones and $\delta$ to zeros is a safe bet. This initialization is called a flat start.

The method described so far shows how to deal with load (PQ) buses, but fails to treat voltage controlled (PV) buses. The real power $P_k$ and voltage magnitude $V_k$ of a PV bus $k$ are assumed to be known. The power flow problem is designed to solve for $V_k$ and $\delta_k$ using power mismatches between the known active and reactive power and the solution output. Since there is no known reactive power $Q_k$ at a PV bus, they only require that the real power equation (Eq. 2.5) be solved in the system of equations. This means that $V_k$ and $Q_k(x)$ can be omitted from vectors in Eq. 2.10, and their corresponding columns and rows can be omitted from the Jacobian. At the end of each iteration, the value of $Q_k$ can be calculated using Eq. 2.6 independently of the Newton’s method optimization. PQ buses often have minimum and maximum constraints on the value of $Q$. If the constraints are violated, the bus is converted to a PQ bus with $Q$ set to its limiting value.
2.1.3 Optimal power flow

Optimal power flow (OPF) uses the bus voltage angles and magnitudes and the generator real and reactive powers as optimization parameters in a cost minimization problem. Operational cost curves are applied to each generator real and reactive output and the sum of generator operational costs are minimized alongside the power mismatches from 2.7.

\[
\min_x f(x) \quad (2.13)
\]

subject to

\[
g(x) = 0 \quad (2.14)
\]

\[
h(x) \leq 0 \quad (2.15)
\]

\[
x_{\text{min}} \leq x \leq x_{\text{max}} \quad (2.16)
\]

\[
l \leq Ax \leq u \quad (2.17)
\]

where

\[
x = \begin{bmatrix}
\theta \\
V_{\text{mag}} \\
P_g \\
Q_g 
\end{bmatrix} \quad (2.18)
\]

The objective function is the summation of the generator real and reactive power injection cost functions and the system losses:

\[
\min_{\theta, V_{\text{mag}}, P_g, Q_g} \left( \sum_{i=1}^{n_g} f^p_i(p^i_g) + f^q_i(p^i_g) \right) + L \quad (2.19)
\]

The equality constraints in Eq. 2.14 are the real and reactive power balance equations given in Eq. 2.7, the inequality constraints in Eq. 2.15 represent the branch flow limits at both the to and from ends of each branch, and the variable limits in Eq. 2.16 define the limits on voltage angle of reference buses, voltage magnitudes on every bus, and the real and reactive power injections by each generator. A set of linear restrictions on the optimization variables is defined by Eq. 2.17, where matrix \( A \) defines a system of relationships of variables \( x \) that are bound by \( l \) and \( u \). This allows for more complex constraints such as a constraint on the difference on the standing voltage phase angles between buses.

Matpower provides extensibility of OPF beyond the formulation described here, where user defined cost functions can be defined and included in the optimization. These would be functions beyond the generator output cost functions, which for example allows for the usage of soft limits on line capacities by applying a cost function to the power injection of each line rather than the hard inequality constraint in Eq. 2.15. This could allow for extreme but unfavorable system states as feasible solutions the OPF that might otherwise cause non-convergence. This would be a very useful tool in restoration modeling because of the extreme states the system often finds itself during restoration.

The algorithm used to solve the OPF was the Matpower Interior Point Solver (MIPS), outlined in [7]. MIPS uses a primal-dual interior point method which can be applied to general nonlinear optimization problems of the form of Eq. 2.13 and its constraints. The interior
point method introduces a barrier function and a vector of slack variables to convert the inequality constraints of Eq. 2.15 to equality constraints. The primal-dual method introduces the Lagrangian variable, and the primal-dual equation is solved for the Lagrangian for which the gradient of the barrier function is zero, which is called the first order optimality condition. The Newton method is used to update the state, slack, and Lagrangian parameters until convergence tolerance is met.

2.1.4 Dynamic system phenomena

A prerequisite to investigating power system restoration is a good understanding of the physical nature of power systems. The static and dynamic physical characteristics must be adequately modeled so that the system response can be predicted with reasonable accuracy. A review on the common modeling techniques and considerations for restoration are outlined in this section. The following subsections describe the types of dynamic phenomena that commonly occur during system restoration.

Dynamic phenomena are not modeled in this thesis. It is important to recognize this simplification and keep possible dynamic hindrances in mind when considering restoration strategies. Dynamic system modeling is a high priority for future work.

2.1.4.1 Frequency response

System frequency response to load pickup and generator loss are of concern during system restoration. It is important to know the maximum allowable load pickup that does not violate frequency limits, and also know the generator reserve needed to meet the loss of the largest generator. Since frequency response is within the field of signal analysis, it is often modeled using block diagrams and transfer functions. A simplified model of a governor within a power system is depicted in Fig. 2.3, where all active controls are wrapped into the proportional constant $K_{sys}$.

![Figure 2.3: Simplified block diagram of generator frequency response.](image)

The transfer function of this model is arranged into a ratio of polynomials:

$$H(s) = \frac{T \alpha s + 1}{2HT\alpha^2 + 2Hs + K_{sys}}$$  \hspace{1cm} (2.20)

A step input may be fed in via the $\Delta P$ parameter to simulate a load pickup. An exploration of the frequency response of prime movers during restoration is performed in [8] using sim-
ilar but more detailed models. Models were developed for combustion turbine units, steam units and hydro units, and the frequency responses to load pickup were analyzed. The paper determines how to find the maximum load pickup corresponding to a maximum allowable frequency dip. An analysis showing the optimal distribution of generation reserve based on the parameters of prime movers is performed. The study shows that the fastest prime movers should have the largest reserves in order to minimize the frequency dip after load pickup.

2.1.4.2 Voltage control

The major voltage concerns during restoration include sustained power frequency overvoltages, switching transients, and harmonic resonance voltages. These phenomena are investigated in [9] and summarized in this section.

Sustained overvoltages occur due to the charging currents of lightly loaded transmission lines. If the lines are lightly loaded for an extended period and overvoltages persist, they cause overheating issues in transformers, generate harmonic distortions, and might cause self-excitation of generators. The overexcitation limit of a transformer and voltage limit of an arrester are illustrated in Fig. 2.4. This specification says that the transformer can handle a voltage of 1.4 p.u. initially, but within one minute, its capacity drops to 1.2 p.u.

![Figure 2.4: Transformer overexcitation limit, adapted from [9]](image)

Methods used to limit sustained overvoltage include connecting reactive loads, operating parallel transformers on different tap levels to increase circulating currents and reactive power loss, only energizing high load lines, maintaining as low a voltage profile as possible during restoration to minimize charging currents.

High voltage transients due to switching are another concern during restoration. These may be severe overvoltages causing flashover and damaging equipment. These may occur during the initial energization of lines and loads if the closing of the switch is poorly timed. Closing a switch on the maximum of the instantaneous source voltage may catapult the voltage of the newly connected line far above 1 p.u. as seen in Fig. 2.5. The thick line illustrates the source voltage and the thin line illustrates the transient caused due to switching.
The danger of voltage switching transients are typically mitigated using arresters and are usually not a limiting factor in restoration. Arrestors allow electricity to flow to ground in instances of high voltage of short duration by usage of varistors, whose resistance decreases as voltage increases.

The third voltage phenomenon, harmonic resonance, occurs when harmonic distortions are produced by saturated transformers and the line capacitance is high with a low system impedance. The resonance may amplify to unstable voltages. To avoid this phenomenon, the operator should connect as many loads and start as many generators as possible. In general, overvoltage can be reduced by increasing the reactive power absorption in the system by enabling shunt reactors, connecting reactive loads, and adjusting generator excitation. Also adjusting transformer taps to low ratios helps reduce the line voltages. Overvoltage is an important factor in the success rate of connecting lines, so it is important to understand what it is and how to reduce it.

A dynamic control scheme was developed in [10] to optimally control sustained overvoltage during restoration. The method minimizes the number of control actions required to keep the voltage stable throughout the restoration with the aim of improving the success rate of the restoration plan and shortening the restoration time. Specifically, the method uses three objective functions: a risk metric associated with the success rate of line reconnection, the time required to implement the chosen control actions, and the voltage deviation in the network. The Strength Pareto Evolutionary Algorithm 2 (SPEA2) was used in the optimization.

### 2.1.4.3 Cold load pickup

The cold load pickup condition pertains to the abnormally high load on distribution systems after prolonged outages. This occurs due to the immediate switching on of thermostatically controlled loads such as air-conditioners, heaters and refrigerators which constitute a large proportion of household consumption. Normally these loads are somewhat randomly distributed among consumers, however following an outage, a simultaneous switching on
occurs until these system reach their temperature thresholds. Thus overloads occur, sometimes followed by a period of underload in a damped sinusoidal pattern. Overloads cause overheating of transformers, unacceptable voltage drops through feeders, and affect the frequency of the system [11].

Several methods have been used to model cold load pickup, including thermodynamic based [12], regression based [13], and delayed exponential decay based [11] models. In [11], a particle swarm approach was used to optimize the placement of distributed generation at the transmission and distribution levels to minimize the load not served and transformer loss of life due to overheating. The algorithm shows the possibility of of single step restoration of distribution networks using distributed generation to supplement the power supplied by the feeder, with the added benefit of improving the voltage profile of the network.

The pickup of cold loads is modeled using a physics based thermodynamic household model in [12]. In the study, careful consideration is given to predicting the magnitude and duration of overloads after load pickup by incorporating a thermodynamic model of house heating. In the study, a cold winter environment is assumed such that houses require heating using electric space heaters. The thermostat is modeled, which starts the heater if the temperature dips below a threshold, yielding a probabilistic model that is used to estimate thermostat behavior after outage. The likelihood of heaters starting and thus system overload increases with the length of outage. Fig. 2.6 shows the load pattern after an outage with a high initial load and damped oscillatory decay. The study shows how the overload due to household space heating becomes more severe with colder weather and with longer outages.

Figure 2.6: Pattern of cold load pickup, adapted from [12]

2.2 Power system restoration

The literature specific to power system restoration is reviewed and discussed in this section. Special attention is given to standing phase angle difference (SPAD) on lines, and traditional versus more modern approaches to the restoration problem.
2.2.1 Standing phase angle difference constraint

When reconnecting a tripped line, it is common practice to first close a circuit breaker on one end of the line to energize the line, then complete the connection at the other end. For a bus $i$ and bus $j$, if the breaker at bus $i$ has been closed, the action of closing the breaker at bus $j$ completes the line reconnection, but it is important to ensure that the standing phase angle difference (SPAD) is within a predetermined threshold to prevent large rotor shaft impacts and equipment flashover during closing [14], [15]. Fig. 2.7 shows the time domain and phasor representations of the voltage on buses $i$ and $j$. The equation

$$ \Theta_{ji} = |\Theta_j - \Theta_i| $$

defines the SPAD between them, where $\Theta_j$ and $\Theta_i$ are the voltage phase angles at the respective buses.

![Diagram showing instantaneous voltage and phasor representation of voltages](image)

Figure 2.7: Left: Instantaneous voltage at buses $i$ and $j$. Right: Phasor representation of voltages showing the SPAD, $\Theta_{ji}$

Typically synchro-check relays are in place to measure the SPAD between the nodes on a breaker. These relays will deny closing a breaker if the SPAD is above the predetermined threshold, as occurred during the 2003 blackout in Italy [1]. Finding appropriate SPAD thresholds for different conditions can be done via dynamic simulation sensitivity studies of line reconnections but typical values used are 60 degrees for 500 kV, 40 degrees for 230 kV and 20 degrees for 115 kV systems [16]. The SPAD correlates with the amount of active power potential between the nodes and is a critical consideration in restoring power systems where many lines need to be reconnected. Additionally, SPADs commonly need reduction during the load restoration phase because of high active power flows [14].

2.2.1.1 SPAD reduction controls

Most studies use re-dispatch of generator active power as the primary control mechanism for reducing SPADs [15], [17]–[20]. Re-dispatching the generators allows the operator to shift power flow through the system in order to reduce the systems tendency to want to push power through the disconnected line. In effect, operators re-rout active power to balance the
2.2. POWER SYSTEM RESTORATION

generation and load on either side of the disconnected line. This reduces the SPAD across the line. An operator with intimate knowledge and experience with the system can intuitively manipulate the power dispatch, but several optimization techniques can be applied to find good re-dispatch solutions which will be discussed in the next section.

Another popular control technique in the literature employs load shedding [18]–[20]. Shedding load provides an alternative or supplementary method alongside generator redispatch in that it provides additional degrees of freedom to power flow control. Load shedding can be done with either fixed or dispatchable loads, however dispatchable loads give more precise power flow control fixed loads if they can occupy a continuum of consumption states. Operators are often load shedding adverse because it directly impacts the monetary income of the transmission system operator [1], even though it may prevent larger scale problems.

Load pickup has also been modeled as a method for SPAD control [17], [21]. During restoration, there are typically unserved loads on system that can actually help facilitate the reconnection of a line if picked up. This works in the same way as generator redispatch and load shedding in rerouting power flow, but has monetary benefits in that some additional load gets served in the system. For any restoration action, the system must be evaluated for stability. If the generator capacity is available and the system is stable, picking up a load may balance out the power flow on either side of the line to be restored, thus reducing the SPAD. Load pickup should be considered in any holistic study of system restoration.

Lastly, a recent study found that manipulating generator terminal voltages provides a faster but limited means of SPAD control [20]. In this study all of the above mentioned methods were combined with terminal voltage control for a more efficient solution. The voltage control has two benefits: The terminal voltage can be changed in seconds whereas generator active power changes are on the order of minutes, and they do not incur any marginal cost to the transmission grid operator.

2.2.1.2 SPAD reduction optimization

Several optimization techniques have been found in the literature for determining good control actions for reducing SPAD. One of the simplest methods involves calculating the sensitivity of the voltage phase angle on each bus to changes in active power output of the generators [15]. This is done by relating change in active power injection at each bus $\Delta P_i$ to change in voltage phase angle at each bus $\Delta \delta_i$:

\[
\begin{bmatrix}
\Delta P_1 \\
\vdots \\
\Delta P_N 
\end{bmatrix}
= 
\begin{bmatrix}
H_{11} & \cdots & H_{1N} \\
\vdots & \ddots & \vdots \\
H_{N1} & \cdots & H_{NN}
\end{bmatrix}
\begin{bmatrix}
\Delta \delta_1 \\
\vdots \\
\Delta \delta_N 
\end{bmatrix}
\]

(2.22)

where

\[
H_{ij} = V_i V_j \left( G_{ij} \sin \delta_{ij} - B_{ij} \cos \delta_{ij} \right)
\text{ for } i \neq j
\]

(2.23)

\[
H_{ii} = V_i \sum_{j=1,j\neq i}^{N} V_j \left( -G_{ij} \sin \delta_{ij} + B_{ij} \cos \delta_{ij} \right).
\]

(2.24)
where $V_i$ is the voltage magnitude at bus $i$, and $G_{ij}$ and $B_{ij}$ are the conductance and susceptance of the line from $i$ to $j$. The off-diagonal $H$ matrix elements represent the apparent power flow between buses and the on-diagonal elements represent the power injection at each bus. The vector $\Delta \delta$ can be manipulated to change the desired SPAD within the system, and Eq. 2.22 will solve for the required changes in $\Delta \mathbf{P}$. The method described here is very direct but only incorporates generator re-dispatch and lacks real world problem constraints such as generator power limits and economic dispatch.

Optimal power flow (OPF) is another tool that can be used for determining optimal generator re-dispatch, load pickup and shedding [18], [22]. Here the OPF method, which is was designed for estimating optimal economic power dispatch as described in section 2.1.3, is applied with special constraints on the SPAD. In the formulation of the OPF,

$$\Theta_{ji} \leq \Theta_{\text{max}}$$  \hspace{1cm} (2.25)

is considered for the list of constraints, where $\Theta_{\text{max}}$ is the predetermined maximum SPAD threshold. Eq. 2.25 is transformed to an equality constraint with a constrained slack variable to fit the OPF formulation:

$$\Theta_{ji,\text{slack}} - \Theta_{ji} = 0$$ \hspace{1cm} (2.26)

where the slack variable $\Theta_{ji,\text{slack}}$ must be within the limits:

$$0 \leq \Theta_{ji,\text{slack}} \leq \Theta_{\text{max}}.$$ \hspace{1cm} (2.27)

Eq. 2.26 and Eq. 2.27 transform the SPAD limit into the format of the OPF problem formulation, thus expanding the usage of OPF to encompass SPAD limits.

In [18], the SPAD constraints are applied and OPF is solved via the primal-dual interior point technique where the minimization objective parameters are the squared deviation from the original generator schedule and load shedding. The method of SPAD constraint applied in this thesis is modeled after this approach. The approach taken in [22] is more sophisticated in that it limits the generator redispatch step size if the OPF solution calls for large setpoint changes. A maximum step size is defined and an iterative method is applied where SPAD constrained OPF is solved and the generator setpoints changes are clipped based on the maximum step size. This iterative approach is applied to protect the generators from large rotor shaft impacts during restoration.

A modified discrete-continuous genetic algorithm (GA) is applied in [19] to solve the SPAD reduction problem. The individual encoding was composed of a binary variable to represent the load switches, and a continuous variable for the active power generation, thus producing a hybrid discrete-continuous gene. The crossover operator was applied separately to the binary and continuous parts of the gene. The method proved successful in limiting SPAD of two tripped lines on the IEEE 118 bus system. A population size of 30 and generation count of 1000 was used in the study, which would require more computation than the primal-dual interior point method traditionally used in the OPF algorithm as in [18].

A mixed integer nonlinear programming algorithm and alternative two-stage decoupled algorithm with load pickup as a control means is implemented in [21]. And a more holistic method for restoration including line restoration order is implemented in [17].
2.2. POWER SYSTEM RESTORATION

The papers [18], [19], [21], [22] all use similar variants of the same objective function: Minimization of the weighted sum of active power generation adjustments and load shedding.

2.2.2 Traditional restoration techniques

Traditional techniques are generally based on human knowledge and experience with the power system in question. Very general rules are established to help guide the system operators through the restoration process, which is useful in reducing stress for operators and perhaps improving their decision quality during restoration. However, the traditional methods lack the mathematical and modeling rigor required to perform optimization on the restoration method, therefore these techniques could result in suboptimal results.

2.2.2.1 Traditional planning

Traditional restoration planning is focused on high level decision making based on the intuition of system operators. There was a push for creating expert systems in the 1990s. Expert systems (ES) can be thought of as hard coded agglomerates of human expertise wrapped into decision making protocols, considered to be the first successful form of artificial intelligence [23]. The rules for restoration developed by human operators, that would eventually make their way into ES are described in this section. The focus in the literature on issues regarding power system restoration have changed over time, generally evolving as a result of greater availability of computational power and improvements in algorithms and AI. The development of power system infrastructure technologies has been slow relative to the development of computation, although advances in data collection of power system parameters have been made in an effort to take advantage of the increasing sophistication of AI. Phasor measurement units (PMU) for instance, measure the real time voltage and current phasors at any system node, where real time means there is less than a second delay between the measurement and its arrival to the supervisory control and data acquisition (SCADA). This section on traditional restoration planning refers to the time period from the late 1980s to the early 2000s, when ES were the technology of choice.

A 1991 paper [14] tabulates the main issues that need to be solved in system restoration. They illustrate the need for rules of thumb or thresholds for action taking on the system addressing different problems. A survey of power system restoration plans was conducted and a synthesis of general rules and concerns was tabulated:

1. Identification of the status of the collapsed system, components, and equipment
2. Restart and supply of station service to plants, substations, cable pumping plants, compressed air, etc.
3. Coordination of power plant startup timings with load pickups to bring generators to their stable minimum levels and within the range of major analog controllers
4. Energizing large sections of transmission lines within the acceptable transient and sustained over voltages
5. Picking up load on large increments without the risk of frequency decline
6. Reintegration of the skeleton of the bulk power supply with the requisite time-consuming switching operation

7. Deactivation of the automatic load shedding and automatic switched capacitors during initial phases of restoration

8. Maintenance of steady-state and transient stability as the system is being restored and when impedances are large

9. Reduction of standing phase angles when closing loops to firm up transmission paths

The paper claims that the problem of system restoration is very complex and contemporary methods involve breaking it up into smaller subproblems, but there is a need for the exploration of advanced technologies, namely ES and better operator training. Many of the listed concerns involve monitoring the dynamic phenomena of the system during restoration.

Another set of restoration tasks and concerns is defined in [24] which is very similar to the above list but in greater detail. In addition, this paper describes general restoration strategies i.e. the build-down and build-up strategies. The baseline of each strategy includes opening all breakers in the affected area after system collapse to simplify the restoration process and determination of network status (closed breakers indicate what parts of the system are energized). The specifics of the two plans are outlined below:

1. Build-down strategy[24]: This strategy prioritizes the re-energization of the bulk power network, followed by the reintegration of generation and loads in a step-by-step manner. A difficulty of this strategy is the high reactive power persistence in the system resulting from the unloaded branches. The excess reactive power typically can not be absorbed by the on-line generators, causing over-voltages throughout the system. This strategy however is effective in smaller systems without long distance high voltage lines, systems with high levels of hydro generation which can absorb large amounts of reactive power, or if there is a partial outage on a large system. The steps of this strategy are:

   a) assessment of power system status and local energy storage status
   b) energization of high voltage lines avoiding sustained overvoltages and harmonic resonances
   c) supply of cranking power to non-black-start plants and/or supply of station service for the hot-restart of time critical units
   d) synchronization of units as they become available for service
   e) full restoration of load and return to normal operation

2. Build-up strategy[24]: This is a more common strategy due to its applicability to large non-hydro systems and is selected in scenarios of complete system collapse with a lack of interconnection assistance. It involves performing parallel rebuilding of islands then reconnecting them after they are sufficiently stable.

   a) assessment of power system status and local energy storage status
   b) sectionalization of the system into subsystems with black-start generation units
   c) supply of cranking power to non-black-start plants
d) interconnection of generating stations within each subsystem  

e) synchronization of subsystems and load pick up to stabilize all units  
f) start-up and synchronization of large super-critical once through units  
g) closing of ties with the interconnection when adequate reactive absorbing capability is reached

The state of the target system as described in [25] is the goal of the restoration process. The target system is not necessarily the same as the predisturbed system because equipment may be damaged and the underlying demand may have changed due to the passage of time or the cold load phenomenon, however the target system can usually be determined after the disturbance occurs. The target system may specifically include in-service units, their capacities, system load, power import/export, and a minimum topological structure. The restoration strategy described in [25] takes a very human approach in its assumptions. The paper breaks restoration into three stages: Preparation, system restoration, and load restoration. The preparation stage is allotted 30 to 60 minutes and included defining the target system, and defining a restoration strategy, the system restoration stage is allotted 3 to 4 hours for restoring the skeleton or islands of the bulk power system. The last stage, or load restoration stage, is allotted up to 12 hours where the loads are restored as fully as possible. These time frames are designed around a large scale system collapse.

2.2.2.2 Expert system

An expert system (ES) requires a hierarchy of modules that represent knowledge based on human experience. The base level modules represent actions that must be granular enough to be generally applicable to any power system, but can be combined to achieve higher level system specific tasks [25]. Three restoration periods are defined in [26] similarly to in [25]: Preparation, system restoration, and load restoration. The requirements of the ES during each period is defined below. Specifically, these requirements are given for a build-up strategy as defined in section 2.2.2.1.

1. Planning: The ES should be able to define unforeseen subsystems or islands on the system after blackout using general restoration modules. The ES should have probabilistic models for the the success rate of execution of actions such as generator start-ups or breaker closings. The ES should define teams of people, including control center operators and field operators, designated to each subsystem.

2. System restoration: The primary objective is to build up the subsystems and ultimately to synchronize and interconnect them. The ES should build up paths between the online and necessary offline stations. The ES needs to be able to energize lines using correct timing and switching sequences, check constraints, and rank alternative paths. There will be time for the use of analytical tools in this stage, which are necessary for monitoring over voltages, frequency transients, line loadings, generator loadings, transformer tap selections, and phase angle determination. Additionally, the ES should be able to perform parallel restoration of each island. Loads should be picked up incrementally and matched with maximum generator ramp rates. Subsystems should be synchronized by the generators where frequency can be regulated.
3. **Load restoration**: This final phase of the restoration involves integrating the islands together and ensuring service to all loads. Only one restoration team is needed in this phase as all islands should be stable and actions are performed in series from this point. At this point, thermal units with large reactive absorbing capacity should be online, allowing for energization of high voltage lines. The ES should use restoration power flow, short circuit, voltage transient, and electro-magnetic transient software to determine the line reconnection constraints. The ES should use a cold load pickup model to estimate the size of loads to be served, and evaluate the frequency effects of picking up those loads.

ES have been described in [27], [28] with more mathematical rigor. Following the general ES requirements above [27] includes a toolbox of *generic restoration milestones* (GRMs) that can be combined by system operators to establish a restoration strategy. Each GRM contains a set of logic that may include the execution of lower level GRMs. There is therefore a hierarchical order to these milestones that is executed in a fixed logic manner, but uses some level of analysis and optimization for the lowest level actions. For example, a high level GRM is tasked with bringing generators and critical loads online, and OPF is used to find acceptable operating points for the generators between actions.

The solution in [28] uses a hierarchical data structure that represents the system and its state. High level abstract information such as island index and energy surplus/deficit state are stored at high levels of the hierarchy whereas granular elements of the system and their corresponding characteristics lie at the bottom of the hierarchy. The basis of the ES deals only with manipulating switches, with a sensitivity framework that predicts the power flow effect of opening or closing each switch. The logic of the ES manipulates the topological information for system bookkeeping and executes predetermined algorithms to restore the network.

### 2.2.3 Modern restoration techniques

Modern restoration techniques are heavily based on modeling and optimization of decision making, whereas traditional techniques promote modeling of the system only to facilitate human understanding. The boundary between traditional and modern techniques as identified in this thesis, is the decision making agent, where traditional methods rely on human decisions perhaps assisted by recommender systems, whereas modern techniques are geared toward relinquishing control to an AI. Modern techniques frame the restoration problem into a mathematical framework that can be optimized. However, there are limitations to these methods arising from model resolution and accuracy, model scale, and availability of computational power. The limits of modern techniques are being pushed further towards the capability of human operators and may surpass the ability of humans with time.

#### 2.2.3.1 Centralized methods

Centralized control methods involve optimization of control actions based on a holistic view of the power system. That is, all relevant information from the power system model is aggregated in once place and a central algorithm uses that data to perform intelligent actions on the system. The studies reviewed here, adopting the centralized approach, vary significantly in scope and control space with few discussing the feasibility or logistics of a real
2.2. POWER SYSTEM RESTORATION

world implementation of their technique. This general shortcoming is perhaps because of the academic nature of these studies, where their purpose is to merely provide a proof of concept for a narrow restoration problem.

One study [29] investigates the stability of island reconnections using Support Vector Machines (SVM) as a binary discriminant. Dynamic system data was simulated collected using virtual phasor measurement units which would constitute the feature matrix, and the post-connection stability served as the binary target variable. The SVM method achieved an 85% prediction accuracy on whether island reconnection would be stable.

A more comprehensive study is performed in [30], where an optimal skeleton network is defined after large scale blackout. However, the objective of the optimization was to find the best skeleton network after a large scale blackout, but the restorative steps to achieve the skeleton network are not within the scope of the study. Graph theory is used to calculate node importance degree which is an index based on the number of connected branches to each node. This index is used in calculating the objective function which they call the network reconfiguration efficiency $\eta$ which is determined by network structure parameters $\alpha$, and $\beta$:

$$\eta = \frac{\alpha}{\beta}. \quad (2.28)$$

Parameter $\alpha$ is the average node importance degree of the loads included in the skeleton network, reflecting the importance of the loads involved in the restoration. Parameter $\beta$ is a clustering coefficient, which reflects how spread out or clustered the skeleton network is. A large $\beta$ is undesirable because the skeleton network should connect the desired nodes with as few lines as possible. So effectively by maximizing the network reconfiguration efficiency $\eta$, the most important loads get connected in the skeleton network while connecting as few energized lines as possible. This in theory should minimize time required to reach those critical loads. A Discrete Particle Swarm Optimization (DPSO) is performed, which is a modification of the conventional continuous space PSO allowing it to work in the binary on/off branch space. The study therefore finds an optimum skeleton network according to their criteria in Eq. 2.28. However, restoration after the skeleton network is achieved is not addressed, nor is the order of restoration actions.

A tree representation of restoration is implemented in [31]. A tree representation is interesting and useful because it defines all restoration action sequences in a memory efficient way because each restoration step is defined only once and all feasible sequences end at the leaf nodes. In [31], the infeasible branches of the tree are cut and discarded in the branch and cut method, which greatly reduces the search space of the algorithm. The branch and cut method requires a branching strategy and a cutting criteria. The pseudo-cost branching strategy is implemented which branches on the control variable that changes the objective function the most. The Gomory rounding cut, knapsack cover cut, and fixing variable cut methods where implemented. This study optimized one scenario with 17 disconnected elements, solving 85,597 actions via OPF, taking 112 minutes on their CPU. Although the tree representation is memory efficient, it does not sufficiently the computational efficiency of the action sequencing problem. The heuristic cutting strategies greatly reduce the computation required, but not enough for use in real time.
The following study [32], uses a Genetic Algorithm (GA) to optimize the restoration action sequence for a given degraded state and then trains a Decision Tree machine learning model to learn a decision strategy based on the results. The methodology of this thesis is inspired by the methodology of this study and is therefore described in good detail here. The argument for this methodology is well outlined in the paper. They claim that optimization of the restoration sequence for a given state is useful for gaining insight, but is not of practical use in a real time setting because of the required computing time. The time required to compute the optimal restoration sequence can take hours and increases at a rate of \( n! \) where \( n \) is the number of required restorative actions. So a more useful model would be one that learns simple decision rules to select the best next action, rather than optimize the entire sequence from scratch. Machine learning techniques solve this problem by extracting the useful information from a set of optimal restorations, therefore the Decision Tree is employed after performing the GA optimization on many degraded states. The general flow of the methodology goes as follows:

1. Many degraded systems states are optimized off-line using the GA. The optimal restoration data is stored and formatted into two matrices: one storing the system state features, the other storing the optimal actions corresponding to each state. This is called the training set because it is used to train the supervised machine learning model.

2. A machine learning model, the Decision Tree, is trained on the training set. The objective of the Decision Tree is to predict the optimal action based on the system state which is a classification problem. It does this by finding thresholds on different state features that maximize the separation of the data, i.e. it wants to split the data into pure groups of the same class, also thought of as minimization of the entropy of the data.

3. The Decision Tree is validated using a separate set of data, the test set. The architecture of the tree may be adjusted for better results.

4. The Decision Tree can be used as a decision support tool by the operator.

An important part of this study is the quality of the system simulator used to perform and evaluate the actions and the performable action resolution. This study uses a dynamic simulator that detects the dynamic response of the system. The simulator can handle different operational states, islanded systems, and an array of different restorative actions. Simple models of system components are used to speed up the simulation time. Dynamic mechanical response of the prime movers is modeled, but simple load flow is used to calculate the system voltages and reactive power levels. The actions available to the simulator are:

1. Incremental generator set point changes (based on their ramp rates and a constant time interval).
2. Generator voltage set point regulation.
3. Load pick up in MW, the MVAr follows according to a constant power factor.
4. Energizing branches.

The fitness function used by the GA optimization is computed as:
\[ F = \left( 1 - \frac{0.5 N_v + 0.5 N_{ol}}{N_{ldf}} \right) \times \left( 1 - \frac{I_{f+} + I_{f-}}{I_f} \right) \times L \]  

(2.29)

where \( N_v \) and \( N_{ol} \) are the number of voltage limit violations, and overload violations respectively, \( N_{ldf} \) is the total number of load flows, \( I_{f+} \) and \( I_{f-} \) are integrals of over and under frequency deviations respectively and \( I_f \) is the total integral of absolute frequency deviation and \( L \) is the total served load. The fitness function is therefore the served load multiplied by penalization factors from voltage, load, and frequency violations.

The study performed sequence optimization on 1000 degraded system states and found 7181 good sequences to those 1000 starting states. The data was split into a training set of 6181 sequences and 1000 test sequences. The test validation was provided by showing the fitness of sequences performed by the Decision Tree, which is a poor choice of result presentation because it does not provide relative performance to the GA. It would have been more instructive to present the classification error of the Decision Tree based on the optimal actions discovered by the GA, or at least include the fitness scores achieved by the GA.

### 2.2.3.2 Multi-agent methods

An agent in multi-agent methods are entities that interact with the power system in an intelligent way based on some information input. These agents can exist in a completely decentralized fashion, where agents only receive local data and can only communicate with their adjacent neighbors, such as in [33], [34]. There can be a hierarchy of agents, where higher level agents exercise influence over lower level agents such as in [35], [36]. Under any hierarchical configuration, the agents are strictly defined and act under a defined set of rules, both in what information they receive and what actions they can perform. Advantages of a multi-agent system include the ability to survive single point failures, and more efficient task distribution due to decentralized data processing, leading to faster decision making and response times. Thus multi-agent systems have become a popular network control solution [33].

There are two ingredients that the multi-agent approach must include: An objective function that is either maximized or minimized, and a method to tweak the internal protocol of each individual agent to improve the global performance. All studies reviewed here on multi-agent systems [33]–[36] use a load function of the form:

\[ \max \sum_{i=1}^{n} w_i L_i \]  

(2.30)

where \( L_i \) is the \( i \)th load and \( w_i \) is the weight associated with the importance of the load. Some studies give all loads the same weight and it is not always clear whether loads are modeled as fixed or have a dispatchable nature.

The method of tweaking the agent protocols varies between studies. In [34], [35], no optimization is performed on the agent protocols, rather they are hard coded in a heuristic manner that makes logical sense. These studies where early attempts at multi-agent systems and do not contain the flexibility required for true optimization and agent learning. In [33], an adaptive weight matrix is implemented that dictates how much information is shared between adjacent buses. The adaptive matrix is optimized using partial swarm optimization,
maximizing the objective function in Eq. 2.30. The particle swarm optimization therefore requires a centralized approach, independent of the agents. This partially undermines the power of the multi-agent approach, because performing a centralized optimization may not survive a single point of failure, and slows down the data processing.

The study performed by [36] however uses the Q-Learning algorithm which is a truly decentralized learning approach. Under this method, well described in [37], each agent performs an independent internal protocol update, and performs actions using an internal reward maximization function. The policy quality or $Q$ is the expected discounted reward for executing action $a$ and following the same policy $\pi$ thereafter. At each iteration of learning, the agent observes its current state $x_n$, selects and performs an action $a_n$, observes the subsequent state $y_n$, calculates and receives a immediate reward $r_n$, estimates its future reward following the same policy $V_{n-1}(y_n)$ and adjusts its $Q_{n-1}$ values according to:

$$Q_n(x, a) = (1 - \alpha)Q_{n-1}(x, a) + \alpha(r_n + \gamma V_{n-1}(y_n))$$ (2.31)

where $\alpha$ and $\gamma$ are the learning and discount rates respectively. The learning rate determines how influential new information is to the overall policy, and the discount rate determines how much future rewards impact the learning. Future rewards always have diminishing value proportional to the discount rate. After learning the hidden quality values $Q$, the agent should always perform the action corresponding to maximal $Q$ for a given state $x_n$.

### 2.3 Machine learning models

Dimensionality reduction techniques and the feed forward neural network (FFNN) are described in this section.

#### 2.3.1 Dimensionality reduction

Reducing dimensionality of datasets is useful for data compression, feature extraction and data visualization. In this thesis, the main purpose of dimensionality reduction is for data visualization.

##### 2.3.1.1 Principal component analysis

Principal component analysis (PCA) can be defined as the orthogonal projection of data onto a low dimensional subspace, called the principal subspace such that the variance of the projected data is maximized [38], [39]. It can also be defined as the linear projection that minimizes the projection cost, but this summary will follow the first definition, although both arrive at the same mathematical result.

Consider a dataset $\mathbf{X}$ of dimensionality $N \times D$. The goal is to project dataset $\mathbf{X}$ onto a subspace of feature dimensionality $M$ where $M < D$. Setting $M = 3$ is typical so that the data can be visualized in a three dimensional plot.

First the covariance of the data set is considered:
2.3. MACHINE LEARNING MODELS

\[
S = \frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n - \bar{\mu}_X)(\vec{x}_n - \bar{\mu}_X)^T
\]  

(2.32)

where \( \bar{\mu}_X \) is the mean of the data and \( N \) is the number of samples. The covariance is projected onto a \( D \) dimensional unit vector \( \vec{u}_1 \) with the intention of orienting it to maximize the variance of the projection onto the vector. The maximization problem must have the constraint \( \vec{u}_1^T \vec{u}_1 = 1 \) which is enforced by a Lagrange multiplier \( \lambda_1 \). The maximization of

\[
\vec{u}_1^T S \vec{u}_1 + \lambda_1 (1 - \vec{u}_1^T \vec{u}_1)
\]

(2.33)

with respect to \( u_1 \) is performed by setting the derivative to zero. This maximization shows

\[
S \vec{u}_1 = \lambda_1 \vec{u}_1
\]

(2.34)

which says \( \vec{u}_1 \) is the first eigenvector of the covariance matrix \( S \), or the first principal component. \( \lambda_1 \) is the first eigenvalue, or the variance of the data projected onto \( \vec{u}_1 \). This method can be extended to find the first \( M \) principal components.

In short, PCA involves calculating the covariance matrix of the data \( X \) and finding its first \( M \) eigenvectors, then projecting the data onto the space defined by those vectors:

\[
X_{PCA} = XU^T
\]

(2.35)

where \( U \) is a \( M \times D \) matrix of principal component vectors.

2.3.1.2 Fisher’s criterion

The following on Fisher’s criterion is summarized from [38]. Fisher’s criterion maximizes the between class covariance in a projection, where the covariance is defined as

\[
S_B = \sum_{k=1}^{K} N_k (\bar{\mu}_k - \bar{\mu})(\bar{\mu}_k - \bar{\mu})^T,
\]

(2.36)

which is the covariance matrix of the means of the classes. Here, \( N_k \) is the number of data points in class \( k \), \( \mu_k \) is the mean of class \( k \), and \( \mu \) is the mean of the dataset as a whole. Fisher’s criterion also minimizes the sum of the within class covariances defined as

\[
S_W = \sum_{k=1}^{K} \sum_{n \in C_k} (y_n - \bar{\mu}_k)(y_n - \bar{\mu}_k)^T.
\]

(2.37)

where \( y_n \) is data point \( n \) of class \( k \). Solving Fisher’s criterion amounts to maximizing the Raleigh coefficient with respect to \( W \):

\[
\max_W \frac{W^T S_B W}{W^T S_W W}.
\]

(2.38)

Where \( W^T S_B W \) and \( W^T S_W W \) are the between class and within class variance of the data \( y \) which is \( x \) projected onto the space defined by \( W \), as defined in Eq. 2.35. The Raleigh coefficient is maximized by differentiating and setting to zero. It is found that the columns
of the weights matrix $\mathbf{W}$ are the eigenvectors of $\mathbf{S}_B^{-\frac{1}{2}} \mathbf{S}_B$ corresponding to its largest $D'$ eigenvalues. The number of columns of $\mathbf{W}$ can be chosen as this dictates the dimensionality of the subspace of the data projection.

### 2.3.2 Feed forward neural networks

Feed forward neural networks (FFNN) are the simplest form of neural nets, the formulation for which is well outlined in [38] and summarized in this section. The FFNN can be visualized as a network of nodes and edges organized into layers that send information forward from the input layer to the output layer, as shown in Fig. 2.8. The input data is organized into a matrix $\mathbf{X}$ with columns $\vec{x}_1, \ldots, \vec{x}_D$ where $D$ is the number of dimensions of the input layer. If a single input data point, a $1 \times D$ vector, is pushed through the network, the evaluation goes as follows: The input to the first hidden layer is calculated first, where $M$ is the number of nodes in that layer. The input to each hidden layer node is a linear combination of the input vector:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (2.39)$$

which can be generalized in matrix form:

$$\vec{a} = \mathbf{W}^{(1)} \vec{x} + \vec{w}_{0}^{(1)} \quad (2.40)$$

The hidden layer input is run through an activation function $\sigma(\cdot)$, typically a nonlinear sigmoidal or rectified linear function, returning the output of the node:

$$\vec{z} = \sigma(\vec{a}) \quad (2.41)$$

Many hidden layers can be stacked on top of each other, where the output of each node becomes the linear combination of inputs to the next layer. The general input for a layer $n$ is given by

$$\vec{a}^{(n)} = \mathbf{W}^{(n)} \vec{z}^{(n-1)} \quad (2.42)$$

where $\vec{z}^{(n-1)}$ denotes the output from the previous or $n-1$ layer.

The output layer is tailored to the specific use case of the model. This layer can contain any number of nodes (node count is typically dictated by the number of target variables in the data set) and can be used for either classification or regression modeling. For regression models, the output layer is typically just a linear or rectified linear function, whereas classification models perform better employing a logistic output for binary classification or the softmax function for multiple outputs. The logistic and softmax functions squeeze the output in the range $[0, 1]$. The softmax function plays an important role in this thesis because its output represents the probabilistic certainty of the model of each class being correct. The function is given:

$$\vec{y}(a_j) = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}} \quad \text{for } j = 1, \ldots, K \quad (2.43)$$

The outputs therefore sum to one.
The error calculation is key in training the model, as this is the utility function i.e. the function that is minimized. A multiclass classification is performed in this thesis, in which case the cross entropy error function is used:

$$E(w) = -\sum_{k=1}^{K} \left[ t_k \log(y_k(\vec{x}, W)) + (1 - t_k) \log(1 - y_k(\vec{x}, W)) \right],$$  

(2.44)

where $t_k$ is the target value for class $k$. The target values are either given as one or zero, where the target is a one-hot vector where a one identifies the correct class. The softmax function penalizes both the predictive error on the correct class $t_k \log(y_k(\vec{x}, W))$ and for falsely predicting incorrect classes, $(1 - t_k) \log(1 - y_k(\vec{x}, W))$. It is important to notice that the output $y_k(\vec{x}, W)$ is a function of the weights of the model. The weights will be modified during training in an attempt to minimize the error $E(W)$, using the back-propagation method as described in the next section.

![Generalized FFNN](image-url)  

Figure 2.8: Generalized FFNN, adapted from [38]

### 2.3.2.1 Error back-propagation

Back-propagation was the key innovation in enabling training of neural nets. It is the method of solving for the error gradient with respect to the model parameters, thus enabling gradient descent on the model. In minimizing the error function, the individual errors of each data point in $X$ are summed to represent the error of the model as a whole:

$$E(W) = \sum_{n=1}^{N} E_n(W)$$  

(2.45)

which is the metric used in evaluating the quality of the model. However, weight updates are done on a finer basis after the evaluation of single data points or batches. Back-propagation is used to find the error gradient $\nabla E_n(W)$ with respect to every weight $w_{ji}^n$ of the model, which is then used to update the weights. Working from the output nodes back to the last hidden layer, we see that $E_n$ depends on $w_{ji}$ only through a linear combination of $a_j$ for node $j$, so the chain rule is applied:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j} + \frac{\partial a_j}{\partial w_{ji}}.$$  

(2.46)
The partial derivative of $E_n$ with respect to $a_j$ is the error contribution of hidden node $j$ and is denoted as $\delta_j$:

$$\delta_j = \frac{\partial E_n}{\partial a_j}. \quad (2.47)$$

Using equation 2.42, the partial derivative of $a_j$ with respect to $w_{ji}$ can be defined:

$$z_i = \frac{\partial a_j}{\partial w_{ji}}. \quad (2.48)$$

Substituting 2.50 and 2.48 into 2.46, the following is obtained:

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i, \quad (2.49)$$

therefore the gradient due to weight $w_{ji}$ is equal to error at the output end of the weight $\delta_j$ multiplied by the value at the input end of the weight $z_i$. The calculation of $\delta_j$ for any hidden node dependent on the $K$ nodes above node $j$, is formulated below:

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sigma'(a_j) \sum_k w_{kj} \delta_k \quad (2.50)$$

where $\sigma'(a_j)$ is the derivative of the activation function $\sigma$ on input $a_j$. This shows that the error contribution of each node $j$ is a linear combination of the errors of the above layer and the weights connecting node $j$ to those layers.

To summarize, training a FFNN using the back-propagation algorithm works as follows:

1. An input vector $\vec{x}$ is forward propagated through the network using Eqs. 2.41 and 2.42.

2. Evaluate the derivative of the total error using Eq. 2.44, this is $\delta_k$ the partial derivative of the error with respect to the partial derivative of the network output.

3. Back-propagate the $\delta$'s for each hidden node using Eq. 2.50.

4. Evaluate Eq. 2.49 for each weight in the network.

5. Update weight values using the weight gradients via a gradient descent algorithm.

### 2.3.3 Genetic algorithm

The Genetic Algorithm (GA) was first introduced in [40], but is given another clear and thorough treatment in [41] which is summarized here. The GA is a family of optimization models inspired by evolution. The formulation of the GA requires an encoding to a potential solution to a problem, represented by a chromosome-like data string of a binary variable. The encoding maps to the input parameters of the function that is being optimized, and thus the problem can be evaluated using a single chromosome-like encoding. Broadly, the GA begins by randomly initializing a population of individual chromosomes, which are then evaluated. Reproductive opportunities are allocated to the individuals based on their performance, where the best performers have the best probability of reproduction. A crossover operator is performed on the reproductive pairs of chromosomes, mixing the two encodings. The resulting child is usually mutated with low probability. This process is iterated such that
the genes of the best performers are carried on to future generations of individuals, and the population evolves over time to perform better.

2.3.3.1 Selection

For the purpose of selection, a fitness function is defined:

\[ f_i = \frac{y_i}{y_{avg}} \]  (2.51)

where \( y_i \) is the evaluation associated with individual \( i \), and \( y_{avg} \) is the average evaluation of the population. This fitness value represents a relative performance of each individual to the rest of the population which is useful in several selection methods.

Many selection methods exist two of which will be described here. One method is called \textit{Remainder stochastic sampling} in which a pool of candidate individuals (also called the intermediate population) is created based on two components of the fitness value \( f_i \): From the integer component, an equal number of duplicates of the individual are added to the pool, and the remainder component is used as the stochastic probability of adding an additional duplicate to the pool. For example if an individual has fitness 1.74, one individual is added to the pool, and there is 74\% chance that another is added.

Another simpler method \textit{tournament selection}, is described in [42]. In this method, a number of individuals are randomly chosen from the population (with or without replacement), and the best individual is chosen to be added to the intermediate pool. This process is repeated until the pool is filled to the desired level. This method has some surprising properties, namely binary tournaments (where two individuals are chosen) have the same selection expectation as a linear ranking selection.

2.3.3.2 Crossover and mutation

Crossover involves the recombination two individuals from the intermediate pool to create the new population. As explained in [41], single point crossover is applied to two randomly selected individuals of the intermediate pool by randomly selecting a single crossover point is selected and fragments of the two individuals are swapped at that point. This is illustrated in Fig. 2.9, with two different binary encodings to help visualize the effect of crossover. More than one crossover point can be selected.

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
y & x & y & y & y & x & y & x & x & y \\
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 0 & x & y & x & y & y & 0 & 1 & 1 & 0 & y \\
y & x & y & y & y & 0 & 1 & 1 & 0 & y \\
\end{array}
\]

Figure 2.9: Single point crossover operator, adapted from [41]
Mutation is applied after recombination. For each bit of each individual, a mutation is possible with some low probability, usually less than 1%. Mutation is performed by simply flipping the binary bit to the opposing value. The purpose of the mutation operator is to introduce added randomness to the algorithm, which increases the local search capability of the GA.
Chapter 3

Methods

Discovering and harnessing the intelligence needed for decision making during power system restoration requires a flexible restoration model, a robust optimization strategy, and effective way to synthesize the information gathered from optimization into a decision making agent. Decision making agents or deep learning agents are on the forefront of machine learning research and are practical in real life scenarios because they can be evaluated extremely quickly. This is a distinct advantage over optimization which can not realistically solve problems in real time with today's computational resources. However, the disadvantage of deep learning agents is that they require large datasets to learn optimal decision policies, which can be hard to come by.

This chapter outlines the implementation of each of those vital elements of developing a deep learning agent. Section 3.1 describes the power system restoration model used in modeling power system restoration. The model is designed to be as general as possible in that it can simulate the restoration of any transmission system starting from any degraded state. Section 3.2 describes the action sequence optimization methods. Two main methods were compared: the Tree Search (TS) and Genetic Algorithm (GA), and their hyperparameters were tested to find the best performing setup. The structure of the decision making agent is outlined in sections 3.4 and 3.5.

3.1 Restoration model

The restoration model (RM) is a python class developed for this thesis in which a power system restoration can be simulated and evaluated, the code for which is open sourced and hosted on Github repository https://github.com/ADM91/PowerSystem-RL [43]. The RM provides a means of performing a holistic optimization on the restoration of a power system given a specific starting degraded state. The model developed here is built around steady state load flow using the Matpower package in Matlab. A broad overview of the operation of the RM is shown in Fig. 3.1. The RM is centered around bringing a power system from a stable degraded state back to its ideal operating state. On the left, the dotted arrows indicate the reality of system degradation: First a disturbance occurs, which could be weather related or due to internal component failure, next the system response kicks in to deal with the disturbance, which could include load shedding, component disconnection, intentional islanding etc. However, modeling the reality of system degradation is outside the scope of this thesis, so a very rough approximation of the situation is used. Instead of
attempting to accurately model the degradation of the system, components are randomly disconnected from the system as an approximation. This is deemed sufficient because the goal of this model is to serve as a general restoration engine starting from the degraded state. Therefore disconnecting components randomly serves as an adequate method.

![Diagram](image)

**Figure 3.1: High level restoration methodology.**

The RM was designed to be very general in the sense that the restoration of any meshed power system starting from any degraded state can be performed. This required the development of a framework into which any system could be inserted and which contained both diagnostic logic to uncover the true topology of the degraded system (including islands and deactivated components within islands), and a set of tools to handle the reconnection of any of the feasibly reconnectable components of the system. Feasibly reconnectable branches require that at least one of its buses be energized, in other words, it is not in a region that is under total blackout. Feasibly reconnectable generators and loads also require that they lie on energized buses. In order to reconnect a load on an unenergized bus, that bus must first be energized by a line reconnection.

The RM must be manipulated by an external function in order to realize a restoration. The external function interacts with the RM by feeding in restorative actions and storing data produced during the restoration process. Data collected during the restoration process can then be used to numerically evaluate the quality of the restoration. Generally, the process of realizing a restoration goes as follows: The RM is initialized to a degraded state by the random disconnection method, meaning topological elements are deactivated on the system. A sequence of restorative actions is selected, and are executed one by one on the RM. The state of the system is recorded before and after each action, and stored. Assuming all actions
were feasible, which implies a successful restoration, the stored system states are evaluated by a cost function to arrive at the cost of the restoration.

The RM can be broken up into many independent parts that are described in the sections below, starting with the delineation between restoration action types in section 3.1.1. Section 3.1.2 describes how OPF is used to meet the load flow constraints which include the SPAD limit, line capacity limits, and generator power limits. Section 3.1.3 describes how the model is initialized and how the initial degraded power system state is created. Sections 3.1.4 and 3.1.5 describe the state variables and functions within the RM respectively. The specifics of the restoration process are described in 3.1.5. Finally 3.1.6 talks about how the cost of the restoration is evaluated.

### 3.1.1 Primary versus secondary actions

It is useful to categorize actions taken during the restoration process into two categories: Primary and secondary. A primary action is defined as a system topology altering action. The available primary actions available to the RM are:

1. Branch reconnection
2. Generator reconnection
3. Load reconnection

Primary actions require that power be available at the bus for generator and load reconnections, and that at least one bus be energized for branch reconnections. This requirement simplifies the modeling but asserts the assumption that there are no black start generators in the system.

Secondary actions are defined as power balancing actions. The available secondary actions are:

1. Generator redispatch
2. Load curtailment
3. Transformer tap changes

The logistics of primary and secondary actions are handled differently by the RM. The primary actions are supplied to the RM as a permuted sequence of the required topographical changes to the network. This comes in the form of a unique set of numbers which are mapped to discrete actions required to return the system to its full topology. In a way, the primary action sequence supplies the footprint of a restoration from the viewpoint of the RM. The secondary actions are passive in that they are not supplied to the RM beforehand, rather performed internally over the course of the restoration to maintain the balance between supply and demand in the system. These secondary actions are performed by running OPF on the system topology given an ideal load profile and scheduled generator states. A schedule preserving OPF is described in 3.1.2 below, and is intended to approximate the load balancing actions of a system operator.
3.1.2 Schedule preserving OPF

The OPF problem is formulated as in section 2.1.3 and is executed in Matpower using the Matpower Interior Point Solver (MIPS). The purpose for using OPF is to approximate the secondary actions that a control room operator might take during restoration to balance supply and demand, and to do so, the traditional economic dispatch OPF is modified to impose ideal state preserving pressures on the system. Specifically, the pressures attempt to conserve the scheduled generator dispatch, and load profile, to better mimic the actions of the control room operator who’s mission is to preserve the scheduled operation of the system. Modeling these human actions in a mathematical formulation is difficult, but the schedule preserving OPF outlined below attempts to model the decision making pressures that an operator might feel.

3.1.2.1 OPF constraints

The OPF constraints dictate the load curtailment and generator redispacth in the OPF solution. To place a generator schedule preserving pressure on the OPF, the curve in Fig. 3.2 is used as the generator cost curve model \((f_P)\) used in Eq. 2.19. This cost curve is at a minimum at the scheduled generator setpoint, \(P_{set}\), and thus forces the OPF to try to maintain the scheduled setpoints. In Fig. 3.2, \(w_G\) is a scalar weight applied to the cost of generator deviation, also applied as the slope of the generator cost curves. \(P_{max}\) refers to the maximum real power output of the generator in question.

Loads in the system may be modeled either as fixed or dispatchable. If a load is modeled as dispatchable, it is modeled as a power absorbing generator as discussed in appendix section B.4. These loads therefore also require a cost curve, which is illustrated in Fig. 3.3. The power output of this curve must be negative (to indicate power absorption rather than injection), and the curve must penalize the OPF for reducing the load. The variable \(w_L\) represents the cost weight applied to loads in the system, where higher values of \(w_L\) cause greater pressure on for the OPF to maintain the loads at their maximum values \((P_L)\). \(w_L\) also manifests as the slope of the load cost curve.

![Generator cost curve for OPF.](image-url)
3.1. RESTORATION MODEL

3.1.3 Model initialization

The RM is implemented as a python class which gets instantiated as an object. The interactive nature of the RM class creates an environment in which experimentation on the restoration process can occur. In the instantiation of an object, its initialization function is always executed, after which the object can be manipulated through its variables and internal functions. In the case of the RM object, the initialization requires the input of a base Matpower case which provides the scheduled generator setpoints and load profile, metadata (outlined in Tab. 3.1), and preselected system elements to be disabled in the system’s degraded state.

A graphical representation of the initialization in Fig. 3.4, shows the initialization sequence: The first step is to instantiate the RM object which is fed a base Matpower case and metadata. The base case is used to generate the ideal system state, a key immutable reference variable that is used for state comparison when calculating cost. A newly instantiated RM receives a list of elements it must deactivate on the power system, and it disables those components. Next an island detection algorithm from Matpower (extract_islands) is performed on the power system, returning a new list of isolated sections of the system as independent Matpower cases. At this point, the complete list of deactivated elements is compiled as the act of deactivating some elements, might trigger the disconnection of further elements. Radial sections of a power system are particularly susceptible to large scale deactivation as often one branch feeds a large section of the grid. The final step in the RM initialization requires running the schedule preserving OPF (outlined in section 3.1.2) on each identified island of the power system. If the OPF converges on every island, the initialization is deemed successful, and the state is evaluated and saved to the current system state variable.

The functions available within the RM object include restorative action functions (connect branch, connect fixed load, connect dispatchable load, connect generator) and reverse and reset functions. Every function is critical to the optimization of the restoration sequence and are outlined in detail in section 3.1.5.
CHAPTER 3. METHODS

Figure 3.4: Initialization of the restoration model object.
Table 3.1: Restoration model metadata

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matpower options</td>
<td>A diverse set of options applied to the Matpower OPF function</td>
</tr>
<tr>
<td>Ramp rates</td>
<td>A float vector defining the ramp rates (MW/min) for each generator in the system</td>
</tr>
<tr>
<td>Dispatchable loads</td>
<td>A binary vector identifying the buses on which the load will be modeled as dispatchable</td>
</tr>
<tr>
<td>SPAD limit</td>
<td>The SPAD limit (degrees) enforced during branch re-connection.</td>
</tr>
<tr>
<td>Loss deviation cost</td>
<td>Cost of load deviation (cost/MWh) from the ideal used in the cost function</td>
</tr>
<tr>
<td>Load cost</td>
<td>Cost of unserved load (cost/MWh) used in the cost function</td>
</tr>
<tr>
<td>Generator schedule deviation cost</td>
<td>Cost of generator schedule deviation (cost/MWh) from the ideal used in the cost function</td>
</tr>
</tbody>
</table>

3.1.4 State variables

The state variables describe features of the system. These features or states can be divided into mutable and immutable variables to facilitate the understanding of the use of each variable. A mutable variable can be changed and are thus used to describe the system state over time. The current state and islands variables are mutable: The islands variable gives the topological description of the system at any given time, and the current state variable describes the state information of the system in time. Immutable variables differ in that they can not be changed, which is useful for creating reference points. The ideal state and action list variables are immutable.

3.1.4.1 Current and ideal system state

The systems state variables in Fig. 3.4 (current state and ideal state) have the same form, but ideal state is immutable while current state is mutable. These variables include several aspects of the system, including voltage angles, load data, branch power injections, generator power injections, and losses. The features of each aspect are described in Tab. 3.2.
Table 3.2: Ideal state and current state variables

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus voltage angle</td>
<td>bus ID, island ID, angle (degrees)</td>
</tr>
<tr>
<td>Dispatchable load</td>
<td>bus ID, island ID, enabled/disabled, load (MW)</td>
</tr>
<tr>
<td>Fixed load</td>
<td>bus ID, island ID, enabled/disabled, load (MW)</td>
</tr>
<tr>
<td>Branch injection reactive</td>
<td>to bus ID, from bus ID, island ID, enabled/disabled, power injection (MVAr)</td>
</tr>
<tr>
<td>Branch injection real</td>
<td>to bus ID, from bus ID, island ID, enabled/disabled, power injection (MW)</td>
</tr>
<tr>
<td>Generation real</td>
<td>bus ID, island ID, enabled/disabled, power injection (MW)</td>
</tr>
<tr>
<td>Losses</td>
<td>losses (MW)</td>
</tr>
</tbody>
</table>

3.1.4.2 Islands

The *islands* variable is simply the list of Matpower case objects that define the current islands of the system. The components of Matpower case objects are described in appendix B. The *islands* variable is mutable and serves as the primary bookkeeping medium for the system topology. That is, when an action is performed on the system, the change is first applied to the *islands* structure, which could involve adding branches to islands, enabling branches generators or loads within islands, or combining islands. State information for the *current state* variable is not derived from the system until after the island structure of the system is updated and each island is passed through the OPF function. At which point, the *current state* variable is updated with the new topology and state info taken from the OPF results.

3.1.4.3 Action list

The *action list* is an immutable variable, defining the primary actions required to fully restore the system. The sequence of these actions is ambiguous but the complete list is necessary in order to define the search space of the restoration optimization described in section 3.2. Specifically *action list* is a dictionary of action type-to-bus identifier pairs shown in Tab. 3.3 below.

Table 3.3: Action list

<table>
<thead>
<tr>
<th>Types</th>
<th>Identifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>to bus ID, from bus ID</td>
</tr>
<tr>
<td>Generator</td>
<td>bus ID</td>
</tr>
<tr>
<td>Fixed load</td>
<td>bus ID</td>
</tr>
<tr>
<td>Dispatchable load</td>
<td>bus ID</td>
</tr>
</tbody>
</table>
3.1.5 Functions

The functions of the RM execute restoration action logic on the power system. A subset of these functions execute reconnection actions, and the others provide custom logistic functionality for the optimizers. An important part of the reconnection functions is the recording of the power system state over the course of action execution. This is done by taking snapshots i.e. saving and storing the system states before, after and sometimes during the execution of an action. The other important aspect of action execution is the precise bookkeeping of the system topology and state evaluation. The evaluation is generally executed by running the schedule preserving OPF on the system and updating the current state variable.

3.1.5.1 Branch reconnection

The reconnection functions are the set of functions within the RM that restore components of the power system. The complexity of branch restorations is greater than that of generator or load restoration as seen when comparing Fig. 3.6 and Fig. 3.7, which describe the logic flow of the branch reconnection and generator and load reconnection functions respectively. The branch reconnection function must be able to handle three different connection scenarios: Blackout-energized, energized-energized, and island-island reconnections, whose differences are outlined below.

The three branch reconnection scenarios are succinctly illustrated in Fig. 3.5. The figure represents a power system in a degraded state with two islands, a large blackout region and the three connection scenarios.

1. **Blackout-energized**: Connects a branch between a de-energized bus and an energized one. The bookkeeping includes adding the blackout bus to the island of the energized bus and removing the bus from the blackout category. The SPAD constraint is not necessary because the blackout bus will not contain a connected load, so the OPF is run only once.

2. **Energized-energized**: This is the connection of a branch internal to an existing island. Bookkeeping only requires enabling the branch on the respective island, however, the SPAD constraint must be met before branch connection. Therefore the OPF must be run twice, once before reconnection to ensure the SPAD constraint is met, and again after the line is connected and the SPAD constraint is relaxed.

3. **Island-island**: This is the connection of a branch whose buses are both energized but lie in separate islands. The bookkeeping requires that the islands be combined into one with the connecting line enabled. It is assumed that the islands are in phase, so this connection is modeled as free, i.e. there are no SPAD constraints set on the branch. Note that this is a simplification and in reality the islands must be synchronized within a certain angle tolerance as described in section 2.2.1.
CHAPTER 3. METHODS

Figure 3.5: Branch reconnection scenarios (a) blackout-energized, (b) energized-energized, (c) island-island

Figure 3.6: Logic of the branch connection function.
3.1.5.2 Generator and load reconnection

The logic of connecting generators, fixed loads, and dispatchable loads is largely the same and is illustrated in Fig. 3.7. The difference between them is solely in the bookkeeping of the topology of the system.

![Figure 3.7: Logic of the connect generator, connect fixed load, and connect dispatchable load functions.](image)

3.1.5.3 Reverse and reset

The *reverse* function is used to revert the state of the system back to a previous state, which is useful for the optimization code because it allows failed actions to be undone. This is important in maintaining a feasible action sequence when a set of the available actions are not possible. The inputs to the *revert* function are the desired island topology and the state. Therefore in the optimization, the state before any given action is always stored so that reversion to that state is possible in the event that the action is not feasible.

The *reset* function is used to revert the system back to its initial degraded state. This allows for the reuse of an RM in testing many restorations on the same degraded state.

3.1.5.4 Infeasible actions

An action is infeasible if it is not practical on the current state of the system. Two avenues exist that can make an action infeasible: (a) The action involves connecting a system element on a de-energized bus, or (b) the action results in a non-converging OPF. The first avenue is easily detected by the RM since the energized bus condition is easily checked as the system state indicates the connectedness of each bus. The case in which the action results in a non-converging OPF, it is difficult to diagnose the cause of the failure. The non-convergence of the OPF means that an adequate solution was not found given the constraints provided...
in section 2.1.3 and the initial conditions of the power system. Some example sources of non-convergence include: insufficient power supply in a region or insufficient line capacity to deliver the necessary power.

If a provided action is infeasible by virtue of bus connectedness, the RM returns nothing as indicated by Figs. 3.6 and 3.7, which is registered by the optimization function. When an action returns nothing to the optimization function, the action is either reverted and substituted by another, or the entire sequence is reshuffled and the restoration process is restarted.

### 3.1.6 Cost evaluation

An objective function has been implemented that assesses the quality of any given restoration sequence on a power system. There are three terms in the cost function: unserved load (MWh), deviation from generation schedule (MWh), and deviation in system losses (MWh). Time of restoration is a latent parameter in the cost evaluation model, as the energy parameters contributing to the cost are summations of power over time. The value of each parameter is listed in Tab. 3.4. The objective is evaluated using the data in power system states, which are saved during the restoration process as seen as snapshots in Figs. 3.6 and 3.7. The state list is fed into the cost function which first evaluates the time between each consecutive state and then uses the power curve between them to calculate energy deviation. Eqs. 3.1, 3.2 and 3.3 describe the energy deviation for the time interval between consecutive states for load $E_{L}^{(1,2)}$, generation $E_{G}^{(1,2)}$ and losses $E_{LD}^{(1,2)}$:

\[
E_{L}^{(1,2)} = \sum_{i=1}^{N_{L}} \int_{t_{1}}^{t_{2}} (P_{d}^{i} - P_{s}^{i}(t)) dt \tag{3.1}
\]

\[
E_{G}^{(1,2)} = \sum_{i=1}^{N_{G}} \int_{t_{1}}^{t_{2}} |P_{set}^{i} - P_{out}^{i}(t)| dt \tag{3.2}
\]

\[
E_{LD}^{(1,2)} = \int_{t_{1}}^{t_{2}} (L_{a}(t) - L_{o}) dt \tag{3.3}
\]

where $N_{L}$ and $N_{G}$ are the number of loads and generators respectively, $P_{d}^{i}$ and $P_{s}^{i}$ are the power demand of load $i$ and the power served to load $i$ respectively, $P_{set}^{i}$ and $P_{out}^{i}$ are the power setpoint of generator $i$ and the power injection of generator $i$ respectively, $L_{a}$ and $L_{o}$ are the actual losses of the system and the ideal losses respectively. Times $t_{1}$ and $t_{2}$ are the instances at which snapshots 1 and 2 were taken. These energies are linearly combined with weights to arrive at a cost for time interval 1-2:

\[
C^{(1,2)} = w_{L}E_{L}^{(1,2)} + w_{G}E_{G}^{(1,2)} + w_{LD}E_{LD}^{(1,2)} \tag{3.4}
\]

The total cost of a restoration with $N_{s}$ snapshots is then:

\[
C_{tot} = \sum_{i=1}^{N_{s}-1} C^{(i,i+1)} \tag{3.5}
\]

In order to evaluate Eqs. 3.4 and 3.5, the time $t$ of each snapshot must be defined. The elapsed time between any two snapshots is:
3.1. RESTORATION MODEL

\[ \Delta t = t_{i+1} - t_i = \max\left\{ \frac{|\Delta \vec{P}_G|}{\vec{R}}, |\Delta \vec{P}_L|S \right\} + 15 \text{ sec} \] (3.6)

where $\Delta \vec{P}_G$ and $\Delta \vec{P}_L$ are vectors representing change in generator output and load consumption between times $i$ and $i+1$. The parameter $S$ is the system settling rate (min/MW) due to changes in load. An additional 15 seconds is added to each timeframe to model the reaction time of the operators. The generator ramp rates $\vec{R}$ are defined as:

\[ \vec{R} = 0.33 \vec{P}_{\text{max}} G_{geo} + 0.66 \vec{P}_{\text{max}} G_{hydro} \] (3.7)

where $G_{geo}$ is a vector of ones and zeros where ones define generators of geothermal plants, $G_{hydro}$ is a similar vector where ones define the generators of hydro plants and $\vec{P}_{\text{max}}$ is a vector of the maximum real capacities of each generator on the system. The ramp rate vector $\vec{R}$ then has the units MW/min.

Table 3.4: Objective function parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_L$</td>
<td>100</td>
<td>weight of unserved load</td>
</tr>
<tr>
<td>$w_G$</td>
<td>1</td>
<td>weight of generator output deviation</td>
</tr>
<tr>
<td>$w_{LD}$</td>
<td>1</td>
<td>weight of system loss deviation</td>
</tr>
<tr>
<td>$\vec{R}$</td>
<td>Eq. 3.7</td>
<td>generator ramp rates (MW/min)</td>
</tr>
<tr>
<td>$S$</td>
<td>0.02</td>
<td>system settling rate (min/MW)</td>
</tr>
</tbody>
</table>

3.1.7 Modeling simplifications

The following is a thorough list of modeling simplifications applied in the RM implementation. These simplifications may be improved upon in future work.

1. No black start units
2. Feasibility preserving OPF sufficiently models operator secondary actions
3. Parallelization of primary actions is not performed
4. Dynamic phenomena are not inhibitory to the actions performed
5. Islanding reconnecting is instantaneous
6. Generators are instantly operable starting from any state whenever called upon
7. Generators can always ramp linearly at their highest ramp rate
8. The target system is identical to the pre-disturbance ideal state, i.e. there are no permanently damaged system elements.

9. Component reconnection never fails

10. A 15 second delay between primary actions is reasonable to simulate real world delays

11. There is a rigid delineation between fixed and dispatchable loads where fixed loads are fully fixed and dispatchable ones can exist in a continuum of states down to zero.

12. Loads are constant i.e. not voltage dependent

13. Loads are the same when they get picked up as before outage

14. The cost function perfectly reflects the interests of the TSO (reliability and risk concerns are ignored)

15. The branches have hard capacity limits

### 3.2 Optimization methods

A key component of the thesis is optimizing restoration action sequences of known degraded states. The optimization framework and algorithms are outlined in this section. Several optimization algorithms were implemented and deployed with the cost minimization objective from section 3.1.6. Each optimization method was compared in an effort to find the most efficient algorithm, i.e. the one that required the least computation. In the comparison between algorithms, the allotted action count was held constant in each optimization for a fair comparison. The effectiveness of each method was evaluated by the mean of the best solutions after a number of optimization runs.

The input to the optimizer is an instance of the power system class, the number of evaluation iterations to perform per optimization and the number of optimization runs to perform. The power system class contains a list of required actions, a key input to the optimizer. The actions are mapped to unique integers, creating an action encoding that is easily manipulated by the optimizers.

#### 3.2.1 Optimization framework

The optimization framework depicted in Fig. 3.8 describes broadly the informational flow of the implemented algorithms. A functional programming technique was used where the algorithm itself was simply input to the optimization framework, and modules for informational storage and visualization where consistent between approaches. The framework was initialized with an optimization algorithm, hyperparameters (labeled optimization inputs in the figure) and the RM from section 3.1. The general flow goes as follows:

1. The framework is initialized.

2. The optimization algorithm searches the action sequence space until the end condition is met: The maximum number of restoration actions are simulated.
3. Simulation data is stored in a standardized data structure as the algorithm searches the space.

4. The data structure is returned after the end condition is met.

3.2.2 Tree representation of restoration

Stochastic tree search methods were implemented as heuristic searches of the action sequence space. The tree search method uses a tree representation of the power system restoration as seen in Fig. 3.9, where the root node is the initial degraded state of the system and each subsequent node represents an action. For any given node, all available actions spawn off as child nodes, such that in the last layer of the tree (the leaf nodes), all action sequence permutations of the restoration are represented. For the example shown in Fig. 3.9, there are only 6 permutations, but this number grows by $n!$ for $n$ required actions. The shaded nodes in the figure represent infeasible action sequences, note that an infeasible action nullifies all permutations following that action. Each node in the tree stores information that is useful to the search algorithm and for evaluating action sequences. The action sequences are evaluated by backwards traversing the tree from the leaf nodes and gathering the power system state and cost data that was previously simulated.
3.2.2.1 Search methods

Three stochastic search methods were implemented based on the tree representation. These methods are based on traversing the restoration path with a probabilistic path selection function. The algorithm always has a defined location as a node of the tree and at each iteration, all child nodes are created, i.e. all remaining actions are simulated and evaluated for cost. The algorithm then looks at each child and stochastically selects one based on assigned probabilities.

The simplest implementation of the search method, applies a uniform probability to every child node. This is effectively a random search algorithm, because the chosen restoration paths are randomly chosen. A second implementation assigns selection probability proportional to the cost of the action. The third implementation assigns probabilities to the children linearly based on their cost rank.

3.2.3 Genetic algorithm

The optimization algorithms implemented was a modified version of the Genetic Algorithm (GA). The typical formulation of the GA, which uses binary vectors as encodings to represent solutions described in section 2.3.3, is not compatible with the action sequencing problem. Coming up with a binary encoding for sequences is a nontrivial problem and therefore the sequence itself was used to represent an individual, where no encoding was necessary. In order to use the sequence itself, an adjustment to the traditional crossover method was implemented where a child is created by averaging the positions of each action between the parent individuals. The average position crossover operator was inspired by the work in [45], where a practical crossover and mutation operators for sequences were developed. Another problem encountered in implementing the GA is that most sequences are infeasible, the degree to which depends on the initial degraded state. A feasibility preserving algorithm was developed to ensure that child sequences were possible within the scope of the RM. Also,
to speed up the execution of the GA, and maximize usage of the CPU, a parallelization of optimizations was implemented.

### 3.2.3.1 Average position crossover operator

The crossover operator in the context of the sequence problem must output a new permutation of the action sequence, in which all actions are included. This is not guaranteed by the traditional crossover operator in which the parents are fractured and recombined, because the recombined fractures of each parent may contain the same actions. To avoid this, the proposed operator finds the average sequence position of each individual action, and orders them by the rank of their averages.

For example, if \([A, B, C, D, E]\) defines five independent actions, two parent individuals can be imagined as in Fig. 3.10. The index of each action for each parent is stored, and an average value is calculated for each action, as in the middle of the figure. The sequence permutation received by the child is the sorted rank of each action. There exist cases where multiple actions have the same average position and thus have the same rank, in which case the sorting algorithm chosen for sorting the ranks will determine the order of those actions. It is desirable then to use a sorting algorithm of a stochastic nature to introduce randomness in the cases where actions have the same rank. Note that this crossover operator produces only one child per parental pair, therefore the intermediate breeding pool must be double the desired population size.

\[
\begin{align*}
\text{Index} & : 1 \ 2 \ 3 \ 4 \ 5 \\
\text{Parent 1:} & \ A \ B \ D \ C \ E \\
\text{Parent 2:} & \ D \ A \ E \ B \ C \\
& \downarrow \text{Avg. Pos.} \\
A & : 1.5 \\
B & : 3 \\
C & : 4.5 \\
D & : 2 \\
E & : 4 \\
& \downarrow \text{Rank by Avg. Pos.} \\
\text{Child:} & \ A \ D \ B \ E \ C
\end{align*}
\]

Figure 3.10: Average position crossover operator.

### 3.2.3.2 Feasibility preserving evaluation

Feasibility preservation of an individual helps maintain the integrity of the optimization. There is a large subset of the solution space which is infeasible, meaning at least one of the actions in the sequence is impossible within the scope of the RM. The conditions under which an action is infeasible is described in detail in section 3.1.5.4, and to work around
this, a feasibility preservation algorithm was implemented. The algorithm delays infeasible actions and executes them as soon as they become feasible when the prerequisite actions have been performed. The pseudocode is outlined below:

**Algorithm 1** Feasibility preserving evaluation

```plaintext
1: function EVALUATE(gene_{initial}, model_{RM})
2:   gene_{final} ← empty array
3:   gene_{store} ← empty array
4:   states ← empty array
5:   n_{actions} ← length of gene_{initial}
6:   while length of gene_{final} ≤ n_{actions} do
7:       for action in gene_{store} do
8:           state ← execute action with model_{RM}
9:           if execution successful then
10:              remove action from gene_{store}
11:              append action to gene_{final}
12:              append state to states
13:       action ← gene_{initial}(0)
14:       state ← execute action with model_{RM}
15:       remove action from gene_{initial}
16:       if execution successful then
17:           append action to gene_{final}
18:           append state to states
19:       else
20:           append action to gene_{store}
21:   cost ← evaluate objective function using states
22:   return cost, gene_{final}
```

The while loop first looks at the actions in the gene_{store} array, which contains failed actions that occurred in prior steps. These are executed first because they are awaiting their turn and have order priority over the remaining actions. After trying all stored actions in gene_{store}, the algorithm moves on to the next gene from the initial sequence or gene_{initial} which is promptly removed from the gene_{initial} array. If this action is successful, the algorithm moves to the next iteration of the while loop, otherwise it moves the failed action to the gene_{store} array. This algorithm greatly increases the likelihood of sequence success, although the final gene is often significantly different from the initial gene.

### 3.2.3.3 Parallelization

Many iterations of the GA were performed on different initial degraded system states to generate a dataset of optimal state-to-action pairs as will be discussed in section 3.4.3. Therefore it was important to maximize the capacity of the CPU when optimizing to obtain as large a dataset as possible. Parallelizing the optimization runs enabled a three times speed up in data generation. This was done by running the optimization code on several cores at once.
3.3 IEEE 30 bus system

The optimization methods were tested on the IEEE 30 bus transmission system, illustrated in Fig. 3.11. This system was chosen for testing because it provides an adequate level of complexity for interesting restorations and is small enough to be relatively easily understood and conceptualized.

![ IEEE 30 bus transmission network](image)

Figure 3.11: The IEEE 30 bus transmission network.

3.4 Icelandic transmission system

The Icelandic transmission system is relatively small when compared to the North American and European interconnected systems and is therefore good for studies such as this. An overview of the transmission system is given in the 2016 annual system plan report from the TSO [46]. In 2015, on the entire system, 18.11 TWh was generated, which amounts to an average of 2.07 GW of instantaneous power production and the maximum power production was 2.301 MW. The majority of the energy produced, 14.36 TWh, went to industrial applications. The system losses are 2.04% of the total production on average.

Approximately 80% of the power generated comes from hydro, and 20% from geothermal, therefore the system is nimble due the relatively fast responsiveness of the hydro plants. Ramp rates of generator types are given in section 3.1.6, and the benefits of hydro for frequency response is introduced in section 2.1.4.1.
The main purpose of this thesis is to provide a proof of concept for a machine learning driven decision making methodology for restoration. This main part of the study was done using the model provided by Landsnet, and a dataset of system states. The data provided included 9,352 minute-by-minute snapshots of the system over a two week period.

### 3.4.1 Modeling simplifications

The following modeling simplifications are exclusive to the Icelandic system. The simplifications from section 3.1.7 also apply.

1. Parallel lines are modeled as one.
2. Substation generators are modeled as single units.
3. Study only observes disconnections in the East (RAN to SIG)

### 3.4.2 Subset system

The subset system referred to in this section is a portion of the Icelandic transmission system in which components were de-energized to create restoration scenarios. The reason for restricting the study to a subset of the system was to reduce the size of the problem search space to a level that agreed with the available computational resources. This reduction allowed for a more thorough demonstration of the methodology, but there is no reason to assume the methodology could not be applied to the full transmission system.

To give an idea of the reduction of search space: The original system model contains 108 branches, 85 buses, 22 generating buses, and 68 load buses. The subset of the system used in the study, illustrated in Fig. 3.13, has 18 branches, 18 buses, 5 generating buses, and 16 load buses.
The Eastern part of the system was chosen because it is lightly meshed and provides enough complexity for an interesting study.

![Subset for element disconnection](image)

**Figure 3.13:** Subset of Icelandic system used in element disconnection and restoration optimization, adapted from [46]

### 3.4.3 Restoration dataset creation

To train machine learning models, large datasets are necessary. Therefore a dataset of optimal or near-optimal restoration sequences was generated. Two major algorithms were necessary to make this possible: One algorithm generated random degraded states, the other (GA) optimized the restoration action sequence.

The degraded state generator has access to a dataset of 9,352 operating states provided by Landsnet. This dataset provides a range of typical states that the system operates under, and served as ideal states for the RM during optimization. The degraded state generator follows the following logic:

1. Randomly choose an operating state from the dataset provided by Landsnet.
2. Randomly pick 4 branches from the subset system without replacement and disable them (there is a change that one line may be picked multiple times)
3. Using the RM: Detect islands and model them independently
4. Using the RM: Detect required restoration actions (in addition to the there may be large areas of blackout if power access is cut off by line disconnection)
5. Run OPF on each island to simulate a starting state before restoration.

Either a brute force search or the GA is used to optimize each scenario created by the degraded state generator. If the scenario requires five or fewer actions, the brute force method is applied to solve every permutation of the action sequence. Five actions result in only 120
sequence permutations, therefore it is faster to run the brute force method on those scenarios. Else, the GA is used. If a feasible restoration sequence cannot be found, the scenario is discarded. The data from each successful optimization is stored, and a restoration dataset of the optimal sequences is extracted.

The dataset was formatted into four matrices: The ideal state matrix, the current state matrix, the topology matrix, the action target matrix. Each row corresponds to a single system state and optimal action. The matrices have the form:

\[
\begin{align*}
X_{\text{ideal}} &= [\vec{P}_{\text{gen},i}, \vec{P}_{\text{inj},i}, \vec{Q}_{\text{inj},i}, \vec{L}_{\text{fix},i}, \vec{L}_{\text{disp},i}, P_{\text{loss},i}] \\
X_{\text{curr}} &= [\vec{P}_{\text{gen},t}, \vec{P}_{\text{inj},t}, \vec{Q}_{\text{inj},t}, \vec{L}_{\text{fix},t}, \vec{L}_{\text{disp},t}, P_{\text{loss},t}] \\
X_{\text{top}} &= [\vec{T}_{\text{gen}}, \vec{T}_{\text{branch}}, \vec{T}_{\text{load,fix}}, \vec{T}_{\text{load,disp}}] \\
Y_{\text{top}} &= [\vec{A}_t]
\end{align*}
\]

where

\[
\begin{align*}
\vec{P}_{\text{gen},i}, \vec{P}_{\text{gen},t} &: \text{Active power production of each generator for ideal and current states} \\
\vec{P}_{\text{inj},i}, \vec{P}_{\text{inj},t} &: \text{Active power injection to each branch for ideal and current states} \\
\vec{Q}_{\text{inj},i}, \vec{Q}_{\text{inj},t} &: \text{Reactive power injection to each branch for ideal and current states} \\
\vec{L}_{\text{fix},i}, \vec{L}_{\text{fix},t} &: \text{Active power served to each fixed load for ideal and current states} \\
\vec{L}_{\text{disp},i}, \vec{L}_{\text{disp},t} &: \text{Active power served to each dispatchable load for ideal and current states} \\
P_{\text{loss},i}, P_{\text{loss},t} &: \text{Active power losses for ideal and current states} \\
\vec{T}_{\text{gen}} &: \text{Binary vector of generator statuses} \\
\vec{T}_{\text{branch}} &: \text{Binary vector of branch statuses} \\
\vec{T}_{\text{load,fix}} &: \text{Binary vector of fixed load statuses} \\
\vec{T}_{\text{load,disp}} &: \text{Binary vector of dispatchable load statuses} \\
\vec{A}_t &: \text{One-hot vector indicating the optimal action for the corresponding state}
\end{align*}
\]

About 5,973 restoration optimizations were performed, producing a state-action dataset of 24,431 data points.

### 3.5 Deep feed forward neural network

A deep feed forward neural network (FFNN) was trained to learn optimal restoration actions. The FFNN was implemented following the theory in section 2.3.2 using the Keras package in python. The network was configured to perform classification, where each class represented an action in a pool of possible actions. Several neural architectures were compared.
3.5. DEEP FEED FORWARD NEURAL NETWORK

3.5.1 Data preprocessing

Careful consideration was taken in preprocessing the dataset to provide the FFNN with as much useful information as possible in a condensed way. Condensing the data is helpful in training the FFNN because it reduces the number of input nodes to the network, thus reducing the required number of model parameters. The dataset used to train the FFNN was relatively small (24,431 points) so parameter reduction was helpful. As in standard machine learning practice, the dataset was split into training and test sets (80% - 20%). A 10% validation set was taken from the training set to check for over-fitting. The following steps were taken in preprocessing the data:

1. The difference from ideal state was calculated: \( X_{\text{diff}} = X_{\text{ideal}} - X_{\text{curr}} \).
2. Each column of \( X_{\text{diff}} \) was normalized to mean zero and standard deviation one and renamed \( X_{\text{norm}} \).
3. Fishers’s criterion and PCA were used to project \( X_{\text{norm}} \) onto a 15 dimensional subspace, now denoted as \( X \). This reduced the feature count from 293 to 15.
4. The topological dataset \( X_{\text{top}} \) was horizontally appended to \( X \) which proved to increase the performance of the FFNN.

3.5.2 Configuration

The FFNN was implemented using the Keras package and several architectures were trained as discussed in section 4.3. The Scaled Exponential Linear Unit (SELU) [47] was chosen as the activation function on all hidden units, and the softmax function was used on the output layer. The categorical cross entropy loss function was applied and the adaptive moment estimation algorithm (ADAM) [48] was used for gradient descent.
Chapter 4

Results

The results of the study are broken up into four sections. First, a comparison between optimization methods is presented and the best approach is selected and validated. Next the results from testing the deep learning agent are shown. Then a case study is presented that compares the performance of the deep learning agent to that of operators of the Icelandic transmission system. Last, a computational comparison is shown between the optimization and deep learning techniques.

4.1 Optimization comparison and validation

An optimization performance comparison was conducted between the three flavors of the TS algorithm and the GA. This comparison was done as a validation of the optimization technique that would ultimately be used to create the state-action dataset for deep learning. In creating the dataset, a computationally efficient method of optimizing restoration sequences was desired to speed up the dataset creation. Therefore an optimization method comparison was an important high level step in the methodology of this thesis.

4.1.1 IEEE 30-bus degraded state

The optimization algorithms were tested on a degraded state of the IEEE 30-bus system shown in Fig. 4.1, where red elements are de-energized. The degraded state was designed to provide an interesting study case with enough complexity to make the restoration non-trivial. From this state, there are 14 actions that need to be taken to fully restore the system, which represent 14! or about 87 Billion possible action sequence permutations. In reality, many sequence permutations are infeasible. These feasibility restrictions significantly reduce the size of the solution space. For instance, the first action is limited to five feasible possibilities, reducing the solution space by about 2/3. There is further reduction of the space at each step where space loss is proportional to the fraction of feasible actions. Rough estimation suggests that there are approximately 10 Billion feasible action sequences, which still poses an interesting optimization challenge.
4.1.2 Comparison

The results of the optimization method comparison are shown in Fig. 4.2, where each line corresponds to the performance of an individual method. The GA was run twice using different mutation rates 25% and 75%, and the three flavors of the TS method were performed as well. The comparison between the methods was established with the following rules:

1. 20 optimization runs were performed using each method.

2. Each of the 20 optimization runs were capped at 1560 total actions, that is, during the sequence evaluation step of the optimization, each action in the sequence counts towards the action count.

3. The best performing individual sequence at each iteration of the optimization is singled out and its cost was stored, this is called the lower envelope.

4. An average of lower envelopes is calculated for the 20 optimization runs of each method and plotted, seen in Fig. 4.2.

As apparent in the figure, the GA outperforms the TS methods. It is also clear that the higher mutation rate 75% gives the best overall performance of the methods tested. Therefore the GA with $\eta = 0.75$ was selected for deep learning dataset creation.
4.2 Validation of genetic algorithm

The optimal solution to the degraded state presented above in 4.1.2, is explored in this section. The solution presented here is the product of the GA $\eta = 0.75$ method. A validation of the method is performed by by showing that the solution is reasonable given knowledge of the cost function.

Tab. 4.1 shows the best action sequence found by the GA $\eta = 0.75$ method, and Fig. 4.3 provides information on the performance of this sequence. The cost function, outlined in section 3.1.6, gives overwhelming weight to unserved load in the cost calculation, so one would expect to see the GA to find a solution in which the lost loads are connected as quickly as possible. The generator deviation and system losses have a much lower but equal weight, so a reduction in these is expected after the loads are served.

Tables A.1, A.2 and A.3 show the idea generator, load, and branch parameters of the system respectively. It is useful to reference these to get a clear picture of the ideal system state.

In Fig. 4.3, subplot (a) shows the the cumulative cost over the time of the restoration. Each curve (unserved load, generator deviation and losses) are cost terms from Eq. 3.4 and their combination provides the total cost as in Eq. 3.5. The numbered areas on (a) show the time frame of each action in the sequence from Tab. 4.1, clearly the most time consuming action is ramping up generator at bus 2, in action number 6. Subplot (b) shows the unserved load in the system over time by bus. The de-energized loads in the system (buses 2, 3 and 4) are all unserved and additionally the load at bus 8 is curtailed due to lack of generation capacity. The largest loads should be recovered first to minimize the restoration cost, so it is crucial that the sequence get the generator and load on bus 2 on-line quickly. Subplot (c) shows how much each generator deviates from its ideal schedule. Note that under-powered and over-powered generators weigh equally in power deviation, so the reduction in deviation from the generator at bus 2 corresponds to the energization of loads at bus 2 and 8. Subplot (d) shows the deviation in losses referenced to the ideal state. Relative to power deviation and unserved load, the system losses typically deviate less by order of magnitude. Subplots (c) and (d) indicate that the restoration does not converge perfectly to the pre-disturbance
system state because the plots do not go to zero. This unfortunately occurs because the schedule preserving OPF finds a local optima outside the global optimal solution.

Figure 4.3: Cost and power deviation results from best performing action sequence discovered by GA $\eta = 0.75$
Table 4.1: Best performing action sequence discovered by GA $\eta = 0.75$

<table>
<thead>
<tr>
<th>Action</th>
<th>Type</th>
<th>Bus(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Line</td>
<td>6, 8</td>
</tr>
<tr>
<td>2</td>
<td>Line</td>
<td>4, 6</td>
</tr>
<tr>
<td>3</td>
<td>Load</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Line</td>
<td>2, 5</td>
</tr>
<tr>
<td>5</td>
<td>Load</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Generator</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Line</td>
<td>2, 4</td>
</tr>
<tr>
<td>8</td>
<td>Line</td>
<td>1, 2</td>
</tr>
<tr>
<td>9</td>
<td>Generator</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Line</td>
<td>1, 3</td>
</tr>
<tr>
<td>11</td>
<td>Load</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>Line</td>
<td>3, 4</td>
</tr>
<tr>
<td>13</td>
<td>Line</td>
<td>2, 6</td>
</tr>
<tr>
<td>14</td>
<td>Line</td>
<td>4, 12</td>
</tr>
</tbody>
</table>

The best performing sequence presented in this section is clearly not the global optimum solution because it could have addressed the load and generator at bus 2 more quickly. However, the result is reasonable within the computational capacity given to the algorithm (1540 action evaluations for a solution space of 10 Billion possible sequences). A general trend of serving the loads quickly is observed, and given more time computational capacity the GA would arrive at a result closer to the global optimum. This exploratory analysis of the GA provided valuable information about its capability which was used to design the scope of the system degradation in the deep learning dataset creation.

4.3 Deep learning results

As outlined in section 3.4.3, a dataset of optimal restorations was collected using the GA on randomized degraded states of the Icelandic transmission system. This data set $X$ has exactly 24,431 points, 326 features, and the target $Y_{top}$ is a sparse matrix of 33 dimensional vectors. The target provides the optimal action in one-hot 33 featured vector. This section goes over the dimensionality reduction and visualization of the system state in section 4.3.1, the training and test performance of the FFNN in section 4.3.2, a case study on the Icelandic transmission system which compares the performance of the GA optimization and FFNN to 8 operators at Landsnet for a restoration scenario in section 4.4, and a summary of the computational expenses of the GA and FFNN in section 4.5.

4.3.1 Dimensionality reduction of system state

Projection using Fisher’s criterion is described in detail in section 2.3.1.2, but in short, it maximizes the variance between class means, and minimizes the variance within each class, an effective class clustering method. Fig. 4.4 shows a projection of the dataset via Fisher’s criterion, note the class separation. The left shows a projection using Principal Component
4.3. DEEP LEARNING RESULTS

Analysis (PCA) which maximizes the variance of the data holistically [39].

![Figure 4.4: Data visualization using PCA and Fishers criterion](image)

In each projection in the figure, each axis represents a linear combination of the raw state data, so each point in the figure represents of a single power system state. The colors represent the optimal action determined by the GA for that state, as indicated by the legend. Note that in the projection via Fisher’s criterion, the extremities of the state projection are dominated by load connection actions, which is expected since a load connection should have a large impact on the system state.

### 4.3.2 FFNN results

The dataset was broken down into 72% for training, 8% for validation, and 20% for testing. These splits where performed by random data point selection from the full set. Three versions of the dataset were compared in training the FFNN: The PCA projection (24,431 × 15), the Fisher’s criterion projection (24,431 × 15), and the full normalized state matrix (24,431 × 293). The topology of the system, a 24431 × 33 matrix, was horizontally appended in each case. It was found that the FFNN performed comparably in each case within a 1% accuracy rate. The projection via PCA was used in the final FFNN model, the results of which are shown in Tab. 4.2 below. The numbers in parenthesis under the Model column represent the number of hidden nodes in each layer of the FFNN. For example FFNN (60, 60) represents a two hidden layer FFNN with 60 nodes in each layer.
Table 4.2: FFNN results

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Accuracy</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFNN (60)</td>
<td>0.766</td>
<td>0.743</td>
</tr>
<tr>
<td>FFNN (250)</td>
<td>0.774</td>
<td>0.754</td>
</tr>
<tr>
<td>FFNN (500)</td>
<td>0.762</td>
<td>0.738</td>
</tr>
<tr>
<td>FFNN (60, 60)</td>
<td>0.782</td>
<td>0.746</td>
</tr>
<tr>
<td><strong>FFNN (100, 100)</strong></td>
<td><strong>0.781</strong></td>
<td><strong>0.755</strong></td>
</tr>
<tr>
<td>FFNN (300, 300)</td>
<td>0.784</td>
<td>0.739</td>
</tr>
<tr>
<td>FFNN (60, 60, 60)</td>
<td>0.795</td>
<td>0.738</td>
</tr>
<tr>
<td>FFNN (100, 100, 100)</td>
<td>0.811</td>
<td>0.751</td>
</tr>
<tr>
<td>FFNN (200, 200, 100)</td>
<td>0.806</td>
<td>0.743</td>
</tr>
</tbody>
</table>

For each architecture that was tested, 30 training epochs were performed. An epoch is one round of training on the entire training dataset. The training error can be seen in Fig. 4.5a. The training error typically continues to decrease as epochs are performed, which indicates that at some point the model will overfit to the training dataset. The point where overfitting occurs can be found by observing the error of validation data set, also shown in the figure. The validation set is not used to train the FFNN, but gets evaluated after each epoch to test for overfitting. When the error ceases to decrease in the validation set, the model is very likely fully trained and further training would cause over fitting and possibly a decreased performance on the test set. Ultimately, the performance on the test set is used as the final model evaluation. The results in Tab. 4.2 show these test results. Fig. 4.5b shows the classification performance of the model **FFNN (100,100)**, the best performing model.

![Training error: FFNN (100,100)](image1)

(a) FFNN training and validation error

![Classification accuracy of test data](image2)

(b) FFNN test accuracy (75.5%)

Figure 4.5: Training and testing of FFNN (100,100)

It is worth noting that the 75.5% test accuracy is a hard binary metric on the classification success rate from a possible 33 actions. The output of the model however is a vector of action confidences in the range [0, 1] summing to 1 (softmax function of Eq. 2.43). For purposes of accuracy evaluation, the action of highest confidence is chosen and compared to
the target. However, often there are several actions that the model assigns high confidence to, as is shown in Fig. 4.9 in the next section. Therefore the FFNN outputs more useful information than the accuracy rate may indicate.

4.4 Case study: Operators vs AI

A practical comparison between human operators and the AI developed in this thesis was performed. In this context, AI refers to both the GA and FFNN. The case study was useful in gathering information for a constructive dialogue on the future of AI in the power system operation space.

The case study compared restoration performances on the Icelandic transmission system under an imagined scenario. The disconnected elements in the restoration scenario are outlined below by Fig. 4.6 and Tab. 4.3. There are 8 disconnected topological elements in the scenario, listed in Tab. 4.3, that require primary restoration actions. Thus about 40,320 different action sequence permutations are possible (there are actually fewer due to some action infeasibilities), which provides enough complexity for a non-trivial restoration.

Figure 4.6: Case study scenario on the Icelandic system.
A few notable characteristics of the scenario:

1. The system is split into two islands
2. A large generator at KRA (56 MW) is disconnected
3. Load at KRA (0.3 MW) is disconnected
4. Load at NSK (3.0 MW) is disconnected

From the standpoint of the cost function implemented for the AI, the penalty for unserved loads far outweighs the penalty for generator deviation at a ratio of 100:1. Therefore the AI should attempt to serve the 3 MW load at NSK first, then it becomes less obvious whether is less costly to try to serve the load or generator at KRA first. The generator is of geothermal type, therefore it has a slow ramp up rate, meaning unserved loads will incur cost for a long period while the generator ramps up since there is no action parallelism implemented in the restoration model.

The performance of the GA is shown in Fig. 4.7. A summary of the performances of the GA, the FFNN and the human operators are given in Tab. 4.5.
4.4. CASE STUDY: OPERATORS VS AI

Table 4.4: System operators vs AI

<table>
<thead>
<tr>
<th>Agent</th>
<th>Action sequence</th>
<th>Time (min)</th>
<th>Restoration cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operator 1</td>
<td>[3, 7, 2, 6, 4, 0, 1, 5]</td>
<td>8.06</td>
<td>870.55</td>
</tr>
<tr>
<td>Operator 2</td>
<td>[6, 4, 3, 1, 7, 2, 0, 5]</td>
<td>7.69</td>
<td>1669.72</td>
</tr>
<tr>
<td>Operator 3</td>
<td>[6, 4, 1, 3, 7, 2, 5, 0]</td>
<td>7.61</td>
<td>1683.19</td>
</tr>
<tr>
<td>Operator 4</td>
<td>[6, 4, 1, 0, 3, 5, 7, 2]</td>
<td>6.97</td>
<td>2365.06</td>
</tr>
<tr>
<td>Operator 5</td>
<td>[3, 1, 0, 4, 6, 7, 2, 5]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Operator 6</td>
<td>[7, 2, 4, 3, 1, 0, 6, 5]</td>
<td>6.19</td>
<td>431.95</td>
</tr>
<tr>
<td><strong>Operator 7</strong></td>
<td>[7, 2, 4, 1, 0, 3, 6, 5]</td>
<td>6.44</td>
<td><strong>371.09</strong></td>
</tr>
<tr>
<td>Operator 8</td>
<td>[4, 1, 0, 3, 6, 7, 2, 5]</td>
<td>6.22</td>
<td>1830.55</td>
</tr>
<tr>
<td><strong>GA</strong></td>
<td>[7, 2, 4, 1, 0, 3, 6, 5]</td>
<td>6.44</td>
<td><strong>371.09</strong></td>
</tr>
<tr>
<td><strong>FFNN</strong></td>
<td>[4, 7, 2, 6, 1, 0, 5, 3]</td>
<td>7.06</td>
<td><strong>632.78</strong></td>
</tr>
</tbody>
</table>

The results show that operator 7 and the GA find the optimal solution based on the cost function defined in section 3.1.6. The FFNN although suboptimal, finds a good solution that is on par with the operators according to the cost function. At this point, it is worth mentioning that the GA and FFNN derive all their knowledge from an imperfect restoration model and cost function. There are many assumptions to the restoration model as stated in section 3.1.7 and 3.4.1, which may skew the results. It is also worth noting that the cost function is imperfect as the cost weights are merely assumed, and system risk is overlooked. Operators have an intuition of system risk that may influence their decision making, meaning they have an internal cost function differing from that presented in this work. However, the precision and consistency of a mathematically formulated decision cost function allows for clear evaluation of decisions, which may be a useful tool for system operation.

Note that the action sequence from operator 5 results in a non-convergence failure in the schedule preserving OPF. This failure is purely within the restoration model i.e. there is no evidence it could not be performed in reality. However, this sequence cannot be evaluated for restoration cost because of this failure.

Figs. 4.8a and 4.8b show how the GA and FFNN performed over the course of the restoration. Note that the GA serves the loads slightly faster than the FFNN and is able to bring the generators back to their scheduled setpoints more quickly. These factors are the biggest contributors to the total cost of the restoration, thus the GA has a better overall performance.

The action output of the deep learning agent can be studied in Fig. 4.9, which shows the outputs of the FFNN through the restoration. Each bar represents a confidence of a given action, and the actions of highest confidence are indicated as green bars. Note that in some system states, the FFNN outputs high confidence for more than one action such as action (1). A post-processing module that tests the ideas provided by the FFNN may therefore be of value, or perhaps an operator could deduce the best course of action given this output. It is interesting that the last action provided by the FFNN is infeasible because it had already been performed in action (4). This shows that the FFNN is far from a perfect solution, and need be trained in a better way, or a post-processing operation is required.
Figure 4.8: Restoration performances on test scenario.

Figure 4.9: Restoration performance by the FFNN.
It is worth noting that some runs of the GA found very good solutions within the first few generations as seen in Fig. 4.7. This is because the data in the figure shows the best solution in every generation, and each individual is forced through the feasibility preserving evaluation function presented in section 3.2.3.2, which conforms each individual to a feasible solution. Since it seems that the GA performs very well even in the first generation, those individuals where taken and compared to the performance of the FFNN:

1. Out of 220 first generation individuals, 19 performed better than the FFNN or 8.6%.

2. The expected value of the GA first generation individual is 1700.87, significantly more expensive than the FFNN which is 632.78.

### 4.5 Computational expense

A comparison of computational expense of each method outlined in this thesis helps illustrate their potential use cases. Specifically, it shows that the GA is not sufficiently fast to be useful in a real time setting, whereas the FFNN can be evaluated in real time, albeit it requires a large training dataset which can be difficult to obtain. Operators require more time than the FFNN to make decisions (FFNN evaluation time is 0.9 ms if given a system state), however their speed is sufficient for real time operation. Operators also require training to understand the system, however they should require fewer data samples than the FFNN.

All computations for this thesis were performed on an Intel Core i7-6700 @ 3.40GHz 8-core CPU.

The computation time of all restoration simulations and evaluations depend on action execution time in the restoration simulator. A histogram of a large sample of action times (~4000) is shown in Fig. 4.10. As described in sections 3.1.5.2 and 3.1.5.1 an action generally consists of:

1. Determining if the bus(s) in question are energized
2. Taking a snapshot of the system state
3. Topological bookkeeping
4. Running schedule preserving OPF
5. Taking a final snapshot of system state

In the case where the SPAD has to be reduced, which occurs when connecting a line between two energized buses, the schedule preserving OPF must run twice as explained in Fig. 3.5. For this reason, the histogram forms a combined distribution of two populations. The average action time taken from this data is 1.18 seconds.
A comparison of computational expenses for different elements of the methodology are shown in Tab. 4.5. This table shows that the evaluation time of the FFNN is about 3 orders of magnitude less that that of the GA. The dataset creation however took an extensive amount of time at around 45 days. For a real application, there would be abundant time to develop such a dataset, and dataset creation should be continuously pursued after training the learning agent to improve its performance. The FFNN training time is negligible compared to creation of the dataset, and the result is a real time decision making agent.

Table 4.5: Computational expenses

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GA on Iceland case study</strong></td>
<td>10.3 hrs</td>
<td>Thorough search (Fig. 4.7)</td>
</tr>
<tr>
<td>State-action dataset creation</td>
<td>~45 days</td>
<td>~40 min per seq. (6 actions on avg.)</td>
</tr>
<tr>
<td>FFNN training</td>
<td>46.4 sec</td>
<td>30 epochs</td>
</tr>
<tr>
<td>FFNN single state evaluation</td>
<td>0.9 msec</td>
<td>One forward pass</td>
</tr>
<tr>
<td><strong>FFNN on Iceland case study</strong></td>
<td>10.1 sec</td>
<td>Performs nearly as well as GA</td>
</tr>
</tbody>
</table>
Discussion

This work is a step in the right direction for improving power system restoration and the general methodology may even have applications beyond restoration. The methodology applied in this thesis is centered on machine learning, closely resembling the work in [32], but with a more complex learning agent: The deep neural network. Several points of discussion were discovered along way in the implementation of the methodology. To start, it is safe to say this method is at best only as good as the power system model used, because all data used to train the learning agent derives from the modeling environment. Therefore extremely accurate power system models will be key in improving work towards automation of power systems. Model improvements are proposed in section 5.1. Also, decision making agents will always conform to their objective functions, therefore it is of critical importance to characterize the objective function in way that is mutually beneficial to all parties involved in power transmission. In characterizing this function, social implications were discovered because different parties have different vested interests. This topic is further discussed in section 5.2. Lastly, it was discovered that although the learning agent presented in this thesis performs remarkably well in classifying states into optimal actions, the real world application of such an agent should demand superhuman decision making capability. Future improvements to the methodology to increase its practicality are discussed in 5.3.

5.1 Restoration model improvements

An accurate restoration model was a prerequisite requirement for this study. Learning agents can only learn from the data they receive from their environment, so it is imperative that the power system model has the appropriate level of detail for the resolution of actions that are to be learned. The level of detail covered by the restoration model implemented for this thesis should not be considered sufficient for training a learning agent for the real world. Many assumptions were made which are listed in sections 3.1.7 and 3.4.1, which skew the realism of the restoration. However, the model was perhaps realistic enough to gain some insights into high level restoration on the Icelandic transmission grid, and more importantly was viable enough to serve as a learning environment for the agent. Taking from the simplifications list of the restoration model, below are some issues that should be resolved in future work:

1. Include black start units: Black start units are vital in restoration scenarios where there is wide area blackout or total system collapse. This can be implemented by
allowing units identified for black start to energize their bus, creating a new islanded system.

2. **Do action parallelization:** Parallel action execution is more efficient than series execution, and is thus parallel actions are performed in real restoration scenarios on islanded systems. Implementation of parallel actions would be difficult and would change the optimization problem formulation because it would no longer be simple sequence optimization. Much thought has to be given to finding an appropriate formulation for a parallel restoration paradigm.

3. **Perform dynamic action simulation:** Dynamic simulations would add an additional validation layer to the model for action feasibility verification. Actions would be simulated dynamically and passed or failed based on some criteria. This may be computationally expensive however.

4. **Model generator ramping behavior:** This would simply require observing real data and developing mathematical models that approximate this.

5. **Estimate the target-system in real time:** The target system may be significantly different from the pre-disturbance system due to component damage and demand changes during outage. A real time estimate of the target would help train the agent in a more realistic way.

6. **Model reconnection failure:** Failure during reconnection is a reality that should be modeled. This would be done by sampling a binary distribution, perhaps based on historical data.

7. **Improve load modeling:** Model the loads as voltage dependent by running an iterative power flow and apply a realistic degree of load shedability.

8. **Use soft branch capacities:** Soft limits will increase the success rate of OPF if it is used, and is more realistic.

Another interesting avenue for further work is increasing the action resolution of the restoration model. This could open up the feasibility of total operational automation from all system operational states. This implies that an agent or agents can perform exceptionally well at a superhuman level with a very carefully defined objective function. Such a learning agent(s) would streamline the operation of the power system.

### 5.2 Cost function improvements

The cost function formulated in this thesis includes three terms: Cost of unserved load, cost of generator dispatch deviation, and cost due to system loss deviation. The relative weights were assumed based on discussions with subject experts [44]. The total cost were calculated as a linear combination of those terms. It was made clear that the TSO heavily emphasizes restoring and securing loads in its operation of the transmission system. The GA demonstrates this emphasis by always prioritizing the reconnection of loads in order of their magnitude.
In reality, there are other considerations that the operators have in mind such as the security or risk of the system. When prioritizing risk for example, it may be more important to restore the system ring than addressing a specific load. Considerations for the cost function used by an AI could become quite political because there are many conflicting interests in the operation of the system: A real example of how ethics has worked its way into the AI space. Note that a cost function is necessary for any computational method, but the importance of getting the cost function right increases as AI gets closer to a real world implementation. Discussion and dialogue between parties is key to landing on an adequate cost function.

5.3 Learning agent improvements

The learning agent in this thesis was trained via supervised learning on a rigid state-action dataset. Although this learning method proved effective, it is most likely not the optimal technique in training a learning agent and is certainly not state-of-the-art. Part of the problem with the supervised learning method is that the learner does not receive any temporal information and is thus not temporally aware. In other words, it does not consider future performance based on its action policy, but merely tries to provide actions that worked well in similar previous situations. Here two new terms are introduced: state quality and action policy. State quality is useful information to a learner because it allows the learner to develop an internal map of good and bad states in a spacial and temporal context. The action policy is the action output based on the state input, which is typically a neural network such as the one trained in this thesis. The combination an action policy and state quality awareness should prove effective in developing better learning agents.

Many parallels can be drawn between power system operation and playing the game of Go, a hugely combinatorially complex game. In Go, at each iteration of action execution, a difficult-to-evaluate system state has to be evaluated and a subsequent action produced. In 2016 a learning agent developed by Google Deepmind defeated the world champion Lee Sedol at Go using reinforcement learning (RL), then continued to improve its performance far beyond the level of humans [49]. The version of their agent AlphaGo that defeated European champion Fan Hui, and world champion Lee Sedol used a policy network, a value network and a Monte-Carlo tree search (MCTS):

1. **Policy network**: Initially trained via supervised learning on a dataset of human professional matches to produce action probabilities, similar to the approach in this thesis. The performance of the network was then refined by RL by playing itself.

2. **Value network**: Trained to output the quality of any game position or state. It trained on trying to predict the outcome of matches.

3. **MCTS**: This is a tree search method thoroughly explained in [50], where each tree node represents a game state. This was used as a post-processing mechanism to boost performance by performing a look-ahead search where the policy network would provide high-probability moves and the value network would evaluate those positions.

A similar approach to the one presented above can be taken to training a power system operating agent. The problem formulation for restoring or operating power systems is similar to that of playing Go in that the action space is huge and the system state is hard to evaluate.
CHAPTER 5. DISCUSSION

The state would be evaluated based on the agent’s future ability to reach the target state. The policy network could be primed using the method used in this thesis, then refined using RL as done for AlphaGo. A MCTS could also be implemented to boost performance by evaluating the suggested actions by look-ahead rollouts. However, an adequate power system implementation would require the possibility for parallel action performance, an important distinction.

Moving on from agent improvement and looking more generally at this work, the following questions may be asked:

**How does this method scale to larger systems?**

The effect of increasing system size was not tested directly in this thesis, but the following can be deduced: A larger power system results in more component failures due to the probabilistic nature of system disturbances, leading to larger outages. Therefore there should be linear relationship between system size and restoration action sequence lengths. The implication of this on the GA is not good, studies [51], [52] show that the time complexity for GA convergence on a solution space growing at rate $2^n$ (equivalent to adding characters on a binary chromosome) is linear, or $O(n)$. However the solution space for the sequencing problem presented to GA in this thesis grows by $n!$, which is a much faster rate than $2^n$, leading to an exponential time complexity. It seems that applying the GA sequencing problem to large systems is intractable. This could plausibly be circumvented by breaking a large power system into smaller subsystems, and training learning agents on each individual subsystem. Also, the effect of training agents via RL on large systems has not been explored, perhaps some heuristics can be learned by a neural network in linear time.

**How does the FFNN react to the implementation of topology changes i.e. new lines?**

Changing the topology of the system introduces new features to the system state, and may have a large impact on the operating modes of the system. A significant change in the operational system state would manifest itself as a reshaping of the amorphous PCA data blob of normal system states. The degree of change matters in this case, since if little change to system operation occurs, old data can still be used to train the FFNN. If a line is added that has a high impact on system operation, such as one over the highlands of the Icelandic transmission system, the operational state might change too drastically to make use of old data. This means that if AI is to be applied with the technique shown in this thesis, or with RL, data will likely have to be continuously generated in order to ensure the AI is up to date and effective.

5.4 Real world application of learning agent

As outlined throughout this discussion, there are real hurdles to developing an AI for operational power system control. However, a real world agent is conceivable and will be talked about here. Some considerations are made for such an intelligent system.

Following the flow in Fig. 5.1, live system data would be fed into a data processor which would organize and format the data into a form familiar to the learning agents. The data would be sent to the agents in packets, representing full system states perhaps somehow transformed to improve agent learning. The agents might be split into multiple entities with different action specialties such as topology control, load shedding control and generator
redispatch control as shown in the figure. There may be many agent bundles that provide service to subsections of the grid in order decrease the size of the action solution domains. This might make control of large systems tractable by combating the exponentially increasing time complexity of increasing action counts. The actions outputs generated by the agents may be passed to a secondary evaluation module, like what was done for AlphaGo. They showed that running look-ahead rollouts using a MCTS boosted the performance of their agents. Finally the best action or actions are passed to an implementation module. This may include running actions past a human operator before implementation, but may be unnecessary. The level of human interaction with the system is unforeseen at this point.

The infrastructure for system operation at TSOs will have to change under the AI paradigm. In order to run an intelligent system as outlined above, humans will have to ensure that the AI is well trained (at least at first). This work will come in the form of continually updating the system model used for training the agents especially after topological system changes, and continual updates and improvements to the agent architectures. Also, large computational capacities will be required to continually train intelligent systems. The computational requirement however should not be a bottleneck due to Moore’s law which observes that the number of transistors in a dense integrated circuit doubles about every two years.

5.5 Conclusion

The methodology of this thesis consisted of finding a suitable optimization algorithm to optimize specific restoration sequences after discovery of stable degraded states. After testing several tree search methods and a genetic algorithm modified for sequence optimization, it was found that the GA with $\eta = 0.75$ was the most efficient methodology. The GA was used to optimize 5,973 action sequences seeking various target states selected from a state database provided by Landsnet. These optimizations provided a dataset of state-action pairs, which were then used to train a FFNN. The performances of the GA, FFNN and operators of the Icelandic system were compared. It was found that the GA performed best, but the FFNN was not far behind, achieving a performance on par with the human operators. As expected, the real advantage of the FFNN came in its evaluation speed. It took only 0.9 msec to evaluate one forward pass of the FFNN. Evaluating the entire restoration the scenario on the Icelandic system took 10.1 secs, which is significantly longer than a forward pass be-
cause each proposed action had to be simulated by the RM. By comparison, the GA took 10.3 hours to optimize the same sequence.

This thesis looks at the application of machine learning to the problem of power system restoration. The motivation for this work comes from the need for better restoration protocol after catastrophic system failures, and a desire to explore the possibilities of machine learning in this problem domain due to recent advances in deep learning techniques. Additionally, it is asserted that the applicability of the methods developed in this thesis may extend beyond restoration to normal system operation. This research shows that a continuous system state space exists which can be characterized based on ease of returning to a targeted ideal state. When state is thought of as a continuous space, power system restoration is similar to normal operation in that the objective is to always push the system to a more perfect state in a way that minimizes operational costs.

The main hurdles to a real world realization of such an AI are outlined in the discussion and include:

1. Developing a detailed restoration model. The model developed for this thesis lacks some important features and its control action resolution is limited to topological actions. A proper model should be highly accurate, include dynamic modeling, and have high resolution of actionable controls.

2. The cost function must be well defined and universally accepted. This has some political implications and is a good example of how ethics play into the development of AI.

3. A superhuman deep learning agent must be developed for broad system control. Parallels were drawn between the problem faced by AlphaGo developed by Deepmind, and the problem of power system control. The methodology applied for AlphaGo could be modified for use in power systems.

4. The problem of scaling to large power systems must be solved. An exponential time complexity of restoration on linear increase in system size poses a problem, but may be circumvented by breaking large systems into smaller subsystems.

The implication of this work is that deep learning for power system control and automation is a feasible paradigm that should be pursued. Work towards this end may lead to a more secure and reliable energy future.
Bibliography


Appendix A

System parameters

Table A.1: Base case generator parameters on the IEEE 30 bus system

<table>
<thead>
<tr>
<th>Bus</th>
<th>Output (MW)</th>
<th>Output (Mvar)</th>
<th>Max real (MW)</th>
<th>Max reactive (Mvar)</th>
<th>Min reactive (Mvar)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25.97</td>
<td>-1.00</td>
<td>80</td>
<td>150</td>
<td>-20</td>
</tr>
<tr>
<td>2</td>
<td>60.97</td>
<td>32.00</td>
<td>80</td>
<td>60</td>
<td>-20</td>
</tr>
<tr>
<td>13</td>
<td>37.00</td>
<td>11.35</td>
<td>40</td>
<td>44</td>
<td>-15</td>
</tr>
<tr>
<td>22</td>
<td>21.59</td>
<td>39.57</td>
<td>50</td>
<td>62</td>
<td>-15</td>
</tr>
<tr>
<td>27</td>
<td>26.91</td>
<td>10.54</td>
<td>55</td>
<td>48</td>
<td>-15</td>
</tr>
<tr>
<td>23</td>
<td>19.20</td>
<td>7.95</td>
<td>30</td>
<td>40</td>
<td>-10</td>
</tr>
</tbody>
</table>
Table A.2: Base case load parameters on the IEEE 30 bus system

<table>
<thead>
<tr>
<th>Bus</th>
<th>Real power (MW)</th>
<th>Reactive power (Mvar)</th>
<th>Dispatchable (1/0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>21.7</td>
<td>12.7</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2.4</td>
<td>1.2</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>7.6</td>
<td>1.6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>22.8</td>
<td>10.9</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>30.0</td>
<td>30.0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>5.8</td>
<td>2.0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>11.2</td>
<td>7.5</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>6.2</td>
<td>1.6</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>8.2</td>
<td>2.5</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>3.5</td>
<td>1.8</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>9.0</td>
<td>5.8</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>3.2</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>9.5</td>
<td>3.4</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>2.2</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>17.5</td>
<td>11.2</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>3.2</td>
<td>1.6</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>8.7</td>
<td>6.7</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>3.5</td>
<td>2.3</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>2.4</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>10.6</td>
<td>1.9</td>
<td>1</td>
</tr>
</tbody>
</table>
Table A.3: Base case branch parameters on the IEEE 30 bus system

<table>
<thead>
<tr>
<th>Buses (from-to)</th>
<th>R (p.u.)</th>
<th>X (p.u.)</th>
<th>Injection (MW)</th>
<th>Injection (Mvar)</th>
<th>Capacity (MVA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - 2</td>
<td>0.02</td>
<td>0.06</td>
<td>10.89</td>
<td>-5.09</td>
<td>130</td>
</tr>
<tr>
<td>1 - 3</td>
<td>0.05</td>
<td>0.19</td>
<td>15.08</td>
<td>4.09</td>
<td>130</td>
</tr>
<tr>
<td>2 - 4</td>
<td>0.06</td>
<td>0.17</td>
<td>16.07</td>
<td>5.21</td>
<td>65</td>
</tr>
<tr>
<td>3 - 4</td>
<td>0.01</td>
<td>0.04</td>
<td>12.56</td>
<td>4.37</td>
<td>130</td>
</tr>
<tr>
<td>2 - 5</td>
<td>0.05</td>
<td>0.20</td>
<td>13.79</td>
<td>4.51</td>
<td>130</td>
</tr>
<tr>
<td>2 - 6</td>
<td>0.06</td>
<td>0.18</td>
<td>20.28</td>
<td>7.42</td>
<td>65</td>
</tr>
<tr>
<td>4 - 6</td>
<td>0.01</td>
<td>0.04</td>
<td>22.50</td>
<td>11.38</td>
<td>90</td>
</tr>
<tr>
<td>5 - 7</td>
<td>0.05</td>
<td>0.12</td>
<td>13.68</td>
<td>6.21</td>
<td>70</td>
</tr>
<tr>
<td>6 - 7</td>
<td>0.03</td>
<td>0.08</td>
<td>9.27</td>
<td>3.17</td>
<td>130</td>
</tr>
<tr>
<td>6 - 8</td>
<td>0.01</td>
<td>0.04</td>
<td>24.82</td>
<td>24.43</td>
<td>32</td>
</tr>
<tr>
<td>6 - 9</td>
<td>0.00</td>
<td>0.21</td>
<td>5.79</td>
<td>-3.36</td>
<td>65</td>
</tr>
<tr>
<td>6 - 10</td>
<td>0.00</td>
<td>0.56</td>
<td>3.31</td>
<td>-1.92</td>
<td>32</td>
</tr>
<tr>
<td>9 - 11</td>
<td>0.00</td>
<td>0.21</td>
<td>0.00</td>
<td>-0.00</td>
<td>65</td>
</tr>
<tr>
<td>9 - 10</td>
<td>0.00</td>
<td>0.11</td>
<td>5.79</td>
<td>-3.46</td>
<td>65</td>
</tr>
<tr>
<td>4 - 12</td>
<td>0.00</td>
<td>0.26</td>
<td>-1.67</td>
<td>-2.02</td>
<td>65</td>
</tr>
<tr>
<td>12 - 13</td>
<td>0.00</td>
<td>0.14</td>
<td>-37.00</td>
<td>-9.26</td>
<td>65</td>
</tr>
<tr>
<td>12 - 14</td>
<td>0.12</td>
<td>0.26</td>
<td>5.39</td>
<td>0.88</td>
<td>32</td>
</tr>
<tr>
<td>12 - 15</td>
<td>0.07</td>
<td>0.13</td>
<td>9.48</td>
<td>-1.06</td>
<td>32</td>
</tr>
<tr>
<td>12 - 16</td>
<td>0.09</td>
<td>0.20</td>
<td>9.26</td>
<td>-0.10</td>
<td>32</td>
</tr>
<tr>
<td>14 - 15</td>
<td>0.22</td>
<td>0.20</td>
<td>-0.85</td>
<td>-0.80</td>
<td>16</td>
</tr>
<tr>
<td>16 - 17</td>
<td>0.08</td>
<td>0.19</td>
<td>5.68</td>
<td>-2.08</td>
<td>16</td>
</tr>
<tr>
<td>15 - 18</td>
<td>0.11</td>
<td>0.22</td>
<td>9.16</td>
<td>0.76</td>
<td>16</td>
</tr>
<tr>
<td>18 - 19</td>
<td>0.06</td>
<td>0.13</td>
<td>5.87</td>
<td>-0.33</td>
<td>16</td>
</tr>
<tr>
<td>19 - 20</td>
<td>0.03</td>
<td>0.07</td>
<td>-3.65</td>
<td>-3.78</td>
<td>32</td>
</tr>
<tr>
<td>10 - 20</td>
<td>0.09</td>
<td>0.21</td>
<td>5.92</td>
<td>4.62</td>
<td>32</td>
</tr>
<tr>
<td>10 - 17</td>
<td>0.03</td>
<td>0.08</td>
<td>3.37</td>
<td>8.01</td>
<td>32</td>
</tr>
<tr>
<td>10 - 21</td>
<td>0.03</td>
<td>0.07</td>
<td>-2.23</td>
<td>-11.67</td>
<td>32</td>
</tr>
<tr>
<td>10 - 22</td>
<td>0.07</td>
<td>0.15</td>
<td>-3.75</td>
<td>-8.48</td>
<td>32</td>
</tr>
<tr>
<td>21 - 22</td>
<td>0.01</td>
<td>0.02</td>
<td>-19.78</td>
<td>-22.97</td>
<td>32</td>
</tr>
<tr>
<td>15 - 23</td>
<td>0.10</td>
<td>0.20</td>
<td>-8.81</td>
<td>-5.25</td>
<td>16</td>
</tr>
<tr>
<td>22 - 24</td>
<td>0.12</td>
<td>0.18</td>
<td>-2.10</td>
<td>7.80</td>
<td>16</td>
</tr>
<tr>
<td>23 - 24</td>
<td>0.13</td>
<td>0.27</td>
<td>7.09</td>
<td>0.88</td>
<td>16</td>
</tr>
<tr>
<td>24 - 25</td>
<td>0.19</td>
<td>0.33</td>
<td>-3.86</td>
<td>1.77</td>
<td>16</td>
</tr>
<tr>
<td>25 - 26</td>
<td>0.25</td>
<td>0.38</td>
<td>3.55</td>
<td>2.37</td>
<td>16</td>
</tr>
<tr>
<td>25 - 27</td>
<td>0.11</td>
<td>0.21</td>
<td>-7.44</td>
<td>-0.66</td>
<td>16</td>
</tr>
<tr>
<td>28 - 27</td>
<td>0.00</td>
<td>0.40</td>
<td>-6.11</td>
<td>-6.08</td>
<td>65</td>
</tr>
<tr>
<td>27 - 29</td>
<td>0.22</td>
<td>0.42</td>
<td>6.17</td>
<td>1.68</td>
<td>16</td>
</tr>
<tr>
<td>27 - 30</td>
<td>0.32</td>
<td>0.60</td>
<td>7.12</td>
<td>1.67</td>
<td>16</td>
</tr>
<tr>
<td>29 - 30</td>
<td>0.24</td>
<td>0.45</td>
<td>3.68</td>
<td>0.61</td>
<td>16</td>
</tr>
<tr>
<td>8 - 28</td>
<td>0.06</td>
<td>0.20</td>
<td>-5.31</td>
<td>-6.08</td>
<td>32</td>
</tr>
<tr>
<td>6 - 28</td>
<td>0.02</td>
<td>0.06</td>
<td>-0.77</td>
<td>-2.70</td>
<td>32</td>
</tr>
</tbody>
</table>
Appendix B

Modeling in Matpower

Matpower is a software package for Matlab that solves various steady state power flow problems as thoroughly summarized in [6]. It employs Newton’s method (as described in section 2.1.2.1), the Gauss-Seidel method, and the Fast Decoupled method to solve traditional power flow. The package is also extended to solve optimal power flow and continuation power flow. The optimal power flow feature is designed to be extensible and flexible making it a useful tool for many power flow problems such as optimal economic generator dispatch, SPAD constraint, optimal economic load shedding, setting zonal reserves and zonal flow limits etc. Matpower is an instrumental tool at the core of this thesis, serving as the engine behind the system restoration algorithm that has been developed. It is important to note what Matpower is not: It is not a dynamic system simulator. All power flow solutions represent a steady state system where the parameters are not changing. Therefore, it does not capture the best and worst case states that occur during dynamic events, such as tripping a load. This fact exposes a weakness of this thesis. The restorations generated may not be feasible because the dynamic behaviors of each action are not modeled.

Matpower organizes networks into four matrices containing system component parameters. These are the bus, branch, generator, and generator cost matrices. The bus matrix contains information about the bus type, power demand (loads), shunt admittance of capacitors or inductors, voltage magnitude and angle (for PV buses), and voltage magnitude and angle limits (for PQ buses). The branch matrix includes model parameters of transmission lines, line ratings, and transformer parameters. The generator matrix contains information about the real and reactive power output of each generator and their power limits. The generator cost matrix defines the marginal cost of operating each generator with respect to its real power output. These matrices and the models behind their contained data are discussed below.

B.1 Bus

The bus matrix contains information about any loads or generators connected to each bus and the voltage magnitudes and phases for PV buses. Generators are modeled as complex power injection at each bus:

\[ S_g = \bar{P}_g + j\bar{Q}_g. \]  

(B.1)

Loads are modeled as the complex power consumed at a bus:
\[ \tilde{S}_d = \tilde{P}_d + j\tilde{Q}_d. \]  
(B.2)

Shunt elements such as capacitors or inductors at buses are modeled as a fixed impedance to ground:

\[ Y_{sh} = G_{sh} + jB_{sh}. \]  
(B.3)

Note that each model parameter above incorporated into the bus matrix is defined by a vector of size \( n_b \times 1 \), where \( n_b \) is the number of buses in the network.

## B.2 Branch

A row of the branch matrix corresponds to a branch, and the columns contain its parameters. Branches include models for the standard \( \pi \) transmission line and a lossless phase shifting transformer in series with the line. The \( \pi \) model of the transmission line with series impedance \( z = r + jx \) shunt admittance \( b \) is illustrated in Fig. B.1. The transformer model, also in Fig. B.1, is placed in series with the transmission line model, but is effectively excluded by setting the windings ratio \( N \) to one. The transformer has a variable tap ratio \( \tau \), and a phase shift angle \( \theta \), which are modeled in the windings variable, \( N = \tau e^{j\theta} \). Matpower defines a from and to end of each line, indicated by subscripts \( f \) and \( t \) in the figure, which get incorporated into the system admittance matrix. This is equivalent to defining an admittance node at each bus. The Matpower model always places transformers at the from end of branches.

![Figure B.1: Branch model adapted from [7].](image)

## B.3 Generator

Generators are given their own matrix in Matpower to provide additional information about generation limits, capability curves and ramp rates. Generators are modeled simply as active and reactive power injection at a bus. Cost curves can be defined for each generator if the user intends to run OPF. These are defined by either a piecewise linear or polynomial functions.
B.4 Dispatchable vs fixed loads

Loads are modeled as dispatchable by defining them as negative power injectors with a negative cost structure. They are defined as generators with negative output i.e. they absorb power instead of injecting it into the system. Cost curves can be assigned to dispatchable loads, the same as with generators. Defining loads this way makes them a useful tool in OPF.