Service Restoration Algorithm in Distribution Networks by using Reinforcement Learning

Pablo Alejandro Parra Gómez

Faculty of Engineering

Los Andes University

A thesis submitted for the B.S. degree of

Electrical Engineering

2020

Abstract

Distribution Networks (DNs) are highly susceptible to faults, which affects their quality and reliability. This thesis proposes a novel Service Restoration approach to help DNs to automatically and intelligently resupply the out-of-service un-faulted customers after a fault occurs. The presented approach is developed with the use of Reinforcement Learning techniques and it is integrated with OpenDSS.

Keywords: DA, Reinforcement Learning, Q-Learning, Service Restoration, OpenDSS.

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Pablo Alejandro Parra Gómez

Dedication

I dedicate this work to my family.

Acknowledgments

I would like to thank my advisor Prof. Gustavo Ramos Lopez Ph.D. for his collaboration and guidance during the development of this thesis. Also, for the knowledge acquired throughout the courses.

Besides my advisor, I would like to thank David Celeita Rodriguez Ph.D. for his support and insightful comments.

I would like to express my greatest gratitude to my family. To Cecilia, Lyda, Margarita, Romelia, Myriam, and Hugo for all of their support throughout my life. In particular, I would like to thank my mother for her willingness to listen, support, and motivate.

Last but not least, I would like to thank my friends and classmates.

Contents

List of Figures

Nomenclature

Acronyms

CAIDI	Customer Average Interruption Duration Index
CI	Customers Interruptions
CMI	Customer Minutes of Interruption
DAS	Distribution Automation System
DG	Distributed Generation
DN	Distribution Network
DNA	Distribution Network Automation
DNO	Distribution Network Operator
EPRI	Electric Power Research Institute
FISR	Fault Isolation and Service Restoration
IEEE	Institute of Electrical and Electronics Engineers
MAS	Multiagent Systems
MDP	Markov Decision Process
ML	Machine Learning
NC	Normally Closed
NO	Normally Open
RL	Reinforcement Learning
SAIDI	System Average Interruption Duration Index
SAIFI	System Average Interruption Frequency Index
SR	Service Restoration

Definition variables and dimensionality

$[f, [ss]]$	State: ordered pair of failure and the state of switches
π	Policy of the Reinforcement Learning algorithm
a_t	Action taken at time step t
$n(IL)_{post}$	The number of isolated loads after restoration
$n(IL)_{pre}$	The number of isolated loads before restoration
$n(LN)$	The number of lines
$n(S)$	The number of states
$n(ST)$	The number of switch status
$n(SW)$	The number of switches
s_t	State at time step t
$[ss]$	The state of switches (ordered list)
f	Failure
n(A)	The number of actions
RR	Restoration rate

Chapter 1

Introduction

This work intends to contribute to Distribution Networks to improve power supply quality and reliability throughout Service Restoration (SR) methodologies. By reviewing and comparing the existing SR methodologies, it is possible to identify the challenges and the limitations of the current methodologies, in order to develop a solution that attempts to overcome them.

In Chapter 1, the background and motivation of this work are presented. Additionally, and based on the identified challenges for SR improvements in distribution networks, the problem statement and research objectives are stated.

1.1 Background and Motivation

Distribution Networks (DNs) allow supplying power to customers, and power quality and reliability ensure customer satisfaction. However, DNs are highly susceptible to failures, which causes more than 85% of power interruptions [3]. That results not only in decreasing reliability indices but also in economic and social impacts [4].

Therefore, there is a necessity to improve the DNs [5], whereby, in recent years, utilities invest in Distribution Automation (DA) intending to create a self-healing smart grid in order to improve SAIDI, SAIFI, and CAIDI indices [4] [6] [7].

Distribution Automation (DA) takes advantage of available technologies to automate DN operations and enhance its performance [8], since DNs became more difficult to design, manage, and sustain [7]. Consequently, DA includes automated ways to locate the fault and restore the service in an optimized approach such as Fault Location, Isolation, and Service Restoration (FLISR) [1] [3].

FLISR operates after a fault detection. Firstly, it locates and isolates the fault as fast as possible. After the fault area is isolated, it transfers customers to healthy feeders and restores power supply to them [4]. Otherwise, customers may experience sustained outages [1].

In a program conducted by the U.S. Department of Energy [1], it was shown that FLISR techniques achieve substantial DN operation improvements, such as a reduction in the number of customers interruptions (CI) by 55%, and in customer minutes of interruption (CMI) by 53%. [1]

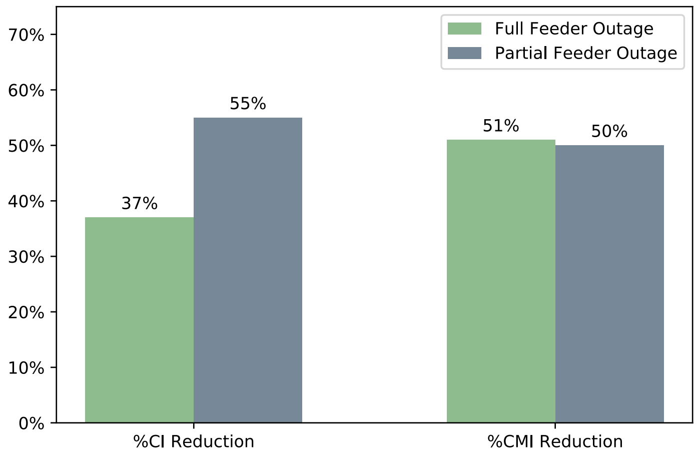


Figure 1.1: FLISR effects on the number of customers interrupted (CI) and customer minutes of interruption (CMI) by the type of outage from USDON2016 [1].

Figures 1.1 and 1.2 show the reduction in CI and CMI by the type of outage and by type of FLISR scheme, respectively. The first figure states the number of customers interrupted decreases by 55% in a partial feeder outage, while the time

[1]Average per event for FLISR operations reported by five utilities over one year [1].

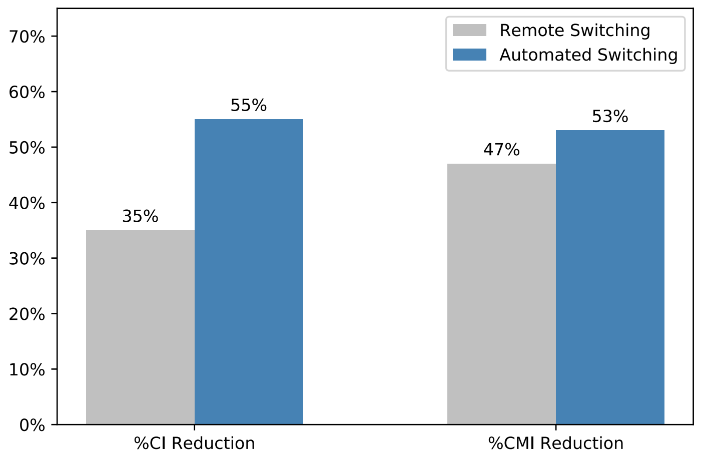


Figure 1.2: FLISR effects on the number of customers interrupted (CI) and customer minutes of interruption (CMI) by the type of FLISR operating scheme from USDON2016 [1].

of a full feeder outage decreases by 51%. On the other hand, the second figure establishes that both CI and CMI have a further reduction when the FLISR scheme is automatic. Lastly, it should be pointed out these reductions improve directly the reliability indices mentioned before.

As aforementioned, three stages constitute the FLISR scheme. The last stage is Service Restoration, one of the most relevant strategies to enhance the resilience of DNs [9], and which is the main topic of this work.

1.2 Problem Statement and Research Objectives

Service Restoration uses network reconfiguration methods to change the DN topology and resupply the out-of-service un-faulted customers [10] [9]. This reconfiguration is carried out through switching operations, considering typical DNs have normally closed (NC) sectionalizing switches and normally opened (NO) tie switches [4] [11]. I.e., the main goal of SR is to find optimal switching sequences for the DN [12].

This procedure must maximize the number of loads restored, minimize the number of switching operations, and maintain operational and topological constraints [13]

[10]. Nevertheless, and as noted in the formulation problem (Section 2.1), SR is computationally demanding because it is multi-objective, non-linear, and operational constrained. And also, it requires to explore a large number of switching operation combinations [9] [11].

SR algorithm uses one or many techniques to obtain a solution, among which are expert systems, heuristic algorithms, meta-heuristic algorithms, graph theory, multi-agent systems, and mathematical programming [9]. Furthermore, they can be implemented under centralized or decentralized approaches [4]. The control approaches and their corresponding techniques are reviewed, in further detail, in section 2.1 and summarized in table 1.1.

Table 1.1: SR techniques comparison [13] [4]

Method	**Optimal solution**	**System dependency**	**Self-learning capacity**
Expert systems	No	Yes	No
Heuristics	No	Yes	No
Meta-heuristics	No	Yes	No
Mathematical programming	Yes	No	No

Table 1.1 compares the different methodologies used to develop SR algorithms and shows their limitations. Most methodologies cannot obtain an optimal solution, but improvements in processing time [13]; thus, the solution optimality is sacrificed to obtain a faster solution. On the other hand, these methodologies, except mathematical programming, are system dependent and do not have the ability to self-learn, becoming out of date when system variations occur [4].

Therefore, these limitations imply a need for human intervention to update their rules or knowledge bases and make the methodologies readapt to the new changes.

1.2.1 General Purpose

According to the previous data, the focus of this work is to develop a scalable Service Restoration algorithm capable of self-healing Distribution Networks by automatic learning the optimal sequence of switching operations to lead DNs to the best possible state after a fault occurs. Moreover, the challenge of automatic learning behavior also includes continuous learning, hence the system is constantly exploring to improve the sequence and adapt to changes in the network. Last but not least, the optimization objectives and constraints formalized in Chapter 2 are also taken into account.

This work also aims to develop a validation system of the SR algorithm, which allows simulations of a practical distribution network and verification of its operational and topological constraints, through the integration of a co-simulation engine.

Objective 1 Develop a Service Restoration algorithm for Distribution Networks capable of self-learning to obtain an optimal solution that accomplishes operational and topological constraints by implementing a Reinforcement Learning technique.

Objective 2 Develop a validation system that emulates and measures the parameters of a Distribution Network, in order to test the proposed SR algorithm performance, by integrating a co-simulation engine.

Chapter 2

Literature Review

2.1 Overview

Many authors have referred and researched the importance of Distribution Automation (DA), and its advantages, to improve the quality and reliability of Distribution Networks [4] [14] [13] [7].

One of the goals of DA is to obtain a self-healing network capable of automatically removing temporary faults throughout the restoration of the un-faulted areas, which provides benefits to both utilities and customers. Utilities will improve their reliability indices and their profits because they reduce penalties from regulators and costs of restoration. In addition, customers will obtain a more reliable and continuous power supply [6].

This self-healing network can be achieved with SR methodologies [1] [15] [16] [17].

2.2 Service Restoration

Service Restoration can be performed by centralized and decentralized approaches [4] [18]. The first one includes expert systems, heuristics, metaheuristics, and mathematical programming methodologies; the second relies on Multiagent Systems (MAS) [19].

The two types of approaches, and their corresponding methodologies, are reviewed in section 2.2.2. However, both approaches tend to solve the Service Restoration problem that can be formalized as a general optimization problem, as suggested in section 2.2.1.

2.2.1 General SR Problem Formulation

Service Restoration can be formalized as an optimization problem [20].

The main objective function aims to maximize the number of loads restored (equation 2.1), while the second aims to minimize the number of operations performed to resupply the aforementioned loads (equation 2.2) [9].

$$\max \sum_{i=1}^{n} L_i \cdot k_i \tag{2.1}$$

where L_i is the load at bus i, and k_i is its status (1: restored; 0: not restored).

$$\min\ n_{ops} \tag{2.2}$$

where n_{ops} is the number of switching operations.

Constraints

The presented objective functions are subject to operational and topological constraints [13].

1. **Radial topology:** Radial network topology should be preserved.
2. **Bus voltage limits:** Nodal voltage V_i should be within the minimum V_{min} and maximum V_{max} voltage limit: $V_{min} \leq V_i \leq V_{max}$
3. **Line current limits:** Line current I_i should be under the maximum acceptable line current I_{max}: $I_i \leq I_{max}$.

2.2.2 Control approaches and methodologies

Centralized approach

The centralized approach is the principal control approach used in Service Restoration [13]. It captures and processes measurements and status information of the entire distribution network to obtain an optimal SR solution [16], because of that it requires an extensive computational capacity and a low latency system communication [4] [9]. Nonetheless, the communication system can be a single point of failure [9].

Among its methodologies are expert system, heuristics, metaheuristic, and mathematical programming [19].

1. **Expert system:** creates rules relying on expert knowledge [9]. It is a successful way to SR, however, its solution depends on the system and does not guarantee optimality. Besides, its maintenance is expensive for large networks [4] [9].

2. **Heuristics:** partition and approximate the SR problem by using expert knowledge to obtain a feasible solution. Heuristics present an improvement in time processing, but a difficulty to obtain an optimal solution [13] [9].

3. **Mathematical programming:** is a direct method to address the SR optimization problem by obtaining an optimal solution. Although, it demands substantial processing time for large networks [13].

4. **Metaheuristics:** solves the SR problem as an optimization problem, as well as Mathematical Programming (MP), but with less processing time than MP and with a better solution than Heuristics [13]. This is performed by simplifying a problem with many objectives into a problem with a single equivalent objective [9]. However, the processing time is still huge for practical networks [4].

Decentralized approach

Decentralized approach distributes the intelligence across the distribution network [9], as opposed to unique control intelligence in the centralized approach. This structure requires a more robust and reliable communication system, but it overcomes the centralized approach disadvantages of huge computational capacity and one single point of failure [4] [13]. However, since in this approach, the agent does not know all the distribution network but its neighbor network, the solution obtained is suboptimal [16]. This approach is implemented through Multiagent Systems (MAS).

Figure 2.1 summarizes the strengths and limitations of each of the approaches discussed above.

Table 2.1: Service Restoration Control approaches

Approach	**Strengths**	**Limitations**
Centralized	- Optimal solution	- Huge computing capacity - Single point of failure
Decentralized	- Fast solution - Less data	- Suboptimal solution

2.3 Reinforcement learning

Accordingly to the main goal of this work, there is a need to provide the ability of autonomous learning to the Service Restoration algorithm, by which it can find the optimal sequence of switching operations independently of system variations and without human interventions. For this reason, Reinforcement Learning plays an important role in the development of the SR algorithm.

Reinforcement Learning is a machine learning technique between supervised and unsupervised learning. It consists of an interaction between an agent and an environment through which the agent learns to achieve RL goals, previously defined. This interaction is two-way communication: first, the environment sends its current state to the agent; then, the agent chooses an action, based on that state, and returns its decision to the environment.

Consequently, actions attempt to modify the environment (i.e., an agent selects the action, and the environment executes it) which changes the previous state and leads to a new one. The new state fixes new environment parameters that the environment itself must reward according to learning objectives. Additionally, these parameters allow the agent to identify a terminal state.

As a result, the agent gets an immediate reward on the last action-state pair and updates its value based on its value function. It is to be noted that the agent's goal is to maximize the reward, and the value function depends on the selected RL algorithm, among which are Dynamic Programming, Monte Carlo, Q-learning, Sarsa, and Dyna Q.

The theory presented in this section is based on the work conducted by Sutton and Barto [21].

2.3.1 Q-learning

Q learning is one of the Temporal Difference algorithms. It learns directly from experience and updates its value function after each interaction, due to it combines the best features of both Monte Carlo and Dynamic Programming methods.

This tabular RL algorithm saves the action-state value estimates in an m by n matrix, where m represents the number of states and n the number of actions, so each position corresponds to an action-state value. After each interaction, the agent updates the corresponding action-state value by using Equation 2.3 [21].

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \cdot [R_{t+1} + \gamma \cdot max_a Q(St + 1, a) - Q(S_t, A_t)] \tag{2.3}$$

Chapter 3

Service Restoration Algorithm

The present chapter describes each block of the Service Restoration Algorithm proposed in this study.

3.1 Overview

The algorithm contains two parts: a training scenario and a production scenario. Both implement Reinforcement Learning and have a co-simulation engine. However, the first one co-simulates with OpenDSS via COM interface and does not have prior knowledge about the distribution network; while, the second one co-simulates with OpenDSS-G through TCP/IP protocol and does know the distribution network dynamics. The co-simulation engine differs from scenarios due to performance. OpenDSS-G adds a graphic user interface which helps to visualize the topology of the distribution network and its changes, but also adds more processing time compared with OpenDSS.

The SR algorithm is written in Python 3.7 and has an Objet Oriented Programming structure. Also, it requires the following input data from the distribution network: lines, switches, the state of the switches, bus voltages, line currents, load powers, and incidence matrix.

According to this, Figure 3.1 presents the general SR algorithm block diagram.

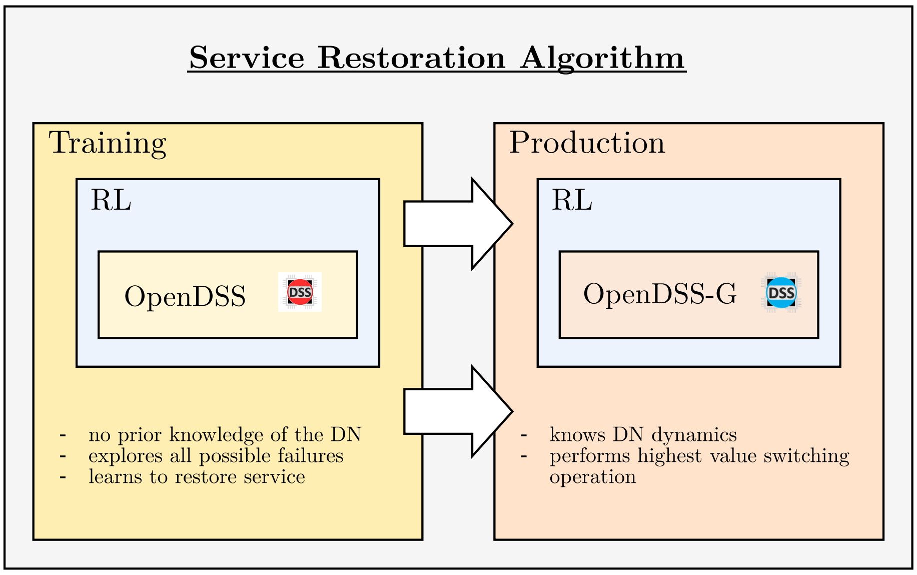

Service Restoration Algorithm
Training
RL
OpenDSS
DSS
- no prior knowledge of the DN
- explores all possible failures
- learns to restore service
Production
RL
OpenDSS-G
DSS
- knows DN dynamics
- performs highest value switching operation

Figure 3.1: RL scenarios

3.2 Reinforcement Learning implementation

Two components support the Reinforcement Learning: the environment and the zone agent. The first one relates to a distribution network and its elements, whereas the second one to a learning agent.

The distribution network is susceptible to failures and switching operations, which modifies its topology and operation values. On the other hand, the zone agent performs Service Restoration on the environment to learn the optimal sequence of switching operations after a failure occurrence.

Figure 3.2 presents the interaction between these two elements, which allows the algorithm to learn and to automate the decision-making process. This interaction is defined in the next section using a finite Markov Decision Process (MDP).

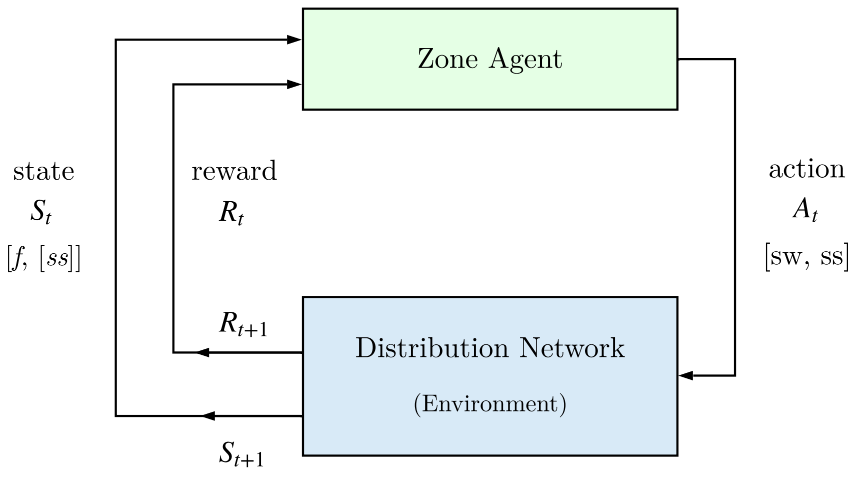


Figure 3.2: MDP block diagram

3.2.1 Markov Decision Process

MDP is a framework that defines the interaction between the agent and the environment in terms of states, actions, and rewards. These terms are defined as follows:

3.2.1.1 MDP Elements

A *state* (s_t) defines how the environment is at a particular time and corresponds to an ordered pair of two elements $[f, [ss]]$. The first element corresponds to a specific failure (f), while the second one is an ordered list of zeros and ones representing the state of switches ($[ss]$). A closed switch takes the value of one, while an open switch takes zero.

Equation 3.1 returns the total number of states $n(S)$, from the numbers of switch status $n(ST)$, switches $n(SW)$, and lines $n(LN)$.

$$n(S) = n(ST)^{n(SW)} * n(LN) \tag{3.1}$$

An *action* (a_t) is a selection, made by the agent, of which switch operates next. However, it does not define the switching operation. The decision if the selected switch opens or closes is determined by the opposite of its status, i.e., the agent chooses a switch instead of a switching operation.

A *policy* (π) tells the agent how to take actions from states. It can be deterministic or stochastic.

A *reward signal* is a numeric value that rates a chosen action, from a given state, based on the Service Restoration constraints, e.g., it assigns a negative number for every violated restriction.

3.2.1.2 MDP Interaction

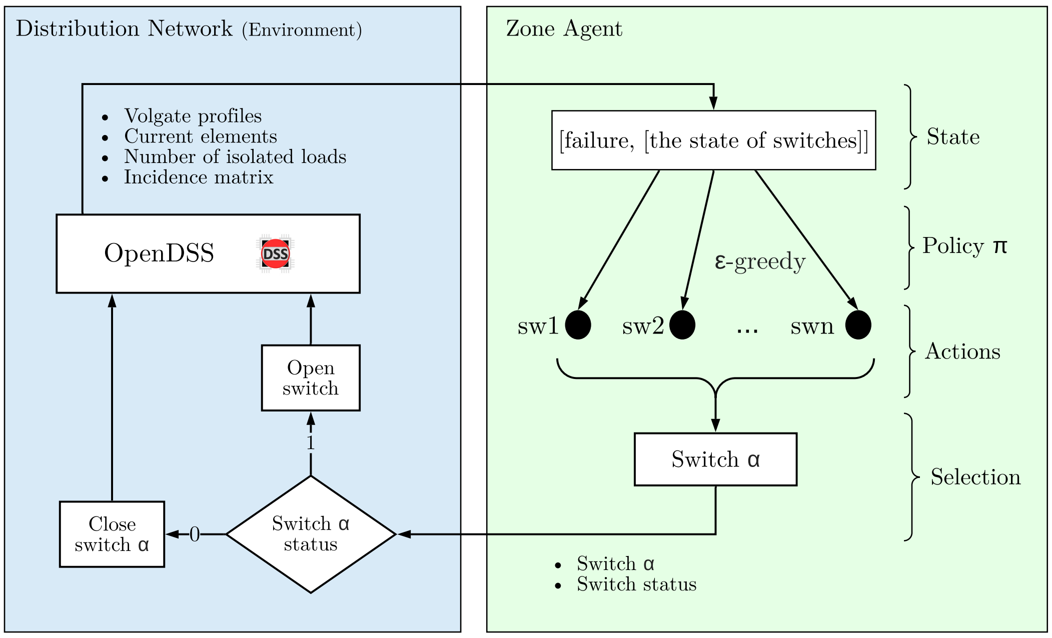


Figure 3.3: MDP block diagram details

A failure occurrence f triggers the interaction between the environment and the agent. The first to initialize is the environment. It gets its current state $[f, [ss]]$ and sends it to the zone agent, who uses it to know which are the possible actions from that state. Secondly, the zone agent starts and chooses its first switch based on the ϵ-greedy policy. And then, it sends the switch to the environment, which checks its status and performs the corresponding switching operation.

After the switching operation, the distribution network reaches the next state, which drives to new network parameters. These parameters include voltage and current profiles, number of isolated loads, and topology. Consequently, the environment compares them with the system constraints to verify their fulfillment and, relying on this, rewards the state-action pair.

The reward depends on the number of violated constraints and their hierarchy. For this reason, the number of loads isolated receives the highest penalty, followed by voltage and current limits, and, lastly, the number of loops within the DN.

To end the first interaction, the agent gets the reward and the next state and uses them to update the value of the previous state-switch pair. The switch-value update depends on the RL algorithm implemented. The present study implements two RL algorithms, which are explained in the following section.

After the action-value update, a new interaction begins, as described above, until the agent reaches the terminal state. As a result, this performs an RL episode. But the agent needs hundreds of episodes to learn an optimal policy.

Figure 3.3 summarized the details for the proposed MDP.

3.2.2 RL episode

The previous section explains the detail of the MDP interaction between the environment and the agent. Each of those interactions corresponds to an episode.

Figure 3.4 shows a block diagram for interaction, but this time from the Service Restoration algorithm perspective.

3.3 OpenDSS Co-simulation

OpenDSS is a fundamental part of the Service Restoration algorithm development. Firstly, it represents the environment in the reinforcement learning model by simu-

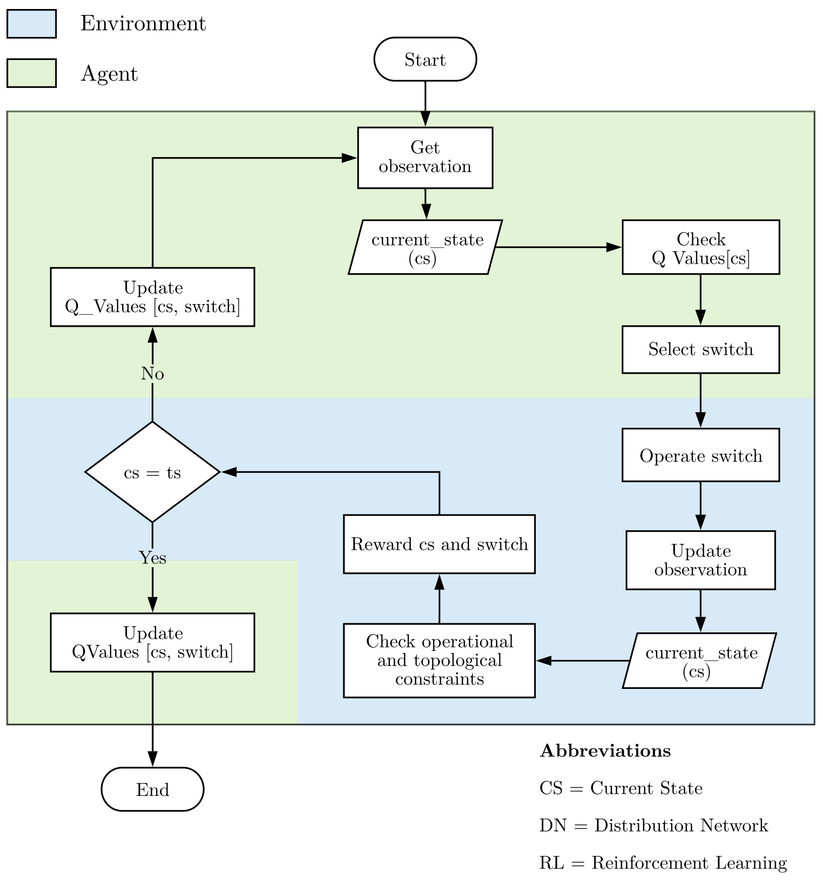


Figure 3.4: Reinforcement Learning Episode

lating the entire distribution network. In this way, it interacts with the agent and performs the switching operations he orders.

On the other hand, it is the analysis tool that performs power flow calculations in each interaction with the agent. It offers voltage and current profiles, the number of isolated loads, and topology information.

The training scenario uses OpenDSS while production uses OpenDSS-G. The connection with the SR algorithm handles COM interface and TCP/IP, respectively.

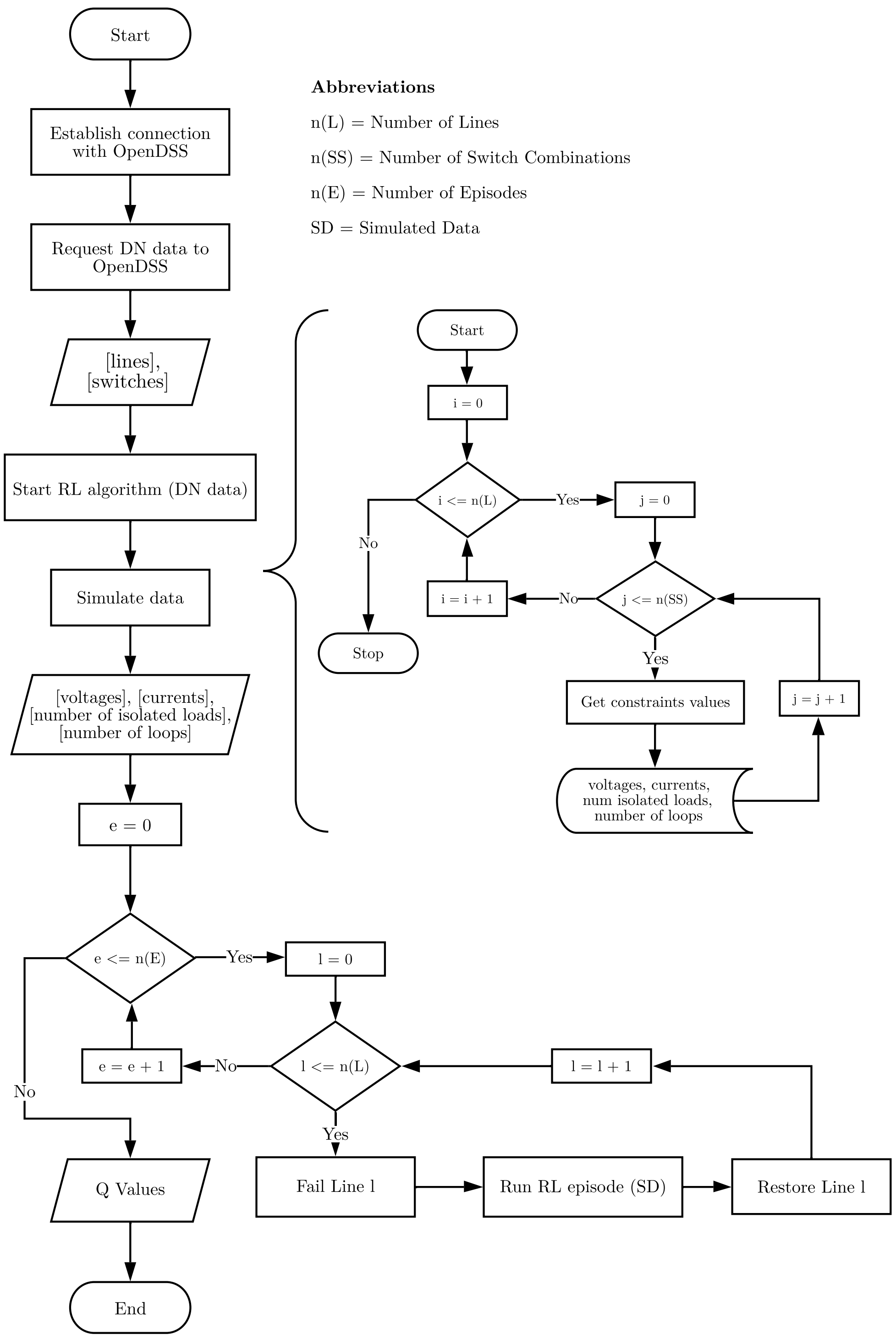


Figure 3.5: SR Algorithm - Training Scenario

3.4 Training scenario

The restoration algorithm starts with Training scenario, which is presented in Figure 3.5. In there, the SR algorithm establishes a connection with OpenDSS and requests the lines and switches data. Then, it initializes the RL class with lines and switches passed as a parameter.

Henceforth, the RL class handles the SR algorithm processes. First, it co-simulates with OpenDSS to obtain the voltage and current profiles, the number of isolated loads, and the number of loops for all possible switch status and faulted line combinations. This simulated data is stored in .ftr files and loaded into the SR algorithm.

Secondly, an episode begins: a line is selected and faulted. So, the RL object performs interactions between the agent and the environment until the environment reaches the terminal state, which is the optimal state after a fault event. Once the terminal state is reached, the RL algorithm restores the faulted line and repeats the aforementioned process by selecting a new line to fail.

This must be done for all lines and during a fixed number of episodes.

As a result, the Q value matrix stores the optimal SR plan, and the training is completed.

3.5 Production scenario

Figure 3.6 shows that Production scenario behaves similarly to Training but with some exceptions. First, the initial connection is carried out with OpenDSS-G. Second, the Q Values matrix has been already computed, so this scenario aims to apply the Service Restoration plan to lead the network to the best possible state after a fault occurrence through self-healing. Third, RL episode runs only once and performs just the necessary switching operations until it reaches the terminal state, instead of switching operations for all possible faults and switch status combinations. And last, this results in a list of actions performed during the SR plan and an updated Q Values matrix.

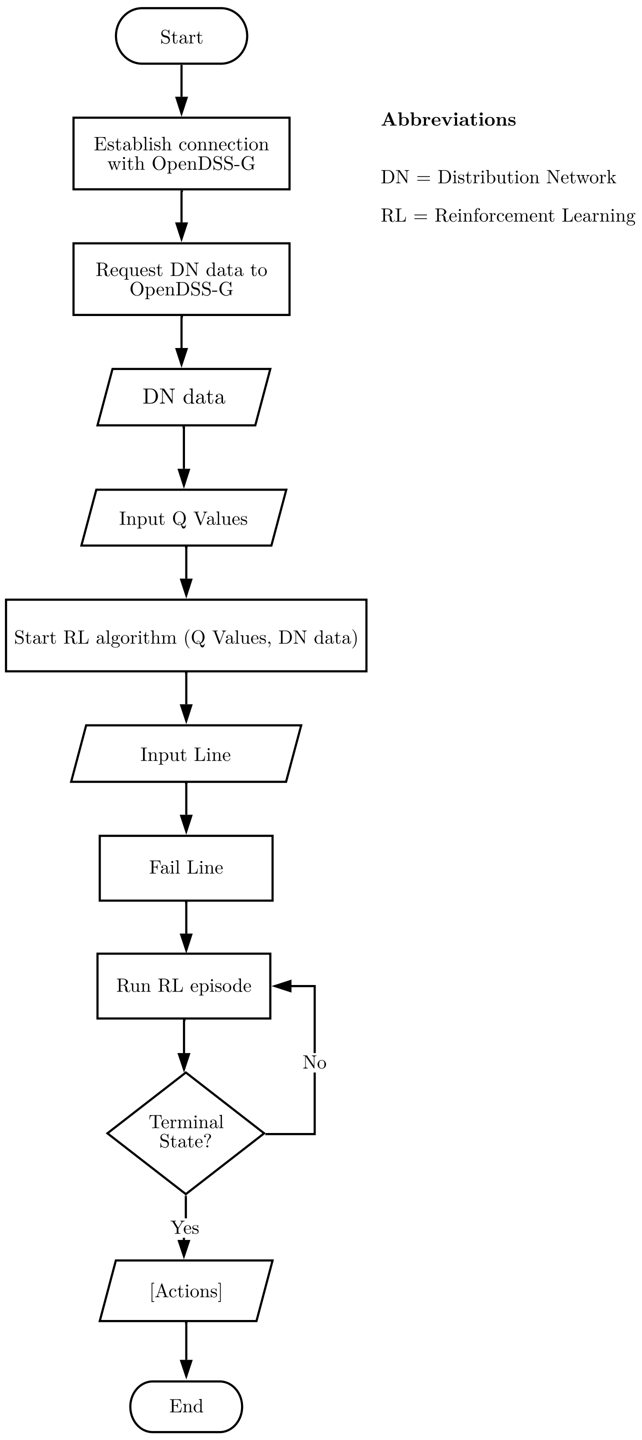

Start
Establish connection with OpenDSS-G
Request DN data to OpenDSS-G
DN data
Input Q Values
Start RL algorithm (Q Values, DN data)
Input Line
Fail Line
Run RL episode
Terminal State?
No
Yes
[Actions]
End
Abbreviations
DN = Distribution Network
RL = Reinforcement Learning

Figure 3.6: SR Algorithm - Production Scenario

Chapter 4

Service Restoration Algorithm Validation

4.1 Overview

As mentioned before in Chapter 2, the Service Restoration algorithm must execute a Training scenario followed by a Production scenario to obtain a restoration plan. Nevertheless, Training and Production can be carried out independently, although the latter requires the Q Values matrix as input data.

To validate the effectiveness and performance of the SR algorithm, two different Distribution Networks provided by the IEEE Power and Energy Society (IEEE-PES) were tested.

In the following sections, the test cases are presented.

4.2 Test Cases

The IEEE Power and Energy Society (IEEE-PES) provides several test cases. The present validation examines and tests the following:

4.2.1 IEEE 33 Node Test Feeder

Figure 4.1 shows the IEEE 33 node test feeder topology and Table 4.1 summaries its parameters.

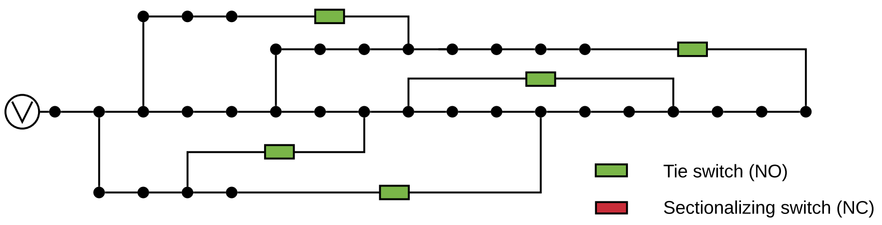


Figure 4.1: IEEE 33 Node Test Feeder from [2]

Table 4.1: IEEE test feeders data

DN Elment	33 Node Test Feeder	123 Node Test Feeder
Lines	32	117
Switches	2 - 5	5 - 15
Loads	32	91

4.2.2 IEEE 123 Node Test Feeder

This system is modified to observe the behavior of the Service Restoration Algorithm and to compare its performance according to the number of switches. For this reason, they have tested ten variants of this test feeder. Each modification increases the number of switches by one, where the minimum is five switches and the maximum is fifteen.

Figure 4.2 shows the IEEE 123 node test feeder topology with 10 switches and Table 4.1 summaries its parameters.

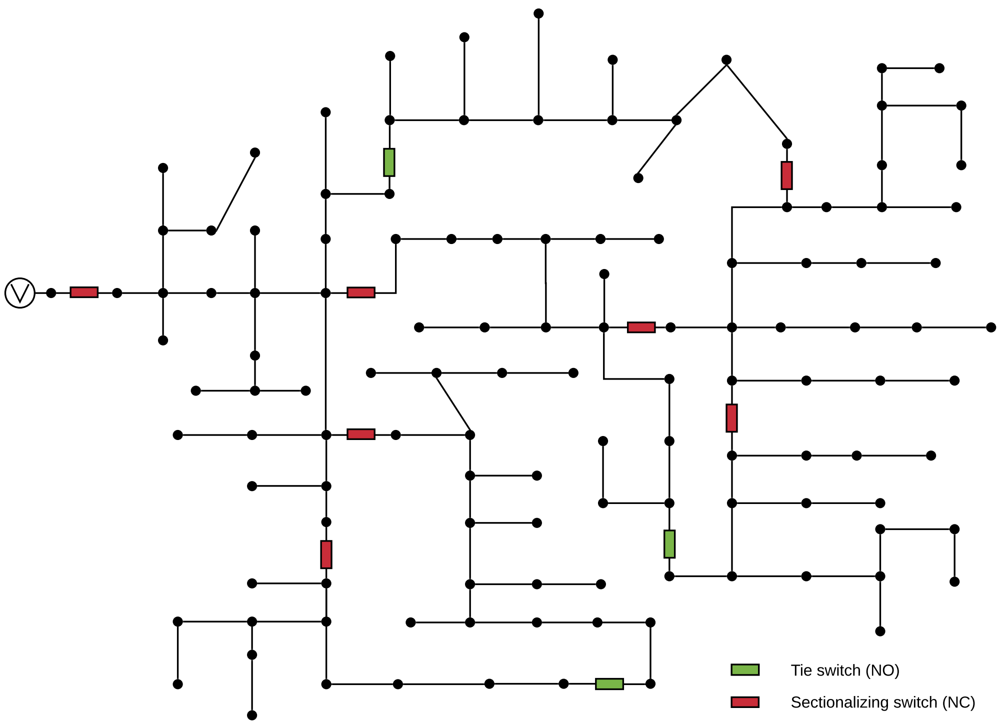


Figure 4.2: IEEE 123 Node Test Feeder

Chapter 5

Service Restoration Algorithm Results

This chapter presents the results of the Service Restoration algorithm proposed in this study for the IEEE test feeders described in the previous Chapter. Firstly, it presents RL parameters obtained from Training scenario. And then, it presents the Service Restoration plans as results of Production scenario.

5.1 RL Training Results

The Service Restoration algorithm obtains the restoration plan through reinforcement learning training, as achieved in Chapter 3. Training scenario turns the network parameters into a Markov Decision Process, which defines the behavior and size of the algorithm operations. Therefore, the numbers of lines and switches of the two test cases presented in Table 4.1 are directly related to the number of states $n(S)$ and actions $n(A)$ presented in the following tables.

Additionally, Tables 5.1, 5.2, and 5.3 show the elapsed time $T[min]$, the number of failures restored $n(FR)$, and the restoration rate RR for each training relying on switch configurations in the IEEE 33 and the IEEE 123 Node Test Feeder.

Table 5.1: IEEE 33 - Training parameters 2sw - 5sw

Data	**IEEE 33 Node Test Feeder**			
	2sw	**3sw**	**4sw**	**5sw**
n(S)	128	256	512	1024
n(A)	12	13	14	15
T [min]	1.3	2.9	5.6	10
n(FR)	20	28	31	31
RR	1	1	1	1

Table 5.2: IEEE 123 - Training parameters 5sw - 10sw

Data	**IEEE 123 Node Test Feeder**					
	5sw	**6sw**	**7sw**	**8sw**	**9sw**	**10sw**
n(S)	3.744	7.588	14.976	29.952	59.904	119.808
n(A)	5	6	7	8	9	10
T [min]	7.3	8.4	11.0	15.5	17.7	41.3
n(FR)	0	0	14	22	22	41
RR	0	0	1	1	1	1

Table 5.3: IEEE 123 - Training parameters 11sw - 15sw

Data	**IEEE 123 Node Test Feeder**				
	11sw	**12sw**	**13sw**	**14sw**	**15sw**
n(S)	239.616	479.232	958.464	1.916.928	3.833.856
n(A)	11	12	13	14	15
T [min]	72.5	145.0	290.3	602.4	1198.2
n(FR)	43	45	45	46	51
RR	1	1	1	1	1

5.2 Service Restoration Plans

The Service Restoration Plan specifies the optimal switching operations to restore the maximum number of loads maintaining the operational and topological constraints after a fault occurs. The execution of this plan is carried out in the Production scenario, that is, once the system is already trained.

Tables 5.4 and 5.5 present the restoration plans for IEEE 33 Node Test Feeder with 5 switches and IEEE 123 Node Test Feeder with 10 switches, respectively. Additionally, they contain the number of actions required $n(A)$, the processing time (T[s]), the number of isolated loads before $n(IL)_{pre}$ and after $n(IL)_{post}$ restoration, and the restoration rate (RR). The latter is calculated as follows: $RR = \frac{n(IL)_{prev} - n(IL)_{post}}{n(IL)_{prev}}$.

The Service Restoration Plans for the other switch configurations of the test cases are presented in Appendix A.

Table 5.4: SR plan - IEEE 33 Node Test Feeder - 5 switches

Faulted line	**Actions**	**N(A)**	**T [s]**	**N(IL)$_{pre}$**	**N(IL)$_{post}$**	**RR**
l1	[('s33', None)]	0	0.005	32	32	0
l2	[('s35', 1)]	1	0.005	26	0	1
l3	[('s37', 1)]	1	0.005	22	0	1
l4	[('s37', 1)]	1	0.005	21	0	1
l5	[('s37', 1)]	1	0.005	20	0	1
l6	[('s36', 1)]	1	0.005	12	0	1
l7	[('s36', 1)]	1	0.004	11	0	1
l8	[('s36', 1)]	1	0.005	10	0	1
l9	[('s36', 1)]	1	0.004	9	0	1
l10	[('s36', 1)]	1	0.004	8	0	1
l11	[('s36', 1)]	1	0.005	7	0	1
l12	[('s36', 1)]	1	0.005	6	0	1
l13	[('s36', 1)]	1	0.004	5	0	1
l14	[('s36', 1)]	1	0.005	4	0	1
l15	[('s36', 1)]	1	0.005	3	0	1
l16	[('s36', 1)]	1	0.004	2	0	1
l17	[('s36', 1)]	1	0.004	1	0	1
l18	[('s35', 1)]	1	0.005	5	0	1
l19	[('s35', 1)]	1	0.005	4	0	1
l20	[('s35', 1)]	1	0.005	2	0	1
l21	[('s35', 1)]	1	0.005	1	0	1
l22	[('s37', 1)]	1	0.005	3	0	1
l23	[('s37', 1)]	1	0.005	2	0	1
l24	[('s37', 1)]	1	0.005	1	0	1
l25	[('s37', 1)]	1	0.004	7	0	1
l26	[('s37', 1)]	1	0.005	6	0	1
l27	[('s37', 1)]	1	0.005	5	0	1
l28	[('s37', 1)]	1	0.005	4	0	1
l29	[('s36', 1)]	1	0.005	3	0	1
l30	[('s36', 1)]	1	0.005	3	0	1
l31	[('s36', 1)]	1	0.005	2	0	1
l32	[('s36', 1)]	1	0.004	1	0	1

Table 5.5: SR plan - IEEE 123 Node Test Feeder - 10 switches

Faulted line	**Actions**	**N(A)**	**T [s]**	**N(IL)**$_{pre}$	**N(IL)**$_{post}$	**RR**
l116	[('sw7', 1)]	1	0.016	52	0	1
l52	[('sw7', 1)]	1	0.016	51	0	1
l53	[('sw7', 1)]	1	0.017	50	0	1
l55	[('sw7', 1)]	1	0.016	48	0	1
l58	[('sw7', 1)]	1	0.018	46	0	1
l117	[('sw7', 1)]	1	0.016	38	0	1
l67	[('sw7', 1)]	1	0.017	21	0	1
l73	[('sw7', 1)]	1	0.016	18	0	1
l77	[('sw7', 1)]	1	0.017	8	0	1
l86	[('sw7', 1)]	1	0.016	7	0	1
l88	[('sw7', 1)]	1	0.017	5	0	1
l90	[('sw7', 1)]	1	0.016	4	0	1
l92	[('sw7', 1)]	1	0.017	3	0	1
l94	[('sw7', 1)]	1	0.016	2	0	1
l17	[('sw7', 1)]	1	0.017	1	0	1
l34	[('sw7', 1)]	1	0.018	2	0	1
l68	[('sw10', 1)]	1	0.016	13	0	1
l118	[('sw10', 1)]	1	0.016	10	0	1
l101	[('sw10', 1)]	1	0.016	7	0	1
l105	[('sw10', 1)]	1	0.016	5	0	1
l63	[('sw10', 1)]	1	0.017	5	0	1
l62	[('sw10', 1)]	1	0.016	6	0	1
l61	[('sw10', 1)]	1	0.017	7	0	1
l114	[('sw9', 1)]	1	0.017	16	0	1
l36	[('sw9', 1)]	1	0.016	12	0	1
l41	[('sw9', 1)]	1	0.016	11	0	1
l43	[('sw9', 1)]	1	0.018	9	0	1
l45	[('sw9', 1)]	1	0.017	7	0	1
l48	[('sw9', 1)]	1	0.016	5	0	1
l49	[('sw9', 1)]	1	0.020	2	0	1
l50	[('sw9', 1)]	1	0.017	1	0	1
l31	[('sw9', 1)]	1	0.017	1	0	1
l30	[('sw9', 1)]	1	0.017	2	0	1
l26	[('sw9', 1)]	1	0.017	3	0	1
l24	[('sw9', 1)]	1	0.017	6	0	1
l22	[('sw9', 1), ('sw6', 0)]	2	0.018	7	0	1
l19	[('sw9', 1)]	1	0.017	8	0	1
l12	[('sw7', 1)]	1	0.018	3	0	1
l115	[('sw5', None)]	0	0.017	91	91	0
l5	[('sw7', None)]	0	0.016	2	2	0
l1	[('sw3', None)]	0	0.016	1	1	0

Chapter 6

Discussion and Conclusion

6.1 Results Discussions

The results obtained in the previous Chapter show that Service Restoration depends on the availability of transferring out-of-service un-faulted loads to healthy feeders, which in turn depends on the number and location of switches, i.e., if the location and number of switches are conducive, the SR algorithm can resupply all out-of-service un-faulted loads.

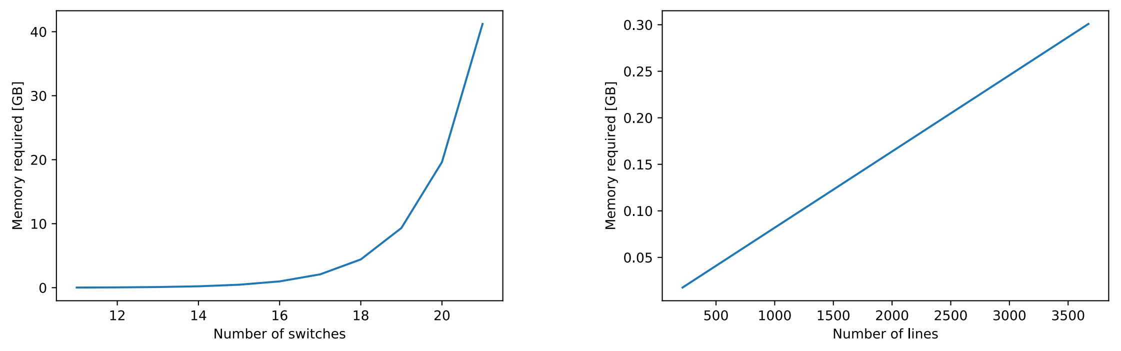
Therefore, the SR algorithm obtained the optimal switch operations sequence for failure scenarios in which there was the possibility of transferring isolated loads to healthy feeders, and for those scenarios it restored all loads.

The optimal sequence is obtained in the Training scenario but constantly updated in the Production scenario. Consequently, the RL algorithm requires retraining when system variations occur that increase or reduce the numbers of lines or switches, but retraining is not required when these changes affect the number of loads.

6.1.1 SR Algorithm performance

The performance of this algorithm relies on computational capacity: memory size and processing.

First, the RL model dictates the memory size required. This is because of the model needs to compute a Q Values matrix, whose size depends on the numbers of actions and states and, in turn, on the numbers of lines and switches, as explained in Chapter 3. Accordingly, Figure 6.1 shows the amount of memory needed under the numbers of lines and switches for the IEEE 123 Node Test Feeder with 10 switches. As a result, memory size grows linearly with an increase in the number of lines and exponentially with an increase in the number of switches.



(a) Memory size according to number of switches

(b) Memory size according to number of lines

Figure 6.1: Memory size comparison according to numbers of lines and switches variations for the IEEE 123 Node Test Feeder - 10 switches

Since the number of switches is the factor that most affects the size of the model, Figure 6.2 presents the required processing time and the number of states according to the number of switches for the IEEE 123 Node Test Feeder.

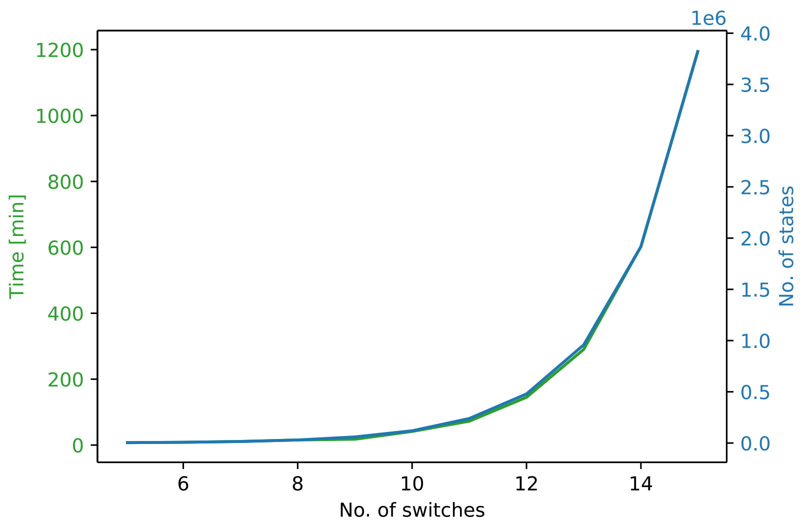


Figure 6.2: No. of switches vs Time [min] and No. of states - IEEE 123 Node Test Feeder

6.2 Conclusion

This thesis aimed to present the development of a Service Restoration Algorithm implementing Reinforcement Learning techniques.

The reviewed literature in Chapter 2 emphasized the need for improving Service Restoration methodologies by developing new approaches that endow Distribution Networks with a self-healing capacity to resupply automatically and intelligently out-of-service un-faulted customers after a fault occurrence.

Chapter 3 explains the Reinforcement Learning technique implemented throughout modeling a Markov Decision Process that captures the Distribution Network parameters, the multiobjective optimization function, and the operational and topological constraints. Furthermore, it states the roles of the co-simulation engine and how it is integrated into the algorithm. As a result, Chapter 3 proposes an algorithm structure that divides the Service Restoration algorithm into Training and Production scenarios, both implementing an RL technique and a co-simulation engine.

Lastly, Chapter 4 defines the SR algorithm validation including test cases and results. The test cases compromise both IEEE 33 bus test feeder and IEEE 123 bus test feeder. In this way, the Service Restoration algorithm was tested for each of the test cases, thus obtaining a restoration plan for a fault under each of its lines.

The service restoration plan obtained for the IEEE 33 Node Test Feeder and the IEEE 123 Node Test Feeder successfully resupply 100% of the out-of-service unfaulted customers, as described by the restoration rate presented in Tables 5.4 and 5.5.

In conclusion, this SR algorithm proposed is capable of self-learning to obtain an optimal solution that fulfills operational and topological constraints. Furthermore, it establishes a new approach in service restoration methodologies by implementing a Reinforcement Learning technique.

References

[1] US Department of Energy, “Distribution Automation: Results from the SGIG Program,” tech. rep., 2016.

[2] A. Landeros, S. Koziel, and M. F. Abdel-Fattah, “Distribution network reconfiguration using feasibility-preserving evolutionary optimization,” *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 3, pp. 589–598, 2019.

[3] J. E. engineer) Liu, X. Dong, X. Chen, X. Tong, X. Zhang, and S. Xu, *Fault location and service restoration for electrical distribution systems.* 2016.

[4] A. Zidan, M. Khairalla, A. M. Abdrabou, T. Khalifa, K. Shaban, A. Abdrabou, R. El Shatshat, and A. M. Gaouda, “Fault Detection, Isolation, and Service Restoration in Distribution Systems: State-of-the-Art and Future Trends,” *IEEE Transactions on Smart Grid*, vol. 8, pp. 2170–2185, sep 2017.

[5] A. Kavousi-Fard and T. Niknam, “Optimal distribution feeder reconfiguration for reliability improvement considering uncertainty,” *IEEE Transactions on Power Delivery*, vol. 29, no. 3, pp. 1344–1353, 2014.

[6] C. Angelo and P. Selejan, “Technologies of the self healing grid,” *IET Conference Publications*, vol. 2013, no. 615 CP, pp. 10–13, 2013.

[7] V. Madani, R. Das, F. Aminifar, J. McDonald, S. S. Venkata, D. Novosel, A. Bose, and M. Shahidehpour, “Distribution Automation Strategies Challenges

and Opportunities in a Changing Landscape," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 2157–2165, 2015.

[8] U.S. Department of Energy, "Smart Grid System Report 2018: Report to Congress," Tech. Rep. November, 2018.

[9] F. Shen, Q. Wu, S. Huang, J. C. Lopez, C. Li, and B. Zhou, "Review of Service Restoration Methods in Distribution Networks," *Proceedings - 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2018*, 2018.

[10] M. Gholami, J. Moshtagh, and N. Ghadernejad, "Service restoration in distribution networks using combination of two heuristic methods considering load shedding," *Journal of Modern Power Systems and Clean Energy*, vol. 3, no. 4, pp. 556–564, 2015.

[11] D. S. Sanches, J. B. A. London Junior, and A. C. B. Delbem, "Multi-Objective Evolutionary Algorithm for single and multiple fault service restoration in large-scale distribution systems," *Electric Power Systems Research*, vol. 110, pp. 144–153, 2014.

[12] N. A. Latare, S. S. Bhat, and I. Srivastava, "Literature review of service restoration in distribution system," *Proceedings of the 2017 2nd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2017*, 2017.

[13] A. E. Abu-Elanien, M. M. Salama, and K. B. Shaban, "Modern network reconfiguration techniques for service restoration in distribution systems: A step to a smarter grid," *Alexandria Engineering Journal*, vol. 57, no. 4, pp. 3959–3967, 2018.

[14] S. Yao, T. Zhao, P. Wang, and H. Zhang, “Resilience-oriented distribution system reconfiguration for service restoration considering distributed generations,” *IEEE Power and Energy Society General Meeting*, vol. 2018-Janua, pp. 1–5, 2018.

[15] T. Yokoyama and T. Nagata, “A multi-agent restoration method for distribution network,” *Energy Procedia*, vol. 14, pp. 726–731, 2012.

[16] C. Koch-Ciobotaru, M. Monadi, A. Luna, and P. Rodriguez, “Distributed FLISR algorithm for smart grid self-reconfiguration based on IEC61850,” *3rd International Conference on Renewable Energy Research and Applications, ICRERA 2014*, pp. 418–423, 2014.

[17] A. Zidan and E. F. El-Saadany, “A cooperative multiagent framework for self-healing mechanisms in distribution systems,” *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1525–1539, 2012.

[18] T. Yip, J. Wang, B. Xu, K. Fan, and T. Li, “Fast self-healing control of faults in MV networks using distributed intelligence,” *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, pp. 1131–1133, 2017.

[19] I. Chellaswamy and J. V. S. Rani, “Automatic fault isolation and restoration of distribution system using JADE based Multi-Agents,” *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 27, no. 3, pp. 2226–2242, 2019.

[20] X. Huang, Y. Yang, and G. A. Taylor, “Service restoration of distribution systems under distributed generation scenarios,” *CSEE Journal of Power and Energy Systems*, vol. 2, no. 3, pp. 43–50, 2016.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction.* IGI Global, 2014.

Appendix A

Additional Results

A.1 Service Resotoration Plans

Table A.1: SR plan for IEEE 123 Node Test Feeder with 7 switches

Faulted line	**Actions**	**n(A)**	**T** [s]	**n(IL)**$_{pre}$	**n(IL)**$_{post}$	**RR**
l19	[('sw9', 1)]	1	0.015	8	0	1
l22	[('sw9', 1)]	1	0.016	7	0	1
l24	[('sw9', 1)]	1	0.016	6	0	1
l26	[('sw9', 1)]	1	0.015	3	0	1
l30	[('sw9', 1)]	1	0.017	2	0	1
l31	[('sw9', 1)]	1	0.016	1	0	1
l50	[('sw9', 1)]	1	0.017	1	0	1
l49	[('sw9', 1)]	1	0.018	2	0	1
l48	[('sw9', 1)]	1	0.016	5	0	1
l45	[('sw9', 1)]	1	0.016	7	0	1
l43	[('sw9', 1)]	1	0.016	9	0	1
l41	[('sw9', 1)]	1	0.017	11	0	1
l36	[('sw9', 1)]	1	0.017	12	0	1
l114	[('sw9', 1)]	1	0.017	16	0	1

Table A.2: SR plan for IEEE 123 Node Test Feeder with 8 switches

Faulted line	**Actions**	**n(A)**	**T [s]**	**n(IL)$_{pre}$**	**n(IL)$_{post}$**	**RR**
l117	[('sw10', 1)]	1	0.015	38	0	1
l68	[('sw10', 1)]	1	0.016	13	0	1
l118	[('sw10', 1)]	1	0.016	10	0	1
l101	[('sw10', 1)]	1	0.016	7	0	1
l105	[('sw10', 1)]	1	0.016	5	0	1
l63	[('sw10', 1)]	1	0.016	5	0	1
l62	[('sw10', 1)]	1	0.016	6	0	1
l61	[('sw10', 1)]	1	0.017	7	0	1
l114	[('sw9', 1)]	1	0.015	16	0	1
l36	[('sw9', 1)]	1	0.016	12	0	1
l41	[('sw9', 1)]	1	0.017	11	0	1
l43	[('sw9', 1)]	1	0.017	9	0	1
l45	[('sw9', 1)]	1	0.017	7	0	1
l48	[('sw9', 1)]	1	0.016	5	0	1
l49	[('sw9', 1)]	1	0.017	2	0	1
l50	[('sw9', 1)]	1	0.017	1	0	1
l31	[('sw9', 1)]	1	0.016	1	0	1
l30	[('sw9', 1)]	1	0.016	2	0	1
l26	[('sw9', 1)]	1	0.018	3	0	1
l24	[('sw9', 1)]	1	0.016	6	0	1
l22	[('sw9', 1)]	1	0.016	7	0	1
l19	[('sw9', 1)]	1	0.016	8	0	1

Table A.3: SR plan for IEEE 123 Node Test Feeder with 9 switches

Faulted line	**Actions**	**n(A)**	**T [s]**	**n(IL)$_{pre}$**	**n(IL)$_{post}$**	**RR**
l68	[('sw10', 1)]	1	0.017	13	0	13
l118	[('sw10', 1)]	1	0.018	10	0	10
l101	[('sw10', 1)]	1	0.016	7	0	7
l105	[('sw10', 1)]	1	0.016	5	0	5
l63	[('sw10', 1)]	1	0.016	5	0	5
l62	[('sw10', 1)]	1	0.017	6	0	6
l61	[('sw10', 1)]	1	0.017	7	0	7
l117	[('sw10', 1)]	1	0.016	38	0	38
l19	[('sw9', 1)]	1	0.016	8	0	8
l22	[('sw9', 1)]	1	0.015	7	0	7
l24	[('sw9', 1)]	1	0.016	6	0	6
l26	[('sw9', 1)]	1	0.016	3	0	3
l30	[('sw9', 1)]	1	0.017	2	0	2
l31	[('sw9', 1)]	1	0.015	1	0	1
l50	[('sw9', 1)]	1	0.016	1	0	1
l49	[('sw9', 1)]	1	0.017	2	0	2
l48	[('sw9', 1)]	1	0.017	5	0	5
l45	[('sw9', 1)]	1	0.017	7	0	7
l43	[('sw9', 1)]	1	0.016	9	0	9
l41	[('sw9', 1)]	1	0.017	11	0	11
l36	[('sw9', 1)]	1	0.016	12	0	12
l114	[('sw9', 1)]	1	0.015	16	0	16

Table A.4: SR plan for IEEE 123 Node Test Feeder with 15 switches - part 1

Faulted line	**Actions**	**n(A)**	**T [s]**	**n(IL)**	**n(IL)**	**RR**
l2	[('sw15', 1)]	1	0.018	3	0	1
l5	[('sw15', 1)]	1	0.017	2	0	1
l6	[('sw15', 1)]	1	0.018	1	0	1
l3	[('sw15', 1)]	1	0.016	86	0	1
l7	[('sw15', 1)]	1	0.016	85	0	1
l9	[('sw14', 1)]	1	0.017	3	0	1
l11	[('sw14', 1)]	1	0.018	2	0	1
l14	[('sw14', 1)]	1	0.016	1	0	1
l20	[('sw14', 1)]	1	0.017	1	0	1
l18	[('sw14', 1)]	1	0.017	2	0	1
l13	[('sw14', 1), ('sw3', 0), ('sw9', 1)]	3	0.019	26	9	0.7
l12	[('sw15', 1)]	1	0.022	3	0	1
l34	[('sw15', 1)]	1	0.018	2	0	1
l17	[('sw7', 1)]	1	0.017	1	0	1
l90	[('sw7', 1)]	1	0.017	4	0	1
l88	[('sw7', 1)]	1	0.017	5	0	1
l86	[('sw7', 1)]	1	0.017	7	0	1
l77	[('sw7', 1)]	1	0.018	8	0	1
l73	[('sw7', 1)]	1	0.017	18	0	1
l67	[('sw7', 1)]	1	0.017	21	0	1
l68	[('sw10', 1)]	1	0.016	13	0	1
l118	[('sw10', 1)]	1	0.017	10	0	1
l101	[('sw10', 1)]	1	0.016	7	0	1
l105	[('sw10', 1)]	1	0.017	5	0	1
l63	[('sw13', 1)]	1	0.017	5	0	1
l62	[('sw13', 1)]	1	0.018	6	0	1
l61	[('sw13', 1)]	1	0.017	7	0	1
l58	[('sw13', 1)]	1	0.022	46	0	1
l55	[('sw13', 1)]	1	0.017	48	0	1
l53	[('sw13', 1)]	1	0.016	50	0	1
l52	[('sw13', 1)]	1	0.017	51	0	1
l116	[('sw13', 1)]	1	0.017	52	0	1
l64	[('sw13', 1)]	1	0.016	4	0	1
l65	[('sw13', 1)]	1	0.017	1	0	1
l40	[('sw13', 1)]	1	0.017	1	0	1
l41	[('sw9', 1)]	1	0.016	11	0	1
l43	[('sw9', 1)]	1	0.017	9	0	1
l45	[('sw9', 1)]	1	0.017	7	0	1
l48	[('sw9', 1)]	1	0.017	5	0	1

Table A.5: SR plan for IEEE 123 Node Test Feeder with 15 switches - part 2

Faulted line	**Actions**	**n(A)**	**T [s]**	**n(IL)**	**n(IL)**	**RR**
l49	[('sw9', 1)]	1	0.017	2	0	1
l50	[('sw9', 1)]	1	0.017	1	0	1
l31	[('sw9', 1)]	1	0.017	1	0	1
l30	[('sw9', 1)]	1	0.019	2	0	1
l26	[('sw9', 1)]	1	0.017	3	0	1
l24	[('sw9', 1)]	1	0.017	6	0	1
l22	[('sw9', 1)]	1	0.017	7	0	1
l36	[('sw9', 1)]	1	0.017	12	0	1
l114	[('sw9', 1)]	1	0.016	16	0	1
l117	[('sw10', 1)]	1	0.017	38	0	1
l19	[('sw9', 1)]	1	0.017	8	0	1
l10	[('sw15', 1)]	1	0.016	81	0	1