2017

A Multiagent Q-learning-based Restoration Algorithm for Resilient Distribution System Operation

Jungseok Hong
University of Central Florida

A MULTIAGENT Q-LEARNING-BASED RESTORATION ALGORITHM FOR RESILIENT DISTRIBUTION SYSTEM OPERATION

by

JUNGSEOK HONG
B.S. South Dakota State University, 2015
B.S. Sung Kyun Kwan University, 2015

A thesis submitted in partial fulfilment of the requirements
for the degree of Master of Science
in the Department of Electrical and Computer Engineering
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2017

ABSTRACT

Natural disasters, human errors, and technical issues have caused disastrous blackouts to power systems and resulted in enormous economic losses. Moreover, distributed energy resources have been integrated into distribution systems, which bring extra uncertainty and challenges to system restoration. Therefore, the restoration of power distribution systems requires more efficient and effective methods to provide resilient operation.

In the literature, using Q-learning and multiagent system (MAS) to restore power systems has the limitation in real system application, without considering power system operation constraints. In order to adapt to system condition changes quickly, a restoration algorithm using Q-learning and MAS, together with the combination method and battery algorithm is proposed in this study. The developed algorithm considers voltage and current constraints while finding system switching configuration to maximize the load pick-up after faults happen to the given system. The algorithm consists of three parts. First, it finds switching configurations using Q-learning. Second, the combination algorithm works as a back-up plan in case of the solution from Q-learning violates system constraints. Third, the battery algorithm is applied to determine the charging or discharging schedule of battery systems. The obtained switching configuration provides restoration solutions without violating system constraints. Furthermore, the algorithm can adjust switching configurations after the restoration. For example, when renewable output changes, the algorithm provides an adjusted solution to avoid violating system constraints.

The proposed algorithm has been tested in the modified IEEE 9-bus system using the real-time digital simulator. Simulation results demonstrate that the algorithm offers an efficient and effective restoration strategy for resilient distribution system operation.

I dedicate this thesis work to my parents Euiin Hong and Haeran Jung who always encourage and support me throughout my school years with their best. I also dedicate this work to my brother Jungho Hong, my aunt Haehi Daud, and my uncle Hary Daud for their continuous support.

I also dedicate this work to my friend Insuk Soh for his contiuous support.

ACKNOWLEDGMENTS

I express my heartfelt gratitude to Dr. Wei Sun for his invaluable and great advice and supports from my undergraduate years. Without his advice and supports, I couldn't accomplish this work. Dr. Sun always encourages me to explore new topics and not to hesitate to use new approaches. I was able to expand my research skills and knowledge with his guidance.

I would like to thank Dr. Qun Zhou and Dr. Qipeng Zheng for their valuable time and advice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Background

Natural Disaster and Blackouts

The stable supply of energy is one of the important issues in power systems. Numerous studies and projects have been conducted to resolve this issue. Despite all these efforts, blackouts are unavoidable due to various threats. The threats can be classified as natural, accidental, malicious, and emerging threats as shown in Fig 1.1. Natural threats are from natural disasters. Accidental threats include operational fault and equipment failure. Malicious threats consist of physical threat, human threat, and cyber threat. Emerging threats include systemic threat and impact from other infrastructures [3].

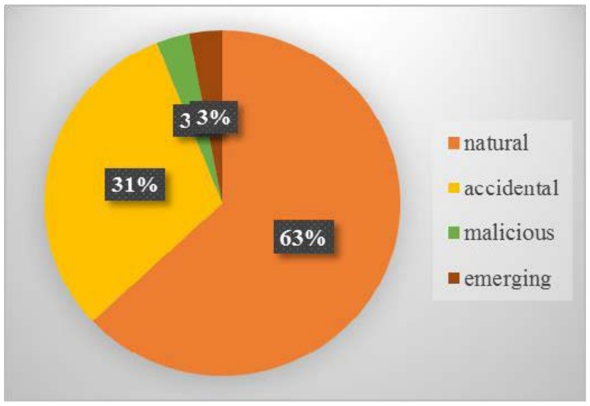


Figure 1.1: Percentages of the causes for historic blackouts

Although natural disasters can not be avoided, preparation is a must to supply electricity in a stable manner. However, modern grids have limited availability to apply the proactive plans for improving the resilience of the grid due to their old infrastructures such as communication, transportation, etc.

Distribution System

An electric power system consists of major components including transmission system, subtransmission system, substation, distribution system, and feeders. Distribution system delivers power to customers and its goal is providing reliable transfer of electricity to customers [5]. Distribution systems typically consist of the distribution substation connected by one or more subtransmission lines. Each distribution substation will supply single or multiple feeders. There are various standard distribution voltage levels. They are in medium voltage and low voltage range. Some of the popular ones are 34.5kV, 23. 9kV, 14.4kV, 13.2kV, 12.47kV, and 4.16kV [4].

In the late twentieth century, most challenges came from transmission systems and interconnected system between them. During last decade, there has been a large development in renewables, batteries, electric vehicles (EVs) and distributed generators (DGs) [4]. The increased penetration level of renewables has brought concerns about the reliability of power system operation and restoration. Integrating energy storage devices such as batteries can improve the reliability of power systems and minimize the restoration time [6].

Microgrid

Microgrid is part of low voltage distribution systems with distributed energy resources (DERs) including DGs, photovoltaic (PV) system, fuel cells, etc. with storage devices such as batteries. When it is interconnected to a main grid, it can be operated in a non-autonomous way. If it is disconnected from the main grid, it can be operated in an autonomous way [7]. Due to the features mentioned, microgrid has the ability to separate and isolate the system itself from the grid in case of blackouts [8]. It enables high penetration of DERs without rebuilding current distribution systems [9]. Furthermore, the microgrid can aid restoration process of distribution system with its surplus power.

Power System Restoration

As mentioned above, the blackout is likely to occur due to various causes such as natural disaster, human error, etc. Therefore, it is provident to prepare and plan restoration after the blackout or faults. The power system's frequency fluctuation is controlled automatically by load shedding, controlled separation, and isolation mechanisms. However, the success rate of the automatic restoration algorithm is about 50%. The challenge is coordinating the control and protective mechanisms with the operation of the power plants and the power system [1]. Figure 1.2 shows general steps for power system restoration.

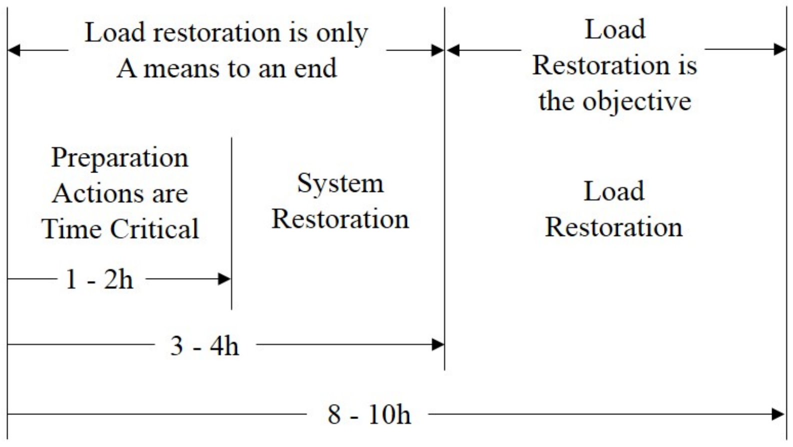

Load restoration is only A means to an end
Load Restoration is the objective
Preparation Actions are Time Critical
System Restoration
Load Restoration
1 - 2h
3 - 4h
8 - 10h

Figure 1.2: General Restoration Stages [1]

Reinforcement Learning

The foundation of all theories related to learning and intelligence is learning from interaction. Among those theories, reinforcement learning is that each agent is learning how to act in each situation in order to maximize the total numerical reward of each agent. There are two main approaches for reinforcement learning, indirect learning, and direct learning. Indirect learning is estimating an explicit model of the environment to learn so they can learn faster compared to direct learning. Direct learning is learning the optimal control policy without an explicit model, and therefore direct learning is not affected by bad models. An agent in an environment takes an action and is given reward and observation from the environment as shown in Fig 1.3. Reinforcement learning aims to maximize the total reward of each agent through a learning process. It has been applied to several fields including control, business management, power systems, etc. [2, 10, 11].

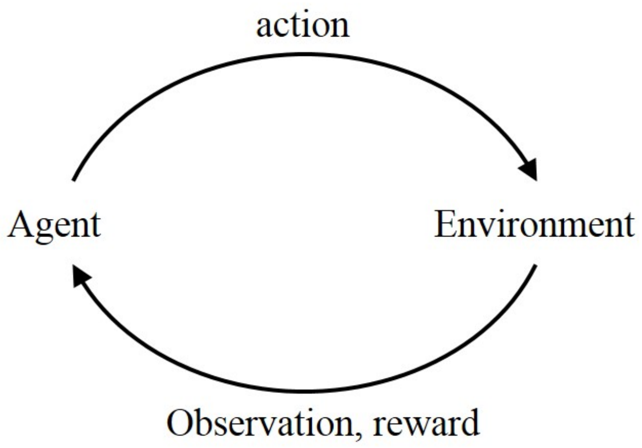


Figure 1.3: Reinforcement Learning [2]

Motivation

Due to the newly introduced technologies to power systems, there are emerging challenges for the power system restoration. The power systems restoration studies based on MAS and Q-learning mostly focused on reinforcement learning and theoretical development without power system physics. There is a need of studying power system restoration with considering both reinforcement learning and power systems perspectives.

Outline of Thesis

This thesis has been organized as follows:
Chapter 1 presents the background of blackouts, distribution system, microgrid, power system restoration, and reinforcement learning followed by the motivation of this study. Chapter 2 presents a literature review and the theory of power system restoration, reinforcement learning, and multiagent. Chapter 3 presents the role of each agent, reward design, system constraints, and restoration algorithm using Q-learning, reduced combination algorithm, and battery algorithm followed by post-restoration algorithm. Chapter 4 presents a testbed and the simulation results using Matlab and Simulink. Finally, Chapter 5 presents conclusions with future work.

CHAPTER 2: LITERATURE REVIEW AND THEORY

Literature Review

Power System Restoration

There have been several studies of power system restoration. Adibi et al. [12] provided an overview of restoration plans with considering power system characteristics and reactive power balance. Gutierrez et al. [13] listed restoration plans in three steps: first of all, isolate the areas and zones which have faults. Secondly, run black-start unit and energize the zones and areas. Thirdly, feed thermal plant auxiliaries and synchronize zones. Lastly, reintegrate the areas and zones into the bulk network. Ancona [14] proposed a framework for power system restoration. The paper specified the goals and objectives of restoration. Also, it listed the sequence of restoration actions: start restoration, prepare initial cranking source, prepare restoration path, and build stable subsystem.

First of all, faults are located and following methods have been studied to locate faults: an expert system [15, 16, 17, 18], multiagent-based distribution automation system [19], genetic algorithm [20], etc. In the second stage, black-start is conducted. This stage includes selecting black-start units and determining black-start schemes. Feltes [21] described black-start operation, planning, and control with an overview of black-start. In the third stage, network reconfiguration is determined to in order to minimize energy loss and maximize power supply to the system. In this stage, the power generation acquired from the previous stage is utilized to restore other units in the power system including non black-start unit, substation, etc. There are several studies of network reconfiguration using following methods: heuristic algorithm [22, 23], genetic algorithm [24], linear programming [25], multiagent [26, 27, 28, 29], etc. In the last stage, load restoration is processed. Cold load pickup is mainly studied for this last stage and Kumar et al. [30] overviewed cold load pickup related issues and models to solve problems. Among all the stages, this study mainly focuses on the second stage of restoration. Besides these conventional methods to restore power systems, new technologies have been studied to restore power systems such as renewables, microgrid, etc. [31]

Adibi and Kafka [32] presented restoration issues including interconnection assistance, load generation balance, fault location, assessment of switching status, etc. Among those issues, this study focuses on how to establish switching strategies to resolve issues to restore power systems. Andrews et al. [33] presented two switching strategies: Controlled operation, and All-open. Although All-open is much simpler to the system operators, it creates a large burden on the power system. However, Controlled operation creates less burden on the power system and requires recurrent attention on switches to restore the power system. In this study, reinforcement learning takes a role of the system operators. Therefore, Controlled operation is chosen.

An overview of power system restoration was covered in [34] with eight topics from the papers in the 1980s and 1990s. A more recent literature review regarding restoration with newer technologies was investigated in [31].

Reinforcement Learning in Power System Restoration

MAS has been used over a decade in power system engineering and it is adopted by several power systems applications: automation [35], energy market [36, 37], smartgrid [38], microgrid [39, 40], load shedding [41], restoration [26, 27, 28, 42, 43, 44, 45, 46], protection [47], etc. McArthur et al. [48] discussed the applications of MAS and the paper suggested any field to adopt MAS if the applications have at least one of the following features:

- An interaction between entities is required.

• A great number of entities needs to interact with each other.

• Enough data from local entities exist.

• Implementing new functions is required within existing systems.

• It is required to add and extend functionality of existing systems.

Power systems restoration requires at least three of the above items, and it can be concluded that MAS can be adopted for power systems restoration. Jayasinghe et al. [29] presented an overview of the application of MAS in power systems restoration. The paper introduced IEC 61850 standards which allow automated systems to share the network for communication within the systems. It also provided an overview of MAS architectures including centralized model, hierarchical model, and hybrid model with their features.

Nagata et al. [26] showed hierarchical structure between each agent. This study considered power system constraints, but renewables and load priorities were not considered in the study. Yan et al. [46] adopted MAS in a centralized manner. Although it has a high speed of the process, it is vulnerable to the loss of central agent. Felix et al. [28] proposed a hybrid model to restore power systems with theoretical perspective. Lin and Chin [45] presented distribution service restoration with centralized MAS model. It is vulnerable to the fault at the central agent. Khamphanchai et al. [44] developed a framework for distributed system restoration with a hierarchical model and JADE. However, it did not consider power system constraints. Solanki et al. [42] proposed the MAS framework with a hierarchical model but the study mostly focused on agent development rather than restoration perspective. Ye et al. [27] developed a hybrid MAS framework using Q-learning to restore power systems. They adopted reward calculation for Q-learning, and it helped agents to take actions with more accurate information. Even if they presented a solid model, power system constraints were not considered in the study.

Theory

In this section, the theories of Q-learning, and MAS are introduced with MAS platforms.

Reinforcement Learning

Q-learning

Q-learning is an off-policy Temporal Difference (TD) control algorithm [49]. Algorithm 1 represents Q-learning algorithm [10] where Q is an action-value function, s is previous state, a is previous action, s' is next state, a' is next action, α is learning rate, $0 < \alpha \leq 1$, r is the reward that an agent receives when it takes an action a in a state s, and γ is discount rate, $0 \leq \gamma \leq 1$. Learning rate α can be referred as step size, and it decides how much an agent will learn from the information. In the case of the learning rate close to 0, an agent is learning almost nothing from the new information. With the learning rate 1, an agent is learning only from the new information. Discount factor γ decides the importance of future rewards. With the discount factor 0, an agent only considers current rewards. In the case of the learning rate 1, an agent only considers future rewards. Line 7 in Algorithm 1 shows the simplest form of Q-learning.

Lauer et al. [50] showed that Q-learning could be used for distributed reinforcement learning in cooperative MAS and it is utilized to develop restoration algorithm with MAS environment in this study.

Algorithm 1 : Q-learning

```
1: Initialize $Q(s, a)$ arbitrarily
2: for each episode do
3:     Initialize $s$
4:     for each step of episode do
5:         Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
6:         Take action $a$, observe $r$, $s'$
7:         $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$
8:         $s \leftarrow s'$;
9:     until $s$ is terminal
```

ϵ-Greedy

Among estimates of the action values, there should be at least one action that has the biggest estimate. This action is called a greedy action. If an agent selects a greedy action, it is said that the agent is *exploiting* its knowledge. If the agent takes a nongreedy action, then it is said that the agent is *exploring*. The agent can improve the estimate of the nongreedy action through this process. Exploiting current knowledge helps to maximize the estimates for one step, but exploring may yield greater total estimates after several steps.

For an agent, an alternative of selecting greedy actions every time is selecting greedy actions most of the time. Every once in a while, the agent selects an action randomly with small probability ϵ as shown in Eq 2.1. This method is called ϵ-Greedy algorithm. The method shows higher average reward after large number of iterations [10, 51].

$$a_t = \begin{cases} a_t^*, & \text{with probability 1-}\epsilon \\ \textit{Random action} & \text{with probability } \epsilon \end{cases} \tag{2.1}$$

Temporal Difference Learning

TD learning is introduced as a combination of Monte Carlo and dynamic programming (DP) concept. TD methods do not require a model of environment, and it can learn from raw information. They are widely used to find solutions in an unknown environment. TD methods can update estimates without waiting for final estimates by using other learned estimates. They can be used to estimate value functions for reinforcement learning [10, 52].

Policy

When an agent is learning, the way of behaving at a given time step is defined by a policy. A policy is linking recognized states of the given environment to actions to be taken by the states. A policy can be as simple as a lookup table, but it can be very complicating which requires extensive computation. In most cases, policies are stochastic.

On-policy

An agent with on-policy learns the value of policy with the exploration steps. State-action-reward-state-action (SARSA) is one of the on-policy reinforcement learning algorithms [53].

Off-policy

Off-policy learns how to perform optimally when an agent is following a non-optimal policy. The agents learns the optimal policy without depending on the agent's action. Q-learning is off-policy algorithm [53].

Agent and Multiagent System

Agent

There are a large number of definitions for an agent and therefore defining an agent can be difficult [48, 54, 55, 56]. Among them, Russell et al. [57] defined an agent as follows: *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.* This is a good way to define the agents assumed in this study along with the following properties presented by Wooldridge and Jennings [58]:

• *autonomy : agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;*
• *social ability : agents interact with other agents (and possibly humans) via some kind of agent communication language;*
• *reactivity : agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;*
• *pro-activeness : agents do not simply act in response to their environment, they are able to exhibit goal-directed behavior by taking the initiative;*

Multiagent System

There are multiple definitions of a MAS. For this study, a MAS can be defined in [55, 59] : *A multi-agent system is a loosely coupled network of problem-solving agents that work together to find answers to problems that are beyond the individual capabilities or knowledge of each agent.* A MAS is useful to power system restoration because of following reasons [59]: *"robustness, scalability, simpler programming, geographic distribution, and cost effectiveness"*.

There are four types of agents for the MAS as follows [59]:

- Homogeneous Non-communicating Agents
- Homogeneous Communicating Agents
- Heterogeneous Non-communicating Agents
- Heterogeneous Communicating Agents

In this study, it is assumed that the given distribution system has heterogeneous communicating agents since each component has a different reward formula and they talk to each other. MAS has three architectures, decentralized, centralized, and hybrid model as shown in Fig 2.1.

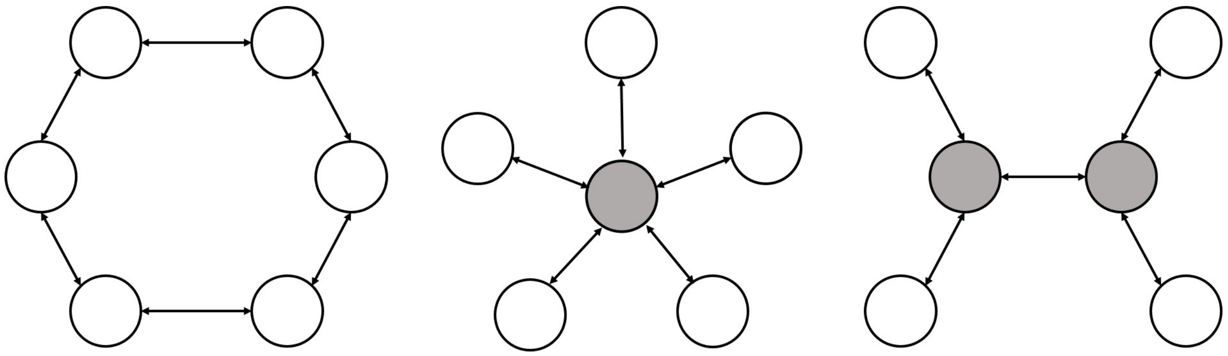


Figure 2.1: MAS Architectures: Decentralized, Centralized, Hybrid Model

In a centralized model, a central agent has access to all information from agents in the model and global information provides accurate information to agents while restoring power systems. However, it is vulnerable to single point of failure when the model doesn't have enough redundancy.

With a decentralized model, the MAS has less communication burden than a centralized model [60]. A decentralized model is more compatible with homogeneous agents due to the nature of its structure. Furthermore, each agent will have less computational burden compared to a centralized model. However, the accuracy of decision is not guaranteed with this model.

A hybrid model combines the advantages from both centralized and decentralized models. It takes less reaction time compared to a decentralized model, and it has less communication burden compared to a centralized model with immunity to single point of failure [27, 29]. In this study, a hybrid model is adopted to exploit these advantages for power system restoration.

Multiagent System Platforms

There are multiple frameworks to implement a MAS such as JADE (Java Agent Development Framework) [61], FIPA-OS (Foundation for Intelligent Physical Agents OS) [62], ZEUS [63], and JaCaMo (Combination of Jason, Cartago, and Moise) [64]. Among them, JADE has been utilized most to study power system restoration [42, 44, 65, 66] since it has great compatibility with Matlab and Simulink through the software MACSimjx [67].

The framework of JADE is based on a middleware that expedites the development process of agent applications. In particular, the following features are useful to a MAS framework for power system restoration [61, 68]: full compliance with the Foundation for Intelligent Physical Agents (FIPA) specifications, a fully distributed system, support for agent mobility, etc. In this study, a MAS platform is not used, but the agent and MAS are used in the Q-learning algorithm.

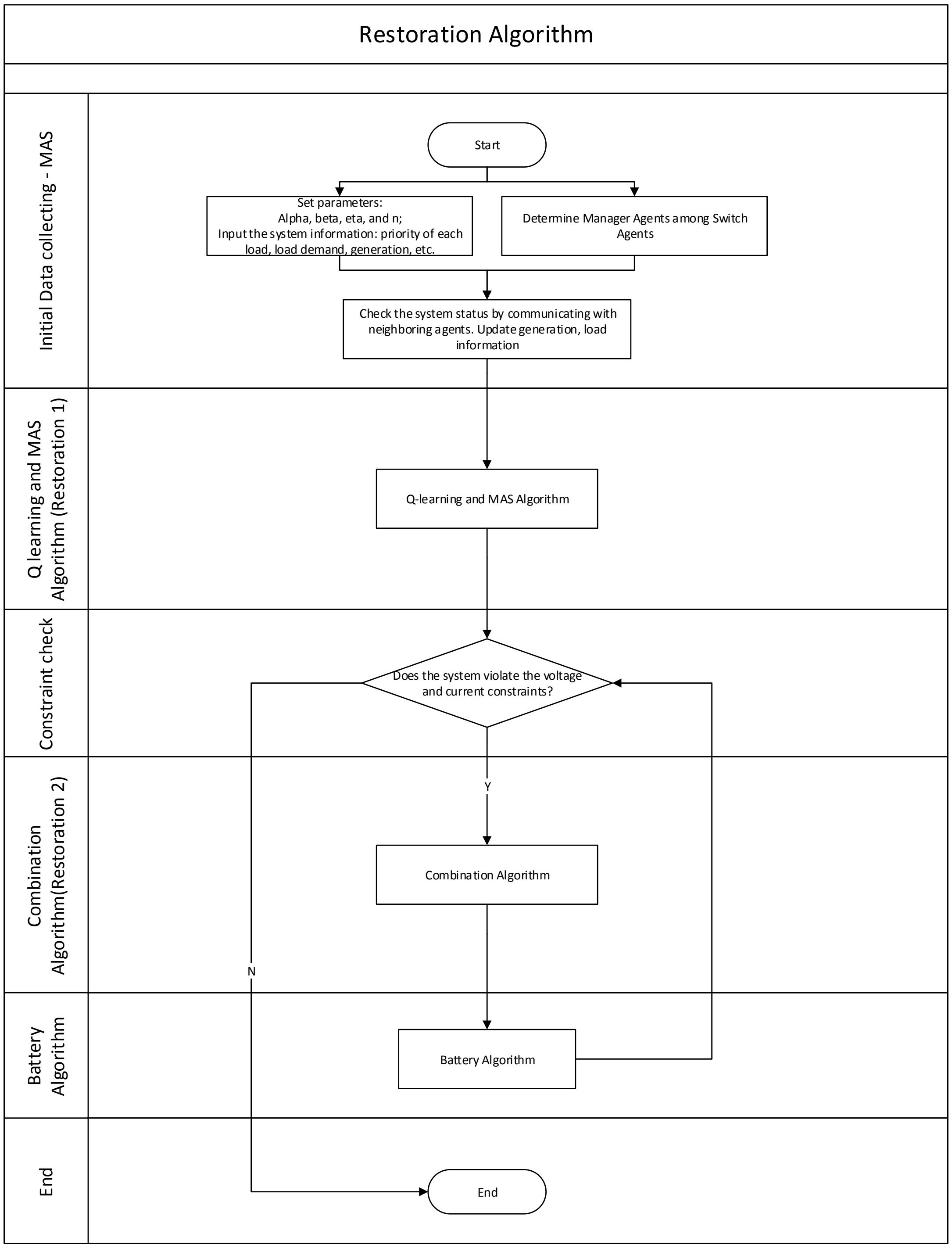
CHAPTER 3: ALGORITHM DEVELOPMENT

Chapter 3 describes the algorithm and model developed to conduct this study.

Algorithm

Objective

The objective of this algorithm is the development of restoration strategy after faults occurred in distribution systems with using several concepts including Q-learning and Multiagent. Additionally, this algorithm is capable of operating distribution systems when they have unexpected system condition changes after they are restored initially. Figure 3.1 shows the entire restoration algorithm to restore the given distribution system from faults. Each component in the algorithm is covered in this chapter.



Restoration Algorithm
Initial Data collecting - MAS
Q learning and MAS Algorithm (Restoration 1)
Constraint check
Combination Algorithm(Restoration 2)
Battery Algorithm
End
Start
Set parameters: Alpha, beta, eta, and n; Input the system information: priority of each load, load demand, generation, etc.
Determine Manager Agents among Switch Agents
Check the system status by communicating with neighboring agents. Update generation, load information
Q-learning and MAS Algorithm
Does the system violate the voltage and current constraints?
Y
N
Combination Algorithm
Battery Algorithm
End

Figure 3.1: Restoration Algorithm

Multiagent

A MAS consists of several distributed intelligent agents. A given distribution system is divided into zones, and each zone has agents. The number of zones is determined by the size of the distribution system. The distribution system restoration is carried out based on the coordination between agents. Two types of agents are used in this study, Manager Agent, and Switch Agent. There are three types of agents that belong to Switch Agent. Each agent is introduced in the following section. This approach has advantages over either fully-centered or local approach [27]. Each zone could have several Switch Agents, and there is only one Manager Agent per zone. It is assumed that the communication channel is existing between each zone and directly connected by a power line. Each agent can share sensations, and policies. Sharing sensations can help each agent to conduct global and local optimization. In this study, the communication between each agent is assumed, and it is covered in Future work section.

Switch Agent

A Switch Agent is located at each switch in the given distribution system. There are three special types of agents among Switch Agents: Load Agent, Generator Agent, and Local Agent. They are defined separately for reward designing purposes. Although Local Agents and Load Agents have their own rewards, Generator Agents do not have rewards since their status is determined by the status of each generator.

Manager Agent

Manage Agents are chosen from Generator Agents since their computational load is less than other types of agents. This helps to distribute a load of computation among agents and improve the efficiency of the algorithm.

Reward Design

Each agent has its own reward depending on its type. The reward from each agent is utilized to implement the reinforcement learning part of the restoration algorithm.

Load Agent

When one load is restored, it could cause the loss of other loads connected in the given distribution system. In order to generalize the reward function for any load in the system, below equations are proposed.

$$LoadValue(i) = Priority(i) * CurrentLoad(i) \tag{3.1}$$

$$LoadAgent(i, on) = LoadValue(i) - \frac{1}{n}\sum_{k \neq i} LoadValue(k) \tag{3.2}$$

where n is the total number of loads in the given distribution system.

$$LoadAgent(i, off) = -LoadAgent(i, on) \tag{3.3}$$

When restoring the load i, there is a possibility that other loads might lose the power. In Eq 3.2, if the summation part of the reward is divided by $n - 1$, the reward can become negative for the load i with the highest priority. Also, if there is only one load in the system, $n - 1$ can be zero, and it causes an error in the given Eq 3.2. Therefore, it is divided by n instead of $n - 1$.

Generator Agent

The status of Generator Agent is predetermined by the condition of the given distribution system. Therefore, there is no need to design reward for Generator Agents. Furthermore, Generator Agents are not included in Q-learning process since their statuses are already decided. It is assumed that generators are connected to the given distribution system unless there are faults at the generators.

Local Agent

Local Agent refers to all Switch Agents in the given distribution system excluding Generator Agents and Load Agents. Therefore, some Local Agents are connected to the switches on lines, and some of them are connected to PV or battery. Therefore, the format of reward can vary depending on the locations and features of switches.

$$LocalAgent(i, on) = \sum_{i=connectedloads} Loadvalue(i) - \sum_{j=disconnectedloads} Loadvalue(j) \tag{3.4}$$

$$LocalAgent(i, off) = -LocalAgent(i, on) \tag{3.5}$$

The result from switching action on, off decides which loads will be connected and disconnected from the given distribution system. The reward of the Local Agent is shown in Eq 3.4 which represents the general format of the reward equation as well.

Manager Agent

As described earlier, Manager Agent is chosen from Generator Agents. Therefore, there is no reward function for this type of agent. Although Manager Agent does not have its own reward function, its function is critical to the Q-learning part of the entire algorithm, and it is described in following sections.

Constraints

In order to present that the solution from this algorithm is realistic, voltage, current, radiality constraints are considered. [28].

Voltage Constraint

Voltage from simulation should stay within $\pm 5\%$ range from normal operating voltage.

$$0.95(p.u.) \leq V_i(p.u.) \leq 1.05(p.u.) \tag{3.6}$$

where V_i is the voltage at the bus i.

Current Constraint

Current from simulation should be less than 110% from current rating of each line of the given distribution system.

$$I_i \leq 1.1 * I_{nom} \tag{3.7}$$

where I_{nom} is nominal current of the given power system.

Radiality Constraint

The system should keep radial structure after the reconfiguration.

Q-learning

Q-learning is one of the popular methods in reinforcement learning. However, the simplest form of Q-learning in Chapter 2 has limited applications to power system since every state needs to be defined prior to the implementation. Therefore, Q-learning without states is necessary to decrease the computational burden and increase the efficiency of the algorithm. Modified Q-learning from [27] is applied to power systems restoration. The function $\pi(a)$ represents the stochastic action policy as shown in Eq 3.8, where ϵ is the small positive number between 0 and 1 and n is is the number of possible actions of an agent. ϵ is set as 0.4 in this study.

$$\pi(a) = \begin{cases} (1-\epsilon) + \frac{\epsilon}{n}, & \text{When Q(a) is the highest} \\ \frac{\epsilon}{n}, & \text{Otherwise} \end{cases} \tag{3.8}$$

Equation 3.9 shows the possible actions of an agent i and therefore n is 2 in this study.

$$Action(i) = \begin{cases} On \\ Off \end{cases} \tag{3.9}$$

Both rewards of Load Agent and Local Agent are from general reward function as shown in Eq 3.10. Eq 3.11 represents the $Degree$ function that is utilized to determine the degree of acceptance. The function ranges from -1 to 1. The sign of Eq. 3.11 shows whether the action is encouraged or discouraged. A positive sign means it is encouraged and a negative sign means it is discouraged to take the $Action(a)$. The magnitude of Eq. 3.11 shows the strength of the suggestion. The closer $|Degree(i,a)|$ to 1 is, the stronger suggestion is.

$$Reward(i,a) = \begin{cases} LoadAgent(i,a), & \text{if } i \in Loads \\ LocalAgent(i,a), & \text{if } i \notin Loads\ or\ Generators \end{cases} \tag{3.10}$$

$$Degree(i,a) = \frac{Reward(i,a) - Average(a)}{max(|Average(a) - Reward(i,a)|)} \tag{3.11}$$

In Eq 3.11, $Average(a)$ is the average of the reward of agents that belong to the same Manager Agent when they take an action a.

$$Suggestion(i) = [Action(i), Degree(i,a)] \tag{3.12}$$

Equation 3.12 represents the form of suggestion. Each agent has this form of suggestion to take an action. Equation 3.13 decides the receptivity of Agent i for suggestions that range from 0 to 1. β is a arbitrary constant and it is 10 in this study. $Info(Switch(i))$ represents the number of messages possessed by Switch Agent i. The messages are acquired during the communication process between neighborhood agents. If an agent has more messages than the other agents during the process, then the agent has less receptivity than the other agents since the value of Eq 3.13 is smaller.

$$\eta(Switch(i)) = \frac{\beta}{\beta + Info(Switch(i))} \tag{3.13}$$

The stochastic policy $\pi(a)$ determined in Eq 3.8 is updated using the equations introduced above.

As shown in Eq 3.14, $\pi'(a)$ is utilized to update Q-learning algorithm.

$$\pi'(a) = \begin{cases} \pi(a) + \pi(a) * \eta * Degree(i,a), & \text{if } Degree(i,a) \leq 0 \\ \pi(a) + (1 - \pi(a)) * \eta * Degree(i,a), & \text{if } Degree(i,a) > 0 \end{cases} \tag{3.14}$$

Learning processes for each Switch Agent from [27] is applied to update Q values and decides actions as shown below.

1. Initialize Q value for each possible action randomly.
2. Set $k = 0$.
3. Calculate $\pi(a)$.
4. Update $\pi'(a)$ with $\pi(a)$.
5. Assign $limit(\pi'(a))$ to $\pi'(a)$.
6. For each possible action a,

$$Q_{k+1}(a) = Q_k(a) + \pi'(a) * \alpha * (\sum_a Reward'(i,a)\pi'(a)) - Q_k(a)) \tag{3.15}$$

7. Repeat step 3 to 6 predetermined $k = x$ times.
8. $a_{opti} \leftarrow argMax(Q)$
9. Switch Agent i takes action a_{opti}.

In step 5, $limit(\pi'(a))$ returns a valid policy that is closet to $\pi'(a)$ in order to normalize π' and make the summation of π' to 1 as shown in Eq 3.16.

$$limit(\pi'(a)) = argMin_{x \in valid(x)} |\pi' - x| \tag{3.16}$$

In this study, there are only two possible actions for each agent so Eq 3.17 is simply adopted to normalize $\pi'(a)$.

$$\pi'(a) = \frac{\pi'(a)}{\sum_{a \in Actions} \pi'(a)} \tag{3.17}$$

From Eq 3.15 in step 6, α is the learning rate of the algorithm and it ranges between 0 and 1. The closer it is to 1, the more weight it has on the future reward than the current reward of each agent. $Reward'(i, a)$ represents the expected reward of each agent by taking an action a. In step 8, the action a that has the highest Q value is determined as a_{opti}. Finally, each agent takes an action a_{opti} in step 9. With the above algorithm, the procedure of restoration using Q-learning and MAS is shown in Fig 3.2.

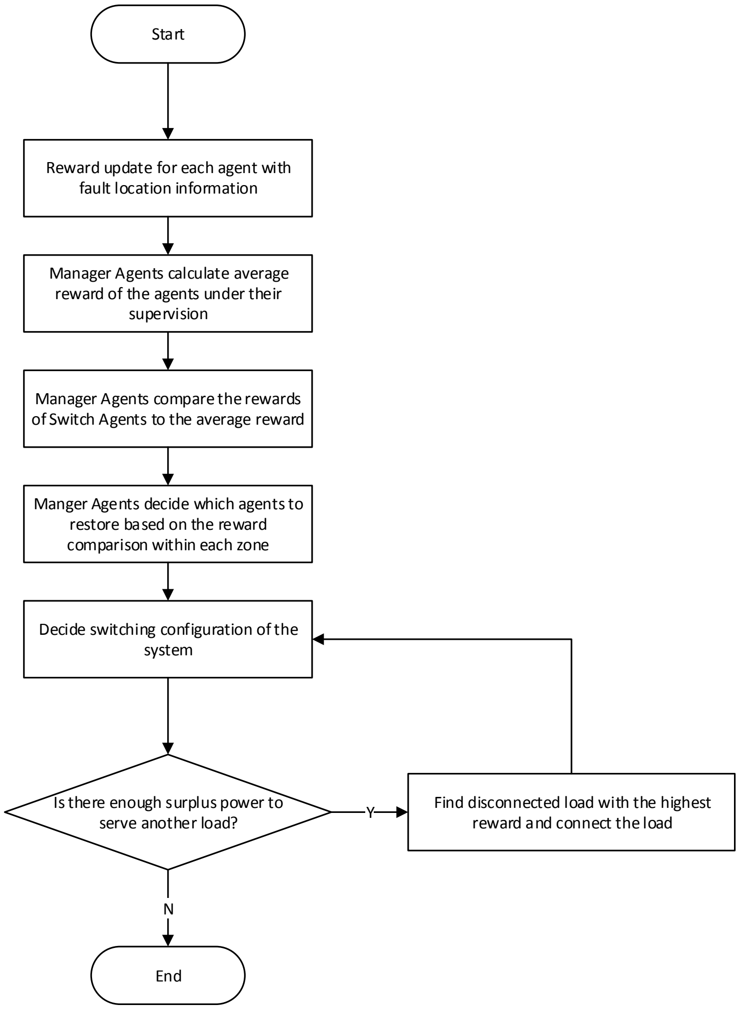

Start
Reward update for each agent with fault location information
Manager Agents calculate average reward of the agents under their supervision
Manager Agents compare the rewards of Switch Agents to the average reward
Manger Agents decide which agents to restore based on the reward comparison within each zone
Decide switching configuration of the system
Is there enough surplus power to serve another load?
Y
Find disconnected load with the highest reward and connect the load
N
End

Figure 3.2: Restoration Algorithm using Q-learning and Multiagent

Reduced Combination Algorithm

From the experiment, it was found that Q-learning does not always provide the solutions free from violating the constraints. Therefore, it is necessary to have supplementary algorithms to provide solutions in case of failure of Q-learning. This algorithm utilizes a combination approach to find the best solution to restore the given distribution system. However, it only considers a reduced number of switch configuration combinations to decrease the computational load of the restoration algorithm.

Preset switches

The following switches are set before the algorithm is implemented: the switches connected to generators, the switches connected to the highest priority loads, and the switches at the fault locations. In this step, regardless of the amount of surplus power, the loads that have the highest priority are only picked up. If all load priorities are equal to each other, then the status of the switches connected to the loads is determined in the next step.

Determine the status of remaining switches

After determining the status of preset switches, the status of remaining switches in given distribution system is decided by following steps.

1. The combination of the status of remaining switches is generated while it meets the minimum requirement such as no loop inside the system.
2. With the reward for each agent calculated in the previous step, the total reward for each combination is calculated.

3. The combination with higher total reward has higher priority than other combinations.

4. The constraints check is implemented for each combination from the combination that has higher priority.

5. If there is any violation with the combination, the algorithm shows the switch (switches) has (have) violation and runs next combination.

6. When the combination has no violation, the algorithm is ended.

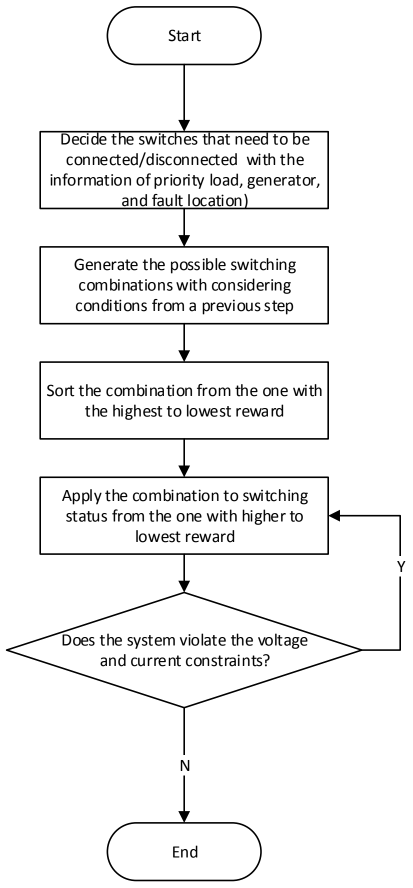

Start
Decide the switches that need to be connected/disconnected with the information of priority load, generator, and fault location)
Generate the possible switching combinations with considering conditions from a previous step
Sort the combination from the one with the highest to lowest reward
Apply the combination to switching status from the one with higher to lowest reward
Y
Does the system violate the voltage and current constraints?
N
End

Figure 3.3: Reduced Combination Algorithm

Battery Algorithm

The goal of the battery algorithm is to use the most surplus power from the given distribution system. Following steps present a general battery algorithm.

1. Calculate the surplus power.

$$TotalGeneration - TotalDemand = SurplusPower \tag{3.18}$$

2. If the surplus power is greater than the battery capacity, the battery is set to be charged. If the surplus power is less than the battery capacity, the battery is set to be discharged.

3. If there is any violation with the combination, the algorithm shows the switch (switches) has (have) violation and run next combination.

4. When the combination has no violation, the algorithm is ended.

For example, if there is 25kW surplus power and are three 10kW batteries in the given system, the algorithm is implemented as follows.

Table 3.1: Battery Algorithm Example

Battery	Capacity (kW)	Generator Variable	Load Variable
1	10	b11	b12
2	10	b21	b22
3	10	b31	b32

Since each battery can be operated as either generator or load, there are following constraints.

$$b11 + b12 = 1 \tag{3.19}$$

$$b21 + b22 = 1 \tag{3.20}$$

$$b31 + b32 = 1 \tag{3.21}$$

Based on the surplus power 25kW, only 2 batteries can be charged. Therefore, following constraints are applied.

$$b11 + b21 + b31 = 1 \tag{3.22}$$

$$b12 + b22 + b32 = 2 \tag{3.23}$$

The combinations meeting above constraints can be implemented and the status of each battery can be determined. Figure 3.4 shows the general battery algorithm.

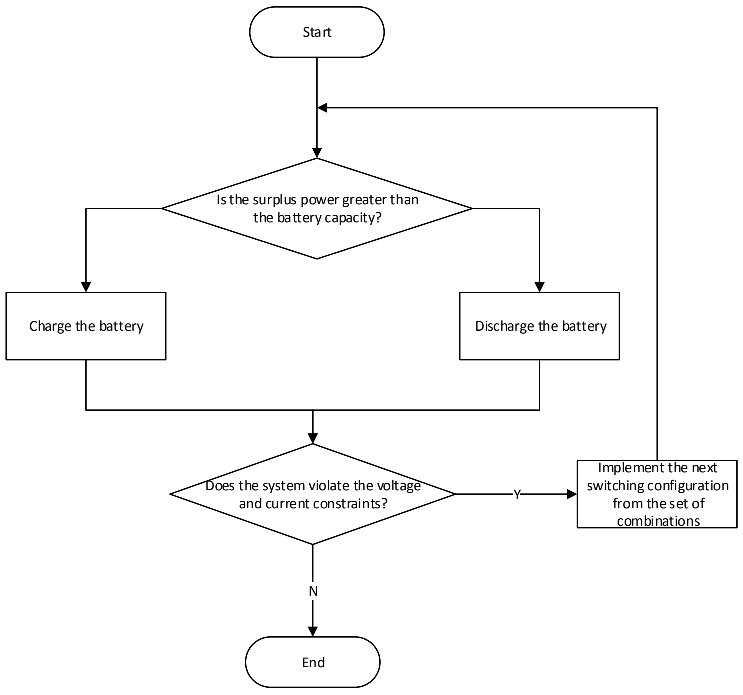

Start
Is the surplus power greater than the battery capacity?
Charge the battery
Discharge the battery
Does the system violate the voltage and current constraints?
Y
Implement the next switching configuration from the set of combinations
N
End

Figure 3.4: Battery Algorithm

Post Restoration Algorithm

After faults, the algorithm in Fig 3.1 is implemented as an initial step. When the restoration is completed, there is a possibility that the conditions of the components in the given system are changed. For example, PV generation, load, the power supply from the substation, and so on. Therefore, it is necessary to consider the post-restoration conditions. The algorithm proposed in Fig 3.1 is applicable to handle these situations.

Summary

The proposed algorithm covers how to restore distribution system using Q-learning, MAS, Combination algorithm, and Battery algorithm. As long as the total generation of the given distribution system is larger than the total amount of loads that are connected to the system, the maximum amount of loads is set to be picked up according to their priorities using the algorithms.

Real-time Simulaton

There are several power systems simulation tools such as PowerWorld, GridLAB-D, OpenDSS, Matlab/Simulink, RTDS, and Opal-RT. Among those tools, Matlab/Simulink and Opal-RT are selected to implement the real-time simulation of the algorithm from this study and this real-time simulation provides rapid prototyping of the algorithm on the test model [69, 70, 71].

Configuration

A real-time simulation testbed consists of two parts: A host computer and a target computer. A host computer is operating Windows OS, and it has a Simulink model and compile the model using RT-

LAB. A target computer is operating Linux OS. It executes a real-time simulation, receives input signals from connected hardware devices, and sends the model output to hardware devices. A host computer and a target computer are communicating using TCP/IP. A configuration of the testbed is shown in Fig 3.5.

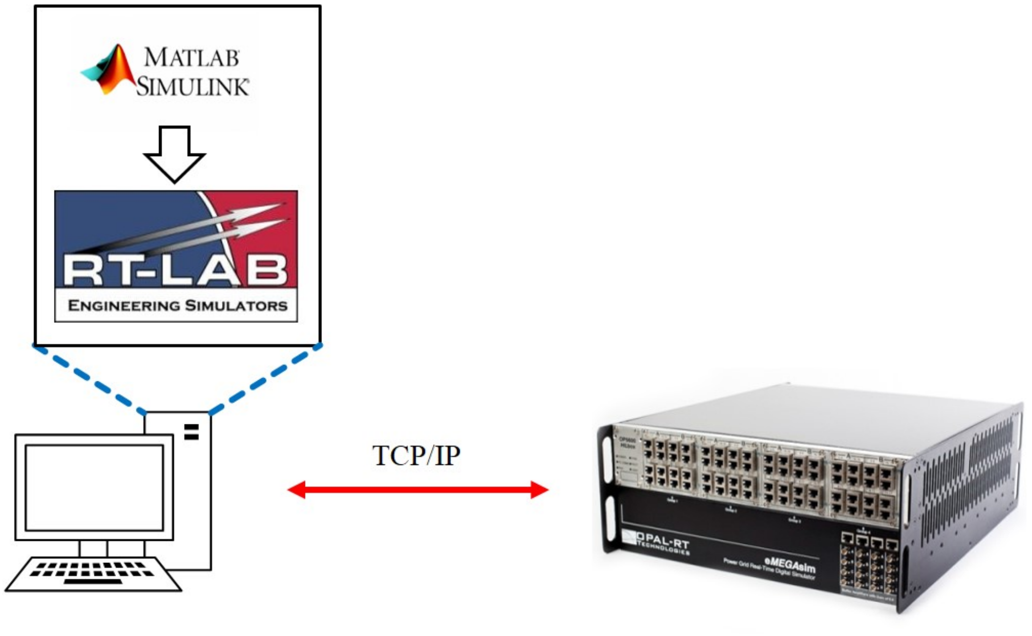


Figure 3.5: A Real-time Simulation Testbed

Procedure

First, the model was built in Simulink with basic and special tool boxes such as SimPowerSystems (SPS) models. The model is tested offline before proceeding to next steps. Next, the Simulink model is separated into two subsystems: Console Subsystem (SC) and Master Subsystem(SM). SC has every interface block of the Simulink model, and it provides interaction with the Simulink model during real-time simulation. SM has all the elements of the Simulink model for computation, I/O, mathematical operation, algorithm, etc. Then, RT-LAB is utilized to create an RT-LAB model, and it separates the Simulink model into SM and SC. With the separated model, RT-LAB generates C code. The RT-LAB model is assigned to a target, and the target executes the real-time simulation of the model.

For a real-time simulation, the target computer is given a preset time such as 10ms, 100us, 10us, etc. to receive input signals from sensors, to execute algorithms, and to send output signals to connected hardware such as relays. The preset time is also known as the step size, and time step (Ts). The preset time involves receiving input signals, executing algorithms and outputting signals as shown in Fig 3.6 [72].

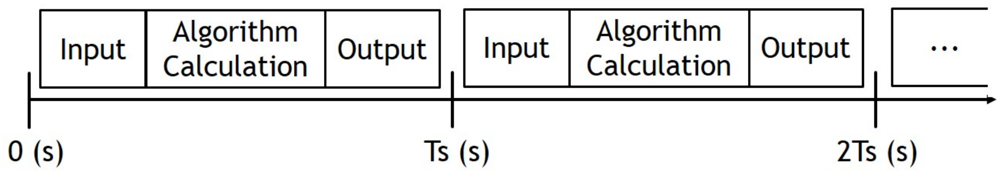


Figure 3.6: The Preset Time for a Real-time Simulation

CHAPTER 4: SIMULATION AND RESULTS

In this chapter, the algorithm and model from Chapter 3 are implemented and tested with various scenarios.

Simulation using IEEE 9-Bus system

The proposed algorithm is tested on the testbed. The testbed is built based on the IEEE 9-Bus system. The model should have a generator, load, switch, PV, and battery. There is no limit to the number of components to run the proposed algorithm. The testbed has two DGs, one battery, one PV, and three loads. Each load has its own priority between 0 and 1. If a load has higher priority than other loads, it is restored first. The substation is located at S12. Table 4.1 shows the main components in the given distribution system. Typically, the voltage of distribution system ranges from 120V to 35kV. The voltage of substation is 24.9kV, and the voltage of remaining network is 4.16kV in the testbed.

Table 4.1: Distribution System Components

Type	Capacity(kW)	Location	Note
Battery	10	S10	
PV	15	S10	
DG	85	S13	
DG	163	S14	
Load	100	S9	Priority:0.8
Load	125	S10	Priority:0.8
Load	90	S11	Priority:0.5

Matlab and Simulink were used to build the testbed shown in Fig 4.1. Fig 4.2, 4.3, and 4.4 show the details of the testbed. Fig 4.5 shows how the testbed is divided to apply MAS algorithm. Table

4.2 represents the default configuration of the given distribution system.

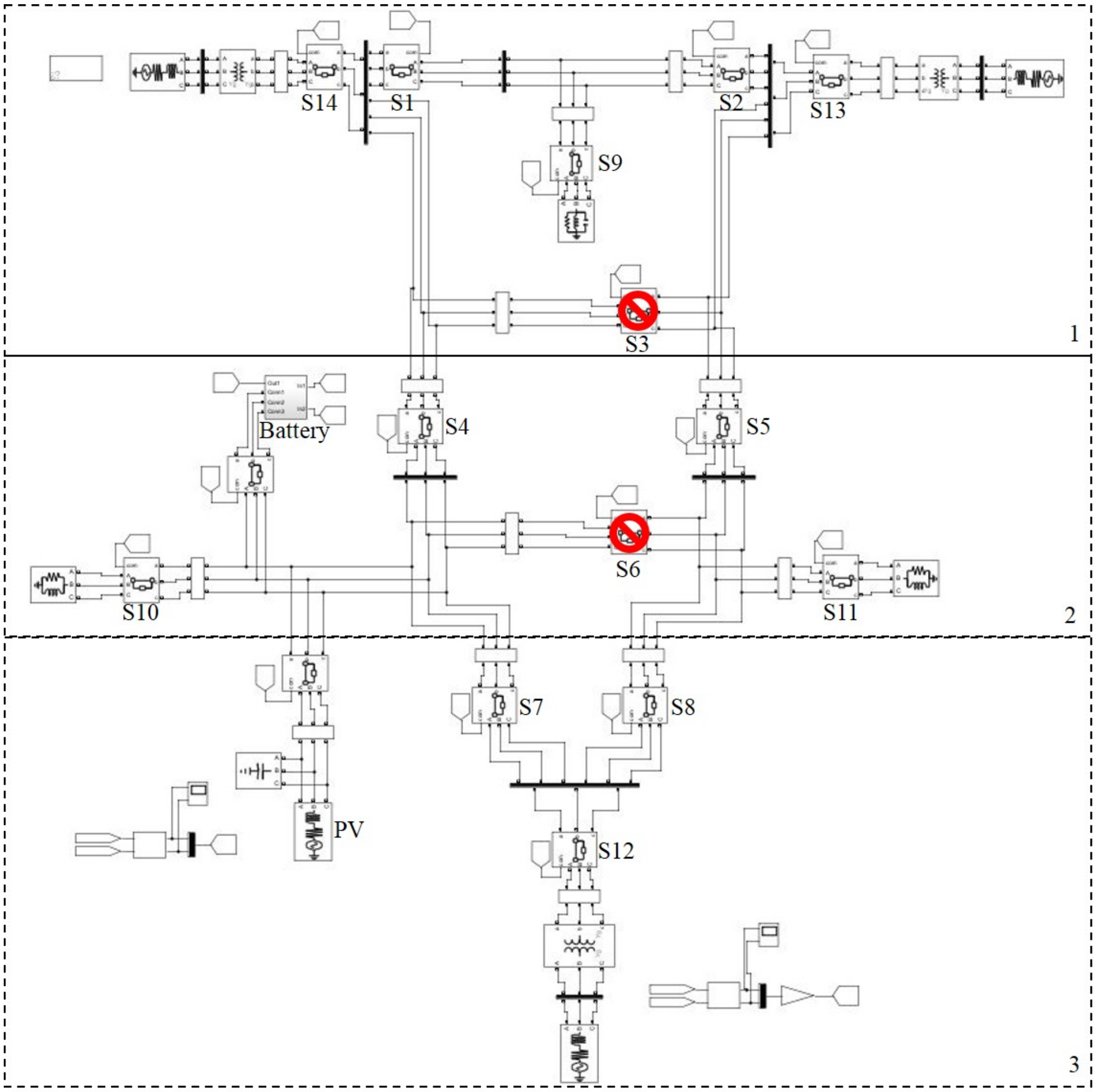


Figure 4.1: Overview of IEEE 9-bus system

Table 4.2: Default Switch Configuration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	On
S6	Off
S7	On
S8	On
S9	On
S10	On
S11	On
S12	On
S13	On
S14	On

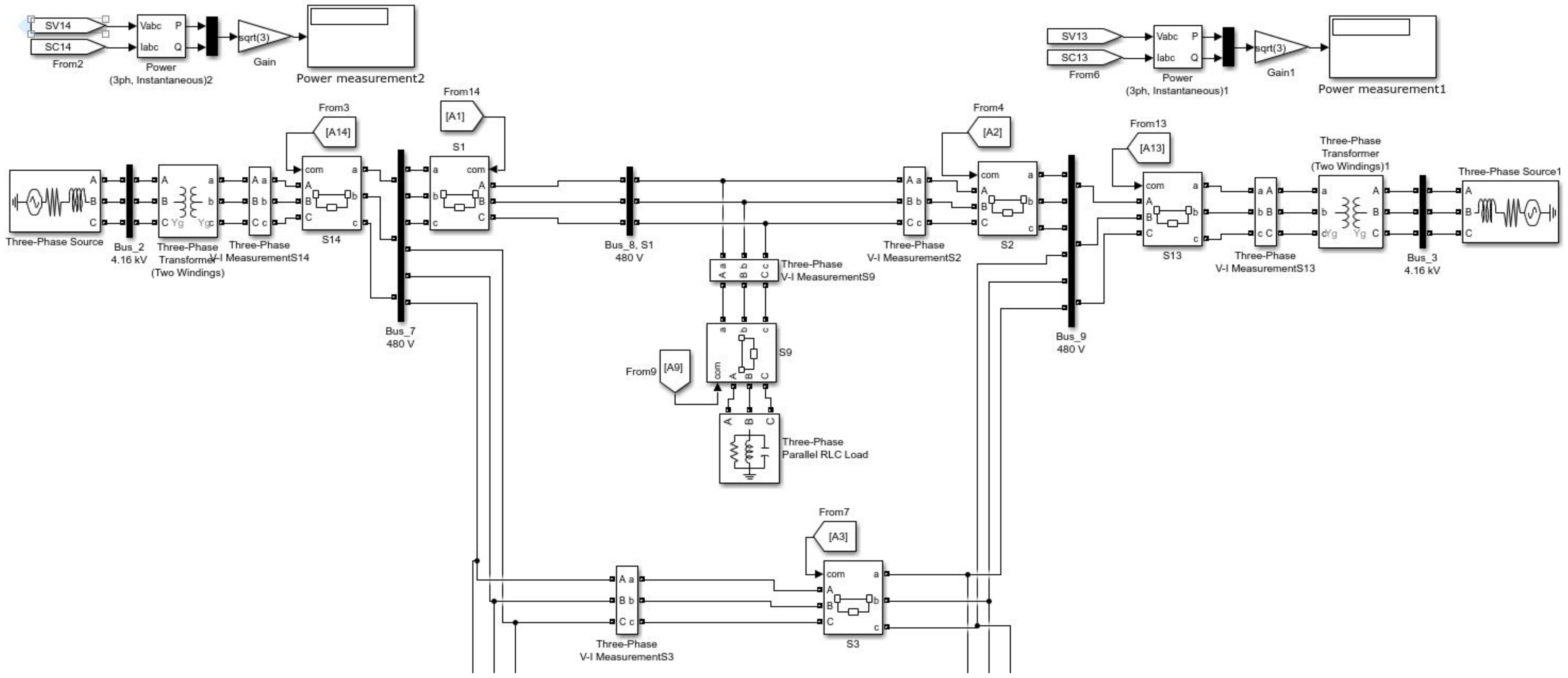


Figure 4.2: IEEE 9-bus System 1

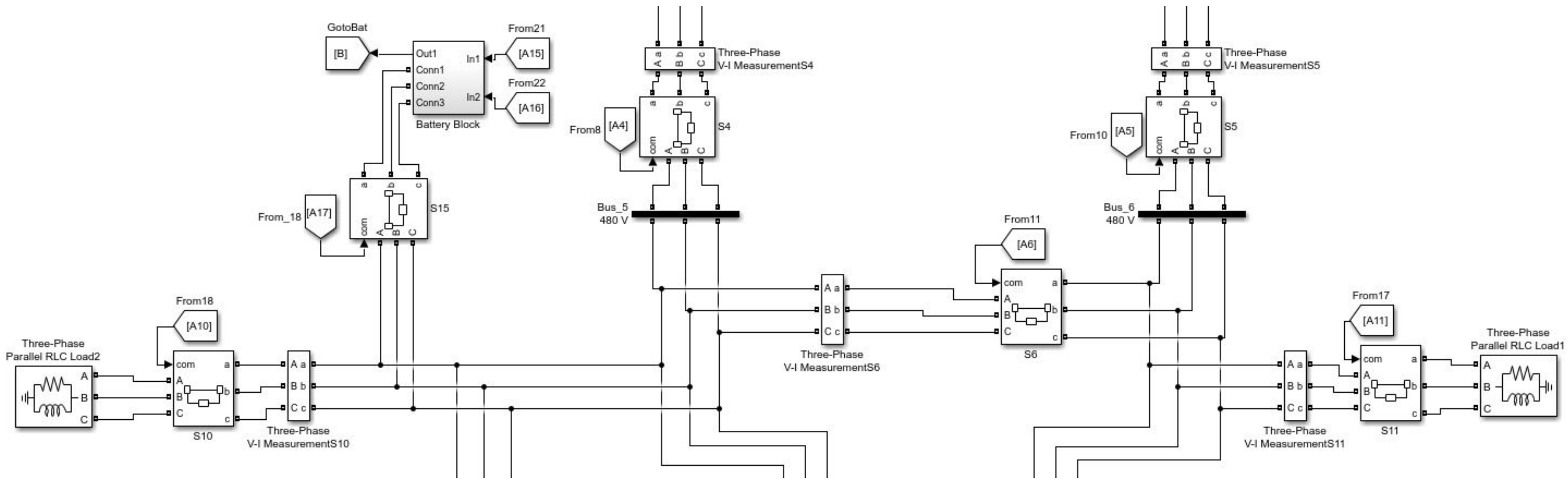

GotoBat
[B]
Out1
Conn1
Conn2
Conn3
In1
In2
Battery Block
From21
[A15]
From22
[A16]
From_18
[A17]
S15
Three-Phase V-I MeasurementS4
From8
[A4]
S4
Bus_5 480 V
Three-Phase V-I MeasurementS5
From10
[A5]
S5
Bus_6 480 V
From11
[A6]
S6
Three-Phase V-I MeasurementS6
From18
[A10]
Three-Phase Parallel RLC Load2
S10
Three-Phase V-I MeasurementS10
From17
[A11]
Three-Phase Parallel RLC Load1
S11
Three-Phase V-I MeasurementS11

Figure 4.3: IEEE 9-bus System 2

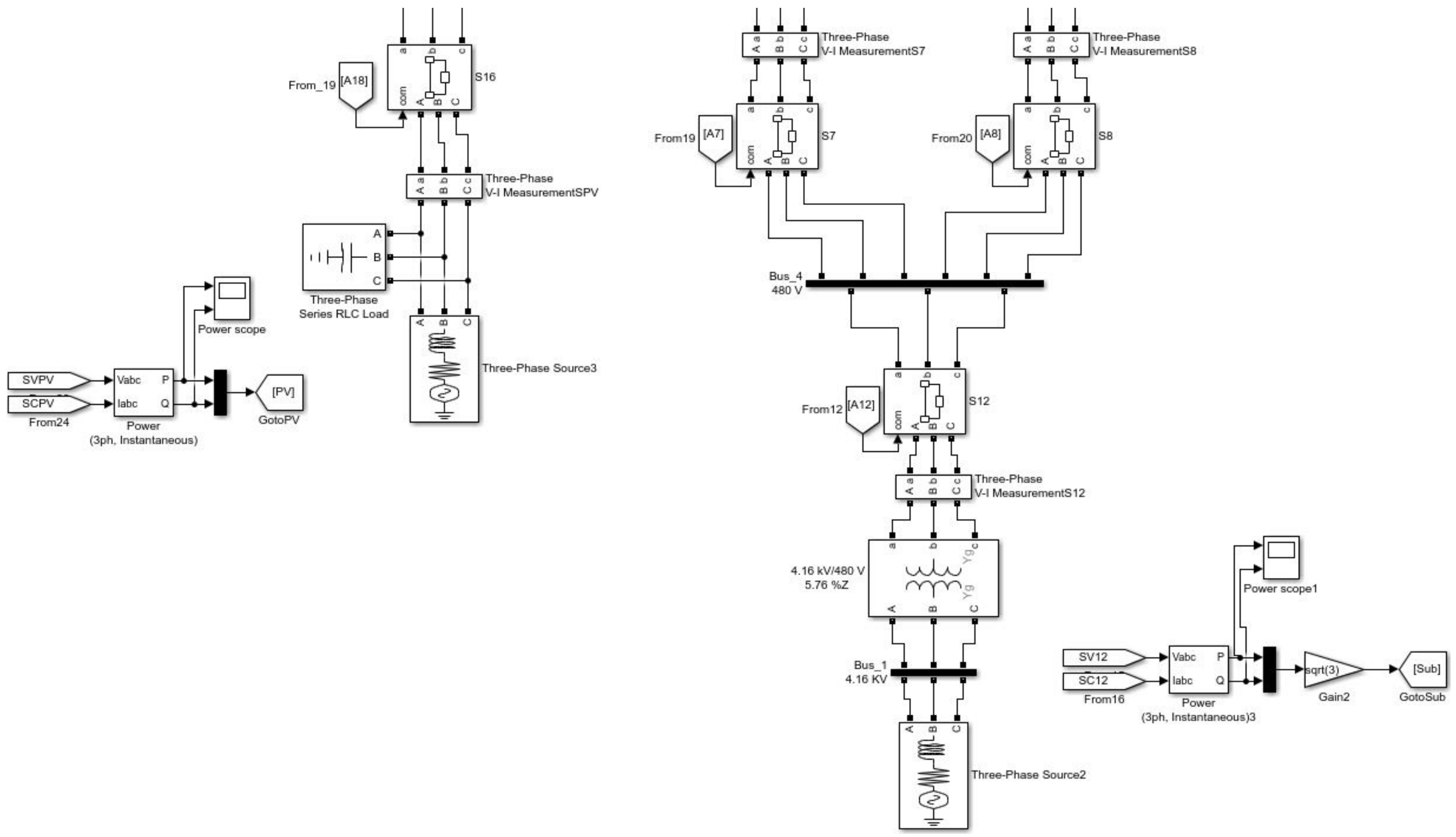


Figure 4.4: IEEE 9-bus System 3

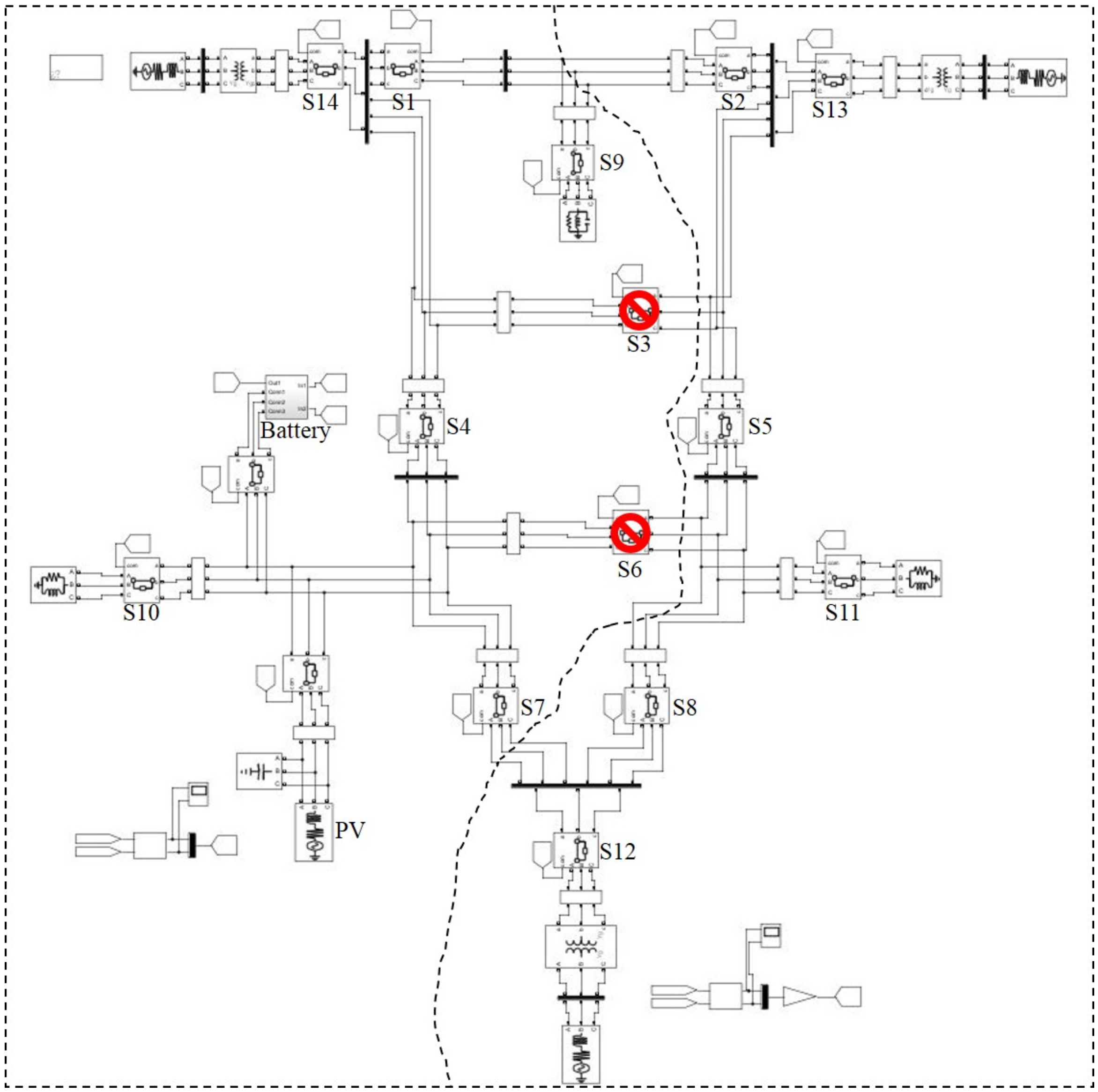


Figure 4.5: Zone Separation of IEEE 9-bus system

As shown in Fig 4.4, PV is simulated using reactive power bank and generator model. Because the focus of this study is not designing inverter and controlling PV, this modeling method is chosen and in this way the power factor can range from 0.95 to 1.05.

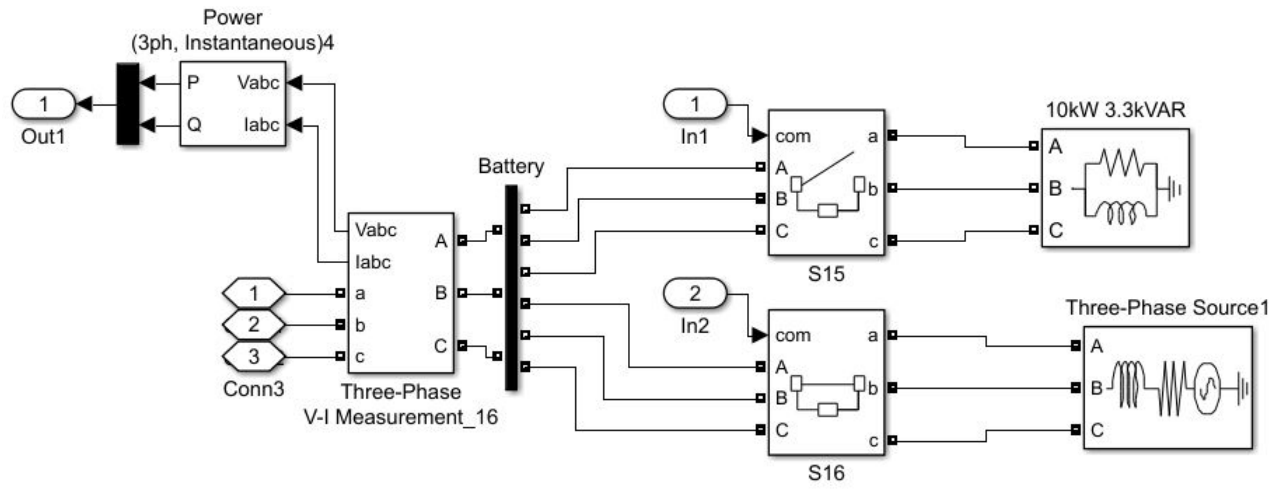


Figure 4.6: Battery Design

The model in Fig 4.6 represents the design of the battery. It has two components, one generator, and one load. The generator and load have 10kW capacity, and also the components are designed in order to keep the power factor between 0.95 and 1.05. As mentioned above, in order to simplify the components, a battery control algorithm is not considered in this study.

Simulation Scenarios for Initial Restoration after Faults and Results

In order to test the compatibility of the proposed algorithm in various situations, five scenarios were tested. The algorithm was implemented with the below assumptions.

1. It is assumed that each switch has communication capability with its neighborhood switches.
2. Each switch receives two messages from neighboring agents during the restoration process.

In this simulation, only three-phase fault cases were tested. The results from each scenario are presented as follows.

Fault at Substation

When a fault happened at the substation, the switching configuration is acquired from Q-learning as shown in Table 4.3. The surplus power is utilized to charge the battery. The fault and system configuration after restoration are shown in Fig 4.7

Table 4.3: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	Off
S6	Off
S7	On
S8	On
S9	On
S10	On
S11	Off
S12	Off
S13	On
S14	On

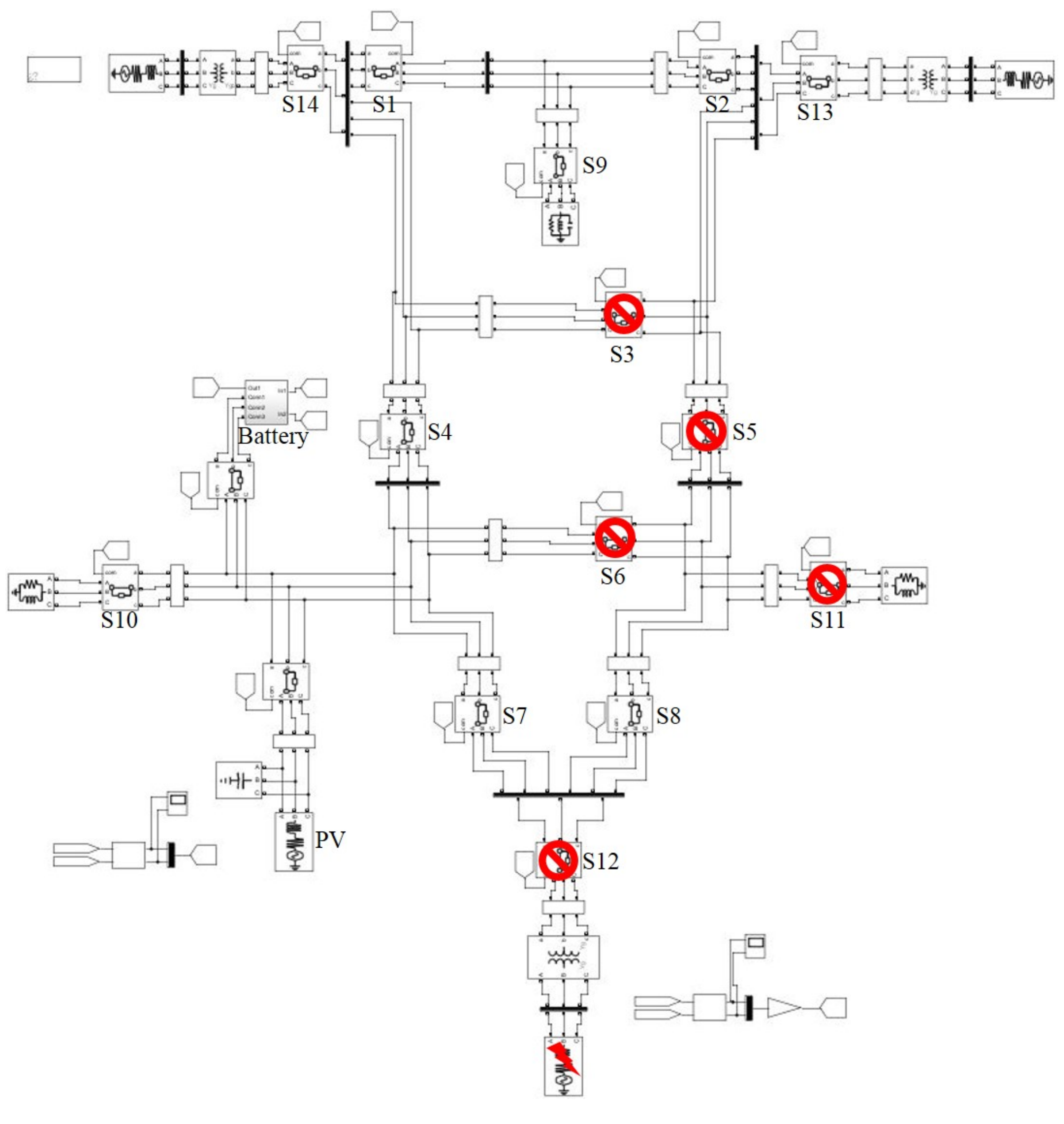


Figure 4.7: Restoration after Fault at Substation

Fault at Line

When a fault happened at the line near S4, the switching configuration from Q-learning violated the voltage constraints. Therefore, the combination algorithm was automatically utilized. After five combinations, no violations were observed. The result is shown in Table 4.4. The surplus power was utilized to charge the battery. The fault and system configuration after restoration are shown in Fig 4.8

Table 4.4: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	Off
S5	On
S6	Off
S7	On
S8	On
S9	On
S10	On
S11	On
S12	On
S13	On
S14	On

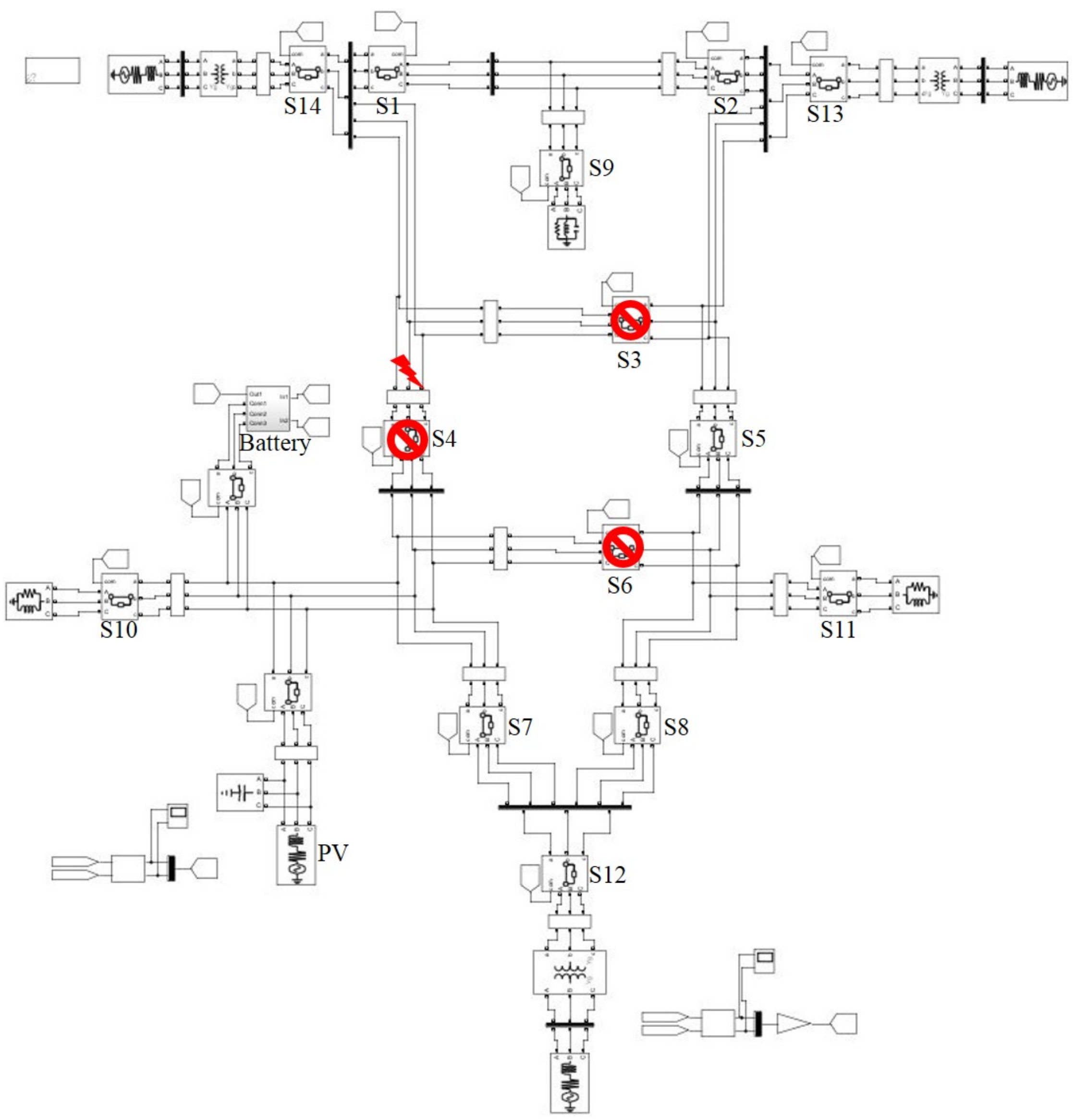


Figure 4.8: Restoration after Fault at Line

Fault at Distributed Generator

When a fault happened at DG near S14, the switching configuration from Q-learning has no violation. The result is shown in Table 4.5. Since the power was not enough to serve all loads in the system, the load at S11 which has lower priority than others was disconnected. The surplus power was utilized to charge the battery. The fault and system configuration after restoration are shown in Fig 4.9

Table 4.5: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	Off
S6	Off
S7	On
S8	On
S9	On
S10	On
S11	Off
S12	On
S13	On
S14	Off

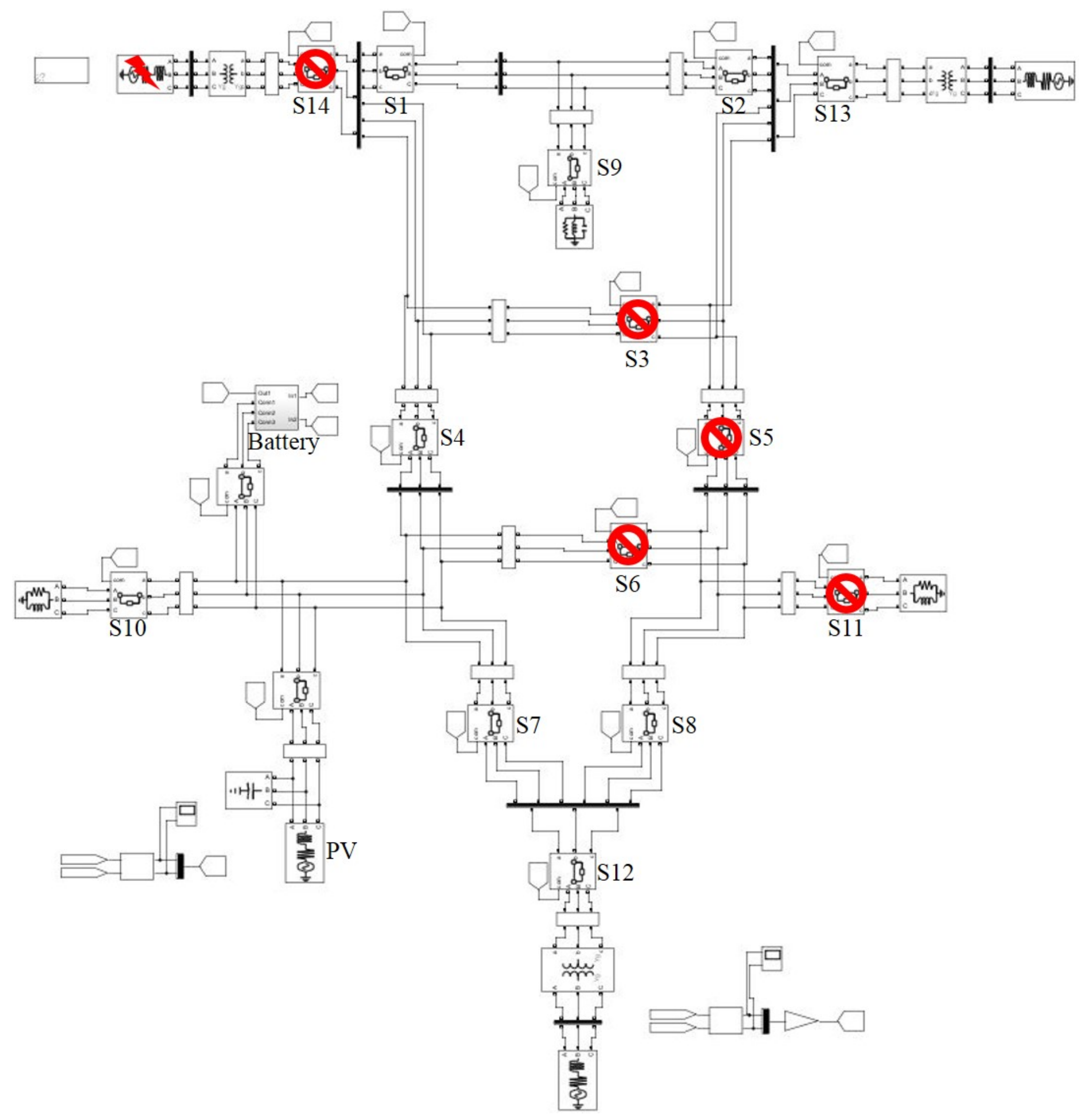


Figure 4.9: Restoration after Fault at DG

Fault at Load

When a fault happened at load near S9, the switching configuration from Q-learning violated voltage constraints. Therefore, the combination algorithm was automatically utilized. After twelve combinations, no violation was observed. The result is shown in Table 4.6. The surplus power was utilized to charge the battery. The fault and system configuration after restoration are shown in Fig 4.10

Table 4.6: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	On
S6	On
S7	Off
S8	Off
S9	Off
S10	On
S11	On
S12	On
S13	On
S14	On

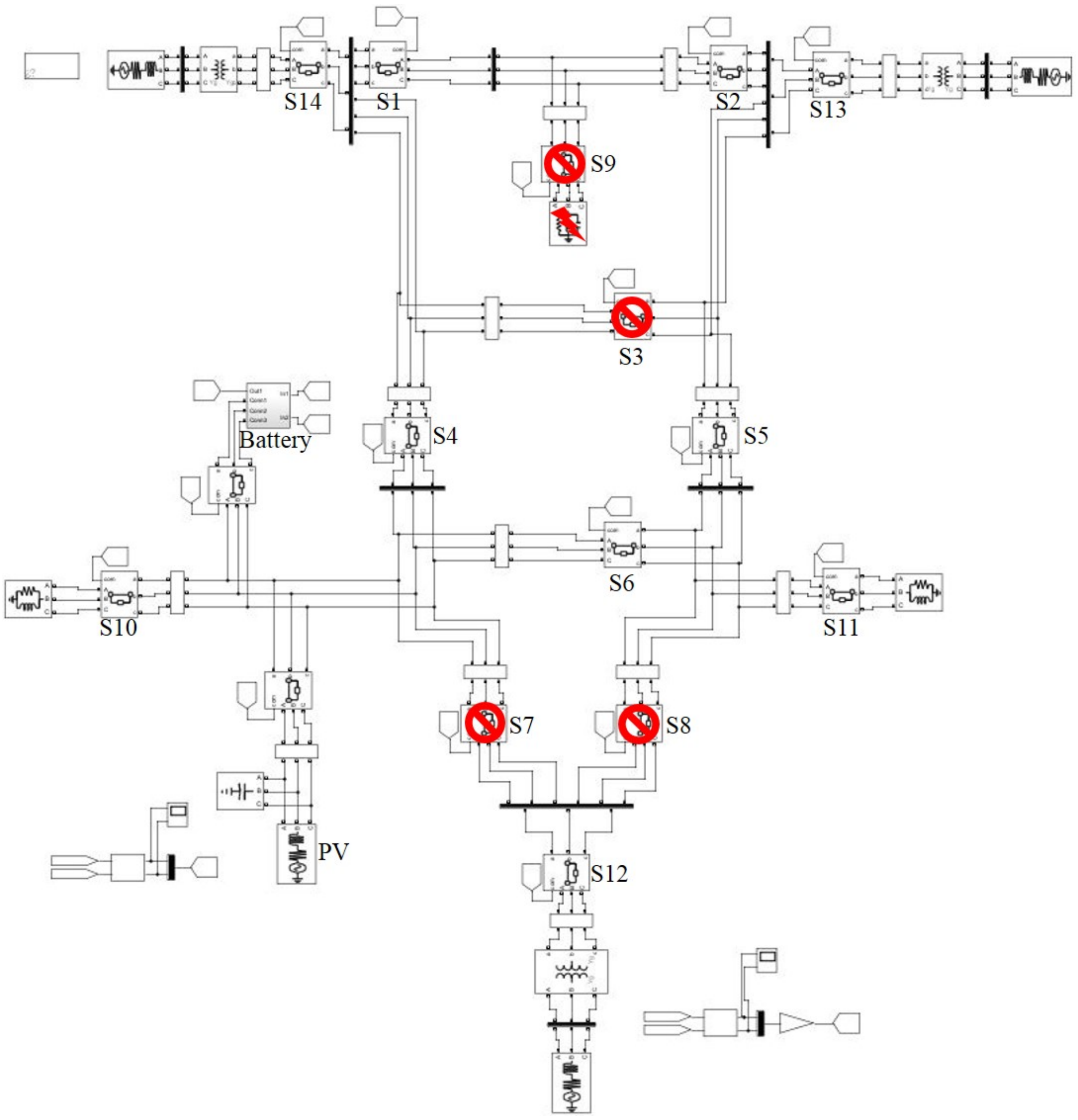


Figure 4.10: Restoration after Fault at Load

Fault at PV and Battery

When a fault happened at PV and Battery, the switching configuration from Q-learning violated voltage constraints. Therefore, the combination algorithm was automatically utilized. After five combinations, no violation was observed. The result is shown in Table 4.7. The fault and system configuration after restoration are shown in Fig 4.11

Table 4.7: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	Off
S6	Off
S7	On
S8	On
S9	On
S10	On
S11	On
S12	On
S13	On
S14	On

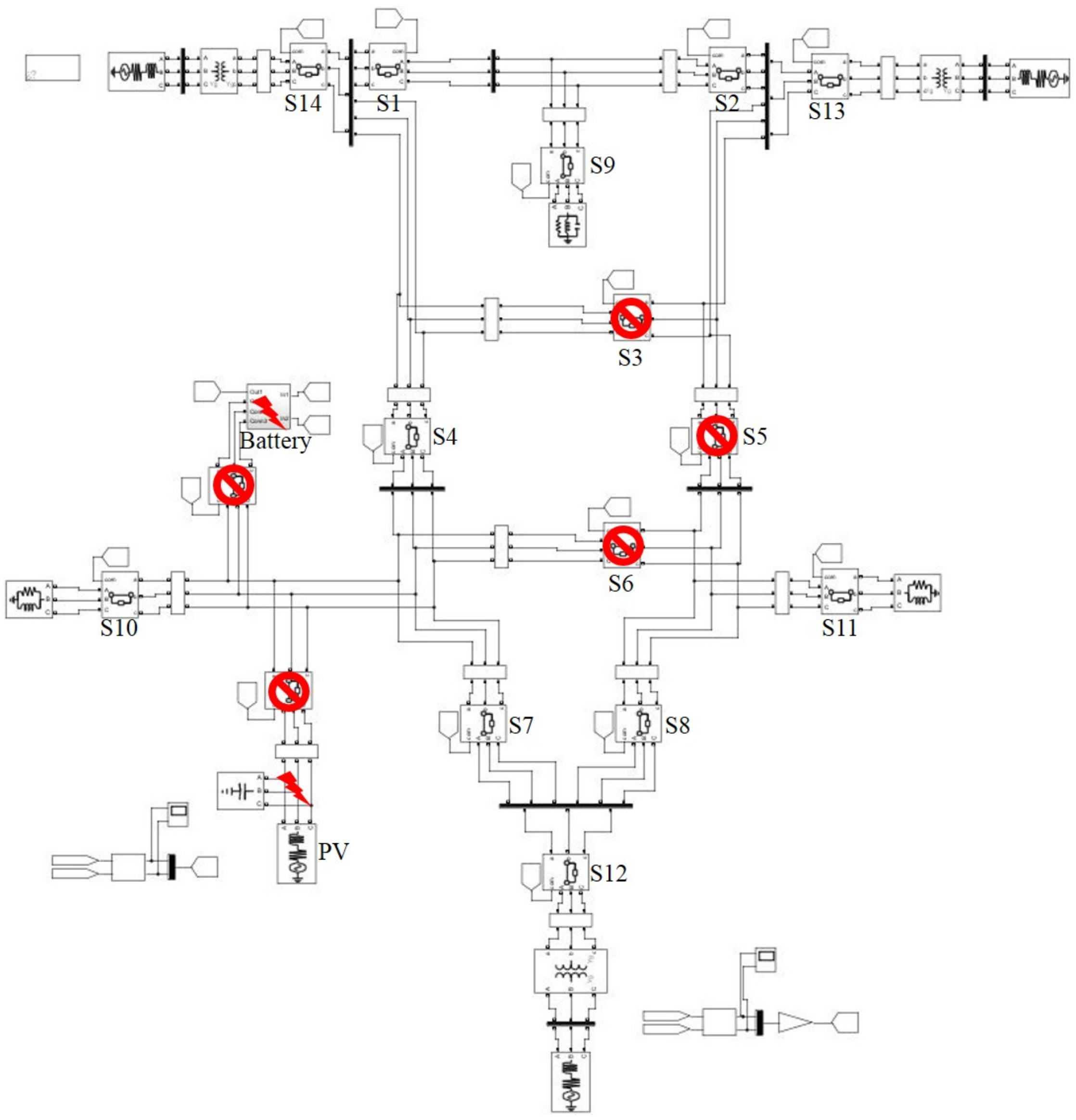


Figure 4.11: Restoration after Fault at PV and Battery

Post Restoration Algorithm Simulation

After the initial restoration of the system, there can be sudden changes from the components in the given system. For example, load demands or power generation from renewables and DGs can change. Post-restoration algorithm adjusts existing switching configuration to these changes. The algorithm keeps monitoring changes of the system and applies the restoration algorithm again to make sure all system constraints are met after the changes.

Fault at Substation

PV Output Changes After Restoration

When a fault happened at the substation, the given system is restored as shown in Table 4.3. The power generation of PV changes from 15kW to 30kW. After the change, the algorithm finds the change, and the restoration algorithm is applied. Switching configuration was acquired from Q-learning as shown in Table 4.8. The system configuration after restoration with the change is shown in Fig 4.12

Table 4.8: Switch Status after Restoration

Switch	Status
S1	On
S2	On
S3	Off
S4	On
S5	Off
S6	On
S7	On
S8	Off
S9	On
S10	On
S11	Off
S12	Off
S13	On
S14	On

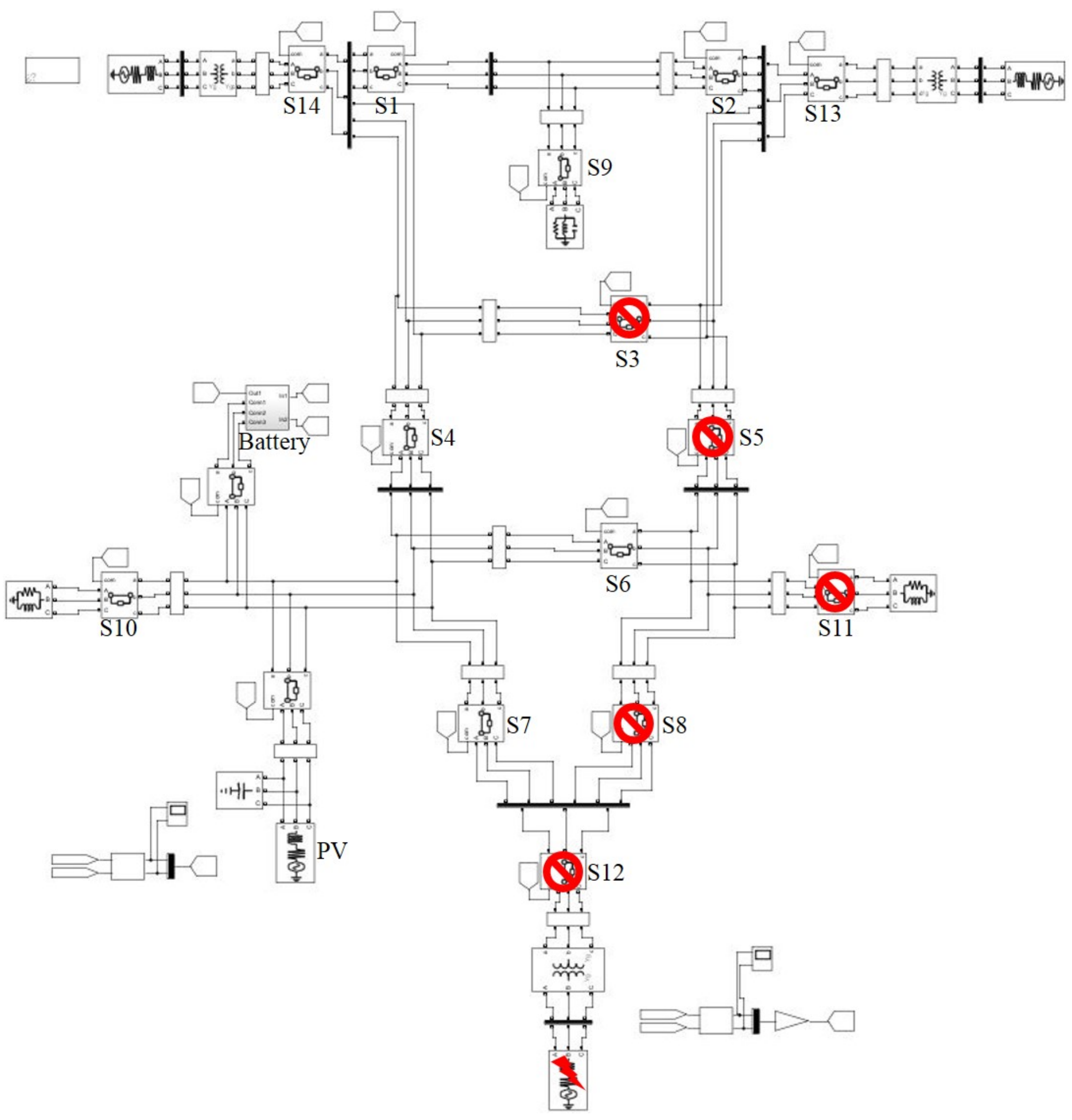


Figure 4.12: Restoration after Fault at Substation with a PV Generation Change

Summary

The algorithm proposed in the previous chapter is implemented in this chapter, and it was tested with six different scenarios including one case that the system condition changed after the initial restoration. The results show that the combination algorithm can backup the Q-learning algorithm, and this makes the restoration algorithm more resilient to fault and system condition changes. The summary of results is shown in Table 4.9.

Table 4.9: Results Summary

Fault location	Applied algorithm(s)	The number of combinations tried
Substation	Q-learning	N/A
Line	Q-learning/Combination	5
DG	Q-learning	N/A
Load	Q-learning/Combination	12
PV/Battery	Q-learning/Combination	5
Substation with a PV output change	Q-learning	N/A

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

Conclusions

This study presents a multiagent and Q-learning based restoration algorithm. The contribution of this study is that it adopts reduced combination algorithm and battery algorithm to the existing MAS Q-learning algorithm to make power system restoration process resilient. Furthermore, this study considers power system physics and provides realistic power system restoration scenarios. Lastly, the proposed algorithm shows that it can reconfigure a distribution system to handle system condition changes after an initial restoration.

Future Work

To develop this study further, first, the MAS communication between neighborhood agents can be developed and established. This will give more realistic solutions, and more research about communication issues between agents can be studied. Second, automatic reward calculation of each agent for the given system can be developed. With this study, larger systems restoration such as transmission system restoration can be developed. Lastly, fully automated restoration algorithm can be studied. A Hardware-in-the-Loop (HIL) testbed can be developed by adding hardware such as relays, satellite clock, amplifier, etc. to the current testbed. The HIL testbed will provide distribution automation system that can protect the system from faults and restore the system after clearing the faults.

APPENDIX A: MATLAB CODE

Related Matlab codes for this study have attached and the codes consist of four parts as follows.

Restoration after Fault at Substation with PVout changes

```
clear all

%Type1: Fault at Substation and restoration after certain change.
%warning('off','all') /turn off and on waring
%2,3,5,6,8,9 load
Qeval = 0;
alpha = 0.1;
beta = 10;
eta = 0.4;
n = 2; % number of available actions
Battery_size = 10;
BatOut = Battery_size;
PVOut = 15;
etc = 1.05;
etcQ = 1;
Pri =   [0 0 0 0 0 0 0 0 0.8 0.8 0.5 0 0 0];
CurLd = [0 0 0 0 0 0 0 0 100 125 90 0 0 0];
Gen =   [0 0 0 0 0 0 0 0 0 PVOut+BatOut 0 200 85 163];
GenStatus = [0 0 0 0 0 0 0 0 0 1 0 1 1 0];
Fault =     [0 0 0 0 0 0 0 0 0 0 0 1 0 0];

Battery_stat = [0 ;1];
```

```
Battery_sw = 1;

PV_sw = 1;


%%%%%%%%%%% Load value calculation

LV = zeros(14,1);

for i=[9 10 11];

    LV(i) = Pri(i)*CurLd(i);

end


%%%%%%%%%%%%%Reward Calculation for each Load

Reward = zeros(14,2);%1,4,7 should be zero

Reward(9,1) = RewardCal_st(9, [10 11]);

Reward(9,2) = -Reward(9,1);

Reward(10,1) = RewardCal_st(10, [9 11]);

Reward(10,2) = -Reward(10,1);

Reward(11,1) = RewardCal_st(11, [9 10]);

Reward(11,2) = -Reward(11,1);


%%%%%%%%%%%%%Reward Calculation for each switch

Reward(1,1) = LV(9)+LV(10)+LV(11);

Reward(1,2) = -Reward(1,1);

Reward(2,1) = LV(9)+LV(10)+LV(11);

Reward(2,2) = -Reward(2,1);

Reward(3,1) = (LV(9)+LV(10)+LV(11))-(LV(9)+LV(10)+LV(11));

Reward(3,2) = -Reward(3,1);
Reward(4,1) = LV(10)-LV(9);
Reward(4,2) = -Reward(4,1);
Reward(5,1) = LV(11)-LV(9);
Reward(5,2) = -Reward(5,1);
Reward(6,1) = (LV(9)+LV(10)+LV(11))-(LV(9)+LV(10)+LV(11));
Reward(6,2) = -Reward(6,1);
Reward(7,1) = (1-Fault(12))*(LV(9)+LV(10)+LV(11));
Reward(7,2) = -Reward(7,1);
Reward(8,1) = (1-Fault(12))*(LV(9)+LV(10)+LV(11));
Reward(8,2) = -Reward(8,1);

%%%%% fault switch reward is set to 0
for i=1:14
    Reward(i,:) = (1-Fault(i))*Reward(i,:);
end

info = zeros(1,14);
info(1:14) = 2;
epsilon = beta./(beta+info);

average = zeros(14,2); %calculate the average reward for close(1)
    and open(2)

%zone1 and zone2 present two Manager Agents
zone1 = [1,3,4,6,7,9,10];
```

```
zone2 = [2,5,8,11];
temp = 0;
for i = zone1
    temp = temp+Reward(i,1);
end
average(14,1) = temp/length(zone1);
temp = 0;
for i = zone1
    temp = temp+Reward(i,2);
end
average(14,2) = temp/length(zone1);
temp = 0;
for i = zone2
    temp = temp+Reward(i,1);
end
average(13,1) = temp/length(zone2);
temp = 0;
for i = zone2
    temp = temp+Reward(i,2);
end
average(13,2) = temp/length(zone2);

degree = zeros(14,2);

MaxD1 = max(abs(average(14,1)-Reward(zone1,1)));
```

```
MaxD2 = max(abs(average(14,2)-Reward(zone1,2)));

for i = zone1
    degree(i,1) = (Reward(i,1)-average(14,1))/MaxD1;
    degree(i,2) = (Reward(i,2)-average(14,2))/MaxD2;
end

MaxDD1 = max(abs(average(13,1)-Reward(zone2,1)));
MaxDD2 = max(abs(average(13,2)-Reward(zone2,2)));

for i = zone2
    degree(i,1) = (Reward(i,1)-average(13,1))/MaxDD1;
    degree(i,2) = (Reward(i,2)-average(13,2))/MaxDD2;
end

%deg, Reward, alpha, eta, epsilon,n
SW = zeros(14,1);
ii=[1 2 3 4 5 6 7 8 9 10 11 12 13 14];
xx = find(Fault(ii)==1); %do not calculate the action for fault
   switch
ii(xx)=[];

%calculate the switching action
for i=ii
```

```
    SW(i) = SAaction1(degree(i,:),Reward(i,:), alpha, eta,
       epsilon(i), 2);
end

% convert close =1 and open =0 from close =1 open = 2 to make
   more intuitive
SWstatus = zeros(14,1);
for i=ii
    if SW(i)==1 % close
        SWstatus(i) = SW(i);
    elseif SW(i)==2 %open
        SWstatus(i) = 0;
    end
end

%%%%%%Check Surplus power and radiality
Recheck=zeros(14,1);
i=[9 10 11];
while dot(GenStatus,Gen)-dot(SWstatus,CurLd)>89
    i=[9 10 11];
    x = find(SWstatus(i)==0);
    k = i(x);
    Recheck(k) = Reward(k,1);
    [c,d]=max(Recheck(k));
    SWstatus(k(d))=1;
```

```
end

if SWstatus(6)+SWstatus(7)+SWstatus(8)==3
    i = [6 7 8];
    Reward(i) = Reward(i,1);
    [e f] = min(Reward(i));
    SWstatus(i(f))=0;
end

if SWstatus(1)+SWstatus(2)+SWstatus(3)==3
    i = [1 2 3];
    Reward(i) = Reward(i,1);
    [e f] = min(Reward(i));
    SWstatus(i(f))=0;
end

j = 1:14;
jx = find(SWstatus(j)==1);
disp('Q-learning solution')
disp('     S1     S2     S3     S4     S5     S6     S7     S8     S9
   S10    S11    S12    S13    S14')
disp(SWstatus')

%%%Verify Q-learning solution
for i=1
```

```
SWstatus_up = SWstatus;
fprintf('Q-learning solution\n')
[viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
   IEEE_9bus_new_PV_0612_simplified_V_auto',0.1, SWstatus_up)
   ;
sum_viol(i) = length(viol_i);
if isempty(viol_i)
    disp('No violation and Switch Status:')
    disp('     S1     S2     S3     S4     S5     S6     S7     S8
          S9     S10    S11    S12    S13    S14')
    disp(SWstatus')

    TotalGen = Gen(10)+Gen(12)*SWstatus(12)+Gen(13)*SWstatus
       (13)+Gen(14)*SWstatus(14);
    TotalLd = CurLd(9)*SWstatus(9)+CurLd(10)*SWstatus(10)+
       CurLd(11)*SWstatus(11);
    AvB = TotalGen - TotalLd;
    fprintf('Surplus Power %i kW\n',AvB)
    if TotalGen - TotalLd >= Battery_size
        Battery_stat = [1 ;0];
    else
        Battery_stat = [0;1];
    end
    [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
       IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
                SWstatus_up);
            if isempty(viol_i)
                disp('No violation from Battery and Switch Status:')
                disp('     S1     S2     S3     S4     S5     S6     S7
                   S8     S9     S10    S11    S12    S13    S14')
                disp(SWstatus')
                disp('     Bat_Load   Bat_Gen')
                disp(Battery_stat')
                Qeval = 1;
                break;
            end
        end

end


SSW = zeros(14,1);
SSW(9:10)=1;%priority load
SSW(12)=1;SSW(13)=1;SSW(14)=1; %generator agent
SSW(14)=0; %fault
UnD = [1 2 3 4 5 6 7 8 11];

combn=0;
for i=5:length(UnD)
    Temp = nchoosek(UnD,i);
    Size_temp = size(Temp);
        combn = combn+Size_temp(1);
end
SW_matrix = zeros(15, combn);

counter = 0;
for i=5:length(UnD)
        Temp = nchoosek(UnD,i);
        Size_temp = size(Temp);
        for j=1:Size_temp(1)
            counter=counter+1;
            SSW(Temp(j,:))=1;
            SW_matrix(1:14,counter)=SSW;
            SumReward=0;%%%%%%%%%%%%%%%%%%%%%%caution!
            for l=1:14
                if SSW(l)==1
                    SumReward = SumReward+Reward(l,1);
                else
                    SumReward = SumReward+Reward(l,2);
                end
            end
            SW_matrix(15,counter) = SumReward;
            SSW(UnD) = 0;
        end
end
```

```
[Y,I]=sort(SW_matrix(15,:),'descend');
SW_reward_sorted=SW_matrix(:,I); %use the column indices from
   sort() to sort all columns of A.

sum_viol = zeros(30,1);

for i=1:60
    while Qeval == 0
        while (SW_reward_sorted(1,i)+SW_reward_sorted(2,i)+
           SW_reward_sorted(3,i) == 3)||(SW_reward_sorted(6,i)+
           SW_reward_sorted(7,i)+SW_reward_sorted(8,i) == 3)
            i = i+1;
        end
        SWstatus_up = SW_reward_sorted(1:14,i);
        fprintf('Combination #:%i\n',i)
        [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
           IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
           SWstatus_up);
        sum_viol(i) = length(viol_i);
        if isempty(viol_i)
            disp('No violation and Switch Status:')
            disp('      S1     S2     S3     S4     S5     S6     S7
               S8     S9     S10    S11    S12    S13    S14')
            disp(SW_reward_sorted(1:14,i)')
```

```
TotalGen = Gen(10)+Gen(12)*SW_reward_sorted(12,i)+Gen
   (13)*SW_reward_sorted(13,i)+Gen(14)*
   SW_reward_sorted(14,i);
TotalLd = CurLd(9)*SW_reward_sorted(9,i)+CurLd(10)*
   SW_reward_sorted(10,i)+CurLd(11)*SW_reward_sorted
   (11,i);
AvB = TotalGen - TotalLd;
fprintf('Surplus Power %i kW\n',AvB)
if TotalGen - TotalLd >= Battery_size
    Battery_stat = [1 ;0];
else
    Battery_stat = [0;1];
end
[viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
   IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
   SWstatus_up);
if isempty(viol_i)
    disp('No violation and Switch Status:')
    disp('     S1     S2     S3     S4     S5     S6     S7
           S8     S9     S10    S11    S12    S13    S14')
    disp(SW_reward_sorted(1:14,i)')
    disp('     Bat_Load    Bat_Gen')
    disp(Battery_stat')
    Qeval = 1;
```

```
                    break;
                end
            end
            i=i+1;
        end
end

%% Monitoring PVOutput change and update system configuration
PVOutnew = 30;
Qeval = 0;
if PVOut ~= PVOutnew
    fprintf('The output of PV changed from %i kW to %i kW\n',
       PVOut,PVOutnew)
    PVOut = PVOutnew;

etc = 1;
etcQ = 1.2;
Pri =   [0 0 0 0 0 0 0 0 0.8 0.8 0.5 0 0 0];
CurLd = [0 0 0 0 0 0 0 0 100 125 90 0 0 0];
Gen =   [0 0 0 0 0 0 0 0 0 PVOut+BatOut 0 200 85 163];%
   %%%%%%%%%%%PV generation
GenStatus = [0 0 0 0 0 0 0 0 0 1 0 1 1 0];
Fault =     [0 0 0 0 0 0 0 0 0 0 0 1 0 0];
Battery_stat = [0 ;1];
Battery_sw = 1;
```

```
PV_sw = 1;

%%%%%%%%%%% Load value calculation
LV = zeros(14,1);
for i=[9 10 11];
    LV(i) = Pri(i)*CurLd(i);
end

%%%%%%%%%%%%%%Reward Calculation for each Load
Reward = zeros(14,2);%1,4,7 should be zero
Reward(9,1) = RewardCal_st(9, [10 11]);
Reward(9,2) = -Reward(9,1);
Reward(10,1) = RewardCal_st(10, [9 11]);
Reward(10,2) = -Reward(10,1);
Reward(11,1) = RewardCal_st(11, [9 10]);
Reward(11,2) = -Reward(11,1);

%%%%%%%%%%%%%%Reward Calculation for each switch
Reward(1,1) = LV(9)+LV(10)+LV(11);
Reward(1,2) = -Reward(1,1);
Reward(2,1) = LV(9)+LV(10)+LV(11);
Reward(2,2) = -Reward(2,1);
Reward(3,1) = (LV(9)+LV(10)+LV(11))-(LV(9)+LV(10)+LV(11));
Reward(3,2) = -Reward(3,1);
Reward(4,1) = LV(10)-LV(9);
Reward(4,2) = -Reward(4,1);

Reward(5,1) = LV(11)-LV(9);

Reward(5,2) = -Reward(5,1);

Reward(6,1) = (LV(9)+LV(10)+LV(11))-(LV(9)+LV(10)+LV(11));

Reward(6,2) = -Reward(6,1);

Reward(7,1) = (1-Fault(12))*(LV(9)+LV(10)+LV(11));

Reward(7,2) = -Reward(7,1);

Reward(8,1) = (1-Fault(12))*(LV(9)+LV(10)+LV(11));

Reward(8,2) = -Reward(8,1);


%%%%% fault switch reward is set to 0

for i=1:14

    Reward(i,:) = (1-Fault(i))*Reward(i,:);

end


info = zeros(1,14);

info(1:14) = 2;

epsilon = beta./(beta+info);

average = zeros(14,2); %calculate the average reward for close(1)
    and open(2)


%zone1 and zone2 present two Manager Agents

zone1 = [1,3,4,6,7,9,10];

zone2 = [2,5,8,11];

temp = 0;
```

```
for i = zone1
    temp = temp+Reward(i,1);
end
average(14,1) = temp/length(zone1);
temp = 0;
for i = zone1
    temp = temp+Reward(i,2);
end
average(14,2) = temp/length(zone1);
temp = 0;
for i = zone2
    temp = temp+Reward(i,1);
end
average(13,1) = temp/length(zone2);
temp = 0;
for i = zone2
    temp = temp+Reward(i,2);
end
average(13,2) = temp/length(zone2);
degree = zeros(14,2);

MaxD1 = max(abs(average(14,1)-Reward(zone1,1)));
MaxD2 = max(abs(average(14,2)-Reward(zone1,2)));

for i = zone1
```

```
    degree(i,1) = (Reward(i,1)-average(14,1))/MaxD1;
    degree(i,2) = (Reward(i,2)-average(14,2))/MaxD2;
end

MaxDD1 = max(abs(average(13,1)-Reward(zone2,1)));
MaxDD2 = max(abs(average(13,2)-Reward(zone2,2)));

for i = zone2
    degree(i,1) = (Reward(i,1)-average(13,1))/MaxDD1;
    degree(i,2) = (Reward(i,2)-average(13,2))/MaxDD2;
end

%deg, Reward, alpha, eta, epsilon,n
SW = zeros(14,1);
ii=[1 2 3 4 5 6 7 8 9 10 11 12 13 14];
xx = find(Fault(ii)==1); %do not calculate the action for fault
   switch
ii(xx)=[];

%calculate the switching action
for i=ii
    SW(i) = SAaction1(degree(i,:),Reward(i,:), alpha, eta,
       epsilon(i), 2);
end
```

```
% convert close =1 and open =0 from close =1 open = 2 to make
   more intuitive
SWstatus = zeros(14,1);
for i=ii
    if SW(i)==1 % close
        SWstatus(i) = SW(i);
    elseif SW(i)==2 %open
        SWstatus(i) = 0;
    end
end

%%%%%%Check Surplus power and radiality
Recheck=zeros(14,1);
i=[9 10 11];
while dot(GenStatus,Gen)-dot(SWstatus,CurLd)>89
    i=[9 10 11];
    x = find(SWstatus(i)==0);
    k = i(x);
    Recheck(k) = Reward(k,1);
    [c,d]=max(Recheck(k));
    SWstatus(k(d))=1;
end

if SWstatus(6)+SWstatus(7)+SWstatus(8)==3
    i = [6 7 8];
    Reward(i) = Reward(i,1);
    [e f] = min(Reward(i));
    SWstatus(i(f))=0;
end

if SWstatus(1)+SWstatus(2)+SWstatus(3)==3
    i = [1 2 3];
    Reward(i) = Reward(i,1);
    [e f] = min(Reward(i));
    SWstatus(i(f))=0;
end

j = 1:14;
jx = find(SWstatus(j)==1);
disp('Q-learning solution')
disp('     S1     S2     S3     S4     S5     S6     S7     S8     S9
   S10    S11    S12    S13    S14')
disp(SWstatus')

%%%Verify Q-learning solution
for i=1
    SWstatus_up = SWstatus;
    fprintf('Q-learning solution\n')
    [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
       IEEE_9bus_new_PV_0612_simplified_V_auto',0.1, SWstatus_up)
       ;
    sum_viol(i) = length(viol_i);
    if isempty(viol_i)
        disp('No violation and Switch Status:')
        disp('     S1     S2     S3     S4     S5     S6     S7     S8
              S9     S10    S11    S12    S13    S14')
        disp(SWstatus')

        TotalGen = Gen(10)+Gen(12)*SWstatus(12)+Gen(13)*SWstatus
           (13)+Gen(14)*SWstatus(14);
        TotalLd = CurLd(9)*SWstatus(9)+CurLd(10)*SWstatus(10)+
           CurLd(11)*SWstatus(11);
        AvB = TotalGen - TotalLd;
        fprintf('Surplus Power %i kW\n',AvB)
        if TotalGen - TotalLd >= Battery_size
            Battery_stat = [1 ;0];
        else
            Battery_stat = [0;1];
        end
        [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
           IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
           SWstatus_up);
        if isempty(viol_i)
            disp('No violation from Battery and Switch Status:')
```

```
            disp('      S1     S2     S3     S4     S5     S6     S7
               S8     S9     S10    S11    S12    S13    S14')
            disp(SWstatus')
            disp('      Bat_Load    Bat_Gen')
            disp(Battery_stat')
            Qeval = 1;
            break;
        end
    end

end

SSW = zeros(14,1);
SSW(9:10)=1;%priority load
SSW(12)=1;SSW(13)=1;SSW(14)=1; %generator agent
SSW(12)=0; %fault

UnD = [1 2 3 4 5 6 7 8 11];

combn=0;
for i=5:length(UnD)
    Temp = nchoosek(UnD,i);
    Size_temp = size(Temp);
    combn = combn+Size_temp(1);
end
```

```
SW_matrix = zeros(15, combn);

counter = 0;
for i=5:length(UnD)
    Temp = nchoosek(UnD,i);
    Size_temp = size(Temp);
    for j=1:Size_temp(1)
        counter=counter+1;
        SSW(Temp(j,:))=1;
        SW_matrix(1:14,counter)=SSW;
        SumReward=0;%%%%%%%%%%%%%%%%%%%%%%caution!
        for l=1:14
            if SSW(l)==1
                SumReward = SumReward+Reward(l,1);
            else
                SumReward = SumReward+Reward(l,2);
            end
        end
        SW_matrix(15,counter) = SumReward;
        SSW(UnD) = 0;
    end
end

[Y,I]=sort(SW_matrix(15,:),'descend');
```

```
SW_reward_sorted=SW_matrix(:,I); %use the column indices from
   sort() to sort all columns of A.

sum_viol = zeros(30,1);
for i=1:60
    while Qeval == 0
        while (SW_reward_sorted(1,i)+SW_reward_sorted(2,i)+
           SW_reward_sorted(3,i) == 3)||(SW_reward_sorted(6,i)+
           SW_reward_sorted(7,i)+SW_reward_sorted(8,i) == 3)
            i = i+1;
        end
        SWstatus_up = SW_reward_sorted(1:14,i);
        fprintf('Combination #:%i\n',i)
        [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
           IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
           SWstatus_up);
        sum_viol(i) = length(viol_i);
        if isempty(viol_i)
            disp('No violation and Switch Status:')
            disp('     S1     S2     S3     S4     S5     S6     S7
               S8     S9     S10    S11    S12    S13    S14')
            disp(SW_reward_sorted(1:14,i)')

            TotalGen = Gen(10)+Gen(12)*SW_reward_sorted(12,i)+Gen
               (13)*SW_reward_sorted(13,i)+Gen(14)*
```

```
       SW_reward_sorted(14,i);
    TotalLd = CurLd(9)*SW_reward_sorted(9,i)+CurLd(10)*
       SW_reward_sorted(10,i)+CurLd(11)*SW_reward_sorted
       (11,i);
    AvB = TotalGen - TotalLd;
    fprintf('Surplus Power %i kW\n',AvB)
    if TotalGen - TotalLd >= Battery_size
        Battery_stat = [1 ;0];
    else
        Battery_stat = [0;1];
    end
    [viol_i,Mea_I,Mea_V] = const_viol_PV0612T1('
       IEEE_9bus_new_PV_0612_simplified_V_auto',0.1,
       SWstatus_up);
    if isempty(viol_i)
        disp('No violation and Switch Status:')
        disp('     S1     S2     S3     S4     S5     S6     S7
              S8     S9     S10    S11    S12    S13    S14')
        disp(SW_reward_sorted(1:14,i)')
        disp('     Bat_Load   Bat_Gen')
        disp(Battery_stat')
        Qeval = 1;
        break;
    end
end
```

```
        i=i+1;
    end
end
end
```

Q-learning algorithm for Switch Agents

```
function [ac_opti ] = SAaction1( deg, Reward, alpha, eta, epsilon
   ,n )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

Pi = [0.5 0.5];
Pi_dot = zeros(1,2);
ac1 = 1;
ac2 = 2;
Q = [0 0];

for k=1:2000
    %action 1:on action 2:off



    %adapt pi to pi_dot

    if deg(ac1)<=0
        Pi_dot(ac1) = Pi(ac1) + Pi(ac1)*eta*deg(ac1);
        Pi_dot(ac2) = Pi(ac2) + (1-Pi(ac2))*eta*deg(ac2);
    else
        Pi_dot(ac1) = Pi(ac1) + (1-Pi(ac1))*eta*deg(ac1);
        Pi_dot(ac2) = Pi(ac2) + Pi(ac2)*eta*deg(ac2);
    end

    %Pi_dot normalization
    Pi_dot = Pi_dot/sum(Pi_dot);

    %Q value update
    for m=1:2
        Q(m) = Q(m) + Pi_dot(m)*alpha*(dot(Reward,Pi_dot) - Q(m))
           ;
    end

    %random initialization
    [~,ac1] = max(Q);
    [~,ac2] = min(Q);
    Pi(ac1) = (1-epsilon)+(epsilon/n);
    Pi(ac2) = epsilon/n;


end

Q;
```

```
[~,ac_opti] = max(Q);




end
```

Reward Calculation

```
function output = RewardCal_st1(connect, disconnect)
sum = 0;
LV = zeros(14,1);
%calculate the reward for picking up load
Pri =   [0 0 0 0 0 0 0 0 0.8 0.8 0.5 0 0 0];
CurLd = [0 0 0 0 0 0 0 0 100 125 90 0 0 0];
for i = connect
    LV(i) = Pri(i)*CurLd(i);
    sum = sum + LV(i);
end
subsum = 0;

%calculate the reward for load that I might lose
for i = disconnect
    LV(i) = Pri(i)*CurLd(i);
    subsum = subsum + LV(i);
end
%multiply the probability to the reward
```

```
subsum = 1/(length(disconnect)+1)*subsum;%%%+1 could be omitted
output = sum - subsum;
end
```

Current and Voltage Constraints Check

```
function [ viol_i,Mea_I,Mea_V ] = const__viol( model, MMargin,
   SWstatus_up )

load_system(model)
sim(model)

%Compare Nominal value to Measured value to check the constraint
   violation.
Nom_I(1:11) = 20;
% Nom_I = zeros(11,1);
% Nom_I(1) = SC1.data(end);
% Nom_I(2) = SC2.data(end);
% Nom_I(3) = SC3.data(end);
% Nom_I(4) = SC4.data(end);
% Nom_I(5) = SC5.data(end);
% Nom_I(6) = SC6.data(end);
% Nom_I(7) = SC7.data(end);
% Nom_I(8) = SC8.data(end);
% Nom_I(9) = SC9.data(end);
% Nom_I(10) = SC10.data(end);
% Nom_I(11) = SC11.data(end);

Nom_V = zeros(11,1);
Nom_V = 1.0e+03 *[3.0835
    3.0835
    3.0836
    3.0836
    3.0836
    3.0835
    3.0835
    3.0835
    3.0835
    3.0835
    3.0835];

% Nom_V = zeros(11,1);
% Nom_V(1) = SV1.data(end);
% Nom_V(2) = SV2.data(end);
% Nom_V(3) = SV3.data(end);
% Nom_V(4) = SV4.data(end);
% Nom_V(5) = SV5.data(end);
% Nom_V(6) = SV6.data(end);
% Nom_V(7) = SV7.data(end);
% Nom_V(8) = SV8.data(end);
% Nom_V(9) = SV9.data(end);
% Nom_V(10) = SV10.data(end);

% Nom_V(11) = SV11.data(end);


Mea_I = zeros(11,1);

Mea_I(1) = SC1.data(end);

Mea_I(2) = SC2.data(end);

Mea_I(3) = SC3.data(end);

Mea_I(4) = SC4.data(end);

Mea_I(5) = SC5.data(end);

Mea_I(6) = SC6.data(end);

Mea_I(7) = SC7.data(end);

Mea_I(8) = SC8.data(end);

Mea_I(9) = SC9.data(end);

Mea_I(10) = SC10.data(end);

Mea_I(11) = SC11.data(end);


Mea_V = zeros(11,1);

Mea_V(1) = SV1.data(end);

Mea_V(2) = SV2.data(end);

Mea_V(3) = SV3.data(end);

Mea_V(4) = SV4.data(end);

Mea_V(5) = SV5.data(end);

Mea_V(6) = SV6.data(end);

Mea_V(7) = SV7.data(end);

Mea_V(8) = SV8.data(end);
Mea_V(9)  = SV9.data(end);
Mea_V(10)  = SV10.data(end);
Mea_V(11)  = SV11.data(end);

SWstatus_temp = SWstatus_up;
i=1:11;
j=[1,2,3,4,5,6,7,8,9,10,11];%4,6,10;
x = find(SWstatus_up(i)==0); %find switches which are opened
i(x)=[];%eliminate the swiches opened
viol_i =[];

for p=i
    if Mea_I(p)>= Nom_I(p)*(1+MMargin)
        fprintf('Switch %i violates I constraint.\n',p)
        viol_i = [viol_i p];
    end
end

for q=i
    if  (Mea_V(q)>=Nom_V(q)*(1+0.05))||(Mea_V(q)<=Nom_V(q)
       *(1-0.05))

        fprintf('Switch %i violates V constraint.\n',q)
        viol_i = [viol_i q];
    end
```

```
end


end
```

LIST OF REFERENCES

[1] M. Adibi and L. Fink, “Overcoming restoration challenges associated with major power system disturbances-restoration from cascading failures,” *IEEE Power and Energy Magazine*, vol. 4, no. 5, pp. 68–77, 2006.

[2] M. Harris. (2016) Train your reinforcement learning agents at the openai gym. [Online]. Available: https://devblogs.nvidia.com/parallelforall/train-reinforcement-learning-agents-openai-gym/

[3] E. Bompard, T. Huang, Y. Wu, and M. Cremenescu, “Classification and trend analysis of threats origins to the security of power systems,” *International Journal of Electrical Power & Energy Systems*, vol. 50, pp. 50–64, 2013.

[4] W. H. Kersting, “Distribution system modeling and analysis,” in *Electric Power Generation, Transmission, and Distribution, Third Edition*. CRC press, 2012, pp. 1–9.

[5] J. D. McDonald, B. Wojszczyk, B. Flynn, and I. Voloh, “Distribution systems, substations, and integration of distributed generation,” in *Electrical Transmission Systems and Smart Grids*. Springer, 2013, pp. 7–68.

[6] K. Divya and J. Østergaard, “Battery energy storage technology for power systemsan overview,” *Electric Power Systems Research*, vol. 79, no. 4, pp. 511–520, 2009.

[7] N. Hatziargyriou, *MicroGrids*. wiley-IEEE press, 2014.

[8] P. Asmus, “Microgrids, virtual power plants and our distributed energy future,” *The Electricity Journal*, vol. 23, no. 10, pp. 72–82, 2010.

[9] R. H. Lasseter and P. Piagi, “Control and design of microgrid components,” *PSERC Publication 06*, vol. 3, 2006.

[10] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” 1998.

[11] N. Shimkin. (2011) Reinforcement learning basic algorithms. [Online]. Available: http://www.ece.iisc.ernet.in/~aditya/E1245_Online_Prediction_Learning_F2014/ch4_RL1.pdf

[12] M. Adibi, P. Clelland, L. Fink, H. Happ, R. Kafka, J. Raine, D. Scheurer, and F. Trefny, “Power system restoration-a task force report,” *IEEE Transactions on Power Systems*, vol. 2, no. 2, pp. 271–277, 1987.

[13] J. Gutierrez, M. Staropolsky, and A. Garcia, “Policies for restoration of a power system,” *IEEE Transactions on power systems*, vol. 2, no. 2, pp. 436–442, 1987.

[14] J. J. Ancona, “A framework for power system restoration following a major power failure,” *IEEE Transactions on Power Systems*, vol. 10, no. 3, pp. 1480–1485, 1995.

[15] Y.-Y. Hsu, F.-C. Lu, Y. Chien, J. Liu, J. Lin, P. Yu, and R. Kuo, “An expert system for locating distribution system faults,” *IEEE Transactions on Power Delivery*, vol. 6, no. 1, pp. 366–372, 1991.

[16] C. Fukui and J. Kawakami, “An expert system for fault section estimation using information from protective relays and circuit breakers,” *IEEE Transactions on Power Delivery*, vol. 1, no. 4, pp. 83–90, 1986.

[17] A. A. Girgis and M. B. Johns, “A hybrid expert system for faulted section identification, fault type classification and selection of fault location algorithms,” *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 978–985, 1989.

[18] H.-J. Cho and J.-K. Park, “An expert system for fault section diagnosis of power systems using fuzzy relations,” *IEEE transactions on power systems*, vol. 12, no. 1, pp. 342–348, 1997.

[19] C.-H. Lin, H.-J. Chuang, C.-S. Chen, C.-S. Li, and C.-Y. Ho, “Fault detection, isolation and restoration using a multiagent-based distribution automation system,” in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on.* IEEE, 2009, pp. 2528–2533.

[20] X. Lin, S. Ke, Z. Li, H. Weng, and X. Han, “A fault diagnosis method of power systems based on improved objective function and genetic algorithm-tabu search,” *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1268–1274, 2010.

[21] J. Feltes and C. Grande-Moran, “Black start studies for system restoration,” in *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE.* IEEE, 2008, pp. 1–8.

[22] D. Shirmohammadi, “Service restoration in distribution networks via network reconfiguration,” *IEEE Transactions on Power Delivery*, vol. 7, no. 2, pp. 952–958, 1992.

[23] Y.-Y. Hsu and H.-C. Kuo, “A heuristic based fuzzy reasoning approach for distribution system service restoration,” *IEEE Transactions on Power Delivery*, vol. 9, no. 2, pp. 948–953, 1994.

[24] X.-m. Li, Y.-h. Huang, and X.-g. Yin, “A genetic algorithm based on improvement strategy for power distribution network reconfiguration [j],” *Proceedings of the CSEE*, vol. 2, p. 009, 2004.

[25] R. A. Jabr, R. Singh, and B. C. Pal, “Minimum loss network reconfiguration using mixed-integer convex programming,” *IEEE Transactions on Power systems*, vol. 27, no. 2, pp. 1106–1115, 2012.

[26] T. Nagata and H. Sasaki, “A multi-agent approach to power system restoration,” *IEEE Transactions on power systems*, vol. 17, no. 2, pp. 457–462, 2002.

[27] D. Ye, M. Zhang, and D. Sutanto, "A hybrid multiagent framework with q-learning for power grid systems restoration," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2434–2441, 2011.

[28] A. Felix, H. K. Nunna, and S. Doolla, "Distribution system restoration-a multi agent approach," in *India Conference (INDICON), 2012 Annual IEEE.* IEEE, 2012, pp. 1014–1019.

[29] S. L. Jayasinghe and K. T. M. U. Hemapala, "Multi agent based power distribution system restorationa literature survey," *Energy and Power Engineering*, vol. 7, no. 12, p. 557, 2015.

[30] V. Kumar, I. Gupta, and H. O. Gupta, "An overview of cold load pickup issues in distribution systems," *Electric Power Components and Systems*, vol. 34, no. 6, pp. 639–651, 2006.

[31] L. Yutian, F. Rui, and V. Terzija, "Power system restoration: a literature review from 2006 to 2016," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp. 332–341, 2016.

[32] M. Adibi and R. Kafka, "Power system restoration issues," *IEEE Computer Applications in Power*, vol. 4, no. 2, pp. 19–24, 1991.

[33] C. Andrews, F. Arsanjani, M. Lanier, J. Miller, T. Volkmann, and J. Wrubel, "Special considerations in power system restoration," *IEEE Trans. Power Syst*, vol. 7, no. 4, pp. 1419–1427, 1992.

[34] D. Lindenmeyer, H. Dommel, and M. Adibi, "Power system restorationa bibliographical survey," *International journal of electrical power & energy systems*, vol. 23, no. 3, pp. 219–227, 2001.

[35] E. M. Davidson, S. D. McArthur, J. R. McDonald, T. Cumming, and I. Watt, "Applying multi-agent system technology in practice: Automated management and analysis of scada

and digital fault recorder data," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 559–567, 2006.

[36] J. Lagorse, D. Paire, and A. Miraoui, "A multi-agent system for energy management of distributed power sources," *Renewable energy*, vol. 35, no. 1, pp. 174–182, 2010.

[37] G. Conzelmann, G. Boyd, V. Koritarov, and T. Veselka, "Multi-agent power market simulation using emcas," in *Power Engineering Society General Meeting, 2005. IEEE.* IEEE, 2005, pp. 2829–2834.

[38] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," in *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES.* IEEE, 2009, pp. 1–8.

[39] A. L. Dimeas and N. D. Hatziargyriou, "Operation of a multiagent system for microgrid control," *IEEE Transactions on Power systems*, vol. 20, no. 3, pp. 1447–1455, 2005.

[40] A. Dimeas and N. Hatziargyriou, "A multiagent system for microgrids," in *Power Engineering Society General Meeting, 2004. IEEE.* IEEE, 2004, pp. 55–58.

[41] Y. Xu, W. Liu, and J. Gong, "Stable multi-agent-based load shedding algorithm for power systems," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2006–2014, 2011.

[42] J. M. Solanki, S. Khushalani, and N. N. Schulz, "A multi-agent solution to distribution systems restoration," *IEEE Transactions on Power systems*, vol. 22, no. 3, pp. 1026–1034, 2007.

[43] F. Ren, M. Zhang, D. Soetanto, and X. Su, "Conceptual design of a multi-agent system for interconnected power systems restoration," *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 732–740, 2012.

[44] W. Khamphanchai, S. Pisanupoj, W. Ongsakul, and M. Pipattanasomporn, "A multi-agent based power system restoration approach in distributed smart grid," in *Utility Exhibition on*

Power and Energy Systems: Issues & Prospects for Asia (ICUE), 2011 International Conference and. IEEE, 2011, pp. 1–7.

[45] W.-M. Lin and H.-C. Chin, "A new approach for distribution feeder reconfiguration for loss reduction and service restoration," *IEEE Transactions on Power Delivery*, vol. 13, no. 3, pp. 870–875, 1998.

[46] X.-W. Yan, L.-B. Shi, L.-Z. Yao, Y.-X. Ni, and M. Bazargan, "A multi-agent based autonomous decentralized framework for power system restoration," in *Power System Technology (POWERCON), 2014 International Conference on*. IEEE, 2014, pp. 871–876.

[47] I. S. Baxevanos and D. P. Labridis, "Implementing multiagent systems technology for power distribution network control and protection management," *IEEE Transactions on Power Delivery*, vol. 22, no. 1, pp. 433–443, 2007.

[48] S. D. McArthur, E. M. Davidson, V. M. Catterson, A. L. Dimeas, N. D. Hatziargyriou, F. Ponci, and T. Funabashi, "Multi-agent systems for power engineering applicationspart i: Concepts, approaches, and technical challenges," *IEEE Transactions on Power systems*, vol. 22, no. 4, pp. 1743–1752, 2007.

[49] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[50] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[51] M. Restelli. (2015) Reinforcement learning - exploration vs exploitation. [Online]. Available: http://home.deib.polimi.it/restelli/MyWebSite/pdf/rl5.pdf

[52] F. Kunz, "An introduction to temporal difference learning," 2000.

[53] D. Poole and A. Mackworth. (2010) 11.3.6 on-policy learning. [Online]. Available: http://artint.info/html/ArtInt_268.html

[54] I. Rudowsky, "Intelligent agents," *The Communications of the Association for Information Systems*, vol. 14, no. 1, p. 48, 2004.

[55] M. Glavic, "Agents and multi-agent systems: a short introduction for power engineers," 2006.

[56] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," *Intelligent agents III agent theories, architectures, and languages*, pp. 21–35, 1997.

[57] S. Russell, P. Norvig, and A. Intelligence, "A modern approach," *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, vol. 25, p. 27, 1995.

[58] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *The knowledge engineering review*, vol. 10, no. 02, pp. 115–152, 1995.

[59] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[60] K. Huang, S. K. Srivastava, D. A. Cartes, and M. Sloderbeck, "Intelligent agents applied to reconfiguration of mesh structured power systems," in *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on.* IEEE, 2007, pp. 1–7.

[61] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE.* John Wiley & Sons, 2007, vol. 7.

[62] S. Poslad, P. Buckle, and R. Hadingham, "The fipa-os agent platform: Open source for open standards."

[63] H. S. Nwana, D. T. Ndumu, L. C. Lee, and J. C. Collis, “Zeus: a toolkit for building distributed multiagent systems,” *Applied Artificial Intelligence*, vol. 13, no. 1-2, pp. 129–185, 1999.

[64] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, “Multi-agent oriented programming with jacamo,” *Science of Computer Programming*, vol. 78, no. 6, pp. 747–761, 2013.

[65] J. M. Solanki and N. N. Schulz, “Using intelligent multi-agent systems for shipboard power systems reconfiguration,” in *Intelligent Systems Application to Power Systems, 2005. Proceedings of the 13th International Conference on*. IEEE, 2005, pp. 3–pp.

[66] K. Huang, D. A. Cartes, and S. K. Srivastava, “A multiagent-based algorithm for ring-structured shipboard power system reconfiguration,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 5, pp. 1016–1021, 2007.

[67] C. R. Robinson, P. Mendham, and T. Clarke, “Macsimjx: A tool for enabling agent modelling with simulink using jade,” 2010.

[68] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, “Jadea java agent development framework,” in *Multi-Agent Programming*. Springer, 2005, pp. 125–147.

[69] S. Abourida, C. Dufour, J. Bélanger, G. Murere, N. Lechevin, and B. Yu, “Real-time pc-based simulator of electric systems and drives,” in *Applied Power Electronics Conference and Exposition, 2002. APEC 2002. Seventeenth Annual IEEE*, vol. 1. IEEE, 2002, pp. 433–438.

[70] C. Dufour, S. Abourida, and J. Belanger, “Hardware-in-the-loop simulation of power drives with rt-lab,” in *Power Electronics and Drives Systems, 2005. PEDS 2005. International Conference on*, vol. 2. IEEE, 2005, pp. 1646–1651.

[71] D. Bian, M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu, "Real-time co-simulation platform using opal-rt and opnet for analyzing smart grid performance," in *Power & Energy Society General Meeting, 2015 IEEE*. IEEE, 2015, pp. 1–5.

[72] OPAL-RT. (2012) Opal-rt training slides. [Online]. Available: http://www.opal-rt.com/academy