AUTOMATIC SYSTEM RESTORATION FOR INDUSTRIAL POWER SYSTEMS

By

ANUSHA PAPASANI

DISSERTATION

Submitted in Partial Fulfillment of the Requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

Arlington, Texas, USA

May 2021

ACKNOWLEDGEMENTS

I am indebted to my supervising professor, Dr. Wei-Jen Lee, for his guidance, perpetual enthusiasm in the subject that helped me in my research and successful completion of my Ph.D.

I also extend my thanks to all my dissertation committee members Dr. David A. Wetz, Dr. Ali Davoudi, Dr. Ramtin Madani, Dr. Rasool Kenarangui and Dr. William Dillon for their comments and feedback on my dissertation.

I also thank my friends and lab mates Dr. Long Zhao, Mr. Yuhao Zhou, Ms. Kaynat Zia and Dr. Igor Matsuo from Energy Systems Research Center (ESRC). Especially special thanks to Kaynat Zia for her constant support.

Lastly, I would like to thank all EE department members of UTA, especially Gail Paniuski, for her assistance during my doctoral program.

DEDICATION

I dedicate this work to my husband Nikhil Gabbi for brainstorming and motivating me to think out of the box. Thanks for constantly reviewing all my work and being with me in ups and downs throughout this journey. I also dedicate this work to my parents Sarada(mother), Ramanatha (Father), and my-in-laws for their endless love and support. Thanks for my sister Greeshma and all my cousins. Finally, I would like to dedicate this work to all my friends for providing endless support and confidence.

ABSTRACT

AUTOMATIC SYSTEM RESTORATION FOR INDUSTRIAL POWER SYSTEMS

By

ANUSHA PAPASANI, Ph.D.

THE UNIVERSITY OF TEXAS AT ARLINGTON

Supervising Professor: Dr. Wei-Jen Lee

The power system industry often operates close to its limits to accommodate the increased demand posing a high risk of blackouts. Power system restoration techniques are utilized post breakout with the focus on load pickup and speedy recovery. In traditional heuristic methods, the load is considered to be constant after it is picked. However, from a system operation point of view, the load varies once picked. This is commonly observed in industrial loads. In Industrial systems, loads, which involve many induction motors, are started in sequence. The high starting currents of the induction motors leads to voltage sags that may affect variable speed drives and cause contactors to drop out. If the load variation and inrush currents are not considered, load at the time of pickup will be significantly underestimated at the time of pickup which might lead to a system re-collapse. Besides, one may have to prioritize the loads to help the operator during restoration. An automatic power system restoration tool is developed by using graph theory to provide an efficient restoration path and considers the priority of the loads, Cold load pickup (CLPU), Inrush currents, and load variation after picking up for a smooth and successful restoration process. Evolution of the smart grid, the Intelligent Electronic Device (IED) has been

deployed throughout the power system network for monitoring and control. Therefore, this dissertation takes advantages on the availability of IEDs to report the loads on the feeder right before the system blackout and the real-time load during the system restoration. The industrial system and IEEE 30 bus system are used as a test cases to demonstrate the effectiveness of the proposed methodology.

Table of Contents

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

TABLE PAGE

LIST OF ABBREVIATIONS

P_{GN}	Real power injected at bus N at each stage
P_{DN}	Real power demand at bus N at each stage
P_{LN}	Real power loss at bus N at each stage
Q_{GN}	Reactive power injected at bus N at each stage
Q_{DN}	Reactive power demand at bus N at each stage
Q_{LN}	Reactive power loss at bus N at each stage
V_i	Voltage at the bus i
V_{min}	Minimum voltage limit at bus
V_{max}	Maximum voltage limit at bus
P_{Gmin}	Minimum active power limit of a generator unit
P_{Gmax}	Maximum active power of a generator unit
Q_{Gmin}	Minimum reactive power limit of a generator unit
Q_{Gmax}	Maximum reactive power of a generator unit
P_{AVA}	Available real power at a given point during restoration
Q_{AVA}	Available reactive power at a given point during restoration
P_{GT}	Total rated active power of the energized generator
Q_{GT}	Total rated reactive power of the energized generator
P_{CR}	Cranking power of generator's real power
Q_{CR}	Cranking power of generator's reactive power
L_P	Total active power of the picked loads after it reaches a steady state
L_Q	Total reactive power of the picked loads after it reaches a steady state
P	Active power of the load appeared after the switch is closed

Q	Reactive power of the load appeared after the switch is closed
P_{LI}	Active power of induction motor when inrush current is observed
Q_{LI}	Reactive power of induction motor when inrush current is observed
P_L	Active power of the load that is obtained from IED right before the blackout
Q_L	Reactive power of the load that is obtained from IED right before the blackout
P_R	Real-time real power load information obtained from IED
Q_R	Real-time reactive power load information obtained from IED
V_L	Rated voltage of a motor
I_{INRUSH}	Inrush current of a motor
CLPU	Cold load pickup

Chapter 1 Introduction

1.1 Background

Loss of power to the end user is referred to as a blackout/power outage. Some power outages are planned for maintenance and upgrades. However, in some cases blackouts might occur and have a drastic impact on the people and the economy. In addition to increased demand, blackouts can also be caused due to weather, accidents, equipment failure, lightning, voltage collapse, faults, etc. [1].

- Weather: Weather is one of the most common reasons for power outages. Storms, hurricanes, floods, wildfires, tornadoes etc., are some of the extreme weather conditions that cause power outages. These natural disasters are especially dangerous because their occurrence is unexpected, and the impact is felt for long periods of time. For example, more than 4.4 million people were affected by power outage due to snowstorm in Texas in 2021 [67]. Furthermore, two-thirds of Florida's electricity customers were affected by power outage due to hurricane Irma in 2017 [11].
- Accidents: Poles and power lines can be damaged due to vehicle accidents.
- Equipment failure: Equipment failure can occur due to several reasons such as age of the equipment, accumulation of salt and dust overtime or natural disasters.

Examples of blackouts:

a. A blackout in one of the world's busiest airports- Hartsfield-Jackson Atlanta on 17 December 2017, affected 30,000 people and more than 1000 flights were canceled [28].

b. A blackout occurred in the Northeastern United States in 2003 affecting 55 million people [1], [2], [15].

c. A blackout in one of the highest populated regions of the world, India in July 2012, affected 620 million people [1].

d. A blackout in the financial district i.e., Manhattan - New York in July 2019 affected 73,000 people, stopped subway trains on tracks, traffic lights were off causing havoc in just five hours [69]. Figure. 1-1 shows Manhattan during the blackout [70].

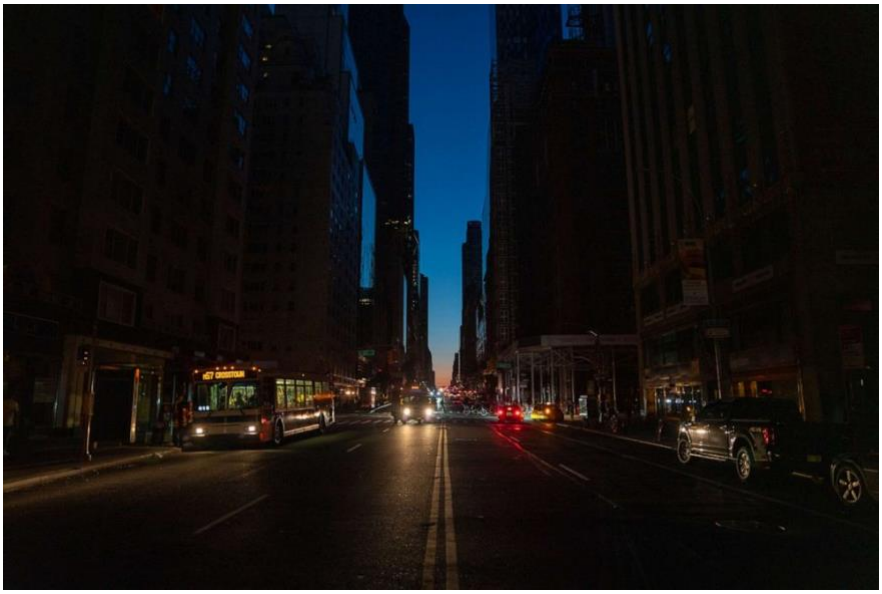


Figure 1-1 Manhattan During Blackout

1.2 Objectives and Steps of Restoration

The process of restoring the entire network i.e., getting the system back to the normal operating state after an outage/blackout is called power system restoration. To minimize the consequences of the blackout, it is essential to have an efficient restoration plan that restores the system as soon as possible.

The objectives of restoration are:

1. Minimize restoration time
2. Minimize service interruptions
3. Maximize load served
4. Maximize generation capacity

Power system restoration involves multiple steps, combinatorial operational decisions, and should comply with many technical constraints making it a very complicated process. The steps involved in restoration are as follows:

1. System status determination
2. Plant preparation
3. Generation restoration
4. Transmission path energization
5. Load restoration

These steps can be broadly classified as preparation, system restoration and load restoration. The first two steps are part of preparation, the third and fourth steps are part of system restoration, and the final step is load restoration.

1.3 Motivation

The complexity of restoring a system depends on many factors, such as the number of loads, the size of the loads, types of loads, voltages observed in the system, and the area covered by the system etc. Industrial systems predominantly use induction motors with heavy loads; the loads are started in sequence and have high voltages. These special characteristics of industrial systems cause many issues during power system restoration as listed below; an explanation of the issues below is discussed in chapter 2.2:

1. Black start capability

2. Voltage control
3. Cold load pickup (CLPU)
4. Inrush currents
5. Priority of the load
6. Load variation

The main goal of restoration is to restore power supply to the users as early as possible and with no further interruptions. Load restoration can be carried out without the entire network and generators being restored. Therefore, load restoration can be done in parallel to other stages of restoration. For example, some loads called ballast loads are required to be restored for balancing unit output and maintaining voltages and power balance within limits.

Single step load restoration activates one or more circuits at the same time, and this might cause a heavy load pickup, leading to lower voltages and frequency excursions. If load pickup is light, the number of steps will be higher resulting in delays in restoration time. Therefore, multi-step load pickup, along with cold load pickup should be considered.

The traditional method of restoration was to follow written instructions that are constructed around general rules and are executed manually [13], [14]. The problem with this approach is that the restoration situation cannot be known beforehand and requires improvisation. With the growing use of computers in every field, the whole or partial restoration process was a focus for automation. Much research has been done to solve the restoration problem; the “Power System Restoration methodologies & implementation strategies” by M. M. Adibi collects many papers that discuss this topic [28]. Many techniques have been explored such as mathematical approach, expert system, heuristic technique, and optimization technique utilizing methods like Particle swarm method, Genetic algorithm, neural network, and tabu search for power system restoration

[12]. Few researchers focused only on black start capability [6] and few on generator starting sequence [7], [8] and others are on Load restorations [64-66].

The restoration issues 1 and 2 mentioned above have been addressed by existing research [16] and [21-23]. The studies on CLPU [17], [18], [19] and [20] have focused on thermostatically controlled loads/residential loads, rather than industrial loads. The effects of inrush currents [27] and load prioritization have been addressed in isolation but are not considered during power systems restoration of industrial systems. The existing restoration techniques [24], [17], [25] and [26] consider the load to be constant after it is picked up and does not account for load variation during restoration.

Therefore, this study provides an effective automatic power system restoration procedure for industrial loads by considering CLPU, Inrush Current, Load Priority and Load Variation. This study also provides an automatic power system restoration tool that identifies the restoration path based on least electrical distance (impedance) and considers the issues discussed above during restoration to assist the operators in the event of a blackout.

1.4 Outline

This research is organized systematically as follows:

a. Chapter 2: Explains power systems restoration and deals with restoration strategies, processes, and issues involved.
b. Chapter 3: Provides the impact of Cold load pickup and Inrush currents on system restoration, mainly focusing on industrial systems.
c. Chapter 4: Provides an algorithm to solve the restoration issues, such as CLPU, inrush currents, load variation, and load priority.
d. Chapter 5: Explains the software development, tools used and critical code snippets.

e. Chapter 6: Provides the case studies used to test the effectiveness of the proposed methodology.

f. Chapter 7: Explains the results of the study.

g. Chapter 8: Summarizes research work and sheds light on potential future work.

Chapter 2 Power System Restoration

The process of restoring an entire network i.e., getting the system back to the normal operating state after an outage/blackout is called power system restoration. During blackout, restoring the system is the highest priority for operators. To achieve this, operators perform a series of time consuming and complex control actions to stabilize the system. Quick and effective system restoration is required to reduce cost and downtime. This chapter discusses restoration approaches, issues and stages of restoration (preparation – black start units, system restoration – energizing the transmission line and load restoration)

2.1 Restoration Approaches

Power system restoration is divided into three stages i.e., preparation, system restoration, and load restoration [31],[32],[33]. The first stage is time-critical and involves preparing the system to restore, i.e., to start the qualified black start generators which do not require external power to start [31] such as Hydropower plants, Diesel generator, aero-derivative gas turbine generator sets, battery storage systems, etc. [34],[35]. The black start unit characteristics are discussed in section 2.4. In stage two, the transmission line is energized, and cranking power is provided to the non-black start generators. Loads can also be picked up in the second stage depending on the generation capacity thereby maintaining the system stability i.e., parallel restoration of generators and loads. The final stage is the load restoration, all the remaining loads are picked up at this stage. The overview of system restoration process is provided in Figure.2-1 [33]

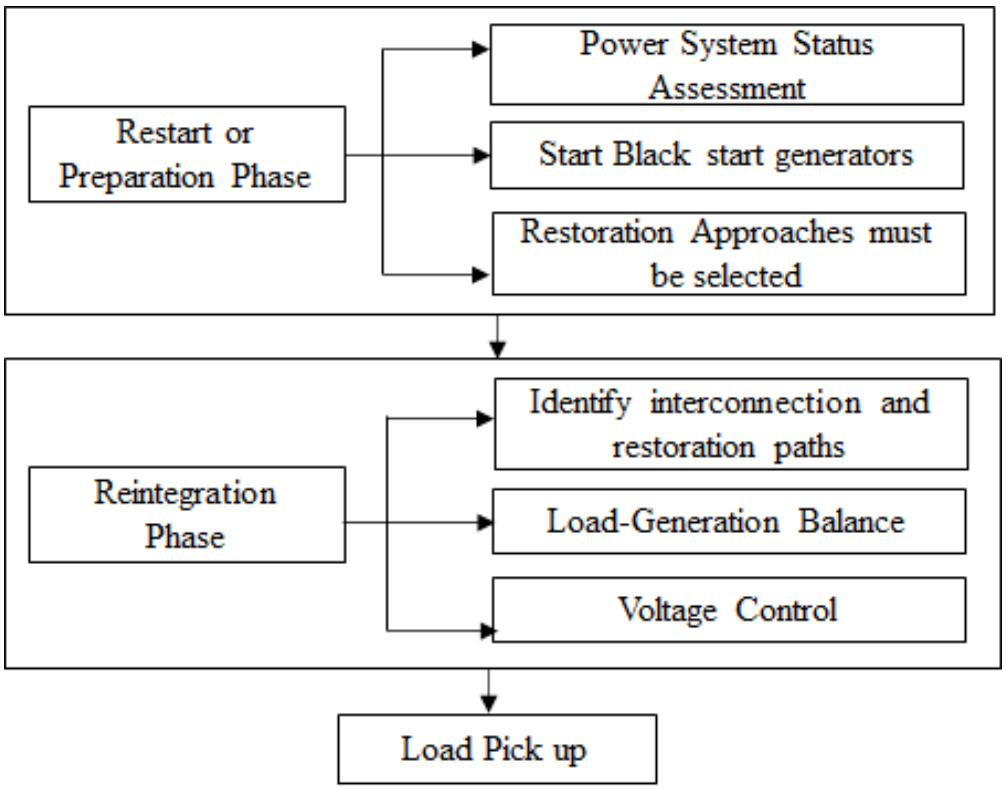


Figure 2-1 Overview of Power System Restoration

There are five main system restoration strategies. Build-Upward, Build-Downward, Build-Inward, Build-Outward and Build-Together [33]. Build-Upward and Build-Downward are the most common strategies.

2.1.1 Build Upward Strategy

This strategy is based on parallel restoration which means that the entire power system is sectionalized into small islands and parallel restoration of islands is done before resynchronizing the islands to a complete system. The islands are divided so that each island has equal or more generation capacity than the load demand of the island. The fact that restoration of islands is carried out in parallel minimizes the overall restoration time. This strategy is typically used for larger systems. The major steps involved are starting black start units, providing cranking power to non-black generators, restoration of islands/subsystems and synchronization of islands [31]. The other

advantage of this approach is to avoid the system outage due to recurrent disturbance in the affected area. Any recollapse or interruption during restoration and before the subsystems are synced will not affect the entire system but only the subsystem that caused the issue. Though this strategy has a lot of advantages, especially with restoration time and handling issues during restoration, the main disadvantage of this strategy is observed during synchronizing the subsystems together. During synchronizing the islands, the voltage magnitudes and frequencies between the subsystems must be matched or identical; voltage phase angle must be within tolerance levels. If not, the restoration might not be successful.

As discussed in [30],[31],[47], [60] the following criteria should be satisfied for sectionalizing the islands:

1. Black start capability: Each island/subsystem must have sufficient black start capability to provide cranking power to the generators.
2. Generation-load balance: Each island should have the ability to match generation and load to within prescribed frequency limits to avoid excessive frequency deviations. The load characteristics and type of loads should be taken into consideration to determine increments of load pickup [35], [47].
3. Generator cohesion: Some generators are cohesive i.e., they have similar rotor angles before the blackout. These generators should be in the same subsystem to make the grid reconnection easier [33], [57], [60].
4. Connectivity: Each island should be capable of coordinating/sharing information with other islands. The connections between the subsystems should be few to minimize the synchronization issues when subsystems are joined back [58-60]

5. Voltage control: Each island should have adequate voltage controls to maintain a suitable voltage profile.
6. Tie lines: The tie lines for islands must be capable of synchronization with adjacent subsystems.

2.1.2 Build Down Strategy

This strategy is based on sequential restoration; the bulk of the system is energized before resynchronization. This strategy is usually applied to small systems with centralized black start capability. The major steps involved are starting black start units, energizing the transmission network, and providing cranking power to non-black generators [31]. The advantage of this strategy is that there is no need of synchronizing the islands. Therefore, the probability of restoration is high. The disadvantages are that it takes longer time to restore the complete system, the high reactive power produced by unloaded high voltage transmission lines, and most importantly, in case of a large disturbance, the entire system might be affected again, leading to a recurrent large-scale blackout [33].

Other restoration strategies are explained in [31]. The factors involved in determining the restoration strategy are system size, black start capability, and location of the black start unit etc. For example, if the system is big in size, the buildup strategy is suitable. The system is sectionalized into small subsystems and for restoring small subsystems the build-down strategy is used.

2.2 Power System Restoration Issues

This research focuses on some of the issues encountered during the restoration process.

1. Black start Capability: The capability to start a generator without using support from grid and these generators are known as black start generators. Black start generators

must have sufficient black start capacity to provide cranking power to non-black start generators.

2. Voltage and Reactive Power Control: For a power system to operate in normal condition the voltage should be within certain limits. High or low voltage has adverse affects on the power system like overheating and damage of the motors and generators, and voltage collapse. The alternating current power system supplies real and reactive power. Real power accomplishes useful work while reactive power is necessary to transmit active power through transmission and distribution systems to customers, and it is also useful in maintaining the voltage. Voltages can be controlled by injecting or absorbing reactive power in the transmission systems. A decrease in reactive power causes under voltage that results in overheating and damaging of the motors, and thereby some loads might not operate properly. Relays and contractors may drop out if the voltage dip is 70-90% of the nominal voltage [36]. An increase in reactive power causes over voltage that might damage the system or decrease the lifetime of the equipment. Therefore, for reliable power system restoration it is important to maintain the voltage within acceptable limits. This can be done by energizing fewer high voltage lines, operating generators at minimum voltage levels, deactivating switched static capacitors, connecting shunt reactors, adjusting transformer taps, and picking up loads with lagging power factors [30].
3. Cold Load Pickup (CLPU): The phenomenon that takes place when the system is re-energized after an extended outage [27]. Chapter 3 provides a detailed explanation of CLPU.

4. Inrush Currents: Inrush currents are high starting currents drawn by equipment when it is turned on. The presence of Inrush Currents and its effects are explained in chapter 3.
5. Load variation: During restoration, loads do not remain constant after they are picked up. The load variation during the restoration process can be obtained from the Intelligent Electronic Device (IED) which is assumed to be present in the system. Smart grid evolution, which include the Intelligent Electronic Devices (IED) are rapidly deployed throughout the power system to offer better information on the system condition. IED is a device that is capable of sending/receiving the data or control. IED provides the real time data such as active, reactive powers, voltage, current, circuit breaker position etc. A detailed explanation of IED is given in [37] and [38]. Load variation is explained in 3.3.
6. Load priority: It is required to restore the critical loads first compared to non critical loads. Load priority is explained in 2.5.3
7. Load and Generation balance: At any point during restoration the load and generation balance should be maintained. If the imbalance occurs during picking up of auxiliary motor loads, then frequency excursions may occur which can cause the loss of already picked up loads and subsequent recurrence of blackouts. If the imbalance occurs during picking up of a cold load, then frequency deviations might occur resulting in load shedding schemes, there by losing the newly picked up loads.

Specially, this study focuses on Cold load pickup, Inrush currents, Load variation and Load priority during restoration of the Industrial system.

2.3. Black Start Generator and Non-Black Start Generators

The generating units are divided into black start and non-black start generators. In the event of a blackout, the electric grid might be out of service and thus unavailable to provide cranking power to power generation plants within its territory. Therefore, the need for a black start generator unit arises. A black start unit is a self-starting unit that can be powered on its own without support from the grid in the event of system blackout. The ideal black start units need minimal time, fuel, and equipment to restart. The black start unit should be able to provide real and reactive power to energize transmission and restart other generators. The controls used on a black start unit include a DC auxiliary support system, an ignition source, a gas turbine, and a diesel generator.

On the other hand, non-black start generators are not self-starting and require starting/cranking power to start from an external source.

2.3.1. Characteristics

The characteristics that favor black start–capability for a generator include the following [39]:

a. Lower power needs for station requirements like cooling, control and monitoring etc.
b. Quick power up of the plant to its rated output.
c. Large on-site real and reactive power capacity
d. Availability of required fuel supply to be able to function efficiently even during frequency fluctuations.
e. Ability to stabilize system frequency (large size and inertia, high ramp rate)
f. Direct transmission connections with other generating plants

2.3.2. Types of Black Start and Non-Black Start Units

Table 2-1 below shows the types of generators and their startup power requirements [55], [39].

Table 2-1 Generator Types and Startup Power

Type of Generator	**Startup Power**
Nuclear	7-8%
Thermal	7-8%
Gas Turbine	1.5-2%
Hydro	0.5-1%

a. Hydroelectric generators: These generators require less cranking power and starts up quickly. Hydroelectric generators supply adequate power to energize the transmission system, provide cracking power to other generators like coal fired units and pickup loads [39]. This generator has the ability to stabilize the system frequency.
b. Gas turbine generating units: These generators are suited for black start as they start and pickup the loads quickly. The aero-derivative gas turbine can be started remotely with local battery power [40]. Most of the industries use gas turbine driven generating units as a black start unit. Gas turbines are large in size and have the capability to pickup major transmission system elements.
c. Battery storage systems: Advancements in battery technology created the possibility of using batteries in the black start process. The advantages and disadvantages of using a battery storage system to black start the gas turbine is discussed in [41].

d. Nuclear power plants: They have large generating capacity and require several days to restart. Therefore, they are not ideal as black start generators.

e. Combined cycle units: These generators are defined as non-black start generators because they require several hours to restart, and they have more complex cooling systems [39].

f. Diesel Generators are small in size and used to supply power to start large generating units and operated by battery power. They generally cannot be used to pickup any major transmission system elements.[40]

Overall, the ability to pickup loads and major transmission systems and very less startup power makes Hydro and Gas Turbine ideal black start units.

2.4 Energizing Transmission Line

The transmission line is used to transmit the power to non-black start generators and loads. During power system restoration procedure, the major step is to energize the transmission system to enable the generation and distribution restoration. The current associated with capacitance of a line is called charging current. The transmission lines with lower voltages have smaller charging currents compared to the transmission lines with higher voltages. Energization of high voltage transmission lines may result in over voltages that is caused due to the line charging current of unloaded transmission lines. Often, multiple paths will be available in the network to the target bus. The transmission lines with lower voltages are preferred as the transmission path to avoid voltage violations [60]. To have steady transmission line energization, the line charging should be balanced which can be achieved by having enough generation that will absorb reactive power and this will also avoid overvoltage problem [42].

2.5 Load Restoration

The main purpose of the power industry is to provide continuous supply to users. In case of a blackout, the load should be restored quickly. Loads are picked up in multiple steps by balancing the voltages and power within acceptable limits. A single step restoration will cause an unusually high load that affects the system voltage constraints, and the system might re-collapse. Load pickup scheme is one of the most important considerations during power system restoration. Loads are described by different load classes (Industrial, commercial, residential) and class compositions (lighting, motors etc.). The study of load models is important to understand the load characteristics.

Load Model is a mathematical representation of the relationship between a bus voltage and the power in a bus load [43]. Load modeling is important for power system analysis, controlling and planning. The accurate representation of load gives the accurate estimate of voltage stability [44]. Load models are classified into static and dynamic loads [43].

2.5.1. Types of Load Models

According to the IEEE task force report, there are two types of load models [44].

1. Static Load model: These models express the active and reactive power at any instant of time as functions of bus voltage magnitudes and frequency. Static load models represent a static load component, e.g., resistive and lighting load, and as an approximation for dynamic load components, e.g., motor-driven loads. [43], [44]

Few static load models are described below:

a. Constant Impedance (Z) Model: This static model represents the relationship between power and voltage i.e., the power variation is directly proportional to square of the voltage magnitude [44].

b. Constant Current (I) Load model: Power variation is directly proportional to the voltage magnitude.

c. Constant Power (P) Load model: The power does not vary with changes in voltage magnitude. It is also known as constant MVA load model or constant PQ model. [44]. This dissertation uses constant PQ model.

d. ZIP model/ Polynomial Load model: This model represents the relationship between the voltage magnitude and power as a polynomial equation. That is the sum of constant impedance(Z), constant current (I) and constant power (P) components [43], [44].

2. Dynamic Load model: These models express the active and reactive powers as function of voltage and time. Differential equations can be used to represent such models. [43], [44]

Most of the literature on power system restoration considers the constant PQ model [5], [14], [24], in addition to the existing model this research considers inrush currents in the restoration process. Reference [43], [44] provides detailed explanation of the load models.2.5.2 Constraints for load restoration.

For stable operation of the system, the restoration should be done systematically by satisfying all the power flow constraints mentioned below:

1. The power balance between generator and demand must be satisfied and given by

$$P_{GN} - P_{DN} = P_{LN} \tag{2.1}$$

$$Q_{GN} - Q_{DN} = Q_{LN} \tag{2.2}$$

2. Power and voltage constraints at all the buses should be within acceptable limits to avoid damage to the power system components and for system stability during restoration.

$$V_{min} \leq V_i \leq V_{max} \quad (2.3)$$

$$P_{Gmin} \leq P_G \leq P_{Gmax} \quad (2.4)$$

$$Q_{Gmin} \leq Q_G \leq Q_{Gmax} \quad (2.5)$$

2.5.3. Load Priorities

Adequate research is done on load prioritization. These techniques discussed the parameters based on which load should be prioritized [45],[46].

Load Priority is essentially stating that during restoration some loads rank on top compared to others. For example, loads like first responders, fire stations, etc., are highly critical and should be prioritized compared to residential loads. Loads can be prioritized by many factors such as load criticality, time of prioritization, cost of interruption, expected outage duration, etc. [46]. The load restoration sequence will be determined by load size and its priority. The factors for prioritization might be different for each industrial system.

Chapter 3 Cold Load Pickup, Inrush Current and Load Variation

3.1 Cold Load Pickup

After a prolonged interruption, loads experience transient behavior when reenergized, thereby the load pickup is a critical issue within restoration. This behavior is known as the cold load pickup. Cold load pickup is the phenomenon that takes place when a distribution circuit is reenergized following an extended outage of that circuit. It is very important to bring the system back to a normal state as that causes inconvenience to the customers and loss of revenue.

At the time of restoration, it is hard to predict the exact amount of load to be restored. Hence, the understanding of the feeders will be complex. The factors that play a part in the feeder load are, large reactive power that may be drawn from the electric grid due to motor starting; typical user pattern may be altered prompting a non-standard behavior [51]. In the case of residential loads, while bringing the system back to normal state, the system demand will be higher compared to the load before the outage due to the loss of diversity. As the CLPU condition tends to put more load on the system, the voltage and current limits are violated during the restoration process. To understand the CLPU phenomenon, researchers provided various types of load models including a physically based CLPU model [48], [49]. The behavior of CLPU will be delayed exponentially on the residential feeder [52].

3.1.1 CLPU Components

CLPU is composed of two different components while restoring the load. The first one is the loss of diversity and the second refers to the Inrush currents (explained in chapter 3.2).

A. Loss of Diversity:

During normal operating conditions, all the loads that are connected to feeders will not be operational at the same time. Therefore, the maximum demand at the feeder is less than the sum

of demands of all individual loads. This concept is called diversified demand. Following an outage for a long duration, when the system is turned on, the system loses its load diversity (to hold a preset value of temperature, pressure etc. a group of loads will be randomly switched that independently cycle on and off). Post outage, all cyclic loads draw current at the same time thereby increasing the load significantly. This is mainly observed in thermostatically controlled loads such as air conditioners, water heater, refrigerators, etc. During normal conditions, the random switching of group of these loads automatically cycling on and off occurs but after a prolonged outage the thermostat contacts will be closed and waiting to run as soon as power is restored [27]. This phase is an enduring phase of CLPU due to the loss of diversity among the thermostatically controlled loads and as soon as the system is restored, that takes several minutes to achieve the diversity of loads (cyclic nature). Due to CLPU conditions, many protective devices activate and de-energizes the faultless circuit and the restoration fails; this event may cause severe voltage drops along the feeders, and transformer overloading. Therefore, it is very important to calculate the CLPU magnitude and duration of the thermostatically controlled loads for smooth restoration and provided in many research papers. [53],

Figure. 3-1 [51] provides an example of power demand demonstrating CLPU and shows three regions [51]

1. Pre-interruption: The normal operating condition where the diversified demand is observed, and the load supplied to the feeder is less than the sum of individual loads connected to that feeder.
2. Interrupted service: During service interruption the supplied load is zero but the estimated load due to loss of diversity keeps increasing depending on the duration of interruption.

3. Restored service: As soon as service is restored the demand will be higher and might be as high as the sum of all individual loads. As time increases the load diversity is restored and the demand is reduced. The peak in demand is directly proportional to the duration of the outage.

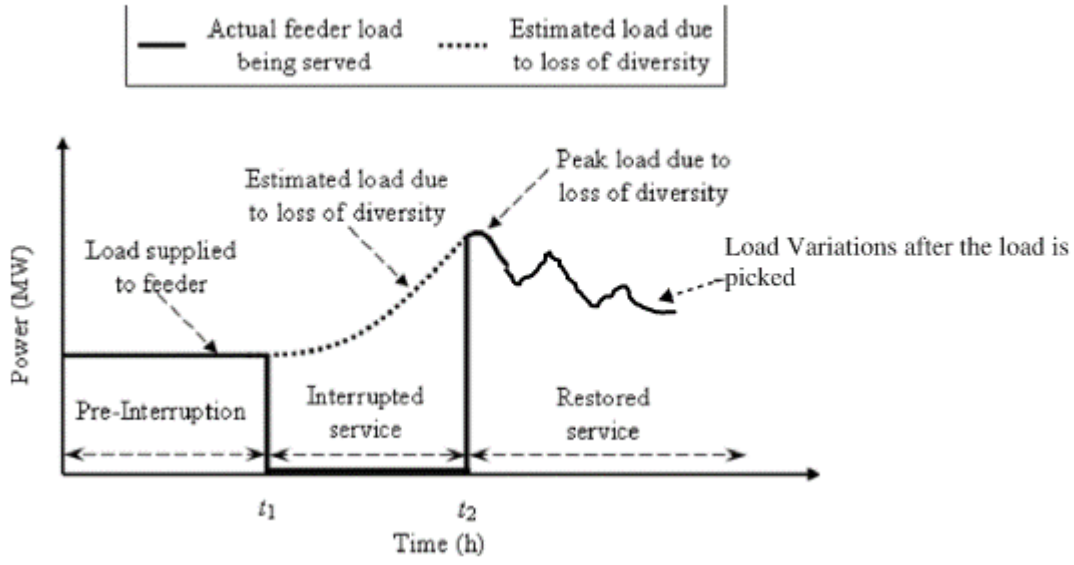


Figure 3-1 An Example of Power Demand Demonstrating CLPU

In thermostatically controlled loads, which are observed in residential loads, the loss of load diversity causes post outage load demand to be higher than pre outage load demand. Hence the cold load pickup ratio given by load under cold load condition divided by load during normal condition is greater than 1.0 [25]. Loss of load diversity is not observed in industrial systems which have a high number of induction motors as opposed to thermostatically controlled loads. Industrial systems have a starting sequence as soon as initial load is picked up other loads in the same bus are gradually added. Therefore, the cold up pickup ratio is less than 1.0 [25] and increases steadily [51].

3.2 Inrush Currents

Initially, in 1940, the behavior of load current was not important, and researchers provided the relay settings and usage of the very inverse characteristic relay to the cold load pickup (CLPU) avoiding inrush currents [19]. With a significant increase in loads, the unfavorable impact of power system elements during restoration occurred. Therefore, researchers conducted a review study in 2006 which shows that from the 1970s many studies were done on behavior of load current [19]. The behavior of load current depends on the characteristics of loads. The load characteristics might differ depending on the load type and might act differently after the outage when the system is re-energized [34].

The behavior of loads once they are energized is called inherent transient behavior. Loads are governed by both internal and external factors. Transient behavior is governed by internal factors and loss of load diversity is governed by external factors. One of the well-known phenomena of transient behavior is inrush current. As soon as the induction motor starts it draws huge amounts of currents called inrush currents often several times that of full load current. During early stages of the motor starting reactive power is higher and reduces as the motor nears the rated speed. Constant/Resistive loads do not have the same behavior as rotating loads where they show relatively constant values and can be modeled using step functions as shown in Figure. 3-2 [51].

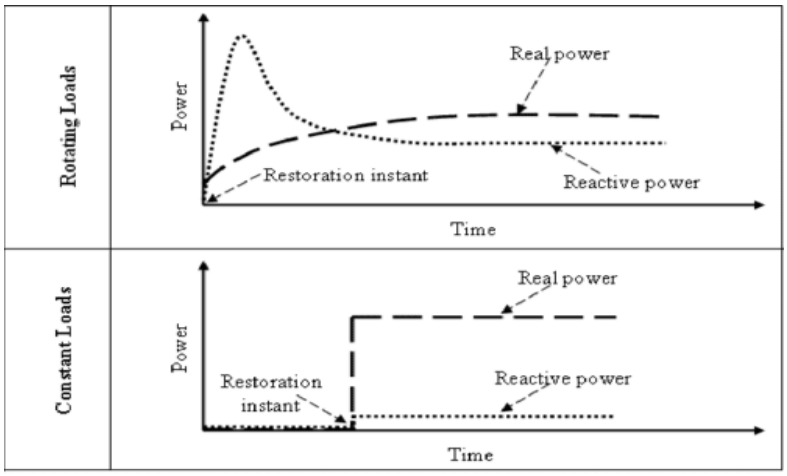


Figure 3-2 Real and Reactive Power Demand for Motor and Constant Loads

If a feeder with large induction motors is energized the high reactive power requirement may cause a voltage drop. Therefore, it is important to account for transient load behavior during restoration.

Transient behavior and loss of load diversity causes load uncertainty. In addition, load tripping, motor starting plan and user load behavior contributes to inaccurate load estimation. During restoration, the loads that have historically been easier to predict or those that deviate very less from the estimation are restored first. The loads that are tough to predict or have high deviation from estimated value are restored at a later stage.

The CLPU is dependent on the type of load/feeder. The Residential and Industrial feeders are compared in the Figure 3.3. The residential loads have high starting CLPU and gradually decreases with time, on the flip side industrial loads demonstrates steady increase in CLPU with time as shown in Figure. 3-3 [51]

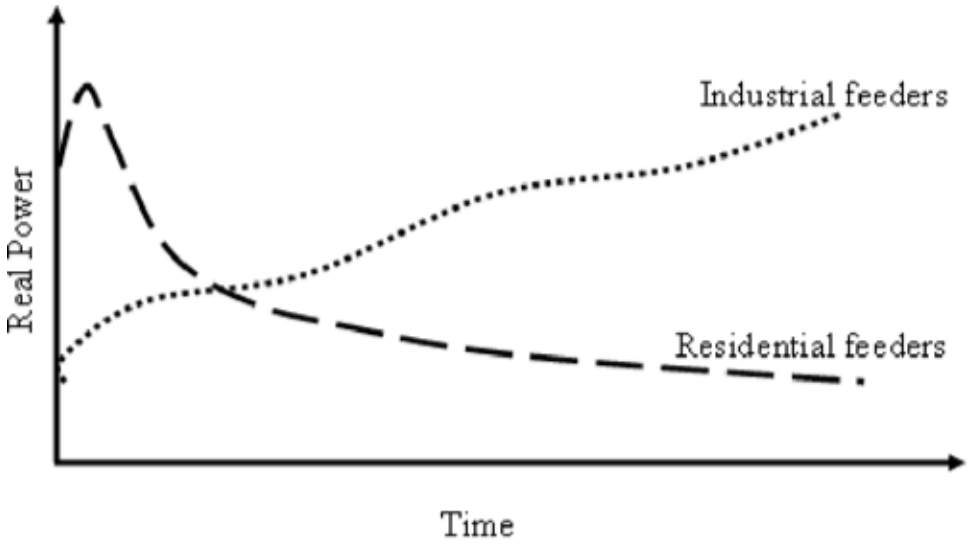


Figure 3-3 CLPU Characteristics for Industrial and Residential Feeders

3.3 Factors Influencing CLPU

This section is taken from the IEEE committee report published in 2009 [27]. There are many factors that determine the magnitude and duration of a CLPU like outage duration, weather, outage causes, types of loads etc.

Cold load pickup is dependent on many factors:

1. Type of Load: Load types can be categorized in many ways. From the perspective of cold load pickup, the most important load characteristics are initial inrush upon re-energization and sustained inrush in excess of normal continuous load. Most types of load incur an initial inrush upon energization. The most common load types are Lighting, Motor load, residential loads etc.
2. Weather: The weather-related factors such as temperature, wind, humidity, sunlight, snow, storm, etc., affect the cold load pickup duration following an outage. For example, extremely cold or hot temperatures increase the duration of cold load pickup. Storms might not have a direct impact on cold load pickup but may cause more systems

to be damaged thereby increasing the duration of outage which increases the duration of the cold load pickup.

3. Load Level and Time of the Day: The amount of load already connected has an impact on cold load pickup. For new facilities with less load connected to the feeders the cold load pickup might not be significant because the distribution lines will be designed estimating future load growth. The load factor for these circuits might be less than 50%. As new customers are added on and existing customers add more load, the load factor and cold load pickup duration increases. Residential, commercial, industrial or agricultural load types depending on region, season, and day of the week have day profiles. A typical day profile for workdays in winter has one peak load period in the morning and another peak load period in the late afternoon. Therefore, estimating the cold load pickup depending on the load before outage would be misleading.
4. Outage Duration: The duration of an outage has a direct relation with duration of cold load pickup. As the duration of outage increases the load demand also increases due to loss of load diversity. The load will lose diversity if the outage exceeds the longest cycle time of equipment. For example, for a device that cycles every 15 minutes if the outage occurs for more than 15 minutes, then its load diversity will be lost. This increased demand has to be considered during cold load pickup.

3.4 Effects of Inrush currents

Induction motors generate high starting currents, called the inrush currents or input surge currents. These inrush currents lead to oscillations that can harm the shafts, gears, and belts, etc. The magnitude and duration of the inrush currents depend on the load type. For example:

- The incandescent bulb has high inrush currents due to its low resistance and lasts for 0.1 seconds (until it reaches its operating temperature [34].
- The thermostatically controlled loads have small, single-phase motors that draw 5-8 times the rated current, but they last only for 0.5 seconds [34].
- The industrial system dominant with induction motors draws 5-8 times the rated current [27], [39] and may take several seconds to reach its operating speed.

75-80% of the motors in industries are induction motors [50]. Induction motors have extremely low power factors during the initial startup phase. The combination of high starting currents and poor power factors cause significant voltage sag, and it is a common power quality problem in industries. Voltage sag may cause the relays and contactors to drop out. The motor controller like a variable speed drive is sensitive to the voltage sag and might shut down. Therefore, it is very important to accommodate the inrush currents during system restoration.

When energizing lightly loaded transmission lines or underground cables, the excessive VARS generated by the undercompensated high voltage lines can increase voltages to unacceptably high levels which are referred to as sustained power frequency overvoltages. If not controlled, these voltages could cause serious reactive power imbalances, resulting in generator self-excitation, transformer overexcitation, and harmonic distortions.

3.5 Load Variation After Pickup

In power system restoration papers [17] and [24-26], the authors considered the load to be constant after it is picked up. However, from a system operation point of view the load varies after it is picked up. If this load variation is not considered, we might be underestimating the load in the system and the system might re-collapse. With the development of smart grid, the real-time load information/changes in load can be obtained from Intelligent Electronic Device (IED) or the load

forecasting can be used to predict the changes in load once picked up by using the historical load information provided by IED. IED stores the data until the blackout, giving us visibility as to how the network was moments before the blackout.

3.6 Discussion

There is no substantial paper that considers inrush currents for the algorithm during load restoration. If these inrush currents are not accounted for, then we will be underestimating the load at the time of pickup. The increased demand (due to inrush currents) of synchronizing power for the system might not satisfy the constraints of power balance, voltage limits, etc., and might eventually end in a system re-collapse. Therefore, this study calculates the CLPU and inrush currents for all the loads and selects the load to pickup.

This research considers the load value reported by IEDs on the feeder right before the blackout. That load value is used as the base value to calculate CLPU value. In industries, 70-80% of the loads are induction motors and 20-30% are remaining loads. Therefore, CLPU power is calculated for 80% of the load given in equation 3.1. CLPU is the function of time, and the load variation with respect to the time comes from 20% of the static load. In other words, the induction motors need static load for operation.

$$P_{CLPU} = 0.8 * P_L \tag{3.1}$$

The inrush currents for these induction motors have a high impact on the restoration process. Therefore, the inrush currents are calculated for 80% of the induction motor loads, and the total load appeared after the switch is closed which is the 80% induction motor load plus the 20% of the static load.

Chapter 4 System Restoration Algorithm and Flow Chart

There are multiple methods to solve the restoration problem that can be broadly classified into knowledge-based approaches and mathematical programming technique.

1. Knowledge-based approach: It was used for restoration planning in the earlier days. Expert system which is a knowledge-based system was regarded as a successful approach. Since it depends on knowledge of practices and simulations which are company specific, it is not always guaranteed to find restoration solution in an unfamiliar setting and there by not scalable [54]
2. Mathematical programming approach: Either optimization technique or graph theory constitutes a mathematical programming approach.
    a. Optimization technique: This technique compares various available solutions iteratively until it arrives at the best/satisfactory solution is reached. Even though computer-based systems are used, the time taken to obtain a solution increases dramatically with complexity of the system [55]
    b. Graph Theory: Power systems can be represented as graphs and graph theory techniques can be utilized to solve restoration problems. The graph theory minimizes the drawback of optimization technique. The shortest path finding algorithm (Dijkstra's algorithm) is used to find the shortest path to restore the network.

According to industry practice, the non-black start generators closer to the energized generating units are started first for efficient and reliable restoration. [4].

The main objective of this study is to provide an efficient restoration method by minimizing restoration duration and mitigating the risks of system recollapse. To achieve the objectives, it is important to identify the restoration path as early as possible. The graph search algorithm can be used to identify the best possible restoration path to provide cranking power to non-black start generating units and pickup loads along the priority order. The bus is considered as a node and the line impedance (electrical distance) is employed as a weight. A path with least line impedance is selected between two buses.

4.1 Shortest Path Algorithm

Dijkstra's algorithm is used to find the shortest path. The algorithm calculates the shortest path between the source node and all other nodes. The steps involved in Dijkstra's, along with an example are given below:

1. Initially all the nodes are set to unprocessed/unvisited and starting node is selected as shown in Figure 4-1.

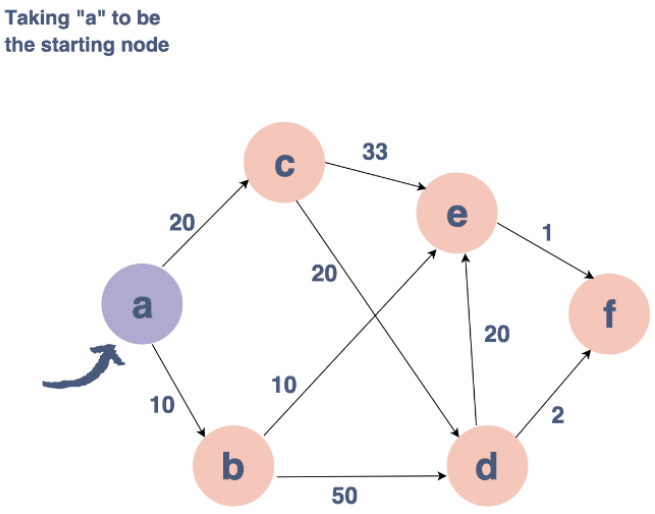


Figure 4-1 Step1: Starting Node Selection

2. The source bus/node is set to current node and marked as 0 and all the other buses/nodes are marked as infinity as shown in Figure 4-2.

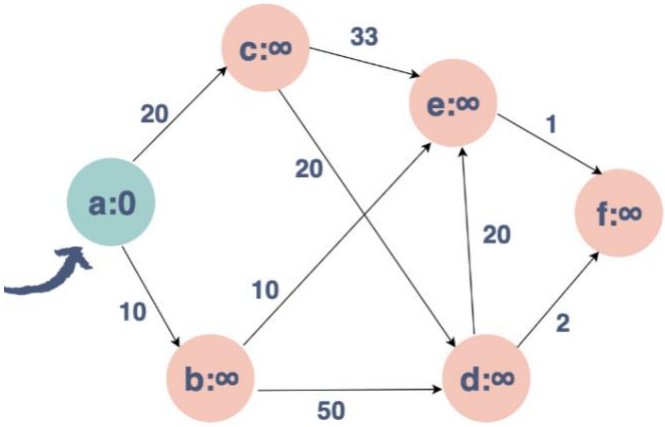


Figure 4-2 Step 2: Initializing Nodes

3. Figure 4-3 demonstrates that the distance between current node and all the immediate neighboring nodes is calculated by adding the current node with weight (line impedance) between the nodes.

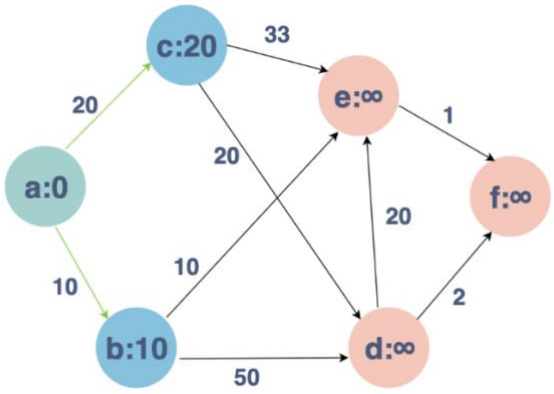


Figure 4-3 Step 3: Processing Neighboring Nodes

4. Compare the distance and update with the least distance and keep track of the path to the processed node. Once all the neighboring nodes are visited then mark the current node as visited/processed as shown in Figure 4-4.

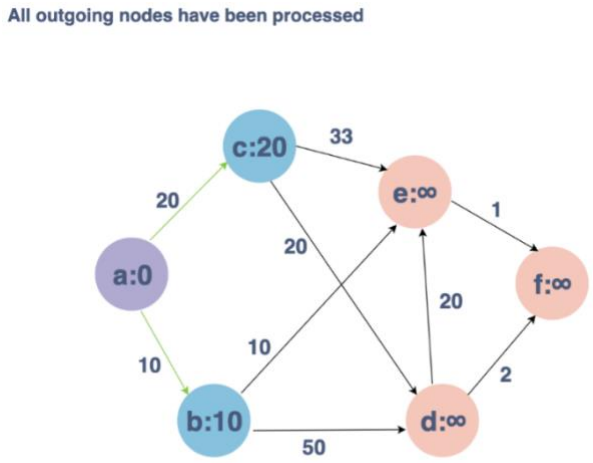


Figure 4-4 Step 4: Identifying Available Paths

5. If the destination node is set as processed/visited then stop, else repeat the steps above by considering the unvisited node with smallest distance as current node. The final state is shown in Figure 4-5.

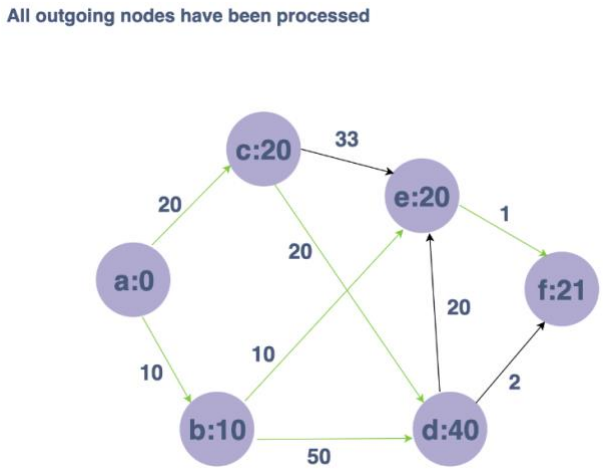


Figure 4-5 Step 5: Process All Nodes

4.2 Power System Restoration Flowchart

In this study, the blackout scenario is simulated by disconnecting the motor contactors of the motor loads and the associated control circuit along with all other system components. The flowchart for the proposed restoration procedure is given in Figure. 4-6 followed by detailed explanation of each step in the restoration sequence.

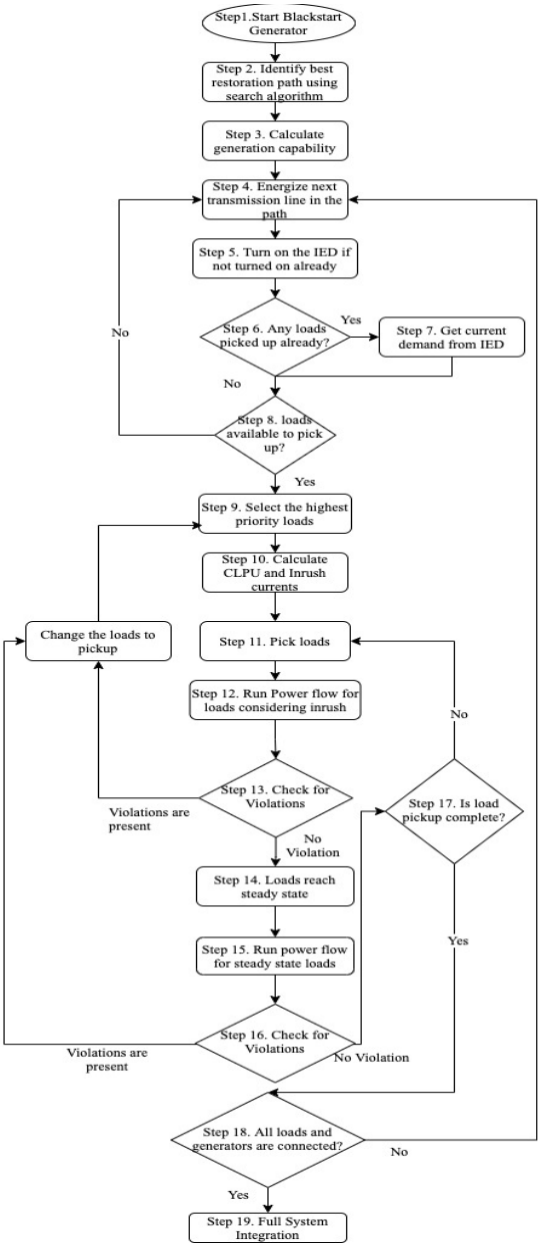


Figure 4-6 Flowchart of the Proposed Procedure

Step 1: Start Black start Generator

The process of restarting the generators after a blackout without relying on external transmission network/power from the grid is known as a black start. After the blackout, the qualified black start generator will be started first to initiate the restoration process. Though it is common to use a hydro generating station as a black start generator due to less starting power requirement, the quick-starting gas turbine-driven generating unit is used as a black start unit in industrial systems [35]. The gas turbine is coupled with an onsite diesel generator set to start the system. Some important characteristics of black start generators to be considered are fast restart, sufficient on-site real and reactive power capacity, ability to stabilize system frequency, transmission path to other generators, etc., [39]. The other available gas turbine or steam turbine variants in the industrial system are considered to be non-black start generators and the startup power requirements for these generators are based on generator type and illustrated in Table 2.1.

Step 2: Dijkstra's - Search Algorithm

The algorithm used in this paper identifies the optimal path in the weighted graph with non-negative weights [24], [46]. The tool developed implements the algorithm in python, and the restoration is carried out along the optimal path.

We are considering buses as edges and electrical distance (impedance) on the line as weight. The algorithm, when applied to the entire network, gives us the ideal traversal path i.e., by considering minimum electrical distance (impedance) from the bus connected to the black start generator to each bus in the network. This path is taken as the shortest restoration path and all the generators and loads are picked up along the path. The loads are picked up based on the priority order and availability capacity if there are multiple loads along the path.

For example, in Figure. 4-2 from reference [26], there are multiple paths connecting buses 2 and 16. The path 2,3,18,17,16 (solid line) is closest but when considering electrical distance, the shortest path is 2,25,26,27,17,16 (dotted line). So, the path 2,25,26,27,17,16 (dotted line) is selected over the 2,3,18,17,16 (solid line).

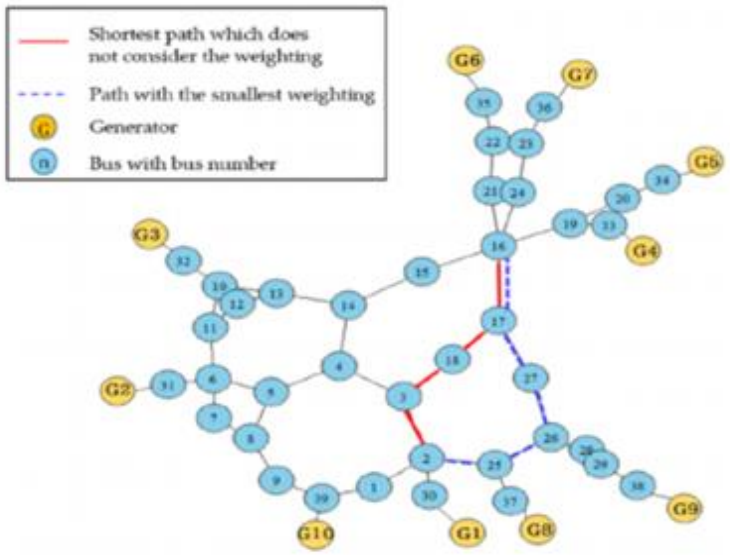


Figure 4-7 An Example of Dijkstra's Algorithm

Step 3: Calculate Generation Capability

Generation capability is the available power that can be used to provide cranking power to non-black start generators and pickup the loads in parallel. Equations 4.1 and 4.2 are used to calculate the available capability of a generator at each stage.

$$P_{AVA} = P_{GT} - P_{CR} - P_R \tag{4.1}$$

$$Q_{AVA} = Q_{GT} - Q_{CR} - Q_R \tag{4.2}$$

Step 4: Energize Transmission Line

The energization is carried out along the shortest path provided by Dijkstra's Algorithm. The line along the shortest impedance path is selected and energized followed by load pick given in further steps. The non-black start generators also provided the cranking power along the restoration path. The system should satisfy the below conditions at any time during the restoration process:

a. The power balance between generation and demand should be satisfied as shown in equations 2.1 and 2.2.
b. The power flow through a line should not exceed its capacity.
c. The voltage limits must be maintained within limits throughout the restoration process illustrated in equation. 2.3.

Step 5,6 and 7: Turn on Intelligent Electronic Device

After the blackout, during restoration, IEDs are energized along with the transmission line in step 4. As soon as IEDs are energized, it will report the pre-blackout load data which is the base value for CLPU of each load. IED will report the load consumption/real-time load after loads are picked up.

Step 6 checks if any loads along the path are already picked up, and if any of the loads are picked up then real-time load information is obtained from IED (step 7). The available generation capability is recalculated considering the real-time load information.

If no loads are picked up, then step 8 is processed, to see if any loads are available for pickup.

Step 8,9: Select the Loads to Pickup

Step 8 checks if any loads are available along the shortest path for pickup. If available, then the loads are selected according to their priority order (step 9), and step 10 is processed else step 4 is processed.

Step 10: Calculate the CLPU and Inrush Currents

It is very important to determine the CLPU and Inrush current values for selected loads to avoid incorrect estimation of loads that can lead to system re-collapse. This paper considers the load value reported by IEDs on the feeder right before the blackout as the base for calculating CLPU value and considers 80% of the loads as induction motors. The CLPU value is given by 0.8* P_L where P_L is the active power of the load obtained from IED right before the blackout and the remaining 20% is the static load.

The active and reactive power of the induction motor loads when inrush current is observed are given by equations (4.3) and (4.4)

$$P_{LI} = \sqrt{3} * V_L * I_{INRUSH} * cos\emptyset \tag{4.3}$$

$$Q_{LI} = \sqrt{3} * V_L * I_{INRUSH} * sin\emptyset \tag{4.4}$$

The total load appeared after the switch is closed is given by (4.5) and (4.6) which is the induction motor power load when inrush current is observed plus the static load:

$$P = P_L - 0.8 * P_L + P_{LI} \tag{4.5}$$

$$Q = Q_L - 0.8 * Q_L + Q_{LI} \tag{4.6}$$

Step 11: Pickup the Loads

The loads are picked up based on priority. According to the industrial setup, each motor load is split into a large number of small, medium, and few large-size motors [35]. It is common in the industry to start the motors of identical size within each load as a group because they perform

a similar operation. This research assumes motors of identical size are grouped together in each load while restoring the load.

The motor startup sequence is very important to avoid voltage sag issues. This voltage sag will cause the contactors of already online motors to open and may cause variable speed drives to shut down which results in system recollapse. Therefore, industries start a large group of induction motors that have high reactive power first to minimize the impact on other loads.

A high value of generation capacity is needed to pickup the motor group with high reactive power; the highest value of generation capacity is present when no motor group is picked up. Therefore, the groups within each load are picked up in decreasing values of reactive power. This way there is a good possibility of successfully picking up the group with high active and reactive power.

A motor/load group is picked up if it has highest priority and its active and reactive power are less than the active and reactive power of available generation capability. If any group cannot be picked up, then it is tagged as not picked and that will be re-visited after sufficient generation capacity is available. Once the group is picked up, it will reach a steady state after a short period i.e., once the loads are in stability limits. Effective generation capability will be reduced since part of it is required to support the load that is picked up.

Step 12,13,14,15,16: Run Power Flow and Check for Violations

Run the power flow for the load picked up in step 11 with inrush current load values. If power flow converges that indicates that the system survives inrush. If violations are present, then change the load and proceed to step 9. After the load reaches the steady state, proceed to step 17.

Step 17,18: Check all the Generators and Loads are Connected

If the current load pickup is not completed, then continue picking up the next group in the decreasing order of active and reactive power, until all the groups are finished processing within a load. After a load is picked up the next load in the priority order will be processed.

If all generators and loads are not connected, then go to step 4. Otherwise, go to step 19 for full integration of the system.

All the steps and constraints discussed in the flowchart are applied to develop the software. Chapter 5 gives the information on software development.

Chapter 5 Software Development

The primary reason for approaching the research from programming point of view is the ability to customize the restoration process and potential for future improvement. This section explains the implementation part of the research. It briefly talks about technology used, environment dependencies, and programming language. This section also walks through the code snippets used in implementation.

5.1 Technology

5.1.1 Backend Coding: Python

Python is an interpreted, object oriented, and high-level programming language. Python is one of the most used programming languages. The syntax is clear and concise, reducing the learning curve and increasing the speed of development. Python has a high number of libraries to cater to a wide variety of specific use cases. The main reason for choosing Python is the availability of the Pandapower library to simulate power systems and its ability of seamless integration with any front-end applications.

5.1.2 Frontend Development: HTML

HTML stands for Hyper Text Markup Language; it is the standard markup language used for developing web pages. HTML is easy to use, understand, and all browsers support it.

5.1.3 Frontend Development: Bootstrap

Bootstrap is used to make the web application responsive, adjusting the display based on screen size to display the content well. Bootstrap is easy to use, light weight, customizable, and supports cross browser compatibility.

5.1.4 Frontend Development: jQuery

jQuery is a lightweight scripting language. It sits on top of JavaScript. It is mainly used for transversal, event handling, animation, etc., in HTML without having to write huge lines of code in JavaScript.

5.1.5 Development Environment/IDE: Visual Studio Code

Integrated Development Environment (IDE) is a software that consolidates the tools needed for software development and testing.

The IDE used for this development is Visual Studio Code, and it has massive library of extensions, supports IntelliSense and it also served as source code repository to backup code for this implementation.

5.2 User Flowchart

5.2.1 User Input:

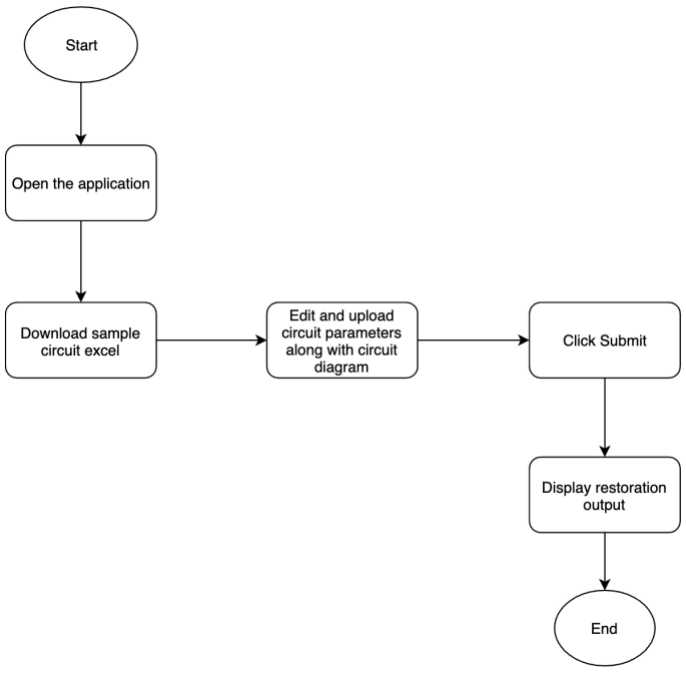


Figure 5-1 User Flow Diagram

Figure 5-1 provides the user flow diagram. The user input to the restoration application is handled by web page. In the landing page the user can upload the network information in an excel sheet and user can upload image of the network. The landing page also has an excel sheet with a preloaded custom system for download as a reference. The user can download the sample system, modify it to the user's system values and upload it as a custom system. The preformatted excel sheet has following tabs.

Generator: The generator tab has information about the generators in the system. The name of the generator, the bus it is connected to and power factor values. Table 5-1 provides the generator parameters description [61].

Table 5-1 Generator Parameters

Parameter	**Description**
name	Name of generator
bus	Index of the bus the generator is connected to
P_mw	Real power of the generator
Vm_pu	Voltage set point
Sn_mva	Nominal power of the generator
Max_q_mvar	Maximal reactive power of the generator
Min_q_mvar	Minimal reactive power of the generator

Transformer: Both step up and step-down transformers information is given in this sheet. The step up and step down is differentiated by combination of fields Hv_bus, Lv_bus and Tap_side. Hv_bus is the bus on the high voltage side of the transformer, Lv_bus is the bus on the low

voltage side of the transformer, Tap_side gives the position of the tap charger. Table 5-2 provides the transformer parameters description [61].

Table 5-2 Transformer Parameters

Parameter	**Description**
name	Name of Transformer
Hv_bus	High voltage bus index of the transformer
Lv_bus	Low voltage bus index of the transformer
Sn_mva	rated apparent power of the transformer [MVA]
Vn_hv_kv	rated voltage at high voltage bus [kV]
Vn_lv_kv	rated voltage at low voltage bus [kV]
Vk_percent	short circuit voltage [%]
Vkr_percent	real component of short circuit voltage [%]
Pfe_kw	iron losses [kW]
I0_percent	open loop losses in [%]
Shift_degree	transformer phase shift angle
Tap_side	defines if tap changer is at the high- or low voltage side
In_service	specifies if the transformer is in service.

Bus: This sheet has the bus index, bus name and rated voltage of the bus. The bus index in this sheet is used to identify the bus to which other components are connected. Table 5-3 provides the bus parameters description [61].

Table 5-3 Bus Parameters

Parameter	Description
name	Name of bus
bus	Index of the bus
Vn_kv	rated voltage of the bus [kV]

Line: Line connected two buses, From_bus and To_bus gives the buses to which the line is connected. Table 5-4 provides the line parameters description [61].

Table 5-4 Line Parameters

Parameter	Description
name	Name of the line
From_bus	Index of bus where the line starts
To_bus	Index of bus where the line ends
Length_km	length of the line [km]
R_ohm_per_km	resistance of the line [Ohm per km]
X_ohm_per_km	inductance of the line [Ohm per km]
C_nf_per_km	capacitance of the line [nF per km]
Max_I_ka	maximal thermal current [kA]
df	derating factor (scaling) for max_i_ka
In_service	specifies if the line is in service.

Load: This sheet has information about the load, the bus it is connected to, power factor values and load priority. The program would consider the loads in the priority order given in this sheet. Table 5-5 provides the load parameters description [61].

Table 5-5 Load Parameters

Parameter	**Description**
name	Name of the load
bus	Index of the connected bus
P_mw	active power of the load [MW]
Q_mvar	reactive power of the load [MVar]
Sn_mva	rated power of the load [MVA]
scaling	scaling factor for active and reactive power
priority	Specifies the order of load pickup
Const_z_percent	percentage of p_kw and q_kvar that is associated to constant impedance load at rated voltage [%]
Const_I_percent	percentage of p_kw and q_kvar that is associated to constant current load at rated voltage [%]
In_service	specifies if the load is in service.

Motor Load: Motor load has the information about motors within each load. This sheet has Load_number and Load_bus to identify the load that the motor corresponds to. Motor_hp is the motor horsepower, number of motors gives the number of motors with same horsepower that are part of the same load. Table 5-6 provides the motor load parameters description [61].

Table 5-6 Motor Load Parameters

Parameter	Description
Load_number	The load that motor belongs to
Load_bus	Index of the load bus
Motor_hp	Horsepower motor rating
No_of_motors	Number of motors connected to the load and are of same horsepower.
Voltage_kv	Rated Voltage of a motor
Full_load_current(A)	Full load Amp (FLA) rating is at which motor will consume power at 100% of rated load
Power_Factor_full_load	Power factor of the motor
Efficiency_full_load	Efficiency at the rated power
Power_factor_locked_rotor	Locked rotor power factor of the motor
Inrush_current(A)	Starting current of the motor

Static Load: This sheet gives information about static component of the load. It has load identifier, active power, and reactive power of the static components. Table 5-7 provides the static load parameters description [61].

Table 5-7 Static Load Parameters

Parameter	Description
Load	The load that motor belongs to
P_mw	active power of the static component of the load [MW]

Q_mvar	reactive power of the static component of the load [MVar]

External Grid: External grid is the higher-level power grid and is considered as slack bus in the system. Table 5-8 provides the external grid parameters description [61].

Table 5-8 External Grid Parameters

Parameter	**Description**
name	name of the external grid
bus	index of connected bus
Vm_pu	reactive power of the static component of the load [MVar]
Va_degree	angle set point [degree]
Max_p_mw	Maximum active power
Min_p_mw	Minimum active power
Max_q_mvar	Maximum reactive power
Min_q_mvar	Minimum reactive power

After the user enters the information in excel and clicks on submit, the python program follows the steps in the flowchart and outputs the restoration data.

5.3 Significant Code Snippets

5.3.1 User Input:

The HTML and Bootstrap code as shown in Figure. 5-2 displays the prompt for the user to enter excel with custom circuit handled in line 32 to 39 below. The user can download sample file which is available in the application files given in code below in lines 51 to 58.

See Appendix B for complete code for creating user input using HTML and Bootstrap.

```
28  </form>
29  <div class="container" id='main_container'>
30      <h3>Please upload the network information</h3>
31  <form class = "" action = "data" enctype="multipart/form-data" method = "post">
32      <div class="col-sm-12" style="padding-left:0px">
33          <div class="col-sm-6">
34              Please upload xlsx file:
35          </div>
36          <span class="btn btn-success fileinput-button">
37              <input type = "file" name="upload-file" value = "">
38          </span>
39      </div>
40      <br/>
41      <div class="col-sm-12" style="padding-left:0px" >
42          <div class="col-sm-6">
43              Please upload Image file:
44          </div>
45          <span class="btn btn-success fileinput-button">
46              <input type = "file" name="upload-image" value = "">
47          </span>
48      </div>
49      <br/>
50      <br/>
51      <div class="col-sm-12" style="padding-left:0px" >
52          <div class="col-sm-6">
53              Click on Excel logo to download sample network data
54          </div>
55          <a href="/static/Restoration_Sample_Data.xlsx" download="Restoration_Sample_Data">
56              <img src="/static/Excel.png" alt="Excel" width="50" height="50" >
57          </a>
58      </div>
59
60          <input type = "submit" name = "" value = "Submit">
61  </form>
```

Figure 5-2 User Input Code Snippet

5.3.2 Creating a Network:

Pandapower library is used to create a network in python [61]. An empty network is created using the method create_empty_network() as described in line 187 in the Figure. 5-3. The file uploaded by the user has one sheet for each electric component in the circuit. Each sheet is read using Read_excel method of the panda's library and looped through passing the values to the created network to create circuit components. Table 5-9 provides the methods create each electric element in the network [61] and Figure. 5-3 is the code snippet for creating electric network.

Table 5-9 Methods to Create Each Electric Element in the Network

Method Name	**Usage**	**Library**
Read_excel	Used to read excel	Pandas
Create_bus	Create bus in the circuit	Pandapower
Create_ext_grid	Create external grid in the circuit	Pandapower
Create_gen	Create generator in the circuit	Pandapower
Create_load	Create load in the circuit	Pandapower
Create_line_from_parameters	Create line in the circuit	Pandapower
Create_transformer_from_parameters	Create transformer in the circuit	Pandapower

```
    import pandapower as pp
    net = pp.create_empty_network()

    restoration_file = pd.ExcelFile(file)

# Creating Buses
    exc_bus = pd.read_excel (restoration_file,sheet_name='bus')

    exc_bus.sort_values(by=['bus'], inplace=True)
    for index, row in exc_bus.iterrows():
        pp.create_bus(net, vn_kv=row['vn_kv'], name=row['name'],in_service = True)

# Creating External grid
    exc_grid = pd.read_excel (restoration_file,sheet_name='externalgrid')
    for index,row in exc_grid.iterrows():
        pp.create_ext_grid(net, bus=row['bus']-1, vm_pu=row['vm_pu'], va_degree = row['va_degree'],
        max_p_mw = row['max_p_mw'],min_p_mw = row['min_p_mw'],max_q_mvar = row['max_q_mvar'],
        min_q_mvar = row['min_q_mvar'],name=row['name'],in_service = True)

        ext_grid_bus = row['bus']

#Creating Generators
    exc_gen = pd.read_excel (restoration_file,sheet_name='generator')

    for index, row in exc_gen.iterrows():
        pp.create_gen(net, bus=row['bus']-1,p_mw=row['p_mw'],vm_pu=1.0,sn_mva= row['sn_mva'],
        name=row['name'],max_q_mvar = row['max_q_mvar'],min_q_mvar=row['min_q_mvar'],in_service=True)
    dat = []

#Creating Loads
    exc_load = pd.read_excel (restoration_file,sheet_name='load')
    for index, row in exc_load.iterrows():
        pp.create_load(net, bus = row['bus']-1, p_mw = row['p_mw'], q_mvar= row['q_mvar'],
        const_z_percent=row['const_z_percent'], const_i_percent=row['const_i_percent'],
        sn_mva = row['sn_mva'], scaling = row['scaling'], name=row['name'], in_service=True)
        dat.append([row['bus']-1,row['priority']])

# Creating Lines
    exc_line = pd.read_excel (restoration_file,sheet_name='line')
    for index, row in exc_line.iterrows():
        pp.create_line_from_parameters(net, from_bus = row['from_bus']-1, to_bus = row['to_bus']-1,
        length_km=row['length_km'], r_ohm_per_km = row['r_ohm_per_km'], x_ohm_per_km = row['x_ohm_per_km'],
        c_nf_per_km = row['c_nf_per_km'], max_i_ka = row['max_i_ka'],name=row['name'],in_service = True)

# Creating Transformers
    exc_trans = pd.read_excel (restoration_file,sheet_name='transformer')
    for index, row in exc_trans.iterrows():
        pp.create_transformer_from_parameters(net, hv_bus=row['hv_bus']-1, lv_bus=row['lv_bus']-1,
        name=row['name'], sn_mva=row['sn_mva'], vn_hv_kv=row['vn_hv_kv'],vn_lv_kv=row['vn_lv_kv'],
        vkr_percent=row['vkr_percent'], vk_percent=row['vk_percent'], pfe_kw=row['pfe_kw'], i0_percent=row['i0_percent'],
        shift_degree=row['shift_degree'],in_service = True,tap_side = row['tap_side'])
```

Figure 5-3 Code for Creating Electric Network

5.3.3 Inrush Current Calculation:

For the motors uploaded the active and reactive power in the presence of inrush currents are calculated using the formula provided in equations 4.3 and 4.4. The inrush current is calculated for each motor and multiplied by number of motors to get the total inrush current for a motor group for each load. As provided in Figure. 5-4, the code depicts the calculation of inrush currents.

```
    motors['irated'] = ((motors['motor_hp']*746)/motors['power_factor_full load'])/
                       (np.sqrt(3)*motors['voltage_kv']*1000*motors['efficiency_full_load'])
    motors['p_inrush'] = motors['voltage_kv']*np.sqrt(3)*motors['irated']*6*
                      motors['power_factor_locked_rotor']*(1/(np.power(10,6)))*1000
    motors['q_inrush'] = motors['voltage_kv']*np.sqrt(3)*motors['irated']*6*
                         np.sin(np.arccos(motors['power_factor_locked_rotor']))*(1/np.power(10,6))*1000
    motors['p_inrush_tot'] = motors['p_inrush'] * motors['no_of_motors']
    motors['q_inrush_tot'] = motors['q_inrush'] * motors['no_of_motors']
```

Figure 5-4 Inrush Current Calculation Code

5.3.3 Dijkstra's Algorithm:

After the circuit is constructed, the Buses are taken as node and the impendence on the line is taken as weight and Dijsktra's algorithm is implemented to calculate the shortest path from black start generator to all the generators and loads in the circuit. It is implemented in python using the code shown in Figure. 5-5.

```
def dijsktra(graph, initial, end):
    # shortest paths is a dict of nodes
    # whose value is a tuple of (previous node, weight)
    shortest_paths = {initial: (None, 0)}
    current_node = initial
    visited = set()

    while current_node != end:
        visited.add(current_node)
        destinations = graph.edges[current_node]
        weight_to_current_node = shortest_paths[current_node][1]

        for next_node in destinations:
            weight = graph.weights[(current_node, next_node)] + weight_to_current_node
            if next_node not in shortest_paths:
                shortest_paths[next_node] = (current_node, weight)
            else:
                current_shortest_weight = shortest_paths[next_node][1]
                if current_shortest_weight > weight:
                    shortest_paths[next_node] = (current_node, weight)

        next_destinations = {node: shortest_paths[node] for node in shortest_paths if node not in visited}
        if not next_destinations:
            return "Route Not Possible"
        # next node is the destination with the lowest weight
        current_node = min(next_destinations, key=lambda k: next_destinations[k][1])

    # Work back through destinations in shortest path
    path = []
    total_weight = 0
    while current_node is not None:
        path.append(current_node)
        next_node = shortest_paths[current_node][0]
        total_weight = total_weight+shortest_paths[current_node][1]
        current_node = next_node

    # Reverse path
    path = path[::-1]
    #Return path as well as total weight
    path_and_weight = {'path':path,'total_imp':total_weight}
#    print("Path&Impedence", path_and_weight)
#    return path
    return path_and_weight
```

Figure 5-5 Dijkstra's Algorithm Code Snippet

5.3.4 Load Priority

The loads are picked up based on the priority order given by the user. The code provided in Figure. 5-6 assigns processed flag as 'N' that is changed to 'Y' once the load is finished picking up. The loads are then sorted by priority and picked up in sequence.

```
269
270     #Create internal table for load priority
271     load_priority = pd.DataFrame(dat, columns = ['load_bus', 'priority'])
272     load_priority['processed'] = 'N'
273     load_priority = load_priority.sort_values(by = ['priority'])
274
```

Figure 5-6 Load Priority Code

Appendix C shows the complete restoration code in python.

5.3.5 Displaying Output:

The output is displayed using jQuery embedded in HTML. The main purpose of jQuery is to traverse the data table received as output from the python program. The jQuery code snippet in Figure. 5-7 loops through the data table and displays result in the form of a HTML table.

```
90      {% for i,row in data.iterrows() %}
91      <tr>
92       <td> {{  row.iteration|int }}</td>
93       <td> {{  row.gen_turned_on }}</td>
94       <td> {{  row.eff_gen_cap_p }}</td>
95       <td> {{  row.eff_gen_cap_q }}</td>
96       <td> {{  row.cranking_power_provided_gen }}</td>
97       <td> {{  row.cranking_power_p  }}</td>
98       <td> {{  row.cranking_power_q  }}</td>
99       <td> {{  row.Load_Name }}</td>
100      <td> {{  row.motor_group }}</td>
101      <td> {{  row.pli_mw  }}</td>
102      <td> {{  row.qli_mvar  }}</td>
103      <td> {{  row.p_mw  }}</td>
104      <td> {{  row.q_mw  }}</td>
105      <td> {{  row.lp_mw }}</td>
106      <td> {{  row.lq_mvar }}</td>
107      <td> {{  row.Voltage_Drop  }}%</td>
108      <td> {{  row.Voltage_Drop_steady }}%</td>
109      </tr>
110      {% endfor %}
```

Figure 5-7 Code for Result Display

See Appendix D for output display code using jQuery embedded in HTML. See Appendix E for code for standardizing UI across the tool.

Chapter 6 Case Study

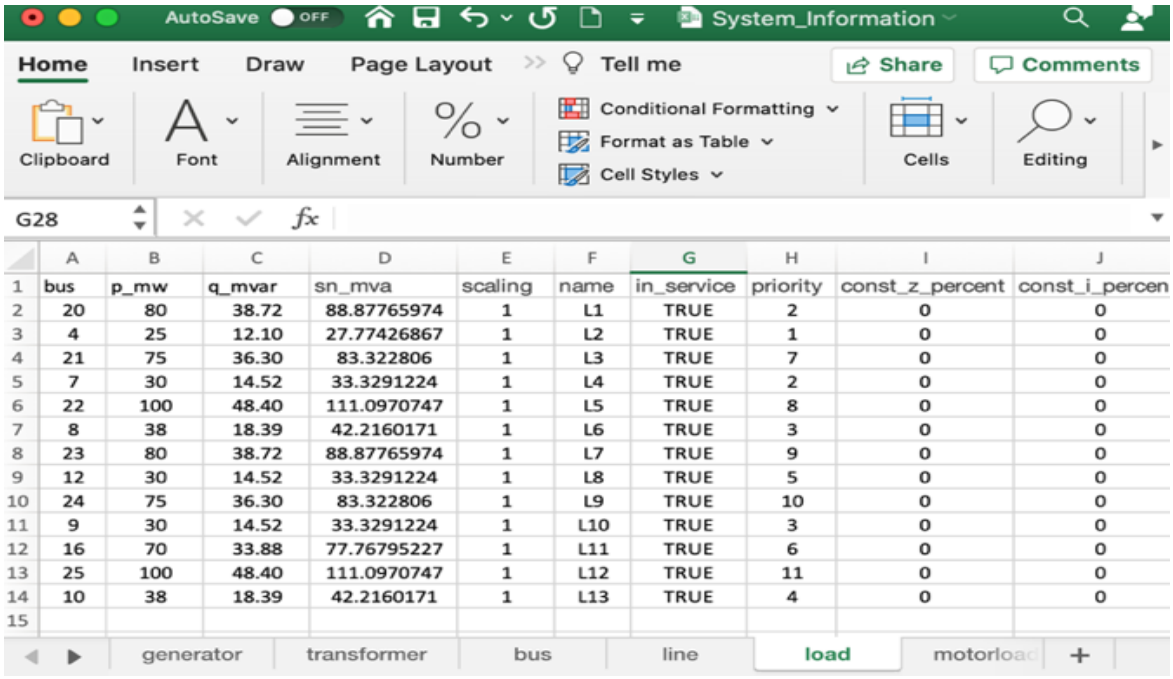
This chapter describes the test cases used to demonstrate the effectiveness of the proposed methodology. This research is based on the industrial power system restoration. However, the algorithm can handle regular power systems. Therefore, IEEE 30 bus system is used as an example to illustrate the capability of the software.

The design of an industrial bus system and procedures for Automatic System Restoration are explained in this chapter.

6.1 Procedures for Automatic System Restoration

As described in chapter 5, a tool is developed to demonstrate the proposed Automatic power system restoration. The procedure of automatic power system restoration is given below:

1. Initialization: The tool when executed loads all the pre-built python libraries that are used to simulate the system, finding the restoration path etc.,
2. Creating system information: A preformatted excel is available for operator to download from the tool and modify it with the desired system information and desired load priority. The excel sheet has multiple tabs, each tab provides information about a specific type of component in the system. A sample tab of preformatted excel sheet is provided in Figure. 6-1. Each tab information is provided in chapter 5.



	A	B	C	D	E	F	G	H	I	J
1	bus	p_mw	q_mvar	sn_mva	scaling	name	in_service	priority	const_z_percent	const_i_percen
2	20	80	38.72	88.87765974	1	L1	TRUE	2	0	0
3	4	25	12.10	27.77426867	1	L2	TRUE	1	0	0
4	21	75	36.30	83.322806	1	L3	TRUE	7	0	0
5	7	30	14.52	33.3291224	1	L4	TRUE	2	0	0
6	22	100	48.40	111.0970747	1	L5	TRUE	8	0	0
7	8	38	18.39	42.2160171	1	L6	TRUE	3	0	0
8	23	80	38.72	88.87765974	1	L7	TRUE	9	0	0
9	12	30	14.52	33.3291224	1	L8	TRUE	5	0	0
10	24	75	36.30	83.322806	1	L9	TRUE	10	0	0
11	9	30	14.52	33.3291224	1	L10	TRUE	3	0	0
12	16	70	33.88	77.76795227	1	L11	TRUE	6	0	0
13	25	100	48.40	111.0970747	1	L12	TRUE	11	0	0
14	10	38	18.39	42.2160171	1	L13	TRUE	4	0	0
15										

Figure 6-1 Sample Excel to Enter System Data

3. Uploading system information: The operator uploads the modified excel sheet containing the system information along with the system image to the tool as shown in Figure.6-2.

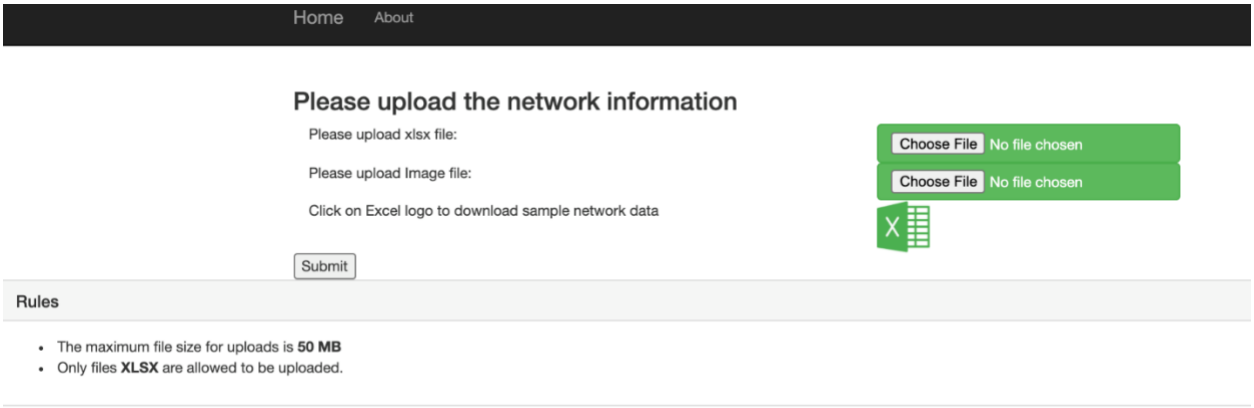


Figure 6-2 User Input for Restoration

4. System Restoration: The tool uses pandapower library for power system modeling and performs automatic system restoration following all the restoration steps as provided in

chapter 4, section 4.2 like identifying the restoration path, simulating the system, calculating inrush currents and load restoration.

5. Results: The tool displays the simulated results with values at each step of the restoration sequence for industrial system and IEEE 30 bus system shown in Figure. 7-1 and Figure. 7-2 respectively.

6.2 Design of the Industrial System

This study designed an industrial bus system to demonstrate the proposed algorithm. A one-line diagram of industrial system is designed in PowerWorld software is shown in Figure. 6-3.

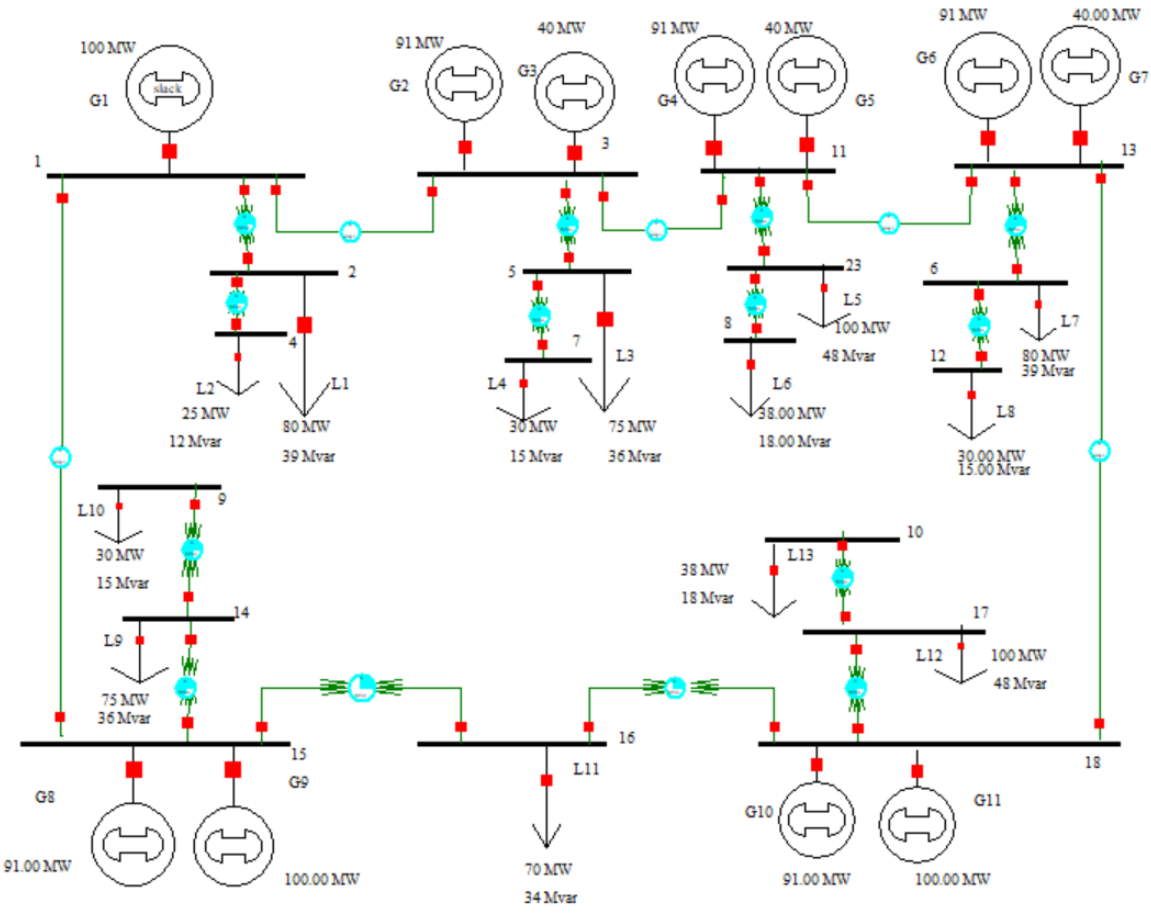


Figure 6-3 One-line Diagram of an Industrial Power System

A. System overview

This section provides information about the industrial system used to validate the algorithm. Table 6-1 provides the generator information of an industrial power system.

Table 6-1 Generator Data

Generator	**Real Rated Power (MW)**	**Reactive Rated Power (MVAR)**		**Generator Type**
		Q_{max}	**Q_{min}**	
G1	100	62	-62	Black start
G2, G4, G6, G8, & G10	91	46	-46	Gas Turbine
G9, G11	100	62	-62	Gas Turbine
G3, G5, G7	40	25	-25	Thermal

G1 is the black start generator. The cranking power required for generators G3 and G5 is 7-8% of its rated capacity, for all the other generators the required cranking power is 1.5-2% of its rated capacity [24].

The loads on the feeder obtained from IED right before the blackout are given in the Table 6-2. Loads in each row have identical real and reactive power and will perform similar operation.

Table 6-2 Load Information

Loads	**P_L (MW)**	**Q_L (MVAR)**
L_1, L_7	80	39

L_3, L_9	75	36
L_2	25	12
L_5, L_{12}	100	48
L_4, L_8, L_{10}	30	15
L_{11}	70	34
L_6	38	18

Each load in the industrial Power system shown above comprises 80% induction loads and 20% static loads.

The total load at each bus connected to the distribution feeder is the combination of several sub feeder loads. Each sub feeder load is the combination of multiple induction motor loads. The inrush currents for induction motors in most cases is considered to be 6 times the rated current [56]. For the industrial system demonstrated in this paper the inrush current is considered to be 6 times rated current for all induction motors. Therefore, the size and variation of the motors considered have the same impact in terms of determining the inrush currents.

Generally, in industries, the identical capacity motors with- in each load are aggregated to one motor group which performs a similar operation in the plant. The range of motors considered for motor groups are between 200HP to 8000HP. The motors below the 200HP are aggregated to one motor group and are considered as a static load, as their magnitude of inrush currents have a negligible impact on restoration. Each induction motor group is combined with 20% of the static load for operation.

The line, bus and transformer data of the industrial system are provided in Appendix A

6.3 IEEE 30-Bus System

The proposed algorithm can also be applied on the regular power system network. Therefore, IEEE 30-bus system is used to demostrate the effectiveness of algorithm and system data is provided in [68]. One-line diagram of IEEE 30-bus system is provided in Figure.6-4 [62].

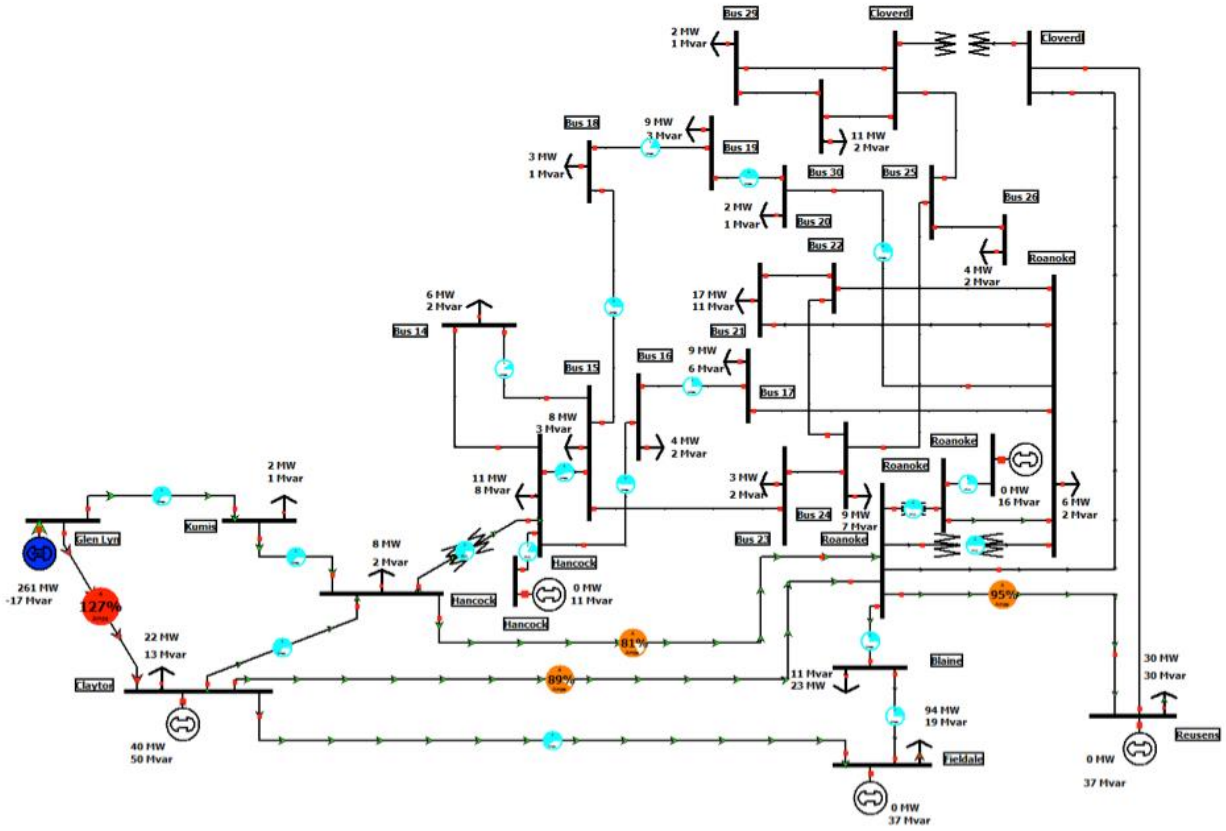


Figure 6-4 One-line Diagram of IEEE 30-Bus System

G1 is the black start generator and other generators in the system are non black start genertor. The cranking power required for all non black start generators is 7% of its rated capacity [24].

As the IEEE 30 bus system loads are lumped loads, this research assumed 15% of each load in the IEEE 30 bus system are induction motors. As described in the chapter 6, section 6-2,

the total load at each bus connected to the distribution feeder is the combination of several sub feeder loads. Each sub feeder load is the combination of multiple induction motor loads. The inrush currents for induction motors in most cases is considered to be 6 times the rated current [56]. Therefore, the size and variation of the motors considered have the same impact in terms of determining the inrush currents. For restoration it is assumed that each load has 15% of motors and that range in between 200HP-3500HP. Each load is aggregated with the induction motor group and static loads.

The automatic power system restoration results are discussed in chapter 7.

Chapter 7 Results

7.1 Tool Implementation for Restoration on Industrial Bus System

After a blackout, the below-mentioned steps are automatically performed by the restoration program developed in Python.

Step 1: Start Black start Generator

Black start generator G_1 is started by closing the switches to the generator.

Step 2: Dijkstra's- Search Algorithm

Applying Dijkstra's Algorithm to the Industrial System given in chapter 6. Figure.6-3 provides the following shortest path to restore the generators (G_1, G_8, G_9, G_2, G_3, G_4, G_5, G_6, G_7, G_{10}, G_{11}).

G_1 must provide cranking power to generators G_8 and G_9.

Step 3: Generation Capability

The available generation capability is calculated using Equations 4.1 and 4.2.

Table 7-1 Generation Capability

$P_{AVA} = P_{GT} - P_{CR} - P_R$	
P_{GT}=100 MW	• G_1 Active Power
P_{CR}=3.83 MW	• 2% of G_8 and 7% of G_9 Rated Active Power
L_P=0	• The real power of load that is already picked up
P_{AVA}=96.17 MW	• Available Active Power
$Q_{AVA} = Q_{GT} - Q_{CR} - Q_R$	

Q_{GT}=62 MVAR	• G_1 Reactive Power
Q_{CR}=2.16 MVAR	• 2% of G_8 and 7% of G_9 Rated Reactive Power
L_Q=0	• Reactive power of load that is already picked up
Q_{AVA}=60.49 MVAR	• Available Reactive Power

Step 4: Energize Transmission Line

Circuit Breakers on the line connecting G_1 and G_8 and G_9 are closed to make it active.

Step 5,6, and 7: Turn on Intelligent Electronic Device and Loads to Pickup

The IEDs are energized along the transmission line from step 4.

A check is performed to see if any loads are already picked up. If yes, then the real-time load value is read from IED.

Step 8 and 9: Select the Loads to Pickup

The circuit is checked to see if any loads are available near G_1 that can be picked up. Loads L_1 and L_2 are available but selected according to their priority order. It is assumed the IEDs report the L_1 and L_2 values before the blackout which are the base values for CLPU.

Load L_1 (80MW and 39 MVAR) and L_2 (25 MW and 12 MVAR) are selected to pickup but according to the priority order load L_2 got highest priority. Therefore, L_2 is picked up first. Each load is picked up in multiple steps, i.e., each group of induction motors at a time in order of decreasing reactive power.

Step 10: Calculate the Inrush Currents for the Induction Motor Loads

80% of L_2 are induction motors. The inrush currents for the induction motors in most cases are considered to be six times the motor rated current [29], [31].

According to the literature [30], the efficiency of the motor varies from 85-90% ignoring the friction and windage losses. Considering the worst-case scenario, the efficiency in this paper is taken as 85%. The full load power factor is 0.9 [13].

Example calculation of inrush current for a 200 HP induction motor considering motor rated voltage, power factor, and efficiency are given in eq. 7.1, eq. 7.2, and eq. 7.3

$$S_{200} = P * cos\theta = \frac{200*746}{0.9} = 165.77kVA \quad (7.1)$$

$$I_{Rated\_200} = \frac{S_{200}}{\sqrt{3}*V_L*\eta} = 28.15\ A \quad (7.2)$$

$$I_{Inrush\_200} = 6 * I_{Rated_{200}} = 168.9\ A \quad (7.3)$$

The starting power factor of a motor varies from 0.2-0.3 [13]. This paper considers the worst-case scenario of the power factor as 0.2. Active and reactive power considering inrush is calculated by using eq. 4.3 and eq.4.4 Therefore,

$$P_{LI} = 9.36\ MW;\ Q_{LI} = 45.86\ MVAR$$

In the same way, Inrush currents for other induction motors for load L_2 are calculated.

Step11,12,13,14,15: Pickup the Loads

The load(L_2) 25MW is a combination of 80% induction motors plus 20% static loads. These 80% induction motors are a combination of several motor groups. As discussed in the flow chart, the highest reactive power motor group is picked up first along with 20% of static load. The

steps below show few picked induction motor groups of loads L_1 and the corresponding voltage drop.

The active and reactive power of available generation capability are given below (calculated from eq. 4.1 and 4.2 provided in restoration flow chart):

$$P_{AVA} = 96.17\ MW;\ Q_{AVA} = 60.49\ MVAR$$

a. The motor group with 700HP has the highest reactive power compared to other induction motor groups of Load L_1. Therefore, it is processed first. The active and reactive power considering inrush for 700HP motor given in Table 7.2.

Table 7-2 Load L_1-700HP Motor Restoration

Step	**Induction motor Group**	**P_{LI8000} (MW)**	**Q_{LI8000} (MVAR)**	**$\%VD_{Inrush}$**	**P_{8000} (MW)***	**Q_{8000} (MVAR)***
1	8000	9.36	45.86	9.98%	10.33	46.33

*Total Active Power Load appeared after the switch is closed (P_{8000}) = P_{static}+P_{LI}

*Total Reactive Power Load appeared after the switch is closed (Q_{8000}) = Q_{static}+Q_{LI}

b. Since, $P_{LI8000}<P_{AVA}$ and $Q_{LI8000}<Q_{AVA}$. The P_{8000} group is picked along with 20% of the static load in L_2.

c. After a while, the 8000HP group reaches a steady state. The total active and reactive power of 8000 HP motor group load plus the static load after reaching steady state are given below

$$L_P = 6.94\ MW; L_Q = 3.36\ MW$$

d. P_{AVA} and Q_{AVA} are adjusted since the steady-state load continuously draws power. Since the load consumption varies after it is being picked up, the real-time load is

obtained from IED. The new available generation capability considering the changed load is given below:

$$P_{AVA} = 89.16\ MW;\ Q_{AVA} = 57.09\ MVAR$$

e. The 6000HP motor group of Load L_2 is picked up in the next step due to its second-highest reactive power. The active and reactive power considering inrush for 6000 HP motor given in Table 7-3

Table 7-3 Load L_1-1000HP Motor Restoration

Step	**Induction motor Group**	**P_{LI6000} (MW)**	**Q_{LI6000} (MVAR)**	**$\%VD_{Inrush}$**	**P_{6000} (MW)***	**Q_{6000} (MVAR)***
2	6000	7.02	34.4	7.3%	7.99	34.87

*Total Active Power Load appeared after the switch is closed (P_{6000}) = P_{static}+P_{LI}

*Total Reactive Power Load appeared after the switch is closed (Q_{6000}) = Q_{static}+Q_{LI}

f. Since, $P_{LI6000}<P_{AVA}$ and $Q_{LI6000}<Q_{AVA}$. The P_{6000} group is picked along with 20% of the static load in L_2. Power flow is executed to check for convergence.

This process is continued until all the loads are picked up.

7.1 Industrial Test System Validation

Figure. 7-1 is the user-interface that displays results along with the circuit diagram. The results screen displays restoration sequence and restoration metrics like generators turned on, available power at any given point, generator to which cranking power is provided, cranking power values, load name, motor group, power values when inrush current is observed, power appeared after the switch is closed, steady state power and voltage drop at each step of the restoration sequence.

Home About

Restoration Sequence:

G1 , G8 , G9 , G2 , G3 , G10 , G11 , G4 , G5 , G6 , G7

Abbreviations:

$P_{AVA_}$mw : Available real power at a given point during restoration

$Q_{AVA_}$mvar : Available reactive power at a given point during

$P_{CR_}$mw Cranking power of generator's real power

$Q_{CR_}$mvar Cranking power of generator's reactive power

$P_{LI_}$mw Active power of induction motor when inrush current is observed

$Q_{LI_}$mvar Reactive power of induction motor when inrush current is observed

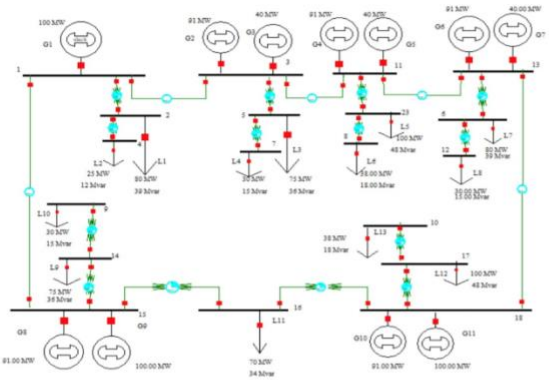
P_mw Active power of the load appeared after the switch is closed

Q_mw Reactive power of the load appeared after the switch is closed

$L_{P_}$mw Total active power of the picked loads after it reaches steady state

$L_{Q_}$mvar Total reactive power of the picked loads after it reaches steady state

%VD Voltage Drop



Restoration Procedure

		Available power		Generator to which cranking power is provided	Cranking power values				Power values when inrush current is observed		Power appeared after the switch is closed		Steady state power		% Voltage Drop	
iteration	Gen_turned_on	$P_{AVA_}$mw	$Q_{AVA_}$mvar	Cr_pwr_to_gen	$P_{CR_}$mw	$Q_{CR_}$mvar	Load_Name	Motor_Group	$P_{LI_}$mw	$Q_{LI_}$mvar	P_mw	Q_mw	$L_{P_}$mw	$L_{Q_}$mvar	$\%VD_{Inrush}$	$\%VD_{Steady\ State}$
1	'G1'	86.61	55.1	'G8', 'G9'	13.39	7.55	L2	8000	9.36	45.86	10.33	46.33	6.94	3.36	7.14%	0.49%
2	-	79.59	51.7	-	-	-	L2	6000	7.02	34.4	7.99	34.87	5.45	2.64	5.17%	0.38%
3	-	74.05	49.02	-	-	-	L2	5500	6.44	31.53	7.41	32.0	5.07	2.46	4.71%	0.36%
4	-	68.9	46.53	-	-	-	L2	4000	4.68	22.93	5.65	23.4	3.96	1.91	3.36%	0.28%
5	-	64.86	44.57	-	-	-	L2	3500	4.1	20.06	5.07	20.53	3.58	1.73	2.92%	0.25%

.

.

20	'G1', 'G8', 'G9'	210.49	131.18	'G2', 'G3'	2.62	1.41	L10	8000	9.36	45.86	10.38	46.36	6.99	3.38	3.91%	0.28%
21	-	203.44	127.77	-	-	-	L10	6000	7.02	34.4	8.04	34.89	5.5	2.66	2.88%	0.22%
22	-	197.85	125.07	-	-	-	L10	5500	6.44	31.53	7.46	32.02	5.12	2.48	2.63%	0.21%
23	-	192.66	122.55	-	-	-	L10	5000	5.85	28.66	6.87	29.16	4.75	2.3	2.38%	0.19%
24	-	187.85	120.23	-	-	-	L10	4000	4.68	22.93	5.7	23.42	4.01	1.94	1.9%	0.16%
25	-	183.76	118.25	-	-	-	L10	3500	4.1	20.06	5.12	20.56	3.63	1.76	1.66%	0.15%
26	-	180.03	116.44	-	-	-	L11	700	7.37	36.12	8.02	36.43	5.35	2.59	2.21%	0.17%
27	-	174.64	113.83	-	-	-	L11	1000	7.02	34.4	7.67	34.71	5.13	2.48	2.1%	0.16%
28	-	169.46	111.33	-	-	-	L11	500	6.44	31.53	7.09	31.84	4.75	2.3	1.93%	0.15%
29	-	164.65	109.0	-	-	-	L11	5000	5.85	28.66	6.5	28.98	4.38	2.12	1.75%	0.14%
30	-	160.24	106.86	-	-	-	L11	600	5.62	27.52	6.27	27.83	4.23	2.05	1.68%	0.13%
31	-	155.96	104.79	-	-	-	L11	4500	5.27	25.8	5.92	26.11	4.01	1.94	1.58%	0.13%
32	-	151.9	102.82	-	-	-	L11	550	5.15	25.22	5.8	25.54	3.93	1.9	1.54%	0.12%
33	-	147.91	100.89	-	-	-	L11	4000	4.68	22.93	5.33	23.24	3.63	1.76	1.4%	0.12%
34	-	144.23	99.11	-	-	-	L11	3500	4.1	20.06	4.75	20.38	3.26	1.58	1.23%	0.1%
35	-	140.9	97.5	-	-	-	L11	250	3.8	18.63	4.45	18.94	3.07	1.49	1.14%	0.1%
36	-	137.77	95.99	-	-	-	L11	800	3.74	18.34	4.39	18.66	3.04	1.47	1.12%	0.1%
37	-	134.67	94.49	-	-	-	L11	450	3.69	18.06	4.34	18.37	3.0	1.45	1.11%	0.1%
38	-	131.64	93.02	-	-	-	L11	3000	3.51	17.2	4.16	17.51	2.89	1.4	1.05%	0.09%
39	-	128.7	91.59	-	-	-	L11	400	3.28	16.05	3.93	16.37	2.74	1.33	0.98%	0.09%
40	-	125.9	90.24	-	-	-	L11	900	3.16	15.48	3.81	15.79	2.66	1.29	0.95%	0.09%
41	-	123.18	88.93	-	-	-	L11	200	3.04	14.91	3.69	15.22	2.59	1.25	0.91%	0.08%
42	-	120.54	87.65	-	-	-	L11	2500	2.93	14.33	3.57	14.65	2.51	1.22	0.88%	0.08%
43	-	117.96	86.4	-	-	-	L11	350	2.87	14.05	3.52	14.36	2.48	1.2	0.86%	0.08%
44	-	115.43	85.17	-	-	-	L11	300	2.81	13.76	3.46	14.07	2.44	1.18	0.85%	0.08%

.

.

169	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	147.7	137.1	-	0.0	0.0	L7	5000	5.85	28.66	6.62	29.03	4.5	2.18	1.87%	0.14%
170	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	143.14	134.89	-	0.0	0.0	L7	300	5.62	27.52	6.38	27.89	4.35	2.1	1.79%	0.14%
171	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	138.74	132.76	-	0.0	0.0	L7	4500	5.27	25.8	6.03	26.17	4.12	2.0	1.67%	0.13%
172	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	134.56	130.74	-	0.0	0.0	L7	400	5.15	25.22	5.92	25.59	4.05	1.96	1.64%	0.13%
173	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	130.46	128.75	-	0.0	0.0	L7	550	5.15	25.22	5.92	25.59	4.05	1.96	1.64%	0.13%
174	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	126.37	126.78	-	0.0	0.0	L7	250	4.97	24.36	5.74	24.73	3.94	1.91	1.58%	0.13%
175	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	122.38	124.84	-	0.0	0.0	L7	4000	4.68	22.93	5.45	23.3	3.75	1.82	1.48%	0.12%
176	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	118.56	122.99	-	0.0	0.0	L7	900	4.21	20.64	4.98	21.01	3.45	1.67	1.33%	0.11%
177	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	115.04	121.29	-	0.0	0.0	L7	3500	4.1	20.06	4.86	20.43	3.38	1.63	1.29%	0.11%
178	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	111.6	119.63	-	0.0	0.0	L7	800	3.74	18.34	4.51	18.72	3.15	1.53	1.18%	0.1%
179	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	108.37	118.07	-	0.0	0.0	L7	350	3.69	18.06	4.45	18.43	3.12	1.51	1.16%	0.1%
180	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	105.21	116.54	-	0.0	0.0	L7	3000	3.51	17.2	4.28	17.57	3.0	1.45	1.11%	0.1%
181	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	102.14	115.05	-	0.0	0.0	L7	200	3.28	16.05	4.04	16.42	2.86	1.38	1.03%	0.09%
182	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	99.25	113.65	-	0.0	0.0	L7	2500	2.93	14.33	3.69	14.7	2.63	1.27	0.92%	0.09%
183	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	96.56	112.35	-	0.0	0.0	L7	2000	2.34	11.47	3.11	11.84	2.26	1.09	0.74%	0.07%
184	'G1', 'G2', 'G3', 'G4', 'G5', 'G6', 'G7', 'G8', 'G9', 'G10', 'G11'	94.24	111.23	-	0.0	0.0	L7	1500	1.76	8.6	2.52	8.97	1.89	0.91	0.56%	0.06%

Figure 7-1 Automatic System Restoration Step by Step Results for Industrial System

7.2 IEEE 30-Bus System Validation

Figure. 7-2 shows the complete automatic restoration sequence for IEEE 30-Bus System. The user-interface that displays results along with the circuit diagram of IEEE 30 bus. In order to show the tool capability this research assumed 15% of the load on each bus consists of induction motors. The tool displays the restoration sequence and restoration metrics like available power at any given point, generator to which cranking power is provided, cranking power values, load name, motor group, power values when inrush current is observed, power appeared after the switch is closed, steady state power and voltage drop at each step of the restoration sequence.

Home About

Restoration Sequence:

G1 , G2 , G3 , G6 , G5 , G4

Abbreviations:

P_{AVA}_mw : Available real power at a given point during restoration

Q_{AVA}_mvar : Available reactive power at a given point during

P_{CR}_mw Cranking power of generator's real power

Q_{CR}_mvar Cranking power of generator's reactive power

P_{LI}_mw Active power of induction motor when inrush current is observed

Q_{LI}_mvar Reactive power of induction motor when inrush current is observed

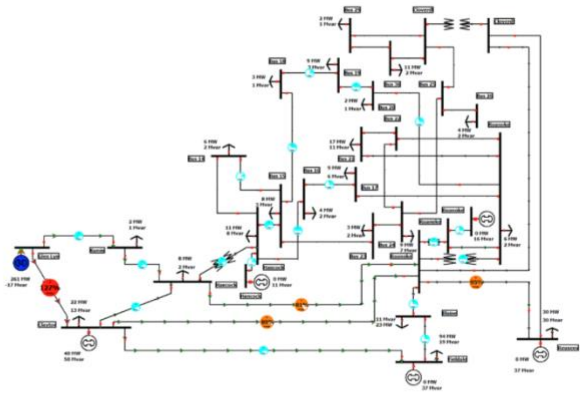
P_mw Active power of the load appeared after the switch is closed

Q_mw Reactive power of the load appeared after the switch is closed

L_P_mw Total active power of the picked loads after it reaches steady state

L_Q_mvar Total reactive power of the picked loads after it reaches steady state

%VD Voltage Drop



Restoration Procedure

		Available power		Generator to which cranking power is provided	Cranking power values				Power values when inrush current is observed		Power appeared after the switch is closed		Steady state power		% Voltage Drop	
iteration	Gen_turned_on	P_{AVA}_mw	Q_{AVA}_mvar	Cr_pwr_to_gen	P_{CR}_mw	Q_{CR}_mvar	Load_Name	Motor_Group	P_{LI}_mw	Q_{LI}_mvar	P_mw	Q_mw	L_P_mw	L_Q_mvar	$\%VD_{Inrush}$	$\%VD_{Steady\ State}$
1	'G1'	359.4	19.3	'G2'	0.8	1.12	L2	250	0.29	1.43	1.32	1.95	1.22	0.61	0.06%	0.14%
2	-	358.12	18.66	-	-	-	L2	200	0.23	1.15	1.27	1.67	1.18	0.59	0.02%	0.15%
3	-	356.85	18.02	-	-	-	L13	250	0.59	2.87	2.2	3.13	1.98	0.44	0.12%	0.02%
4	-	354.7	17.55	-	-	-	L13	200	0.47	2.29	2.08	2.55	1.91	0.4	0.1%	0.02%
.																
.																
28	-	327.99	39.08	-	-	-	L20	200	0.47	2.29	2.22	2.68	2.05	0.53	1.01%	0.4%
29	-	325.8	38.52	-	-	-	L20	250	0.29	1.43	2.05	1.82	1.94	0.47	0.75%	0.37%
30	-	323.75	38.02	-	-	-	L21	300	0.7	3.44	2.44	3.92	2.19	0.69	1.53%	0.68%
31	-	321.45	37.3	-	-	-	L21	250	0.59	2.87	2.33	3.34	2.12	0.66	1.35%	0.65%
32	-	319.23	36.62	-	-	-	L21	350	0.41	2.01	2.15	2.48	2.0	0.6	1.09%	0.61%
33	-	317.09	35.98	-	-	-	L21	200	0.23	1.15	1.98	1.62	1.89	0.55	0.84%	0.57%
34	-	315.04	35.39	-	-	-	L3	250	0.59	2.87	2.07	3.64	1.86	0.95	1.02%	0.39%
35	-	313.06	34.37	-	-	-	L3	200	0.23	1.15	1.72	1.92	1.64	0.85	0.59%	0.35%
.																
.																
83	-	134.75	11.29	-	-	-	L9	200	0.47	2.29	1.83	2.98	1.66	0.83	1.18%	0.48%
84	-	133.03	10.43	-	-	-	L9	250	0.29	1.43	1.65	2.12	1.54	0.77	0.88%	0.45%
85	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	131.36	72.94	-	0.0	0.0	L17	250	1.17	5.73	4.36	9.21	3.93	3.84	7.63%	7.08%
86	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	127.27	68.93	-	0.0	0.0	L17	200	1.17	5.73	4.36	9.21	3.93	3.84	7.63%	7.08%
87	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	123.15	64.88	-	0.0	0.0	L17	450	1.05	5.16	4.24	8.64	3.86	3.8	7.63%	7.08%
88	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	119.04	60.8	-	0.0	0.0	L17	400	0.94	4.59	4.12	8.06	3.78	3.77	7.63%	7.08%
89	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	115.03	56.8	-	0.0	0.0	L17	350	0.82	4.01	4.0	7.49	3.71	3.73	7.62%	7.08%
90	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	111.04	52.75	-	0.0	0.0	L17	300	0.7	3.44	3.89	6.92	3.63	3.69	7.62%	7.08%
91	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	107.08	48.71	-	0.0	0.0	L17	550	0.64	3.15	3.83	6.63	3.6	3.68	7.62%	7.07%
92	'G1', 'G2', 'G3', 'G4', 'G5', 'G6'	103.27	44.79	-	0.0	0.0	L17	500	0.59	2.87	3.77	6.34	3.56	3.66	7.62%	7.07%

Figure 7-2 Automatic System Restoration Step by Step Results for IEEE 30 Bus System

Chapter 8 Conclusion and Future Work

When a blackout occurs, it is highly critical to reduce the restoration time and to restore the system efficiently.

This dissertation presents the algorithm for automatic power system restoration. The proposed restoration algorithm and practical implementation provides operator guidance to make better decisions during restoration thereby saving time and effort.

To decrease the restoration time, it is important to minimize the time taken for selection of restoration path. This study utilizes an advanced graph search technique called Dijkstra's algorithm, which is time efficient and scalable, to identify the restoration path based on electrical distance (impedance).

To restore the system efficiently this research proposes automatic power system restoration procedure by considering inrush currents, cold load pickup, load priority and load variation. A tool is developed which helps the user to enter any network information with the generator, load composition data, load priority, etc.

The proposed restoration algorithm is simple that can be changed/extended with any set of rules/constraints.

An industrial system and IEEE 30 bus system are used to demonstrate the proposed algorithm.

8.1 Limitations of this Study

1. This research concentrates on the load restoration in the event of the blackout and assumes that the grid does not receive assistance/power from neighboring grid.

2. A large system can be sectionalized into small subsystems and parallel restoration can be done in each subsystem. According to the partitioning strategies each subsystem will have one black start generating unit. Therefore, the restoration process in each subsystem is similar. This study focuses on the restoration of one subsystem and does not provide solutions for sectionalizing the system.
3. This study considers that the black start and non-black start units are known.

8.2 Potential Future work

1. The proposed algorithm is best suited to restore individual subsystems. This work can be combined with any automated mechanism to divide the entire system into subsystems making the end-to-end process fully automatic.
2. The algorithm can be extended to identify the black start generator and non-black start generators in the system automatically.
3. Having a mechanism to identify if the load has reached a steady state instead of assuming that the loads reach a steady state after a certain period.
4. The algorithm can be upgraded to provide the user an option on selecting the load to be restored if two or more loads satisfy all the constraints and are ready for pickup.

References

[1] H. H. Alhelou, M. Hamedani-Golshan, T. Njenda, and P. Siano, “A Survey on Power System Blackout and Cascading Events: Research Motivations and Challenges,” *Energies*, vol. 12, no. 4, p. 682, 2019.

[2] E. H. Allen, R. B. Stuart and T. E. Wiedman, "No Light in August: Power System Restoration Following the 2003 North American Blackout," in *IEEE Power and Energy Magazine*, vol. 12, no. 1, pp. 24-33, Jan.-Feb. 2014.

[3] Cechin, A.L., Canto dos Santos, J.V., Mendel, C.A. *et al.* Genetic algorithms to solve the power system restoration planning problem. *Engineering with Computers* **25,** 261–268 (2009). https://doi.org/10.1007/s00366-009-0128-3

[4] Bretas, Arturo & Phadke, A.G.. (2003). Artificial neural networks in power system restoration. IEEE Transactions on Power Delivery. 18. 1181-1186. 10.1109/TPWRD.2003.817500.

[5] R. kumar Mishra and K. S. Swarup, "Power system restoration in smart grid environment," *2014 Eighteenth National Power Systems Conference (NPSC)*, Guwahati, India, 2014, pp. 1-6, doi: 10.1109/NPSC.2014.7103885.

[6] Jiang, Yazhou & Chen, Sijie & Liu, Chen-Ching & Sun, Wei & Luo, Xiaochuan & Liu, Shanshan & Bhatt, Navin & Uppalapati, Sunitha & Forcum, David. (2016). Blackstart capability planning for power system restoration. International Journal of Electrical Power & Energy Systems. 86. 10.1016/j.ijepes.2016.10.008.

[7] C. Liu, Minhao Wu and Yingsong Deng, "Start-up sequence of generators in power system restoration avoiding the backtracking algorithm," 2013 IEEE Power & Energy Society General Meeting, Vancouver, BC, 2013, pp. 1-5, doi: 10.1109/PESMG.2013.6672678.

[8] Qiang Liu, Libao Shi, Ming Zhou, Gengyin Li and Yixin Ni, "A new solution to generators start-up sequence during power system restoration," *2008 Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies*, Nanjing, 2008, pp. 2845-2849, doi: 10.1109/DRPT.2008.4523894.

[9] W. Sun, C.-C. Liu and S. Liu, "Black start capability assessment in power system restoration", *Proc. IEEE Power Energy Soc. General Meeting*, pp. 1-7, Jul. 2011.

[10] N. Saraf, K. McIntyre, J. Dumas and S. Santoso, "The annual black start service selection analysis of ERCOT grid", IEEE Trans. Power Syst., vol. 24, no. 4, pp. 1867-1874, Nov. 2009.

[11] Tyler Hodge, April Lee, Hurricane Irma cut power to nearly two-thirds of Florida's electricity customers, 2017 [online], Available: https://www.eia.gov/todayinenergy/detail.php?id=32992

[12] Abu Talib, Dian & Mokhlis, Hazlie & TALIP, Mohamad & Naidu, Kanendra. (2017). Parallel power system restoration planning using heuristic initialization and discrete evolutionary programming. Journal of Modern Power Systems and Clean Energy. 5. 1-13. 10.1007/s40565-017-0320-1.

[13] J. Figueiredo, J. Sauve, P. Nicolletti, E. Rocha, S. Araujo, F. Amorim, A. Feitosa, R. Agra, and W. Ribeiro. Smart action: A tool to help power system restoration. In Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), pages 167–167, 2008.

[14] L. Lindgren, "Automatic power system restoration" in Lund University, Lund, 2009, [online] Available: http://www.iea.lth.se/publications/Theses/LTH-IEA-1060.pdf

[15] Y. Liu, R. Fan, 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.

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

[17] V. Kumar, H. R. Kumar, I. Gupta, and H. Gupta, “Stepwise Restoration of Power Distribution Network under Cold Load Pickup,” *2006 International Conference on Power Electronic, Drives and Energy Systems*, 2006.

[18] K. P. Schneider, E. Sortomme, S. S. Venkata, M. T. Miller, and L. Ponder, “Evaluating the Magnitude and Duration of Cold Load Pick-up on Residential Distribution Feeders_newline Using Multi-State Load Models,” *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3765–3774, 2016.

[19] 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.

[20] E. Agneholm and J. Daalder, “Cold load pick-up of residential load,” *IEEE Proceedings - Generation, Transmission and Distribution*, vol. 147, no. 1, p. 44, 2000.

[21] M.M. Adibi, "Special consideration in power system restoration. The second working group report," in *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 15-21, Feb. 1994.

[22] O. Akwukwaegbu, O. G. Ibe, "Concepts of Reactive Power Control and Voltage Stability Methods in Power System Network", *IOSR J. Comput. Eng.*, vol. 11, no. 2, pp. 15-25, 2013.

[23] Y. Jiang, S. Chen, C.-C. Liu, W. Sun, X. Luo, S. Liu, N. Bhatt, S. Uppalapati, and D. Forcum, “Blackstart capability planning for power system restoration,” *International Journal of Electrical Power & Energy Systems*, vol. 86, pp. 127–137, 2017.

[24] S. Hemalatha, R. Srinivasan, and A. Ruban Raja, (2019). “A graph theory-based power system restoration plan,” *International Journal of Innovative Technology and Exploring Engineering*, pp.551–556, 2019.

[25] V. Widiputra and J. Jung, “Development of Restoration Algorithm under Cold Load Pickup Condition using Tabu Search in Distribution System,” *2018 IEEE Power & Energy Society General Meeting (PESGM)*, 2018.

[26] B. Goo, S. Jung, and J. Hur, “Development of a Sequential Restoration Strategy Based on the Enhanced Dijkstra Algorithm for Korean Power Systems,” *Applied Sciences*, vol. 6, no. 12, p. 435, 2016.

[27] F. Friend, “Cold load pickup issues,” *2009 62nd Annual Conference for Protective Relay Engineers*, 2009.

[28] E.Grinberg, J.Ostrower, M.Park and C.Zdanowicz. “Atlanta’s Hartsfield-Jackson airport restores power after crippling outage.” CNN.com. https://www.cnn.com/2017/12/17/us/atlanta-airport-power-outage/index.html (accessed Dec. 5, 2020).

[29] M.M. Adibi. Power System Restoration, methodologies & implementation strategies. IEEE Press, 2000.

[30] M. M. Adibi, J. N. Borkoski and R. J. Kafka, "Power System Restoration - The Second Task Force Report," in *IEEE Transactions on Power Systems*, vol. 2, no. 4, pp. 927-932, Nov. 1987, doi:10.1109/TPWRS.1987.4335278.

[31] Y. Hou, C. Liu, Pei Zhang, and K. Sun, "Constructing power system restoration strategies," 2009 International Conference on Electrical and Electronics Engineering -ELECO 2009, pp. I-8-I-13, 2009.

[32] M.M. Adibi and N. Martins, “Power system restoration dynamics issues,” 2008 IEEE Power and Energy Society General Meeting -Conversion and Delivery of Electrical Energy in the 21st Century, 2008.

[33] Tortos, J. Q. and V. Terzija. “A smart power system restoration based on the merger of two different strategies.” *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)* (2012): 1-8.

[34] J. Duncan Glover, Thomas J. Overbye, and Mulukutla S. Sarma, “Transient Analysis,” in Power System Analysis & Design, 6thed., Boston, MA: Cengage Learning, pp. 675-684.

[35] J. W. Feltes and C. Grande-Moran, "Black start studies for system restoration,"2008 IEEE Power and Energy Society General Meeting -Conversion and Delivery of Electrical Energy in the 21st Century, Pittsburgh, PA, 2008, pp. 1-8, doi: 10.1109/PES.2008.4596565.

[36] R.J. Buschart, “Electrical Safety for Chemical Processes,” in Electrical and Instrumentation Safety for Chemical Processes, Springer US, 2012, pp.128. [online]. Available: https://www.springer.com/gp/book/9781468466225?utm_campaign=3_pier05_buy_print&utm_content=en_08082017&utm_medium=referral&utm_source=google_books#otherversion=9781468466201.

[37] F. Zavoda, C. Abbey, Y. Brissette, and R. Lemire, “Universal IED for distribution smart grids,” *22nd International Conference and Exhibition on Electricity Distribution (CIRED 2013)*, 2013.

[38] Hannu Laaksonen, “IED Functionalities Fulfilling Future Smart Grid Requirements,” *International Journal of Distributed Energy Resources and Smart Grids*, vol. 9, July 2013.

[39] J. R. Gracia, L. C. Marke, D. T. Rizy, P. W. O’Connor, R. Shan, and A. Tarditi, “Hydropower plants as Black start resources”, Hydrowires U.S department of energy, 2019.

[40] Sun, Kai, et al. *Power System Control under Cascading Failures : Understanding, Mitigation, and System Restoration*, John Wiley & Sons, Incorporated, 2019. *ProQuest Ebook Central*, https://ebookcentral.proquest.com/lib/utarl/detail.action?docID=5612923.

[41] I. Beil, A. Allen, A. Tokombayev and M. Hack, "Considerations when using utility-scale battery storage to black start a gas turbine generator," *2017 IEEE Power & Energy Society General Meeting*, Chicago, IL, 2017, pp. 1-5, doi: 10.1109/PESGM.2017.8274529.

[42] M. M. Adibi and L. H. Fink, "Power system restoration planning," in *IEEE Transactions on Power Systems*, vol. 9, no. 1, pp. 22-28, Feb. 1994, doi: 10.1109/59.317561.

[43] A. Arif, Z. Wang, J. Wang, B. Mather, H. Bashualdo and D. Zhao, "Load Modeling—A Review," in *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5986-5999, Nov. 2018, doi: 10.1109/TSG.2017.2700436.

[44] "Load representation for dynamic performance analysis (of power systems)," in *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 472-482, May 1993, doi: 10.1109/59.260837.

[45] D. T. Vedullapalli, R. Hadidi and L. S. Bozeman, "Priority based restoration of unbalanced electric distribution systems after multiple faults," 2018 IEEE/IAS 54th Industrial and Commercial Power Systems Technical Conference (I&CPS), Niagara Falls, ON, 2018, pp. 1-5, doi: 10.1109/ICPS.2018.8369992.

[46] M. AlOwaifeer and M. AlMuhaini, "Load Priority Modeling for Smart Service Restoration," in Canadian Journal of Electrical and Computer Engineering, vol. 40, no. 3, pp. 217-228, Summer 2017, doi: 10.1109/CJECE.2017.2705174.

[47] M. M. Adibi, "Power System Restoration A Task Force Report," in *Power System Restoration: Methodologies & Implementation Strategies*, IEEE, 2000, pp.3-9, doi: 10.1109/9780470545607.ch1.

[48] S. Ihara and F. C. Schweppe, "Physically Based Modeling of Cold Load Pickup," in *IEEE Power Engineering Review*, vol. PER-1, no. 9, pp. 27-28, Sept. 1981, doi: 10.1109/MPER.1981.5511825.

[49] R. C. Leou, Z. L. Gaing, C. N. Lu, B. S. Chang and C. L. Cheng, "Distribution system feeder cold load pickup model", *Electr. Power Syst. Res*, vol. 36, no. 3, pp. 163-168, 1996.

[50] Leonard L.Grigsby, “Distribution Short circuit Protection,” *Electric Power Generation, Transmission, and Distribution*. United States: CRC Press,2018, ch.30 pp.30-4

[51] C.-C. Liu, V. Vittal, G. T. Heydt, and K. Tomsovic, “Development and Evaluation of System Restoration Strategies from a Blackout,” *Power Syst. Engi-neering Res. Cent.*, no. Technical report, pp. 10–22, 2009.

[52] V. Kumar, R. Kumar H. C., I. Gupta and H. O. Gupta, "DG Integrated Approach for Service Restoration Under Cold Load Pickup," in *IEEE Transactions on Power Delivery*, vol. 25, no. 1, pp. 398-406, Jan. 2010, doi: 10.1109/TPWRD.2009.2033969.

[53] Widiputra, Victor & Jufri, Fauzan & Jung, Jaesung. (2020). Development of service restoration algorithm under cold load pickup condition using conservation voltage reduction and particle swarm optimization. International Transactions on Electrical Energy Systems. 30. 10.1002/2050-7038.12544.

[54] T. Nagata, S. Hatakeyama, M. Yasouka and H. Sasaki, "An efficient method for power distribution system restoration based on mathematical programming and operation strategy," *PowerCon 2000. 2000 International Conference on Power System Technology. Proceedings (Cat. No.00EX409)*, Perth, WA, Australia, 2000, pp. 1545-1550 vol.3, doi: 10.1109/ICPST.2000.898201.

[55] S. R. Kurup and S. Ashok, "Performance of a hydropower plant during black start and islanded operation," 2015 IEEE International Conference on Signal Processing, Informatics, Communication, and Energy Systems (SPICES), Kozhikode, 2015, pp. 1-5.

[56] M. Habyarimana and D. G. Dorrell, “Methods to reduce the starting current of an induction motor,” *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, 2017.

[57] C. Wang, V. Vittal and K. Sun, "OBDD-Based Sectionalizing Strategies for Parallel Power System Restoration," in *IEEE Transactions on Power Systems*, vol. 26, no. 3, pp. 1426-1433, Aug. 2011, doi: 10.1109/TPWRS.2010.2074216.

[58] J. Quirós-Tortós and V. Terzija, "A graph theory based new approach for power system restoration," *2013 IEEE Grenoble Conference*, Grenoble, France, 2013, pp. 1-6, doi: 10.1109/PTC.2013.6652108.

[59] J. Quiros-Tortos, P. Wall, L. Ding, and V. Terzija, "Determination of sectionalising strategies for parallel power system restoration: A spectral clustering-based met hodolog y," Electr. Power Syst. Res., vol. 116, pp. 381–390, Nov. 2014.

[60] F. Qiu and P. Li, "An Integrated Approach for Power System Restoration Planning," in *Proceedings of the IEEE*, vol. 105, no. 7, pp. 1234-1252, July 2017, doi: 10.1109/JPROC.2017.2696564.

[61] L. Thurner, A. Scheidler, F. Schäfer et al, pandapower - an Open Source Python Tool for Convenient Modeling, Analysis and Optimization of Electric Power Systems, IEEE Transactions on Power Systems, DOI:10.1109/TPWRS.2018.2829021, 2018.

[62] K. Carr, ICSEG Power Case 1 - IEEE 30 Bus Systems, 2013 [online], Available: https://icseg.iti.illinois.edu/ieee-30-bus-system/ [Accessed: 2- FEB- 2021].

[63] Edpresso Team, What is Dijkstra's algorithm? [online] Available:https://www.educative.io/edpresso/what-is-dijkstras-algorithm

[64] Qu HB, Liu YT (2012) Maximizing restorable load amount for specific substation during system restoration. Int J Electr Power Energy Syst 43(1):1213–1220

[65] Qu HB, Liu YT (2011) Load restoration optimization during unit start-up stage. Autom Electr Power Syst 35(8):16–21

[66] Qu HB, Liu YT (2011) Load restoration optimization during last stage of network reconfiguration. Autom Electr Power Syst 35(19):43–48.

[67] Tim Stelloh, Adela Suliman, Kurt Chirbas and Colin Sheeley, Millions in Texas without power as deadly storm brings snow, freezing weather2021 [online], Available https://www.nbcnews.com/news/weather/knocked-out-texas-millions-face-record-lows-without-power-new-n1257964

[68] Power Systems Test Case Archive. Illinois University of Washington. Available: http://www.ee.washington.edu/research/pstca/pf30/pg_tca30bus.htm.

[69] Emma Bowman, Utility Says Power Restored In New York City After Outage Hits 73,000 Customers. Available: https://www.npr.org/2019/07/13/741524829/new-york-city-power-outage-hits-73-000

[70] Emily Shapiro, New York facing 'Russian roulette' with future power outages, Gov. Andrew Cuomo says. Available: https://abcnews.go.com/US/york-facing-russian-roulette-future-power-outages-gov/story?id=64344642

APPENDICES

APPENDIX A- INDUSTRIAL SYSTEM DESIGN DATA

Parameters of the Transmission Lines

Line	**Bus**	**Resistance (p.u)**	**Reactance (p.u)**	**Susceptance (p.u)**
Line 1	Bus_1 to 15	0.007575	0.031680	0.000038
Line 2	Bus_1 to 3	0.025684	0.041305	0.000021
Line 3	Bus_3 to 11	0.013327	0.058325	0.000041
Line 4	Bus_11 to 13	0.038429	0.087106	0.000014
Line 5	Bus_13 to 18	0.461119	0.104521	0.000017

Parameters of the Transformers

Type	From	To	Vn_LV_kV	Vn_HV_kV	Sn_MVA
Step up	Bus 1	Bus 2	13.2	69	170
Step up	Bus 3	Bus 5	13.2	69	170
Step up	Bus 11	Bus 19	13.2	69	160
Step up	Bus 13	Bus 6	13.2	69	160
Step up	Bus 15	Bus 14	13.2	69	230
Step up	Bus 18	Bus 17	13.2	69	230
Step Down	Bus 2	Bus 4	13.8	69	75
Step Down	Bus 5	Bus 7	13.8	69	170
Step Down	Bus 19	Bus 8	13.8	69	75
Step Down	Bus 6	Bus 12	13.8	69	130

Step Down	Bus 14	Bus 9	13.8	69	130
Step Down	Bus 17	Bus 10	13.8	69	130
Step Down	Bus 2	Bus 20	4.16	69	170
Step Down	Bus 5	Bus 21	4.16	69	170
Step Down	Bus 19	Bus 22	4.16	69	170
Step Down	Bus 6	Bus 23	4.16	69	170
Step Down	Bus 17	Bus 25	4.16	69	170
Step Down	Bus 14	Bus 24	4.16	69	170
Step Down	Bus 18	Bus 16	13.2	4.16	170
Step Down	Bus 15	Bus 16	13.2	4.16	170

Bus Information

Bus	**Voltage (kV)**
Bus 1	13.2
Bus 2	69
Bus 3	13.2
Bus 4	13.8
Bus 5	69
Bus 6	69
Bus 7	13.8
Bus 8	13.8
Bus 9	13.8

Bus 10	13.8
Bus 11	13.2
Bus 12	13.8
Bus 13	13.2
Bus 14	69
Bus 15	13.2
Bus 16	4.16
Bus 17	69
Bus 18	13.2
Bus 19	69
Bus 20	4.16
Bus 21	4.16
Bus 22	4.16
Bus 23	4.16
Bus 24	4.16
Bus 25	4.16

APPENDIX- B CODE FOR CREATING USER INPUT USING HTML AND BOOTSTRAP

```
{% extends "base.html" %}
{% block title %}File Uploader{% endblock %}
{% block page_content %}

<!-- <div id='prompt' class="container">
   <h3>Please select an option below</h3>
      <button class="btn btn-primary" id='standard_radio'>Standard</button>
      <button class="btn btn-success" id='custom_radio'>Custom</button>
   <div id = "radiobuttons">
      <form class = "" action = "data_radio" enctype="multipart/form-data" method = "post">
      <div class = "radio">
      <label><input class="form-check-input" type="radio" name="standardradio" id="9_bus"
value="9_bus" checked>
         9 Bus System
      </label>
      </div>
      <div class = "radio">
         <label><input class="form-check-input" type="radio" name="standardradio"
id="30_bus" value="30_bus" checked>
            30 Bus System
         </label>
      </div>

      <input type = "submit" name = "radiosubmit" value = "Submit">

      </form>
   </div>
</div> -->

</form>
<div class="container" id='main_container'>
   <h3>Please upload the network information</h3>
<form class = "" action = "data" enctype="multipart/form-data" method = "post">
   <div class="col-sm-12" style="padding-left:0px">
      <div class="col-sm-6">
         Please upload xlsx file:
      </div>
      <span class="btn btn-success fileinput-button">
         <input type = "file" name="upload-file" value = "">
      </span>
```

```
    </div>
    <br/>
    <div class="col-sm-12" style="padding-left:0px" >
        <div class="col-sm-6">
            Please upload Image file:
        </div>
        <span class="btn btn-success fileinput-button">
            <input type = "file" name="upload-image" value = "">
        </span>
    </div>
    <br/>
    <br/>
    <div class="col-sm-12" style="padding-left:0px" >
        <div class="col-sm-6">
            Click on Excel logo to download sample network data
        </div>
        <a href="/static/Restoration_Sample_Data.xlsx" download="Restoration_Sample_Data">
            <img src="/static/Excel.png" alt="Excel" width="50" height="50" >
        </a>
    </div>

        <input type = "submit" name = "" value = "Submit">
</form>
</div>
<div class="panel panel-default">
    <div class="panel-heading">
        <h3 class="panel-title">Rules</h3>
    </div>
    <div class="panel-body">
        <ul>
            <li>The maximum file size for uploads is <strong>50 MB</strong> </li>
            <li>Only files <strong>XLSX</strong> are allowed to be uploaded.</li>
        </ul>
    </div>
</div>

{% endblock %}
{% block scripts %}
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
<script>
    $('#main_container').show()
    // $('#main_container').hide()
    // $('#custom_radio').on('click', () => {
    //     $('#main_container').show();
    //     $('#radiobuttons').hide();
    // })
```

```
    // $('#standard_radio').on('click', () => {
    //     $('#main_container').hide()
    //     $('#radiobuttons').show();
    // })
</script>
{% endblock %}
```

APPENDIX C RESTORATION CODE IN PYTHON

```
from flask import Flask, render_template, request,jsonify
import os
from os.path import join, dirname, realpath
from werkzeug.utils import secure_filename
import pandas as pd
from flask_bootstrap import Bootstrap
import pandapower as pp
import numpy as np
from collections import defaultdict
import os
import math
import random
import copy as cp
import time

app = Flask(__name__)
data_table = []
Bootstrap(app)
app.config['UPLOAD_FOLDER'] = '/Users/anu/Downloads/Anusha3/PSR/static/'
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/data',methods=['GET','POST'])
def data():
    if request.method == 'POST':
        file = request.files['upload-file']
```

```
        data,path = restore (file)
        image = request.files['upload-image']
        print("image",image.filename)
        image.save(os.path.join(app.config['UPLOAD_FOLDER'],
secure_filename(image.filename)))
        filename = os.path.join(app.config['UPLOAD_FOLDER']) + image.filename
        print("file", filename)
        return render_template ('data.html',data=data,data1=path,data2 = image.filename)


def restore(file):
    import pandapower.networks as pn
    import pandapower as pp
    import pandas as pd
    import numpy as np
    from collections import defaultdict
    import os
    import math
    import random
    import copy as cp
    import json
    #Graph class containing edge and weight (impedance) and algorithm to identify
    #shortest path between edges based on Impedance
    class Graph():
        def __init__(self):
            self.edges = defaultdict(list)
            self.weights = {}

        def add_edge(self, from_node, to_node, weight):
            # Note: assumes edges are bi-directional
            self.edges[from_node].append(to_node)
```

```
        self.edges[to_node].append(from_node)
        self.weights[(from_node, to_node)] = weight
        self.weights[(to_node, from_node)] = weight

    def dijsktra(graph, initial, end):
        # shortest paths is a dict of nodes
        # whose value is a tuple of (previous node, weight)
        shortest_paths = {initial: (None, 0)}
        current_node = initial
        visited = set()

        while current_node != end:
            visited.add(current_node)
            destinations = graph.edges[current_node]
            weight_to_current_node = shortest_paths[current_node][1]

            for next_node in destinations:
                weight = graph.weights[(current_node, next_node)] + weight_to_current_node
                if next_node not in shortest_paths:
                    shortest_paths[next_node] = (current_node, weight)
                else:
                    current_shortest_weight = shortest_paths[next_node][1]
                    if current_shortest_weight > weight:
                        shortest_paths[next_node] = (current_node, weight)

            next_destinations = {node: shortest_paths[node] for node in shortest_paths if node not in
visited}
            if not next_destinations:
                return "Route Not Possible"
            # next node is the destination with the lowest weight
```

```
        current_node = min(next_destinations, key=lambda k: next_destinations[k][1])

    # Work back through destinations in shortest path
    path = []
    total_weight = 0
    while current_node is not None:
        path.append(current_node)
        next_node = shortest_paths[current_node][0]
        total_weight = total_weight+shortest_paths[current_node][1]
        current_node = next_node

    # Reverse path
    path = path[::-1]
    #Return path as well as total weight
    path_and_weight = {'path':path,'total_imp':total_weight}
#    print("Path&Impedence", path_and_weight)
#    return path
    return path_and_weight
class scratch(object):
    pass
##
# Python program to print all paths from a source to destination.
# This class represents a directed graph
# using adjacency list representation
class Graph1:
    def __init__(self, vertices):
        # No. of vertices
        self.V = vertices

        # default dictionary to store graph
```

```
        self.graph = defaultdict(list)
        self.path_list = []

    # function to add an edge to graph
    def addEdge(self, u, v):
        self.graph[u].append(v)

    '''A recursive function to print all paths from 'u' to 'd'.
    visited[] keeps track of vertices in current path.
    path[] stores actual vertices and path_index is current
    index in path[]'''
    def printAllPathsUtil(self, u, d, visited, path):

        # Mark the current node as visited and store in path
        visited[u]= True
        path.append(u)

        # If current vertex is same as destination, then print
        # current path[]
        if u == d:
#           print (path)
            self.path_list.append(path.copy())
        else:
            # If current vertex is not destination
            # Recur for all the vertices adjacent to this vertex
            for i in self.graph[u]:
                if visited[i]== False:
                    self.printAllPathsUtil(i, d, visited, path)

        # Remove current vertex from path[] and mark it as unvisited
```

```
        path.pop()
        visited[u]= False

    # Prints all paths from 's' to 'd'
    def printAllPaths(self, s, d):
        self.path_list = []
        # Mark all the vertices as not visited
        visited =[False]*(self.V)

        # Create an array to store paths
        path = []

        # Call the recursive helper function to print all paths
        self.printAllPathsUtil(s, d, visited, path)
        return self.path_list

net = pp.create_empty_network()
import pandapower as pp
net = pp.create_empty_network()
restoration_file = pd.ExcelFile(file)

# Creating Buses
exc_bus = pd.read_excel (restoration_file,sheet_name='bus')

exc_bus.sort_values(by=['bus'], inplace=True)
for index, row in exc_bus.iterrows():
    pp.create_bus(net, vn_kv=row['vn_kv'], name=row['name'],in_service = True)

# Creating External grid
exc_grid = pd.read_excel (restoration_file,sheet_name='externalgrid')
```

```
    for index,row in exc_grid.iterrows():
        pp.create_ext_grid(net, bus=row['bus']-1, vm_pu=row['vm_pu'], va_degree =
row['va_degree'],
        max_p_mw = row['max_p_mw'],min_p_mw = row['min_p_mw'],max_q_mvar =
row['max_q_mvar'],
        min_q_mvar = row['min_q_mvar'],name=row['name'],in_service = True)


        ext_grid_bus = row['bus']-1


    #Creating Generators
    exc_gen = pd.read_excel (restoration_file,sheet_name='generator')


    for index, row in exc_gen.iterrows():
        pp.create_gen(net, bus=row['bus']-1,p_mw=row['p_mw'],vm_pu=row['vm_pu'],sn_mva=
row['sn_mva'],
        name=row['name'],max_q_mvar =
row['max_q_mvar'],min_q_mvar=row['min_q_mvar'],in_service=True)
    dat = []


    #Creating Loads
    exc_load = pd.read_excel (restoration_file,sheet_name='load')
    for index, row in exc_load.iterrows():
        pp.create_load(net, bus = row['bus']-1, p_mw = row['p_mw'], q_mvar= row['q_mvar'],
        const_z_percent=row['const_z_percent'], const_i_percent=row['const_i_percent'],
        sn_mva = row['sn_mva'], scaling = row['scaling'], name=row['name'], in_service=True)
        dat.append([row['bus']-1,row['priority']])


    # Creating Lines
    exc_line = pd.read_excel (restoration_file,sheet_name='line')
    for index, row in exc_line.iterrows():
```

```
        pp.create_line_from_parameters(net, from_bus = row['from_bus']-1, to_bus = row['to_bus']-
1,
        length_km=row['length_km'], r_ohm_per_km = row['r_ohm_per_km'], x_ohm_per_km =
row['x_ohm_per_km'],
        c_nf_per_km = row['c_nf_per_km'], max_i_ka =
row['max_i_ka'],name=row['name'],in_service = True)


    # Creating Transformers
    exc_trans = pd.read_excel (restoration_file,sheet_name='transformer')
    for index, row in exc_trans.iterrows():
        pp.create_transformer_from_parameters(net, hv_bus=row['hv_bus']-1,
lv_bus=row['lv_bus']-1,
        name=row['name'], sn_mva=row['sn_mva'],
vn_hv_kv=row['vn_hv_kv'],vn_lv_kv=row['vn_lv_kv'],
        vkr_percent=row['vkr_percent'], vk_percent=row['vk_percent'], pfe_kw=row['pfe_kw'],
i0_percent=row['i0_percent'],
        shift_degree=row['shift_degree'],in_service = True,tap_side = row['tap_side'])
    pp.runpp(net)


    # Create internal table for motors
    motors = pd.read_excel (restoration_file,sheet_name = 'motorload')
    motors['p'] = motors['motor_hp'] * 0.000746
    motors['q'] = motors['p'] * np.tan(np.arccos(motors['power_factor_full load']))
    motors['p_total'] = motors['p'] * motors['no_of_motors']
    motors['q_total'] = motors['q'] * motors['no_of_motors']
    motors['p_inrush'] = motors['q'] * motors['no_of_motors'] * np.sqrt(3)
    motors['irated'] = ((motors['motor_hp']*746)/motors['power_factor_full
load'])/(np.sqrt(3)*motors['voltage_kv']*1000*motors['efficiency_full_load'])
    motors['p_inrush'] =
motors['voltage_kv']*np.sqrt(3)*motors['irated']*6*motors['power_factor_locked_rotor']*(1/(np.
power(10,6)))*1000
    motors['q_inrush'] =
motors['voltage_kv']*np.sqrt(3)*motors['irated']*6*np.sin(np.arccos(motors['power_factor_locke
d_rotor']))*(1/np.power(10,6))*1000
```

```
    motors['p_inrush_tot'] = motors['p_inrush'] * motors['no_of_motors']
    motors['q_inrush_tot'] = motors['q_inrush'] * motors['no_of_motors']
    motors['processed'] = 'N'
    motors_renamed = motors.rename(columns={'motor_hp':'motor'})
    motors_renamed['load_bus'] = motors_renamed['load_bus']-1
    static_motor = []
    static_motor_row = []
    sorted_motor = motors_renamed.groupby(["load_bus"]).apply(lambda x:
x.sort_values(["q_inrush_tot"], ascending = False)).reset_index(drop=True)
    for index,row in net.load.iterrows():
        no_of_motors = len(sorted_motor[sorted_motor['load_bus']==row['bus']])
        for i in range(0,no_of_motors):
            static_motor_row = [(row['p_mw']-
sum(sorted_motor.loc[sorted_motor['load_bus']==row['bus'],'p_total']))*(1/no_of_motors),
                                (row['q_mvar']-
sum(sorted_motor.loc[sorted_motor['load_bus']==row['bus'],'q_total']))*(1/no_of_motors),row['b
us'],i,'N']
            static_motor.append(static_motor_row)
    static_data = pd.DataFrame(static_motor, columns = ['p', 'q','load_bus','id','processed'])


    #Create internal table for load priority
    load_priority = pd.DataFrame(dat, columns = ['load_bus', 'priority'])
    load_priority['processed'] = 'N'
    load_priority = load_priority.sort_values(by = ['priority'])


    #Create Table with Cranking Power
    gens =[]
    gens_power_p=[]
    gens_power_q=[]
    busind = []
    gens_name = []
```

```
# Considering few as thermal and few as gas turbine generators
    for index,row in net.gen.iterrows():
        if row['name'] in ['G2','G3']:
            cranking_power_p=0.02*(abs(row['p_mw']))
            if math.isnan(net.gen['sn_mva'][0]):
                cranking_power_q = 0
            else:
                cranking_power_q=0.02*(math.sqrt(row['sn_mva']**2-row['p_mw']**2))
        else:
            cranking_power_p=0.07*(abs(row['p_mw']))
            if math.isnan(net.gen['sn_mva'][0]):
                cranking_power_q = 0
            else:
                cranking_power_q=0.07*(math.sqrt(row['sn_mva']**2-row['p_mw']**2))
        gens.append(index)
        gens_name.append(row['name'])
        busind.append(row['bus'])
        gens_power_p.append(cranking_power_p)
        gens_power_q.append(cranking_power_q)

    d =
{'gen':gens,'gen_name':gens_name,'bus':busind,'pow':gens_power_p,'pow_q':gens_power_q}
    c_pow = pd.DataFrame(data = d)
    q = len(net.line)

    #create switches between bus and line
    for frombus in net.line.from_bus:
            lineinfo = net.line.loc[net.line['from_bus']==frombus]
            for index, lines in lineinfo.iterrows():
```

```
            duplicatecheck_frombus = net.switch[(net.switch['bus']==frombus) &
(net.switch['element']==index)]
            if duplicatecheck_frombus.empty:
                pp.create_switch(net, bus= frombus, element= index, et = 'l')
            duplicatecheck_tobus = net.switch[(net.switch['bus']==lines.to_bus) &
(net.switch['element']==index)]
            if duplicatecheck_tobus.empty:
                pp.create_switch(net, bus=lines.to_bus, element=index, et='l')
            del duplicatecheck_frombus, duplicatecheck_tobus


    #Create switches between bus and transformer
    for index, lines in net.trafo.iterrows():
        load_index = net.load.loc[net.load['bus']==frombus]
        pp.create_switch(net, bus= lines.hv_bus, element= index, et = 't')
        pp.create_switch(net, bus= lines.lv_bus, element= index, et = 't')


    net.switch.drop_duplicates(keep=False,inplace=True)


    #Create Impedance
    q = len(net.line)


    impedance = {}
    for a in range(0,q):
        impedance[net.line.name[a]] =
abs(net.line.r_ohm_per_km[a]+1j*net.line.x_ohm_per_km[a])
    l=[]
    for s in net.bus.name:
        l.append(s)
    nodes=l
    distances = {}
    to_bus_data = {}
```

```
from_bus_data = {}
#Create Edges and Weights for the graph structure
#Edges are buses and weights is impedance
edges_outer = []
each_record = ()
for t in net.line.from_bus:
        result= net.line.loc[net.line['from_bus']==t]
        to_bus_data ={}
        for index, lines in result.iterrows():
            each_record = (t,lines.to_bus,impedance[lines['name']])
            edges_outer.append(each_record)
            to_bus_data[lines.to_bus] = impedance[lines['name']]
        distances[t] = to_bus_data
        del result
        del to_bus_data



for index, row in net.trafo.iterrows():
    each_record = (row['hv_bus'], row['lv_bus'],0)
    edges_outer.append(each_record)
graph= Graph()
edges_outer = list(set(edges_outer))
for edge in edges_outer:
    graph.add_edge(*edge)
# Create a graph given in the above diagram
no_of_buses = len(net.bus.name.unique())
g = Graph1(no_of_buses)
for index, row in net.line.iterrows():
    g.addEdge(row['from_bus'], row['to_bus'])
    g.addEdge(row['to_bus'],row['from_bus'])
```

```
    for index, row in net.trafo.iterrows():
        g.addEdge(row['hv_bus'], row['lv_bus'])
        g.addEdge(row['lv_bus'],row['hv_bus'],)
    gen_load_all_paths = []
    all_gen_load_paths = []
    all_gen_load_paths = pd.DataFrame(columns=['gen', 'bus', 'all_paths'])
    for index, row in net.gen.iterrows():
        for index1,row1 in net.load.iterrows():
            all_paths = g.printAllPaths(row['bus'],row1['bus'])
            gen_load_path = {'gen':row['bus'],'bus':row1['bus'],'all_paths':all_paths}
            gen_load_all_paths.append(gen_load_path)


    # Between each pair of generators perform Dijsktra to find out shortest path
    prev_gen = None
    result = []
    processed = []
    load_temp = net.load
    #Creating temp variable for net.gen.bus to include external grid bus in the list
    gen_incl_ext_grid = []
    #for each_ext_grid in net.ext_grid.bus:
    #    gen_incl_ext_grid.append(each_ext_grid)
    for each_gen in net.gen.bus:
        gen_incl_ext_grid.append(each_gen)
    gen_incl_ext_grid = list(set(gen_incl_ext_grid))
    for prev_gen in gen_incl_ext_grid:
        for curr_gen in gen_incl_ext_grid:
            if prev_gen != curr_gen and curr_gen not in processed:
                short_path = dijsktra(graph,prev_gen,curr_gen)
    # Append Source, Dest, ShortestPath, Impedence along the path, Cranking power and
generator power
```

```
            shortest_path_result = dict()
            shortest_path_result['source_gen'] = prev_gen
            shortest_path_result['dest_gen'] = curr_gen
            shortest_path_result['path'] = short_path["path"]
            shortest_path_result['imp'] = short_path["total_imp"]
            shortest_path_result['c_pow'] = c_pow.loc[c_pow['bus']==curr_gen,'pow'].sum()
            shortest_path_result['c_pow_q'] = c_pow.loc[c_pow['bus']==curr_gen,'pow_q'].sum()
            if any(net.gen.bus==prev_gen):
                shortest_path_result['source_pow'] =
abs(net.gen.loc[net.gen['bus']==prev_gen,'p_mw'].values[0])
                if math.isnan(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]):
                    shortest_path_result['source_pow_q'] = 0
                else:


                    shortest_path_result['source_pow_q'] =
abs(math.sqrt(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]**2-
                                net.gen.loc[net.gen['bus']==prev_gen,'p_mw'].values[0]**2))
                math.sqrt(row['sn_mva']**2-row['p_mw']**2)
            else:
                shortest_path_result['source_pow'] =
abs(net.res_ext_grid.loc[net.ext_grid['bus']==prev_gen,'p_mw'].values[0])
                if math.isnan(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]):
                    shortest_path_result['source_pow_q'] = 0
                else:
                    shortest_path_result['source_pow_q'] =
abs(math.sqrt(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]**2-
                                net.gen.loc[net.gen['bus']==prev_gen,'p_mw'].values[0]**2))
            if any(net.gen.bus==curr_gen):
                shortest_path_result['dest_pow'] =
abs(net.gen.loc[net.gen['bus']==curr_gen,'p_mw'].values[0])
                if math.isnan(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]):
```

```
                shortest_path_result['source_pow_q'] = 0
            else:
                shortest_path_result['dest_pow_q'] =
abs(math.sqrt(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]**2-
                        net.gen.loc[net.gen['bus']==prev_gen,'p_mw'].values[0]**2))
        else:
            shortest_path_result['dest_pow'] =
abs(net.res_ext_grid.loc[net.ext_grid['bus']==curr_gen,'p_mw'].values[0])
            if math.isnan(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]):
                shortest_path_result['source_pow_q'] = 0
            else:
                shortest_path_result['dest_pow_q'] =
abs(math.sqrt(net.gen.loc[net.gen['bus']==prev_gen,'sn_mva'].values[0]**2-
                        net.gen.loc[net.gen['bus']==prev_gen,'p_mw'].values[0]**2))

        result.append(shortest_path_result)
    processed.append(prev_gen)

# Opening switches based on Lowest Impedence First and lowest cranking power
# Gen1 is the black start
black_start = 0
#Filtering the data based on condition that source generator is 1. Since we have to process
from gen1
conditions = {'source_gen':0}
bs_result = [one_dict for one_dict in result if
        all(key in one_dict and conditions[key] == one_dict[key]
            for key in conditions.keys())]
# Sort the filtered data on Impedance and cranking power in ascending order
#bs_result_sorted = sorted(bs_result, key = lambda i: (i['imp'],i['c_pow']))
bs_result_sorted = sorted(bs_result, key = lambda i: (i['imp']))
```

```
path = []
total_imp = 0
for eachrow in bs_result_sorted:
    path.append(eachrow['dest_gen'])
    total_imp = total_imp+eachrow['imp']
short_path['path'] = path
short_path['total_imp'] = total_imp
net.switch['closed']=False
net.gen['in_service']=False
filecount = 0
####### Looping Swith Logic to be added ##########

#Loop on filtered data where source generator is 0
#for eachrow in bs_result_sorted:
#Get the shortest path between two generators
net.gen.loc[net.gen['bus']==ext_grid_bus,'in_service']=True
net.gen.loc[net.gen['bus']==ext_grid_bus,'slack']=True
slack_vm_pu = net.gen.loc[net.gen['slack']==True,'vm_pu']
net.load.in_service = False
processed_bus = []
inrush_bus = []
load_temp = net.load.copy()
indexnames = None
picked_load1 = 0
picked_load2 = 0
available_gen = pd.DataFrame(columns = net.gen.columns.values)
unprocessed_temp = []
avail_list_gen = []

unprocessed_gen = cp.deepcopy(net.gen)
```

```
    for index, row in unprocessed_gen.iterrows():

        if math.isnan(row['sn_mva']):
            row['q'] = 0
        else:
            row['q'] = pd.eval(np.sqrt(row['sn_mva']**2-row['p_mw']**2))

    cond = unprocessed_gen['name'].isin(available_gen['name'])
    unprocessed_gen = unprocessed_gen.drop(unprocessed_gen[cond].index)
    unprocessed_load = cp.deepcopy(net.load)
    not_completed_load = cp.deepcopy(net.load)
    net_copy = pp.create_empty_network()
    net_copy = cp.deepcopy(net)
    iteration = float(0)
    rest_output = []
    current_load_processed = False
    current_gen = None
    next_gen = None
    rest_col_names = ['iteration','gen_turned_on','eff_gen_cap_p', 'eff_gen_cap_q',

'cranking_power_provided_gen','cranking_power_p','cranking_power_q','Load_Name','motor_gr
oup','pli_mw',
    'qli_mvar','p_mw',
'q_mw','lp_mw','lq_mvar','pr_mw','qr_mvar','Voltage_Drop','Voltage_Drop_steady']

    for eachrow in bs_result_sorted:
        restoration_path = eachrow.get('path')
        for rest_var in range(0,len(restoration_path)):
            if restoration_path[rest_var] not in list(available_gen.bus):
                current_gen = restoration_path[rest_var]
                if rest_var < len(restoration_path)-1:
```

```
            next_gen = restoration_path[rest_var+1]
        else:
            next_gen = None
        if next_gen == None:
            sp = short_path['path']
            if sp.index(current_gen) < len(sp) - 1:
                if black_start == sp[sp.index(current_gen)+1]:
                    next_gen = sp[sp.index(current_gen)+2]
                else:
                    next_gen = sp[sp.index(current_gen)+1]


        net_copy.gen.loc[net.gen['bus']==current_gen,'in_service']=True
        available_gen =
available_gen.append(net_copy.gen.loc[net.gen['bus']==current_gen],sort=True)
        available_gen['q'] = pd.eval(np.sqrt(available_gen['sn_mva']**2-
available_gen['p_mw']**2))
        cond = unprocessed_gen['name'].isin(available_gen['name'])
        unprocessed_gen = unprocessed_gen.drop(unprocessed_gen[cond].index)


        gen_capacity = float(0)
        gen_capacity_q = float(0)
        cranking_power = None
        cranking_power_q = None


    ## Calculate Available generation capacity, processed load and effective generation
capability
        gen_capacity = gen_capacity + abs(available_gen['p_mw'].sum())
        gen_capacity_q = gen_capacity_q + abs(available_gen['q'].sum())


        cranking_power = abs(c_pow.loc[c_pow['bus']==next_gen,'pow'].sum())
        cranking_power_q = abs(c_pow.loc[c_pow['bus']==next_gen,'pow_q'].sum())
```

```
            try:
                processed_load_steadystate_p = static_data.query("processed == 'Y'")['p'].sum()
                processed_load_steadystate_q = static_data.query("processed == 'Y'")['q'].sum()
            except IndexError:
                processed_load_steadystate_p = 0
                processed_load_steadystate_q = 0
            try:
                processed_load_steadystate_mot_p = sorted_motor.query("processed ==
'Y'")['p_total'].sum()
                processed_load_steadystate_mot_q = sorted_motor.query("processed ==
'Y'")['q_total'].sum()
            except IndexError:
                processed_load_steadystate_mot_p = 0
                processed_load_steadystate_mot_q = 0
            eff_gen_cap = gen_capacity - cranking_power - processed_load_steadystate_p -
processed_load_steadystate_mot_p
            eff_gen_cap_q = gen_capacity_q - cranking_power_q - processed_load_steadystate_q
- processed_load_steadystate_mot_q
            load_processed = False
            current_load_completed = False
            insufficient_capacity = False
            for l_index, l_row in load_priority.iterrows():
                if insufficient_capacity == False:
                    current_load = l_row['load_bus']
                else:
                    break
                for eachload_paths in gen_load_all_paths:
                    if ((available_gen[available_gen['bus'] == eachload_paths.get('gen')].any().any())
& (eachload_paths.get('bus')==current_load)) == True:
                        all_paths_arr = eachload_paths.get('all_paths')
                        valid_path = []
```

```
unprocessed_gen_set = set(unprocessed_gen['bus'])
unprocessed_load_set = set(unprocessed_load['bus'])
trans_hv = set(net_copy.trafo['hv_bus'])
trans_lv = set(net_copy.trafo['lv_bus'])
for i in range(0,len(all_paths_arr)):
    valid_path_flag = True
    single_path = all_paths_arr[i]
    single_path_set = set(single_path)
    single_path_set_load = set(single_path[:-1])
    if unprocessed_gen_set.intersection(single_path_set):
        valid_path_flag = False
    if valid_path_flag == True:
        if unprocessed_load_set.intersection(single_path_set_load):
            valid_path_flag = False
    if valid_path_flag == True:
        if trans_hv.intersection(single_path_set):
            valid_path_flag = False
        if trans_lv.intersection(single_path_set):
            valid_path_flag = False


    if valid_path_flag == True:
        valid_path = all_paths_arr[i]
        break
if valid_path_flag == False:
    for i in range(0,len(all_paths_arr)):
        valid_path_flag = True
        single_path = all_paths_arr[i]
        single_path_set = set(single_path)
        single_path_set_load = set(single_path[:-1])
        if unprocessed_gen_set.intersection(single_path_set):
```

```
                    valid_path_flag = False
                if valid_path_flag == True:
                    if unprocessed_load_set.intersection(single_path_set_load):
                        valid_path_flag = False
                if valid_path_flag == True:
                    for j in range(0,len(single_path)-1):
                        if (net_copy.trafo.loc[(net_copy.trafo.hv_bus == single_path[j]) &
(net_copy.trafo.lv_bus == single_path[j+1]) & (net_copy.trafo.tap_side == 'hv')].any().any() or
                            net_copy.trafo.loc[(net_copy.trafo.hv_bus == single_path[j+1]) &
(net_copy.trafo.lv_bus == single_path[j]) & (net_copy.trafo.tap_side == 'lv')].any().any()):
                            valid_path_flag = False
                            break
                if valid_path_flag == True:
                    valid_path = all_paths_arr[i]
                    break

            if valid_path_flag == False:
                continue

            if valid_path_flag == True:
                for i in range(0, len(valid_path)-1):
                    if (valid_path[i] is not None) & (valid_path[i+1] is not None):
                        line_bw_buses = net_copy.line.loc[(net_copy.line['from_bus'] ==
valid_path[i]) & (net_copy.line['to_bus'] == valid_path[i+1])]
                        if len(line_bw_buses) == 0:
                            line_bw_buses = net_copy.line.loc[(net_copy.line['from_bus'] ==
valid_path[i+1]) & (net_copy.line['to_bus'] == valid_path[i])]
                        if len(line_bw_buses) > 0:
                            net_copy.switch.loc[(net_copy.switch['element'] ==
line_bw_buses.index[0]) & (net_copy.switch['et'] == 'l'),'closed'] = True
                        trafo_bw_buses = net_copy.trafo.loc[(net_copy.trafo['hv_bus'] ==
valid_path[i]) & (net_copy.trafo['lv_bus'] == valid_path[i+1])]
```

```
                    if len(trafo_bw_buses) == 0:
                        trafo_bw_buses = net_copy.trafo.loc[(net_copy.trafo['lv_bus'] ==
valid_path[i]) & (net_copy.trafo['hv_bus'] == valid_path[i+1])]
                    if len(trafo_bw_buses) > 0:
                        net_copy.switch.loc[(net_copy.switch['element'] ==
int(trafo_bw_buses.index[0])) & (net_copy.switch['et'] == 't'),'closed'] = True
                    temp_trafo_switch = net_copy.trafo.loc[net_copy.trafo.hv_bus ==
valid_path[i]]
                    if len(temp_trafo_switch) == 0:
                        temp_trafo_switch = net_copy.trafo.loc[net_copy.trafo.lv_bus ==
valid_path[i]]
                    if len(temp_trafo_switch) >0:
                        for ts_iter,ts_row in temp_trafo_switch.iterrows():
                            if ts_row['hv_bus'] == valid_path[i] and
net_copy.load.loc[(net_copy.load.bus == ts_row['lv_bus']) & (net_copy.load.in_service ==
True)].any().any():
                                trans_index = net_copy.trafo.loc[(net_copy.trafo.hv_bus ==
ts_row['hv_bus'])& (net_copy.trafo.lv_bus == ts_row ['lv_bus'])].index.values
                                net_copy.switch.loc[(net_copy.switch['element'] ==
trans_index[0]) & (net_copy.switch['et'] == 't'),'closed'] = True
                            elif (ts_row['lv_bus'] == valid_path[i]) and
(net_copy.load.loc[(net_copy.load.bus == ts_row['hv_bus']) & (net_copy.load.in_service ==
True)].any().any()):
                                trans_index = net_copy.trafo.loc[(net_copy.trafo.hv_bus ==
ts_row['hv_bus'])& (net_copy.trafo.lv_bus == ts_row ['lv_bus'])].index.values
                                net_copy.switch.loc[(net_copy.switch['element'] ==
trans_index[0]) & (net_copy.switch['et'] == 't'),'closed'] = True
                    unprocessed_load.drop(unprocessed_load[unprocessed_load['bus'] ==
int(current_load)].index,inplace = True)
                    try:
                        motor = sorted_motor[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'N')&(sorted_motor.p_inrush_tot < eff_gen_cap)
&(sorted_motor.q_inrush_tot < eff_gen_cap_q)].iloc[0]
                    except IndexError:
                        motor = None
```

```
                except TypeError:
                    motor = None
                    print("test")
                while 1==1:
                    static = None
                    motor = None
                    try:
                        static = static_data[(static_data.load_bus ==
int(current_load))&(static_data.processed == 'N')].iloc[0]
                    except IndexError:
                        static = None
                    except TypeError:
                        static = None


                    if static is not None:
                        static_p = static['p']
                        static_q = static['q']
                    else:
                        static_p = 0
                        static_q = 0


                    try:
                        motor = sorted_motor[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'N')&(np.floor(sorted_motor.p_inrush_tot +
static_p)+5 < math.ceil(eff_gen_cap)) &(np.floor(sorted_motor.q_inrush_tot + static_q)+5 <
math.ceil(eff_gen_cap_q))].iloc[0]
                    except IndexError:
                        motor = None
                    except TypeError:
                        motor = None
```

```
                    if motor is not None:

net_copy.load.loc[(net_copy.load['bus']==current_load),'in_service']=True
                        print(iteration)
                        picked_total_load1 = motor['p_inrush_tot']+static_p
                        picked_total_load1_q = motor['q_inrush_tot']+static_q

net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'p_mw']=picked_total_load1
                        + sum(sorted_motor.loc[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'Y'),'p_total'])

net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'q_mvar']=picked_total_load1_q
                        + sum(sorted_motor.loc[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'Y'),'q_total'])
                        net_copy.load.sn_mva =
np.sqrt(np.power(net_copy.load.p_mw,2)+np.power(net_copy.load.q_mvar,2))
                        picked_steady_load1 = static_p+motor['p_total']
                        picked_steady_load1_q = static_q+motor['q_total']
                        random_multi = round(random.uniform(0.05,0.1),2)
                        powerflow_Inrush = pp.runpp(net_copy)
                        if net_copy.converged == True:
                            line_index = net_copy.line[(net_copy.line['from_bus']==valid_path[-
2]) & (net_copy.line['to_bus']==valid_path[-1])].index.tolist()
                            if len(line_index) == 0:
                                line_index = net_copy.line[(net_copy.line['to_bus']==valid_path[-
2]) & (net_copy.line['from_bus']==valid_path[-1])].index.tolist()
                                if len(line_index) == 0:
                                    line_index =
net_copy.trafo[(net_copy.trafo['lv_bus']==valid_path[-2]) &
(net_copy.trafo['hv_bus']==valid_path[-1])].index.tolist()
                                    if len(line_index) == 0:
                                        line_index =
net_copy.trafo[(net_copy.trafo['hv_bus']==valid_path[-2]) &
(net_copy.trafo['lv_bus']==valid_path[-1])].index.tolist()
```

```
                                    line_result = net_copy.res_trafo.iloc[line_index,]
                                    try:
                                        inrush_vd = round(((line_result['vm_hv_pu']-
line_result['vm_lv_pu'])/line_result['vm_hv_pu'])*100,2).values[0]
                                    except IndexError:
                                        inrush_vd = 0
                                else:
                                    line_result = net_copy.res_trafo.iloc[line_index,]
                                    try:
                                        inrush_vd = round(((line_result['vm_lv_pu']-
line_result['vm_hv_pu'])/line_result['vm_lv_pu'])*100,2).values[0]
                                    except IndexError:
                                        inrush_vd = 0
                            else:
                                line_result = net_copy.res_line.iloc[line_index,]
                                try:
                                    inrush_vd = round(((line_result['vm_from_pu']-
line_result['vm_to_pu'])/line_result['vm_to_pu'])*100,2).values[0]
                                except IndexError:
                                    inrush_vd = 0
                        else:


                            line_result = net_copy.res_line.iloc[line_index,]
                            try:
                                inrush_vd = round(((line_result['vm_from_pu']-
line_result['vm_to_pu'])/line_result['vm_to_pu'])*100,2).values[0]
                            except IndexError:
                                inrush_vd = 0


net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'p_mw']=picked_steady_load1
                                + sum(sorted_motor.loc[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'Y'),'p_total'])
```

```
net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'q_mvar']=picked_steady_load1_q
                    + sum(sorted_motor.loc[(sorted_motor.load_bus ==
int(current_load))&(sorted_motor.processed == 'Y'),'q_total'])
                    net_copy.load.sn_mva =
np.sqrt(np.power(net_copy.load.p_mw,2)+np.power(net_copy.load.q_mvar,2))
                    powerflow_Inrush = pp.runpp(net_copy)
                    iteration = iteration + 1
                    rest_row = None
                    rest_df = None
                    if len(rest_output) > 0:


                        line_index =
net_copy.line[(net_copy.line['from_bus']==valid_path[-2]) &
(net_copy.line['to_bus']==valid_path[-1])].index.tolist()
                        if len(line_index) == 0:
                            line_index =
net_copy.line[(net_copy.line['to_bus']==valid_path[-2]) &
(net_copy.line['from_bus']==valid_path[-1])].index.tolist()
                            if len(line_index) == 0:
                                line_index =
net_copy.trafo[(net_copy.trafo['lv_bus']==valid_path[-2]) &
(net_copy.trafo['hv_bus']==valid_path[-1])].index.tolist()
                                if len(line_index) == 0:
                                    line_index =
net_copy.trafo[(net_copy.trafo['hv_bus']==valid_path[-2]) &
(net_copy.trafo['lv_bus']==valid_path[-1])].index.tolist()
                                    line_result = net_copy.res_trafo.iloc[line_index,]
                                    try:
                                        normalvd = round(((line_result['vm_hv_pu']-
line_result['vm_lv_pu'])/line_result['vm_hv_pu'])*100,2).values[0]
                                    except IndexError:
                                        normalvd = 0
                                else:
```

```
                    line_result = net_copy.res_trafo.iloc[line_index,]
                    try:
                        normalvd = round(((line_result['vm_lv_pu']-
line_result['vm_hv_pu'])/line_result['vm_lv_pu'])*100,2).values[0]
                    except IndexError:
                        normalvd = 0
                else:
                    line_result = net_copy.res_line.iloc[line_index,]
                    try:
                        normalvd = round(((line_result['vm_to_pu']-
line_result['vm_from_pu'])/line_result['vm_from_pu'])*100,2).values[0]
                    except IndexError:
                        normalvd = 0
            else:

                line_result = net_copy.res_line.iloc[line_index,]
                try:
                    normalvd = round(((line_result['vm_from_pu']-
line_result['vm_to_pu'])/line_result['vm_to_pu'])*100,2).values[0]
                except IndexError:
                    normalvd = 0



            if
rest_output.loc[rest_output['cranking_power_provided_gen']==str(c_pow.loc[c_pow['bus']==nex
t_gen,'gen_name'].tolist()).strip('[]')].any().any():
                rest_row = [[iteration,
                '-',
                round(eff_gen_cap,2),
                round(eff_gen_cap_q,2),
                '-',
```

```
                                    '-',
                                    '-',

net_copy.load.loc[(net_copy.load.bus==int(current_load)),'name'].values[0],
                                    motor['motor'],
                                    round(motor['p_inrush_tot'],2),
                                    round(motor['q_inrush_tot'],2),
                                    round(picked_total_load1,2),
                                    round(picked_total_load1_q,2),
                                    round(picked_steady_load1,2),
                                    round(picked_steady_load1_q,2),
                                    round(picked_steady_load1+(static_p*random_multi),2),
                                    round(picked_steady_load1_q+(static_q*random_multi),2),
                                    inrush_vd,
                                    normalvd]]
                                    rest_df = pd.DataFrame(rest_row,columns = rest_col_names)
                                else:
                                    rest_row = [[iteration,

str(net_copy.gen.loc[(net_copy.gen.in_service==True),'name'].tolist()).strip('[]'),
                                    round(eff_gen_cap,2),
                                    round(eff_gen_cap_q,2),
                                    '-' if next_gen is None else

str(c_pow.loc[c_pow['bus']==next_gen,'gen_name'].tolist()).strip('[]'),
                                    round(cranking_power,2),
                                    round(cranking_power_q,2),

net_copy.load.loc[(net_copy.load.bus==int(current_load)),'name'].values[0],
                                    motor['motor'],
                                    round(motor['p_inrush_tot'],2),
```

```
                                round(motor['q_inrush_tot'],2),
                                round(picked_total_load1,2),
                                round(picked_total_load1_q,2),
                                round(picked_steady_load1,2),
                                round(picked_steady_load1_q,2),
                                round(picked_steady_load1+(static_p*random_multi),2),
                                round(picked_steady_load1_q+(static_q*random_multi),2),
                                inrush_vd,
                                normalvd]]
                                rest_df = pd.DataFrame(rest_row,columns = rest_col_names)
                            else:
                                line_index =
net_copy.line[(net_copy.line['from_bus']==valid_path[-2]) &
(net_copy.line['to_bus']==valid_path[-1])].index.tolist()
                                if len(line_index) == 0:
                                    line_index =
net_copy.line[(net_copy.line['to_bus']==valid_path[-2]) &
(net_copy.line['from_bus']==valid_path[-1])].index.tolist()
                                    if len(line_index) == 0:
                                        line_index =
net_copy.trafo[(net_copy.trafo['lv_bus']==valid_path[-2]) &
(net_copy.trafo['hv_bus']==valid_path[-1])].index.tolist()
                                        if len(line_index) == 0:
                                            line_index =
net_copy.trafo[(net_copy.trafo['hv_bus']==valid_path[-2]) &
(net_copy.trafo['lv_bus']==valid_path[-1])].index.tolist()
                                            line_result = net_copy.res_trafo.iloc[line_index,]
                                            try:
                                                normalvd = round(((line_result['vm_hv_pu']-
line_result['vm_lv_pu'])/line_result['vm_hv_pu'])*100,2).values[0]
                                            except IndexError:
                                                normalvd = 0
                                        else:
```

```
                                    line_result = net_copy.res_trafo.iloc[line_index,]
                                    try:
                                        normalvd = round(((line_result['vm_lv_pu']-
line_result['vm_hv_pu'])/line_result['vm_lv_pu'])*100,2).values[0]
                                    except IndexError:
                                        normalvd = 0
                                else:
                                    line_result = net_copy.res_line.iloc[line_index,]
                                    try:
                                        normalvd = round(((line_result['vm_to_pu']-
line_result['vm_from_pu'])/line_result['vm_from_pu'])*100,2).values[0]
                                    except IndexError:
                                        normalvd = 0
                            else:

                                line_result = net_copy.res_line.iloc[line_index,]
                                try:
                                    normalvd = round(((line_result['vm_from_pu']-
line_result['vm_to_pu'])/line_result['vm_to_pu'])*100,2).values[0]
                                except IndexError:
                                    normalvd = 0
                            rest_row = [[iteration,

str(net_copy.gen.loc[(net_copy.gen.in_service==True),'name'].tolist()).strip('[]'),
                                round(eff_gen_cap,2),
                                round(eff_gen_cap_q,2),

str(c_pow.loc[c_pow['bus']==next_gen,'gen_name'].tolist()).strip('[]'),
                                round(cranking_power,2),
                                round(cranking_power_q,2),

net_copy.load.loc[(net_copy.load.bus==int(current_load)),'name'].values[0],
```

```
                    motor['motor'],
                    round(motor['p_inrush_tot'],2),
                    round(motor['q_inrush_tot'],2),
                    round(picked_total_load1,2),
                    round(picked_total_load1_q,2),
                    round(picked_steady_load1,2),
                    round(picked_steady_load1_q,2),
                    round(picked_steady_load1+(static_p*random_multi),2),
                    round(picked_steady_load1_q+(static_q*random_multi),2),
                    inrush_vd,
                    normalvd]]
                    rest_df = pd.DataFrame(rest_row,columns = rest_col_names)
                try:
                    rest_output = rest_output.append(rest_df,ignore_index = False)
                except:
                    rest_output = rest_df.copy()


net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'p_mw']=picked_steady_load1

net_copy.load.loc[(net_copy.load['bus']==int(current_load)),'q_mvar']=picked_steady_load1_q
                    sorted_motor.loc[(sorted_motor['load_bus'] == int(current_load)) &
(sorted_motor['motor'] == motor['motor']),'processed'] = 'Y'

                    powerflow_Inrush = pp.runpp(net_copy)

                    if static is not None:
                        static_data.loc[(static_data['load_bus'] == int(current_load)) &
(static_data['id'] == static['id']),'processed'] = 'Y'
                        static_data.loc[(static_data['load_bus'] == int(current_load)) &
(static_data['id'] == static['id']),'p'] = static_p+(static_p*random_multi)
```

```
                    static_data.loc[(static_data['load_bus'] == int(current_load)) &
(static_data['id'] == static['id']),'q'] = static_q+(static_q*random_multi)


                cranking_power =
abs(c_pow.loc[c_pow['bus']==next_gen,'pow'].sum())
                cranking_power_q =
abs(c_pow.loc[c_pow['bus']==next_gen,'pow_q'].sum())
                try:
                    processed_load_steadystate_p = static_data.query("processed ==
'Y'")['p'].sum()
                    processed_load_steadystate_q = static_data.query("processed ==
'Y'")['q'].sum()
                except IndexError:
                    processed_load_steadystate_p = 0
                    processed_load_steadystate_q = 0


                try:
                    processed_load_steadystate_mot_p = sorted_motor.query("processed
== 'Y'")['p_total'].sum()
                    processed_load_steadystate_mot_q = sorted_motor.query("processed
== 'Y'")['q_total'].sum()
                except IndexError:
                    processed_load_steadystate_mot_p = 0
                    processed_load_steadystate_mot_q = 0


                eff_gen_cap = gen_capacity - cranking_power -
processed_load_steadystate_p - processed_load_steadystate_mot_p
                eff_gen_cap_q = gen_capacity_q - cranking_power_q -
processed_load_steadystate_q - processed_load_steadystate_mot_q


            else:
                if ((sorted_motor['load_bus']==int(current_load)) &
(sorted_motor['processed'] == 'N')).any() == False:
```

```
load_priority.loc[load_priority['load_bus']==int(current_load),'processed'] = 'Y'
                                current_load_completed = True

not_completed_load.drop(not_completed_load[not_completed_load['bus'] ==
int(current_load)].index,inplace = True)

load_priority.drop(load_priority[load_priority['load_bus']==int(current_load)].index,inplace =
True)
                                net_copy.load.loc[net_copy.load['bus']==int(current_load),'p_mw'] =
net.load.loc[net.load['bus']==int(current_load),'p_mw']
                                net_copy.load.loc[net_copy.load['bus']==int(current_load),'q_mvar']
= net.load.loc[net.load['bus']==int(current_load),'q_mvar']
                            else:
                                insufficient_capacity = True
                            break


                        total_static_p = 0
                        total_static_q = 0
                        break



    print (rest_output)
    abs_path = []
    for x in short_path['path']:
        for i,row in net.gen.iterrows():
            if row['bus'] == x:
                if len(abs_path) == 0:
                    abs_path.append(net.ext_grid['name'].iloc[0])
                if row['name'] != net.ext_grid['name'].iloc[0]:
                    abs_path.append(row['name'])
    return rest_output,abs_path
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

APPENDIX D CODE FOR OUTPUT DISPLAY USING JQUERY EMBEDDED IN HTML

```
{% extends "base.html" %}
{% block title %}Results Page{% endblock %}
<!-- <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>   -->
<script src="../static/js/jquery.min.js"></script>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>

{% block page_content %}

<!-- <div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
   <div class="card-header">Current Load</div>
   <div class="card-body">
     <p class="card-text"></p>
   </div>
</div> -->
<body>
<!-- <img src= "{{ data2 }}" alt="Circuit Image"> -->
<div class="container-fluid">
 <div class="row">
  <div class = "col-sm-5">
  <p><h1>Restoration Sequence:</h1>  </p>
  {% for sp1 in data1 %}
    {% if loop.last %}
       <b>{{ sp1 }} </b>
    {% else %}
       <b>{{ sp1 }} ,</b>
    {% endif %}
{% endfor %}
<h3>Abbreviations:</h3>
<p>P<sub>AVA</sub>_mw : Available real power at a given point during restoration </p>
<p> </p>Q<sub>AVA</sub>_mvar :  Available reactive power at a given point during </p>
<p>P<sub>CR</sub>_mw  Cranking power of generator's real power </p>
<p>Q<sub>CR</sub>_mvar Cranking power of generator's reactive power </p>
<!-- Cr_pwr_to_gen   -->
<!-- P<sub>CR</sub>_mw   -->
<!-- Q<sub>CR</sub>_mvar   -->
<p>P<sub>LI</sub>_mw  Active power of induction motor when inrush current is observed
</p>
```

```
<p>Q<sub>LI</sub>_mvar  Reactive power of induction motor when inrush current is observed
</p>
<p>P_mw Active power of the load appeared after the switch is closed </p>
<p>Q_mw Reactive power of the load appeared after the switch is closed </p>
<p>L<sub>P</sub>_mw Total active power of the picked loads after it reaches steady state </p>
<p>L<sub>Q</sub>_mvar Total reactive power of the picked loads after it reaches steady state
</p>
<p>%VD  Voltage Drop</p>
</p>
</div>
<div class = "col-sm-2">
  <img src="{{url_for('static',filename = data2)}}" width = "760" height = "512">
</div>
 </div>
 <div>
 <p><h2>Restoration Procedure</h2>  </p>
 <table id="tbl" class="table table-striped table-bordered" style="width:100%">
      <thead>
         <tr>
           <th></th>
           <th></th>
           <th colspan="2" scope="colgroup">Available power</th>
           <th>Generator to which cranking power is provided</th>
           <th colspan="2" scope="colgroup">Cranking power values</th>
           <th></th>
           <th></th>
           <th colspan="2" scope="colgroup">Power values when inrush current is observed </th>
           <th colspan="2" scope="colgroup">Power appeared after the switch is closed</th>
           <th colspan="2" scope="colgroup">Steady state power</th>
           <th colspan="2" scope="colgroup">% Voltage Drop</th>
         </tr>
         <tr>
            <th>  iteration </th>
            <th>  Gen_turned_on </th>
            <th>  P<sub>AVA</sub>_mw  </th>
            <th>  Q<sub>AVA</sub>_mvar  </th>
            <th>  Cr_pwr_to_gen </th>
            <th>  P<sub>CR</sub>_mw </th>
            <th>  Q<sub>CR</sub>_mvar </th>
            <th>  Load_Name </th>
            <th>  Motor_Group </th>
            <th>  P<sub>LI</sub>_mw  </th>
            <th>  Q<sub>LI</sub>_mvar </th>
            <th>  P_mw  </th>
            <th>  Q_mw  </th>
            <th>  L<sub>P</sub>_mw  </th>
```

```
            <th>  L<sub>Q</sub>_mvar  </th>
            <th>  %VD<sub>Inrush</sub>  </th>
            <th>  %VD<sub>Steady State</sub>  </th>

          </tr>
        </thead>
  <tbody id = "tbody">

    {% for i,row in data.iterrows() %}
    <tr>
     <td> {{  row.iteration|int }}</td>
     <td> {{  row.gen_turned_on }}</td>
     <td> {{  row.eff_gen_cap_p }}</td>
     <td> {{  row.eff_gen_cap_q }}</td>
     <td> {{  row.cranking_power_provided_gen }}</td>
     <td> {{  row.cranking_power_p  }}</td>
     <td> {{  row.cranking_power_q  }}</td>
     <td> {{  row.Load_Name }}</td>
     <td> {{  row.motor_group }}</td>
     <td> {{  row.pli_mw  }}</td>
     <td> {{  row.qli_mvar  }}</td>
     <td> {{  row.p_mw  }}</td>
     <td> {{  row.q_mw  }}</td>
     <td> {{  row.lp_mw }}</td>
     <td> {{  row.lq_mvar }}</td>
     <td> {{  row.Voltage_Drop  }}%</td>
     <td> {{  row.Voltage_Drop_steady }}%</td>
     </tr>
     {% endfor %}

  </tbody>
 </table>
 </div>


</div>
</body>


{% endblock %}
```

APPENDIX E CODE FOR STANDARIZING UI ACROSS THE TOOL

```
{% extends "bootstrap/base.html" %}

{% block title %}File Uploader{% endblock %}

{% block head %}
{{ super() }}

{% endblock %}

{% block navbar %}
<div class="navbar navbar-inverse" role="navigation">
   <div class="container">
      <div class="navbar-header">
         <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
         </button>
         <a class="navbar-brand" href="/">Home</a>
      </div>
      <div class="navbar-collapse collapse">
         <ul class="nav navbar-nav">
            <li><a href="/">About</a></li>
         </ul>
      </div>
   </div>
</div>
{% endblock %}

{% block content %}
   {% block page_content %}{% endblock %}
{% endblock %}

{% block scripts %}
{{ super() }}

{% endblock %}
```