

Optimal Reconfiguration of Smart Grid using Graph Theory

A Project Report

*Submitted in partial fulfilment of requirements
for the award of the degree of*

Bachelor of Technology and Master of Technology

In

Electrical Engineering

By

**GaikwadAshwin Ashok
(EE12B082)**

Under the guidance of

Dr. K. S. Swarup



**Department of Electrical Engineering
Indian Institute of Technology, Madras
May 2017**

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude and indebtedness to my project guide Dr. K. S. Swarup, Assistant Professor in Department of Electrical Engineering, for his most valuable guidance, discussions, suggestions and encouragement, from the conception to the completion of this project. His moral support, unreserved co-operation and generosity, which enabled me to complete the work successfully, will be everlasting in my memory.

I would also like to thank my professors and friends from my undergraduate studies while at Indian Institute of Technology, Madras. The preparation and experience I gained were invaluable.

GaikwadAshwin Ashok

CERTIFICATE

This is to certify that the project report entitled “**Switching Optimization in Smart Power Grid using Graph Theory**”, submitted by **Mr. Gaikwad Ashwin Ashok (EE12B082)**, to the **Indian Institute of Technology, Madras**, in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology and Master of Technology in Electrical Engineering**, and is a bonafide record of work carried by him under my supervision.

Date:
Station:

Dr. K. S. Swarup
Department of Electrical
Engineering
Indian Institute of Technology
Madras, Chennai 600036

ABSTRACT

To enhance system reliability and and improve system robustness “Smart Grid” technology we need to focus on two aspects, automatic distribution system and intentional area partitioning. In this research work we have mainly focused on Graph theory implementation to analyze and solve the problem of system reconfiguration in the real world.

In this work we have proposed an efficient methodology for system reconfiguration which helps to attain an optimal result. We have defined a multiobjective function to which considers minimization of line losses with respect to system reconfiguration. We used IEEE33 bus system as the test case in this research to implement all the methodology and algorithms proposed. Load flow analysis of the test case is also conducted to analyze the line losses of the system which is done by Newton-Raphson method

We developed a generalized algorithm for Load Connectivity Matrix which helps us to analyze which buses to be not considered when fault at certain line occurs. Load connectivity matrix plays an important role in this research which helps us to attain an optimal reconfiguration.

Keywords: Optimal reconfiguration of power system, Graph Theory, Prim’s Algorithm, Minimization of losses, Load connectivity matrix, Load flow Analysis.

Contents

ACKNOWLEDGEMENT	2
CERTIFICATE	3
ABSTRACT	4
LIST OF FIGURES.....	7
LIST OF TABLES	8
CHAPTER 1 Introduction.....	9
1.1.1 " Self-Healing” Power Distribution Network.....	9
1.2 Motivation.....	12
1.3 Contribution of this dissertation.....	14
1.4 Dissertation organization	15
CHAPTER 2 LITERATURE REVIEW ON OPTIMAL RECONFIGURATION	18
CHAPTER 3 FORMULATION OF OPTIMAL RECONFIGURATION IN SMART GRIDS.....	18
3.1.1 Basic concept in Graph Theory.....	19
3.1.2 Determination of switching status.....	22
3.2.1 Prim's algorithm flowchart	26
CHAPTER 4 METHODOLOGY OF RECONFIGURATION USING GRAPH THEORY	34

4.1 Proposed methodology	46
4.2 Flow Chart of proposed methodology	46
CHAPTER 5 IMPLEMENTATION AND CASE STUDY	38
CHAPTER 6 SIMULATIONS AND RESULTS	41
6.1 Results.....	41
6.2 Reconfiguration Results	48
6.3 Line Losses Reduction Results.....	53
CHAPTER 7 CONCLUSIONS	54
7.1 Conclusion	54
7.1 Future work.....	55
Appendix A - Program code for LCM matrix	59
Appendix B - Program code for Load Flow analysis	62
Appendix C - Program code for findind new nodes	68
Appendix D - Program code for generating minimum spanning tree of IEEE 33 bus system	68

LIST OF FIGURES

Figure 3.1 Prim's algorithm Flow Chart.....	26
Figure 3.2 15 bus system.....	31
Figure 3.3 LCM matrix of 15 bus system.....	31
Figure 4.1 Proposed Methodology Flow chart.....	37
Figure 5.1 IEEE 33 bus system 33 sectionalizing and 5 tie switches.....	39
Figure 6.1 Graph of 33 bus sytem.....	42
Figure 6.2 Minimum Spanning tree Graph.....	33
Figure 6.3 Voltage Vs Bus Graph without Prim's algorithm.....	46
Figure 6.4 Voltage Vs Bus Graph with Prim's Algorithm.....	47
Figure 6.5 Voltage Vs Bus comparisons in both the cases.....	47

LIST OF TABLES

Table 3.1 Example of Prim's Algorithm.....	27
Table 5.1 Line data and load data of IEEE 33 Bus System.....	40
Table 6.1 Bus Voltage results after load flow analysis in both the cases.....	44
Table 6.1 LCM matrix of 33 bus system without tie Switches.....	48
Table 6.2 LCM matrix of 33 bus system tie Switches.....	49
Table 6.3 Reconfiguration result.....	51
Table 6.3 Reconfiguration result.....	52
Table 6.3 Reconfiguration result.....	53
Table 6.3 Reconfiguration result.....	54

CHAPTER 1

INTRODUCTION

1.1 Smart Grid and Self-Healing Power Network

Development of smart power network with self-healing features are necessary because of recent blackouts in North America, Europe, and other regions in the world, which will be able to respond to unguarded operating conditions and prevent them from calamitous outage. In future one of the important characteristics of the smart grid is the ability to "self-heal" by anticipating and responding to system disturbances.

As the immune system of the smart grid, "self-healing" is aimed at detecting patterns that are predecessor to interference and reducing the impact of the events to the smallest load area by preventive action for both before and after disturbances.

As Reduction of frequency and magnitude of power outages, minimization the system restoration time is expected by self healing smart grid which is helpful to increase customer satisfaction and the system reliability.

1.1 " Self-Healing" Power Distribution Network

To empower distribution system's "self-healing" capabilities distribution automation (DA) technologies are installed. Real time monitoring, control and automated operation of distribution feeders are the main functions of distribution automation. With the help of well designed system

DA system, a distribution system is able to optimize its operation and improve its reliability in a number of areas,e.g.

- Accelerate fault detection, fault isolation and service restoration
- Increase infrastructure reliability
- Reduce operating and maintenance cost
- Improving power quality by remote voltage and power factor control
- Efficiently reconfiguration distribution system to reduce losses
- Increases customer satisfaction

DA application should be extended to coordinate with new customer applications, such as demand response management, main reasons are growing demands of renewable energy and increasing penetration of distribution generations. Application of DA and substation automation (SA) at local level is done using local information for decision making and control. Hence, the network configuration for restoration purpose is limited to neighbouring feeders. With the help of available high speed communication and IT technologies, more advanced DA system involving global control operations in a large area with multiple feeders and substation is expected.

In future power grids development of a self-healing power network which is capable to forecast and response to disturbances has been visualized to boost system reliability and customer satisfaction. To enable the power network at distribution level a self-healing capability, development of distribution of restoration plays a critical role in the future "Smart Grid". In fault isolation action such distribution restoration process refers to the process of changing open/closed states of tie switches and sectionalizing switches to restore loads that are disconnected from the grid. After the fault is isolated the load in the out of service area should be

restored as quickly as possible. To reduce the impact of outage and increase system reliability we need a restoration plan with minimized switching and an optimized switching sequence.

The distribution system restoration problem is a multi-objective, non-linear, combinatorial problem with numerous constraints, including topology constraints, electrical and operating constraints. A large number of components in the distribution system and complicated generation and load models add to the complexity of a difficult problem.

A distribution network can be represented by a Graph $G(V, E)$ that contains a set of vertices V and a set of edges E . By identifying desired graph topology subjected to various constraints the distribution system restoration problem can be formulated. In this research, a graph theory search algorithm is used to identify a post-outage distribution system topology that will require minimum number of switching operations.

The proposed algorithm and methodology has following features:

1. The proposed methodology helps you to minimize the line losses of a system and helps it to maintain it as a radial system.
 2. This research gives us LCM matrix which gives information about which nodes cannot restore when fault at certain switch or line occurs.
 3. A minimum number of switching actions is achieved given the distribution system topology after remedial actions that may be needed to remove constraint violations are taken into account.
- That is, if a system topology that can restore all out-of-service loads is found, the number of switching operations leading to such a minimum system topology

1.2 Motivation

Large disturbance occurring in power system may make them unstable leading to an uncontrolled system separation. Unlike uncontrolled system separations, an intentional and control system separation can balance generation and load within the island.

Hence, the system frequency within the island can be regulated within allowable limits and the islands formed by control system separation would be less prone to a system collapse. The impact of the disturbance can be contained in a small area if an intentional system separation scheme is conducted appropriately. As a result, cascading failure and large-scale blackouts can be avoided.

To improve the robustness of a power infrastructure and improve the system reliability such a flexible power network configuration by partitioning is "smart grid" technology. To improve the distribution system reliability the automated distribution system restoration is also an important "Smart grid" technology. After a fault occurs the most important task is to restore loads by altering the topological structure of the distribution network while meeting electrical and operational constraints. We can obtain network reconfiguration by changing open/close state of a number of tie-switches and sectionalizing switches in a distribution system. To quickly determine and optimize restoration plans which can significantly reduce the overall customer outage time, thereby enhancing customer satisfaction and quality of the power supplying service is obtained by an efficient distribution system restoration strategy. Recent developments of the Intelligent Electronic Device(IEDs) for protection, control and communication make it possible to realize the automated distribution system restoration in a real time environment.

The distribution system restoration problem is large scale and complex optimization problem because of the large size and nonlinearity of power system. The distribution system can be

represented by graph $G(V, E)$ that contains a set of vertices V , and a set of edges E . Hence, the distribution system restoration problem can be formulated as problem of identifying the desired graph topology subjected to a variety of complex constraints.

To analyse and solve the real-world problem in power system, significant results are achieved to gain and implement efficient graph-theory algorithms.

The main objective of graph theory methods in this research includes:

1. Power networks are naturally modelled as graphs
2. Topology constraints can be handled by graph theoretic algorithms
3. In comparison of local suboptimal solution well designed graph theory algorithms provide better solution
4. Well developed and efficient graph-theory algorithms are powerful tools for solution of combinatorial problems
5. Data structure for graph and network programming technique can facilitate the implementation of graph theory algorithm

1.3 Contribution of this Dissertation

First objective of this research is to conduct and execute efficient, graph theory algorithm to analyze and solve the distribution system configuration. The main contribution of this dissertation is as follows

- This dissertation believes first to propose the method of minimization of line losses of a distribution system. This achieved by using graph theory algorithm and load flow analysis. We first find Minimum spanning tree(MST) of the distribution network which is

converted into graph and do load flow analysis of both the distribution networks normal and after finding MST which results in same voltages of all the busses and MST distribution network gives less line losses than normal

- In this research we have proposed a algorithm to find Load Connectivity matrix(LCM), It helps to identify the nodes which can't be reconnected if fault at certain nodes appears
- One of the main objectives of this dissertation is reconfiguration of a system when fault at certain line of switch occurs; this dissertation gives a methodology to achieve optimal reconfiguration which minimizes line losses as well.

1.4 Dissertation Organization

The next sections of this dissertation are organized as follows: Chapter2 gives a formulation of the concept of distribution power system reconfiguration considering both real and reactive power. In this chapter we proposed basic graph theory concepts which is been used to achieve various objective of this topic and we will demonstrate with example of the entire proposed graph theory algorithm. Basic load flow theory is discussed in this chapter to understand load flow analysis and achieve all the proposed outcomes. We have used Newton-Raphson method to do load flow analysis and it is been discussed briefly in this chapter. This chapter has discussed mainly about the formulation and objective function of achieving optimal reconfiguration of a distribution system.

Chapter 3 describes about the methodology of all the algorithms used to achieve our objective function and discusses about the entire proposed and used algorithm to reach optimal

reconfiguration. In this chapter we will deeply discuss about the algorithm flowcharts which been used in this topic

Chapter 4 gives the simulation and results of all the proposed and used algorithms on IEEE 5 Bus System and IEEE 33 Bus System. In this part all the achieved results will be discussed thoroughly with detailed explanation of each outcome.

In Chapter 5 finally we will conclude all the achieved results and technique used and we will discuss future work briefly.

CHAPTER 2

LITERATURE REVIEW ON OPTIMAL RECONFIGURATION

To resolve the distribution system restoration problem researches have used various methods and techniques with respect to their perspective. T D SUDHAKAR [1] has used graph theory techniques to solve the problem of reconfiguration in distribution system. This paper mainly describes about the implementation of graph theory based methodology for electrical distribution network. The methodology used in paper is Kruskal's algorithm based which considers the overall reliability of a distribution system to find an optimal switching sequence even after varying load conditions. IEEE 16 bus distribution system is used for testing and evaluating proposed methodology. Line and bus data of IEEE 33 Bus system is been taken from reference[2]. Reference [3] discusses about the finding a shortest path between two nodes in a given network. This paper proposes an algorithm to find kth shortest path, between nodes which works by building up electric computational analogy that consists of an ideal diode in series with a DC voltage source. The computational operation of this algorithm increases linearly with order of k and it works efficiently for finding all paths between any two nodes in a given network system.

In order to obtain a network topology with minimum energy losses for distribution system Cavellucci[5] has discussed how the problem can be formulated as generalization of the minimum spanning tree problem. Hiroyuki Mori [4] presents a mathematical method for gaining minimum spanning tree in order to analyse network topology. Method proposed in this paper deals not only with identification of network topological observability but also pseudo measurements of branches or nodes which are necessary observe the recovery of the whole network topology. This proposed methodology is tested on 14, 30, 57, 118, 293, 585 and 1169 bus systems to demonstrate its application to real-sized power system.

Reference [6] demonstrates the use of prim's algorithm used in service restoration part of the fault occfured. The solution propsed in this paper reduces the grid topology to a unidirected weighted graph and perform implementation of Prim's algorithm to solve the problem.

Selection of weights for the graph is one of the key parts for find the correct pattern for loss minimization and servise restoration of a distribution system. In reference [7] the method of selecting weights are been specified with respect to loss minimization. In this paper graph theory based Prim's algorithm is used which helps for formulating a methodology to reconfigure distribution system for loss minimization and service restoration. Selection of weights is one of the important tast which is depended on your objective function.As wehave to do reconfiguration of a distribution system with respect to loss minimization we choose R value of the distribution line as the weights of the graph. The achieved minimum spanning tree helps us observe loss minimization in a distribution system. In this paper author has poposed a very well defined methodology to achieve loss minimization and service restoration.

There are various methods for power flow analysis but method proposed in reference[8] is developed with help of some present methodology and algorithms for radial network analysis like oriental element ordering, power summation method for power flow, statistical representation of load variation, and a recently developed energy summation for computing load flow analysis. These methods, combined with the heuristic rules developed to lead the iterative process, make the energy loss minimization method effective, robust and fast. It presents an altemative to the power minimization methods for operation and planning purposes.

CHAPTER 3

FORMULATION OF OPTIMAL RECONFIGURATION IN SMART GRIDS

When electric system is disrupted following a fault distribution system restoration is intended to immediate restore as much load as possible in that area. After all the reconfiguration and loads are restored, the final network topology should maintain radial structure. In order to reduce outage period the restoration process should be completely efficient which lead to increase customer satisfaction. In order to take remote control actions or dispatch maintenance crews to implement the restoration plan an efficient restoration plan should be established quickly to guide operators in distribution operating centres. Besides, the established should not provide only a workable system topology but to order reach final optimal configuration it should also have a sequence of switching operations.

There are two types of typical distribution system structures which are adopted by electric utilities i.e. Radial and loop system.

Radial system: A radial system is identified by have one connecting path to the power source and each customer. It is most widely used by utilities because radial systems are one of the cheapest systems. Power would be interrupted when a power failure, short-circuit or a downed power line occurs so it should be fixed before power can be restored.

Loop system: A loop system is as its name implies, it has at-least two paths it loops through the service area and returns to the original point. Loops generally have alternate power source and that makes it more reliable than a radial system. But as a fact loop system is more costly than radial system. Additional cost is consisting of more need for conductors, higher conductor

capacity, switching and protection of devices. The design of a protection scheme in the loop system requires more coordination in both the direction.

Objective function is as followed:

1. Minimum Losses
2. Choosing the most optimal path when fault at certain line or switch occurs.

Where P_l and Q_l are the real and reactive line losses of the system

The objective of the algorithm includes minimization of line losses and to reconfigure the system by choosing the most optimal path when fault at certain switch or line occurs by maintaining it as radial network structure.

3.1 Graph Theory Foundation

3.1.1 Basic concept in Graph Theory

Definition of some basic concepts of graph theory is defined as follows:

Definition 3.1: Vertex

Vertex is a junction where multiple lines meet. It is also known as a node in a graph. A vertex is denoted by a point and an alphabet.

Definition 3.2: Edge

A line that connects two vertices is called Edge in mathematical form. We can generate many edges from a single vertex. To form an Edge we should have starting vertex and an ending vertex.

Definition 3.3: Weight of Edge

Each edge of a graph has an associated numerical value, known as weight. Weighted graphs may be either directed or undirected. The weight of an edge is often referred to as the "cost" of the edge.

Definition 3.4: Path

A sequence of vertices such that from each of its vertices there is an edge connecting to the consecutive vertex in a sequence is called **Path**

Definition 3.5: Cycle

A cycle is bi-directional path such that both the starting and end points are identical.

Definition 3.6: connectivity

In an undirected graph G , two vertices are said connected when there is a path contained in G which connects these two vertices. If every pair of distinct vertices in the graph is connected through some paths then a graph G is said to be connected.

Definition 3.7: Tree and spanning Tree

A graph that contains no cycles is called a **tree** T . A spanning tree S of a tree connected, unidirectional graph G is graph which contains the entire vertex and some or all the edges which are present in the graph G . The graph in which edges have no orientation is called an undirected graph G . An edge of G that is absent in a spanning tree S of this graph is called a chord.

Let S be a spanning tree with n vertices then these following statements are valid:

- S contains no cycles and it is connected
- S has $n-1$ edges and n vertices
- S can be disconnected if any edge is removed from S
- Exactly one path connects any two vertex of S
- Addition of any nodes and create a cycle in S

Characteristics of spanning tree:

Any spanning tree of any connected graph G of vertices n and edges e has $n-1$ tree branches and $e-n+1$ chords.

Minimum spanning tree (MST):

A minimum spanning tree is subset of a connected, edge-weighted undirected graph which connects all the vertices together, without any cycles and with the minimum possible total edge weight.

There are various algorithms developed to find the Minimum spanning tree(MST). We have prim's Algorithm to find MST of AN Graph G.

3.1.2 Determination of switching status

Closing or Opening of any tie or sectionalizing switches are performed to have various network configuration without any closed loops or leaving out any branches unconnected. A network configuration is considered infeasible switching combination for network configuration when a closed loop or leaving one more braches unconnected. Connectivity from the source to all the nodes is checked in order to avoid any infeasible switching combination. A configuration is characterized as feasible one if a valid path exists or it is characterized as infeasible one. It is important to note that if we are selecting any tie or sectionalizing switch for opening or closing for reconfiguration then the immediate neighbouring two switches are also considered for reconfiguration. A closed loop will be formed if a tie switch is closed. To maintain it as radial

structure, one of the neighbouring sectionalizing switches should be open. To obtain this configuration here prim's algorithm explained

3.2 Prim's algorithm

There are two serious obstacles arise if an exhaustive-search approach is used to construct a minimum Spanning tree

- The number of spanning trees grows exponentially with graph size
- Generating all spanning trees for a given graph is not easy

In this section we have discussed Prim's Algorithm deeply with an example which will help us to understand it better.

Prim's algorithm constructs a minimum spanning tree through a sequence of expanding sub-trees. The initial sub-tree in such a sequence consists of a single vertex selected arbitrarily from the set V of the graph's vertices. In the following iterations the current tree expands in the greedy manner by simply attaching to it the nearest vertex not in that tree. (By the nearest vertex, a vertex not in the tree connected to a vertex in the tree by an edge of the smallest weight.) The algorithm stops after all the graph's vertices have been included in the tree being constructed. Since the algorithm expands a tree by exactly one vertex on each of its iterations, the total number of such iterations is $n - 1$, where n is the number of vertices in the graph. For the tree expansion the algorithm used generates the trees which are obtained as the set of edges used for the tree expansion

Pseudo code for Prim's Algorithm:

```
// Prim's Algorithm for constructing an MST
```

```
// Input: A weighted connected graph  $G = (V, E)$  //  $V$  is node ,  $E$  is edge
```

```
//Output:  $E_t$ , the set of lines composing the MST of  $G$ 
```

//the set of tree nodes can be initialized with any node

$ET \leftarrow O$

For $i=1$ to $(v-1)$ do

Find a minimum weighted tree line $e^* = (v^*, u^*)$ among all the lines (v,u)

Such that v is in V_t and u is in $v-V_t$

$V_t \leftarrow V_t \cup \{u^*\}$

$E_t \leftarrow E_t \cup \{v^*\}$

Return ET

The nature of Prim's algorithm makes it necessary to provide each vertex not in the current tree with the information about the shortest edge connecting the vertex to a tree vertex. The information of the vertex are provided by labelling it as the name of the nearest treevertex and the length (the weight) of the corresponding edge. We can give the infinity label to the vertices that are not adjacent to any of the tree vertices point at them saying that they are at infinite distance to the tree vertices and we will label null for the name of the nearest vertex. With such notations, we can find the next vertex to be added to our current tree $T=(V_t,E_t)$ which helps us to make our task simple to find another vertex with smallest distance which is labelled in the set $v-V_t$.

Two operations has to be performed when a vertex u^* is identified which is to be added to the tree

- To set a tree vertices V_t move u^* from the set $v-V_t$
- For all the remaining vertex u in $V - V_t$ that is connected to u^* by a shorter edge than the u 's current distance label, update its labels by u^* and the weight of the edge between u^* and u , respectively

Prim's algorithm always provides a minimum spanning tree, which is proved by induction, that each of the sub-tree T_i , $i=0, 1, 2, \dots, n-1$ generated by Prim's algorithm is a part of some minimum spanning tree. This implies that the last tree in the sequence, T_{n-1} , is a minimum spanning tree because all the n vertices are present. The basis of the induction is insignificant because T_0 consists of a single vertex and hence it must be a part of any minimum spanning tree. For the inductive step, let us assume that T_{i-1} is part of some minimum spanning tree T . It has to be proved that T_i , generated from T_{i-1} by Prim's algorithm, is also a part of a minimum spanning tree. This is proved through contradiction by assuming that no minimum spanning tree of the graph can contain T_i . Let $e_i = (v, u)$ be the minimum weight edge from a vertex in T_{i-1} to a vertex not in T_{i-1} used by Prim's algorithm to expand T_{i-1} to T_i . By our assumption, e_i cannot belong to the minimum spanning tree T . Therefore, if we add e_i to T , a cycle must be formed. In addition to edge $e_i = (v, u)$, this cycle must contain another edge (v', u') connecting a vertex $v' \in T_{i-1}$ to a vertex u' that is not in T_{i-1} . (It is possible that v' coincides with v or u' coincides with u but not both.) If we now delete the edge (v', u') from this cycle, we obtain another spanning tree of the entire graph whose weight is less than or equal to the weight of T since the weight of e_i is less than or equal to the weight of (v', u') . Hence, this spanning tree is a minimum spanning tree, which contradicts the assumption that no minimum spanning tree contains T_i . Here table 2.1 explains the step by step procedure of obtaining the minimum spanning tree from the starting vertex 'a' for the above sample network.

3.2.1 Prim's algorithm flowchart

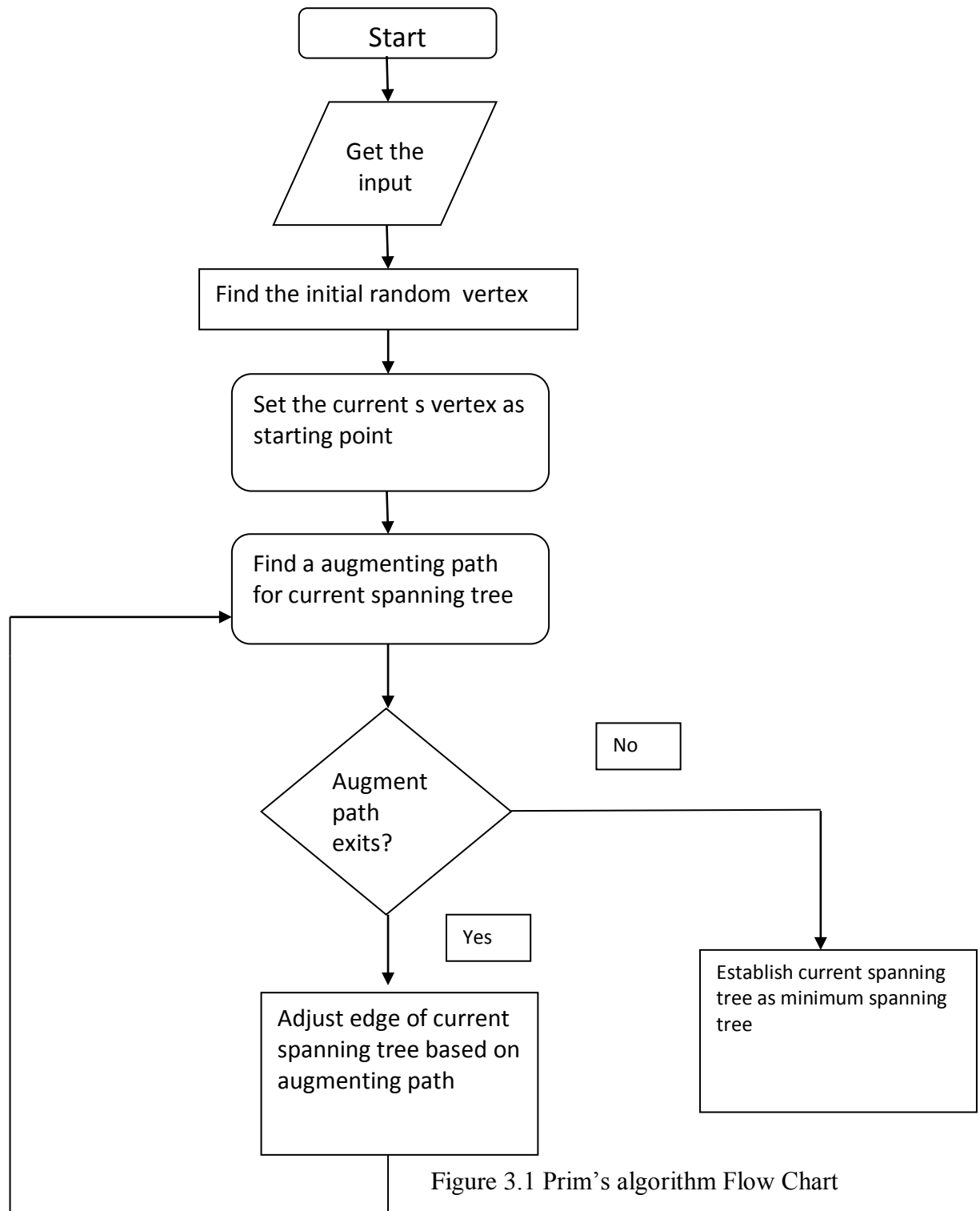
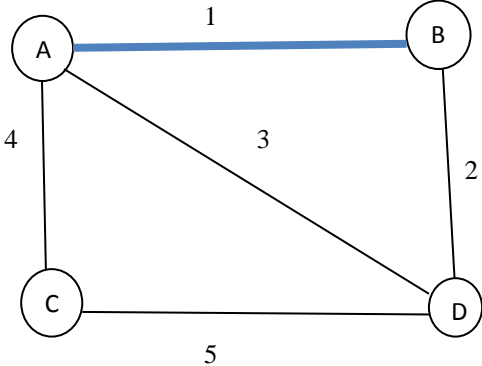
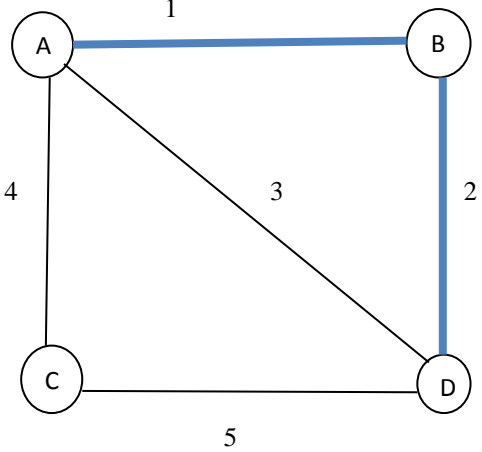
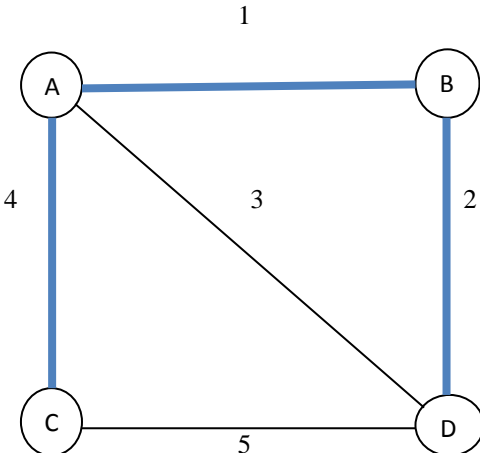


Figure 3.1 Prim's algorithm Flow Chart

Table 3.1 Example of Prim's algorithm

Tree vertices	Remaining vertices	Illustration
B(-,-)	A(B,1) D(B,2) C(-,infinity)	
B(A,1)	D(B,2) C(-,infinity)	
A(B,1)	C(A,4)	

3.3 Load Flow Analysis

In an interconnected system numerical analysis of the flow of electric power is called power-flow study or load-flow study. Power flow study focuses on various aspects of AC power parameters like voltage, voltage angle, real power and reactive power. It analyses the power system in normal steady state operation.

For planning future expansion of power system as well as in determining the best operation of existing system power-flow or load-flow studies are important. The key information obtained from the power-flow study is the magnitude and phase angle of each bus, and the real and reactive power flowing in each line.

The present power systems are too complex to allow for hand solution of the power flow. Special purpose network analyzers were built between 1929 and the early 1960 to provide laboratory-scale physical models of power system. Analog methods were replaced by Large-scale digital computers with numerical solutions.

Computer Programs perform various related calculation other than power-flow study like short-circuit fault analysis, stability studies (Transient and steady-state), unit commitment and economic dispatch. In particular, some programs use linear programming to find optimal power flow, the condition which gives the lowest cost per kilowatt hour delivered:

3.3.1 Newton Raphson method

To solve nonlinear equations Newton Raphson method is efficient algorithm. The procedure of solving nonlinear equations is transformed into the procedure of repeatedly solving linear equations. This sequential linearization process is the core of the Newton-raphsonmethod.

Load flow algorithm

The Newton-Raphson procedure is as follows

Step-1: Choose the initial values of the voltage magnitudes $|V|^{(0)}$ of all n_p load buses and $n - 1$ angles $\delta^{(0)}$ of the voltages of all the buses except the slack bus.

Step-2: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to calculate a total $n - 1$ number of injected real power $P_{calc}^{(0)}$ and equal number of real power mismatch $\Delta P^{(0)}$.

Step-3: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to calculate a total n_p number of injected reactive power $Q_{calc}^{(0)}$ and equal number of reactive power mismatch $\Delta Q^{(0)}$.

Step-3: Use the estimated $|V|^{(0)}$ and $\delta^{(0)}$ to formulate the Jacobian matrix $J^{(0)}$.

Step-4: Solve (4.30) for $\delta^{(0)}$ and $\Delta |V|^{(0)} \div |V|^{(0)}$.

Step-5 :Obtain the updates from

$$\delta^{(1)} = \delta^{(0)} + \Delta \delta^{(0)}$$

$$|V|^{(1)} = |V|^{(0)} \left[1 + \frac{\Delta |V|^{(0)}}{|V|^{(0)}} \right]$$

Step-6: Check if all the mismatches are below a small number. Terminate the process if yes.

Otherwise go back to step-1 to start the next iteration with the updates given by (1) and (2)

3.4 Load connectivity matrix

Load connectivity matrix is a matrix which gives us information about which branches can be restored or which branches can't restore when fault at certain line or switch occurs in a Distribution system. Load connectivity matrix is very important in system reconfiguration. Load connective matrix helps us to achieve a optimal reconfiguration which is one of the biggest necessity in today's Distribution system. Dimensions of Load connectivity matrix are $m \times n$ where m rows represent opening of that branch and n column represents the connectivity of busses in the given system.

Example: Take this 15 bus system with two tie switches. we can see that The tie switch T1 is connected between 2 and 7 and another tie switch T2 is connected between 10 and 14. Numbers are representing bus number and alphabets are representing branches/feeder.

In this example taken we have considered bus 1 and 2 as substation so fault at any one of them is not taken

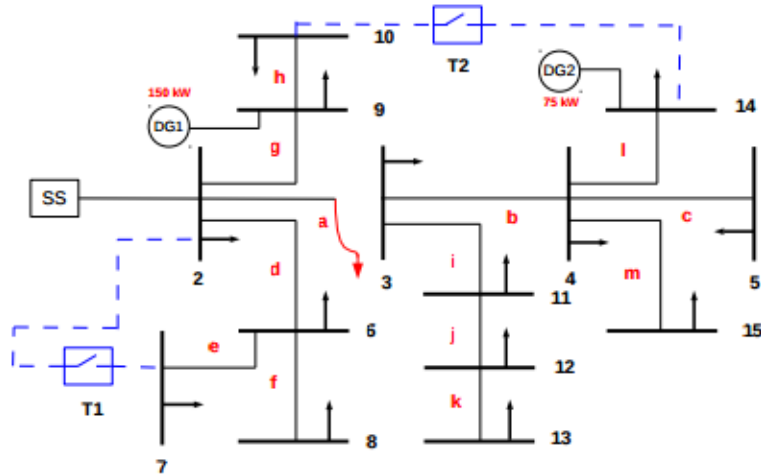


Figure 3.2 15 bus system

Now let's try to write Load connectivity matrix for this 15 bus system

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
c	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
d	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
e	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
f	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
g	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
h	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
i	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1
j	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
k	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
l	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
m	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

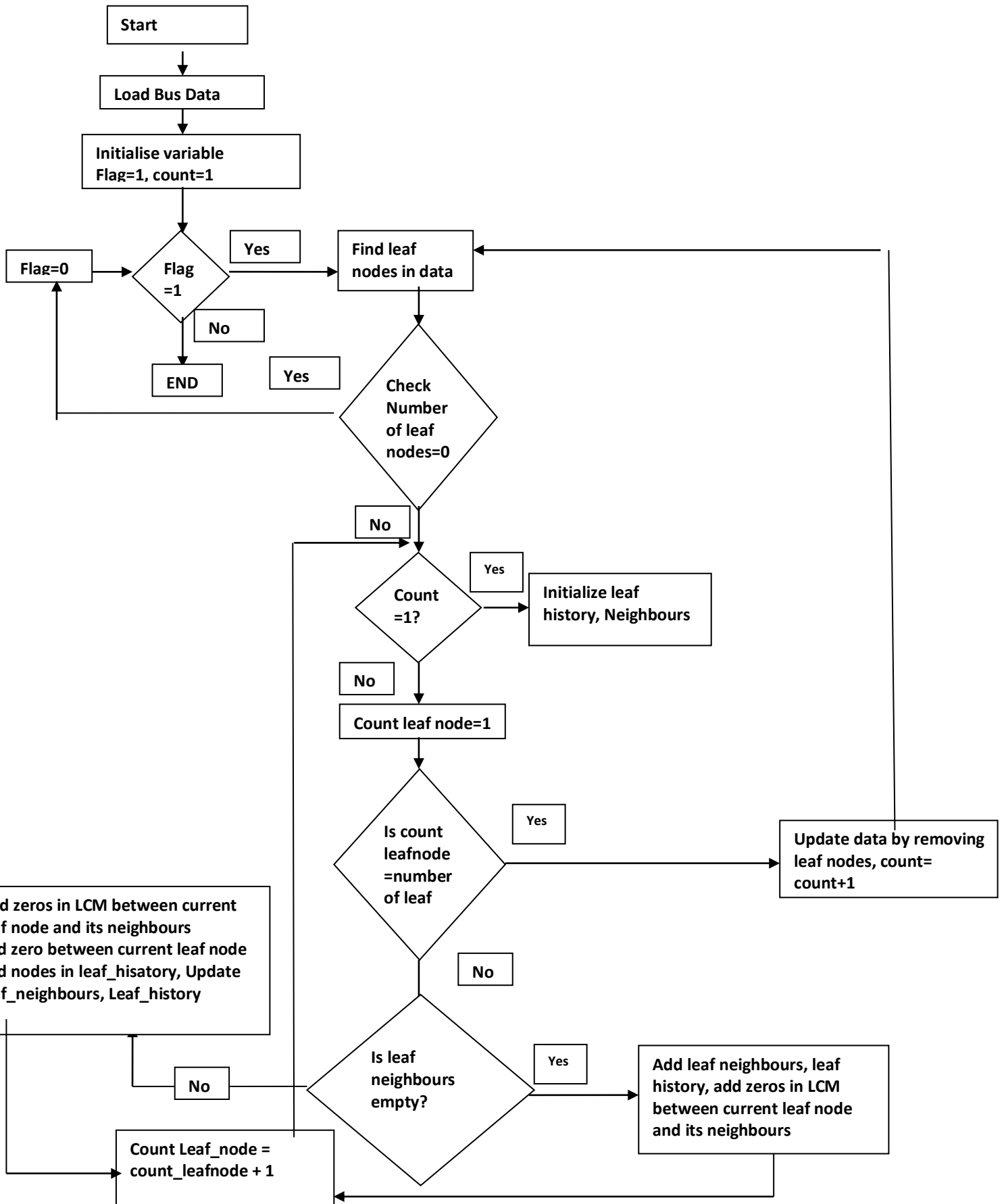
Figure 2.3 LCM matrix of 15 bus sytem

1. Opening of branch 'a' is represented by the first row in LCM matrix . We can observe that all the entries in the row are 1 which means when fault occurs at branch 'a' still all the buses are recovered because Tie switch T2 is closed

2. We can observe a 0 in 5th column of 3rd row of LCM matrix because when the branch 'c' gets disconnected due to any disturbance or fault or by any other reason bus 5 get islanded and load in that bus cannot be recovered with closing of any existing switches. The amount of load which is not served is L5

3. When fault occurs at branch 'i' , buses {11,12,13} cannot be recovered and load in those buses cannot be served with all the existing switches . All the buses which cannot be served back are represented by 0's in that row.

2.4.1 Flow chart of Load connectivity matrix algorithm



CHAPTER 4

METHODOLOGY OF RECONFIGURATION USING GRAPH THEORY

In the previous chapter we achieved formulation to our objective with respect to the all constrains. As we discussed about graph theory in the previous chapter graph theory is one of the key tool in this dissertation. We will use graph theory for reducing the line losses which helpful is any distribution system. Reduction in line losses helps us to increase the system reliability and overall cost. Further we are using graph theory to obtain the optimized reconfiguration of a distribution system. In this modern world distribution system we have experienced my blackouts and load shedding. Graph theory helps us to achieve an optimal way to attain it. In this dissertation we are using Prim's algorithm to find minimum spanning tree which helps us for minimising line losses and analysis system restoration when fault at certain line or switch occurs.

4.1 Proposed methodology

First the network which we are using is modelled as an edge-weighted Graph. Each bus in this network system is a vertex of the graph and each distribution line is an edge of the graph. The weights of the graphs are the resistance value of the line connecting two busses.

After modelling the network in an edge-weighted graph we will use Prim's algorithm for finding the minimum spanning tree of the graph. Minimum spanning treewill help us to attain our first objective of minimizing the line losses. When we attain our minimum spanning tree we will assign data to only those buses which are present in minimum spanning tree when compared to

original data we have. After assigning data to the lines present in minimum spanning data we do our load flow analysis using Newton-Raphson method. In load flow analysis we will observe the voltages vs Bus in both cases i. e. Normal case without finding the minimum spanning tree and with the minimum spanning tree. We will observe the line losses as well in both the cases. According to our desired result we should observe reduction in line losses. If we observe the line losses reduction we will consider it as the best case for attaining it. So obtained minimum spanning tree which helped to attain losses reduction is considered as the Base case for network reconfiguration.

For network system reconfiguration our minimum spanning tree is very important but before going restoration we have to analysis which buses can't be restored when there is disturbance or outage occurring that certain distribution line. For analysis of this we have developed Load connectivity matrix which will give us direct information of which busses to leave out of tree when fault or outage at certain line occurs.

After forming the Load connectivity matrix it becomes easy for us to attain distribution system restoration. We will a follow a procedure to attain system restoration which help of our base case graph and Load connectivity matrix.

We will generate fault on each line sequentially and for each case we will find the minimum spanning tree. After attaining minimum spanning tree we will note down the sum of all the weights of the graph and compare it with our base case and try to observe new connection in this

minimum spanning tree. These new connections which we will get after comparing should be penalized so that our graph should try to find more optimized path for reconfiguration. After penalizing we will find minimum spanning tree of that graph and note down the overall weight of it. Now for each case we will have two weights which propose an optimized path when fault at certain line or switch occurs. We will choose the path for reconfiguration with less overall weight and it will be the best optimized path for reconfiguration when at certain line of switch occurs.

4.2 Flow chart of proposed methodology

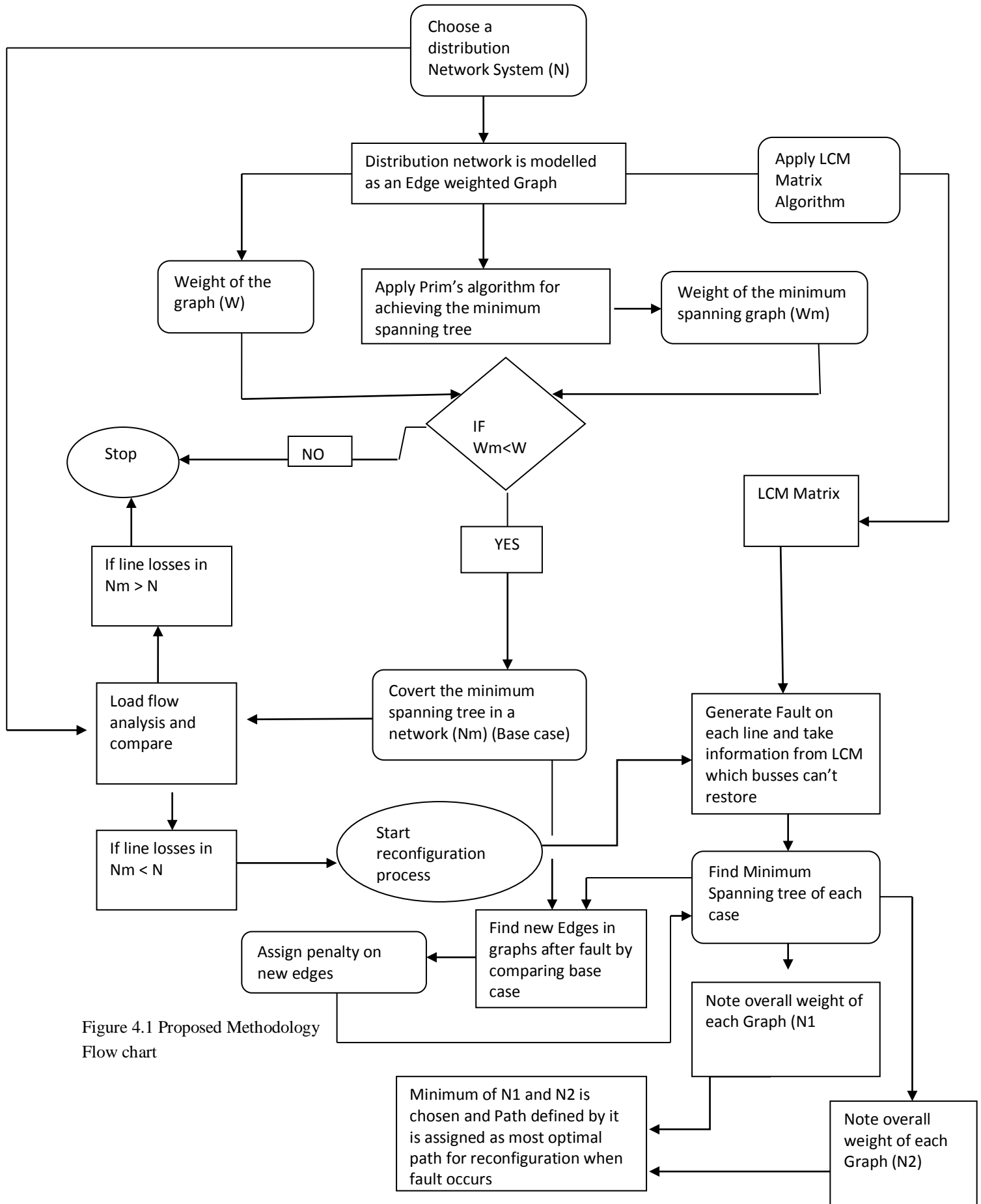


Figure 4.1 Proposed Methodology
Flow chart

CHAPTER 5

IMPLEMENTATION AND CASE STUDY

In this dissertation we have proposed a new methodology for reconfiguration of a distribution system when fault at certain line occurs with respect to loss minimization. We have implemented this method on IEEE 33 bus system which helps to achieve all our objective function and gives a deep record of most optimal path of a distribution system when fault at certain line occurs.

These are the following steps we have taken to apply our methodology:

Step1: Take the test case and convert it into graph and do load flow analysis

Step2: Apply prim's algorithm to get the minimum spanning tree of the graph and fix this as base case

Step3 : Load flow analysis of the graph gained in step2

Step4: Compare bus voltages and line losses of the system from step1 and step3 if line losses reduces from step2 to step1 then proceed

Step 5: Produce fault at each line and find minimum spanning tree of it

Step 6: Compare base case and minimum spanning tree attained in step 5 mark all the common nodes and apply penalty on that

Step 7: After applying penalty find minimum spanning tree again

So now we have two paths or two set of reconfiguration strategy available from step7 and step5.

We choose the best one with minimum weights when compared to both.

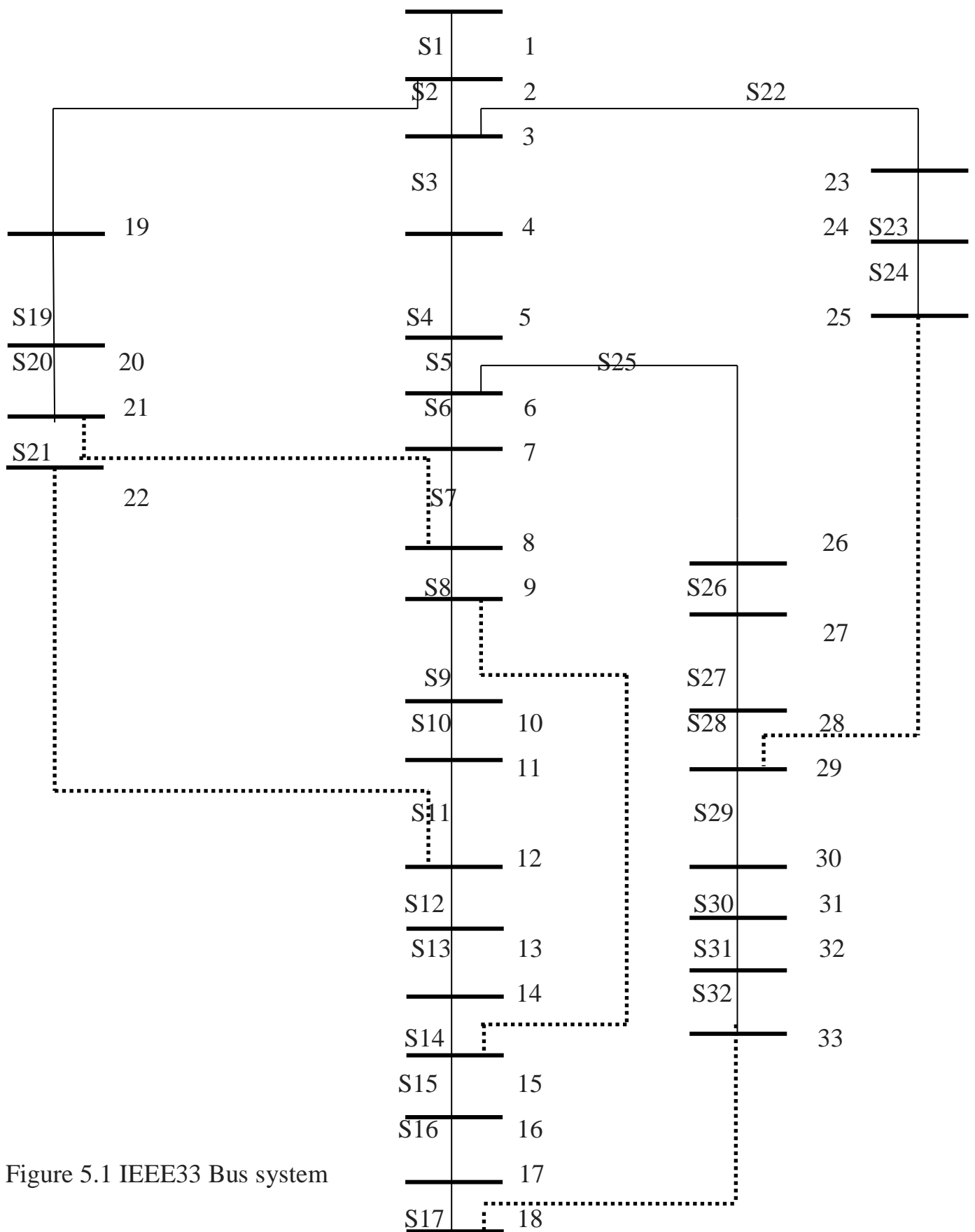


Figure 5.1 IEEE33 Bus system

Table 5.1 IEEE 33 bus system data

Branch No	Sending End	Receiving end	R(Ohms)	X(Ohms)	PL(KW)	QL(KVA)
1	0	1	0.0922	0.0477	100	60
2	1	2	0.4930	0.2511	90	40
3	2	3	0.3660	0.1864	120	80
4	3	4	0.3811	0.1941	60	30
5	4	5	0.8190	0.7070	60	20
6	5	6	0.1872	0.6188	200	100
7	6	7	1.7114	1.2351	200	100
8	7	8	1.0300	0.7400	60	20
9	8	9	1.0400	0.7400	60	20
10	9	10	0.1966	0.0650	45	30
11	10	11	0.3744	0.1238	60	35
12	11	12	1.4680	1.1550	60	35
13	12	13	0.5416	0.7129	120	80
14	13	14	0.5910	0.5260	60	10
15	14	15	0.7463	0.5450	60	20
16	15	16	1.2890	1.7210	60	20
17	16	17	0.7320	0.5740	90	40
18	1	18	0.1640	0.1565	90	40
19	18	19	1.5042	1.3554	90	40
20	19	20	0.4095	0.4784	90	40
21	20	21	0.7089	0.9373	90	40
22	2	22	0.4512	0.3083	90	50
23	22	23	0.8980	0.7091	420	200
24	23	24	0.8960	0.7011	420	200
25	5	25	0.2030	0.1034	60	25
26	25	26	0.2842	0.1447	60	25
27	26	27	1.0590	0.9337	60	20
28	27	28	0.8042	0.7006	120	70
29	28	29	0.5075	0.2585	200	600
30	29	30	0.9744	0.9630	150	70
31	30	31	0.3105	0.3619	210	100
32	31	32	0.3410	0.5302	60	40

CHAPTER 6

Simulation and results

6.1 Results

After proposing the methodology in the previous chapter for line losses minimization and system reconfiguration we will implement it on certain distribution bus system to have numerical and graphical proof. IEEE 33 bus system which has 33 sectionalizing and 5 tie switches.

First we will proceed with finding the graph of the bus system and then the minimum spanning tree of each of them. So in this section we will first find graph of the system followed by the minimum spanning tree of the system. next we will find the bus voltages of the system in both the cases and try to compare them . Figure 6.1 shows the graph of the IEEE 33 bus system which has all 32 sectionalizing switches and 5 tie switches. This graph is achieved in using matlab and the overall weight of the graph is 21.576. Figure 6.2 shows the minimum spanning tree of the IEEE 33 bus system which is obtained using matlab and the overall weights obtained in 18.08.

In this chapter we will further discuss about the bus voltage outputs in both the cases i. e. With prim's algorithm and normal case. The voltage results are been shown in table 6.1. Figure 6.3 shows the voltage in normal case figure 6.4 shows the minimum spanning tree case and figure 6.5 shows the comparison of both the cases. Table 6.4 gives the optimal reconfiguration results which helps us to understand which path to choose when fault at certain line occurs.

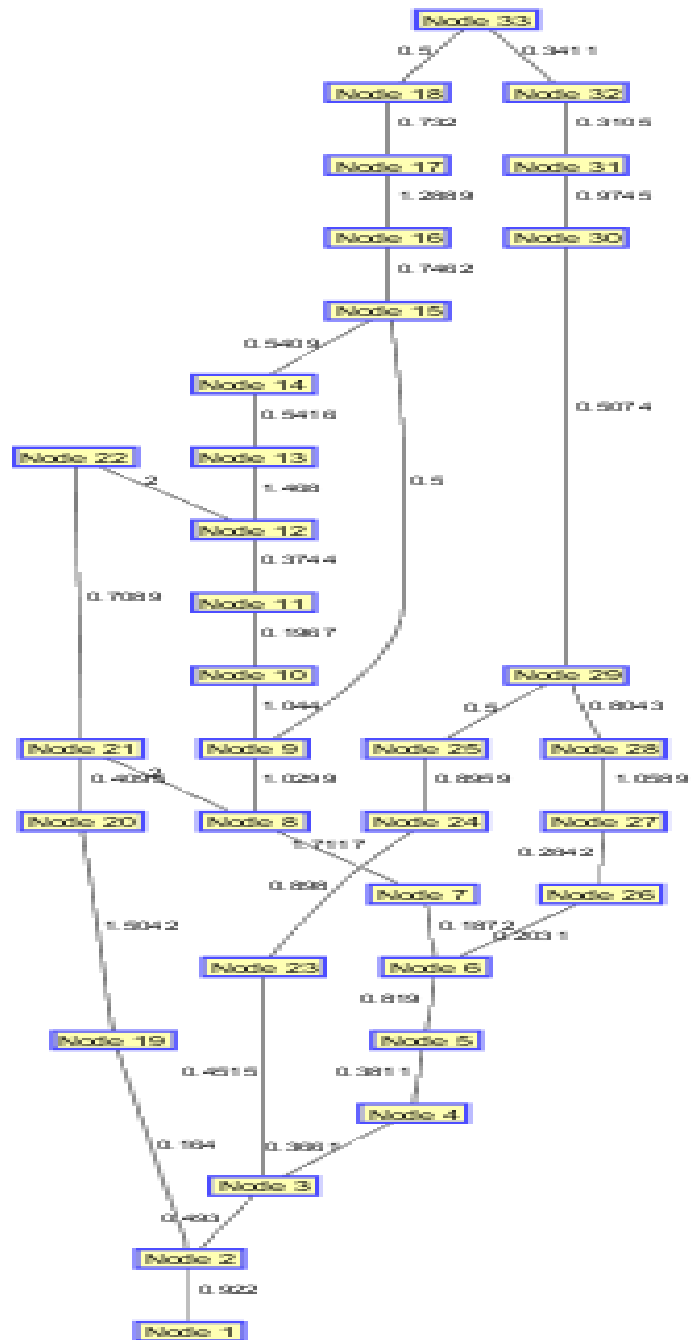


Figure 6.1 Graph of 33 Bus system with a weight of 21.576

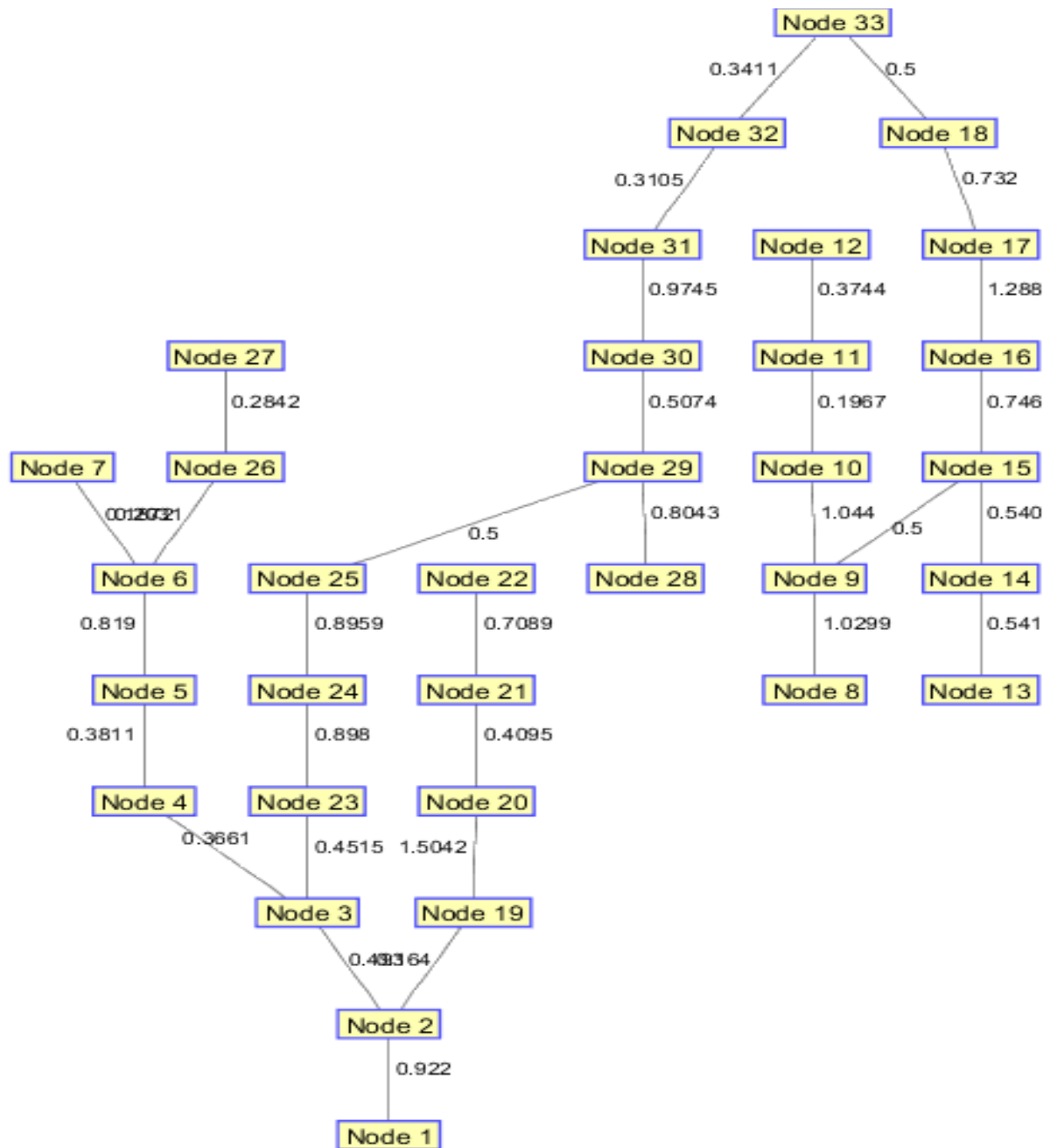


Figure 6.2 Minimum Spanning tree Graph with weight of 18.08

Table 6.1 Bus Voltage results after load flow analysis in both the cases

Bus Number	Voltage in normal case	Voltage after prim's Algorithm
1	1	1
2	0.996	0.996
3	0.977	0.980
4	0.9668	0.997
5	0.9568	1
6	0.9318	0.988
7	0.9271	0.990
8	0.9205	0.988
9	0.9119	0.996
10	0.9040	1
11	0.9028	0.98
12	0.9008	0.996
13	0.8894	1
14	0.8874	0.992
15	0.8856	0.996
16	0.8856	0.98
17	0.8828	0.992
18	0.8820	0.998
19	0.9953	1
20	0.9906	0.996
21	0.9896	0.998
22	0.988	0.992
23	0.9722	0.98
24	0.9632	0.996

25	0.9588	1
26	0.9292	0.996
27	0.9257	0.996
28	0.9101	0.996
29	0.8989	0.998
30	0.8941	0.98
31	0.8840	0.998
32	0.8871	0.987

Voltage Vs Bus Graph

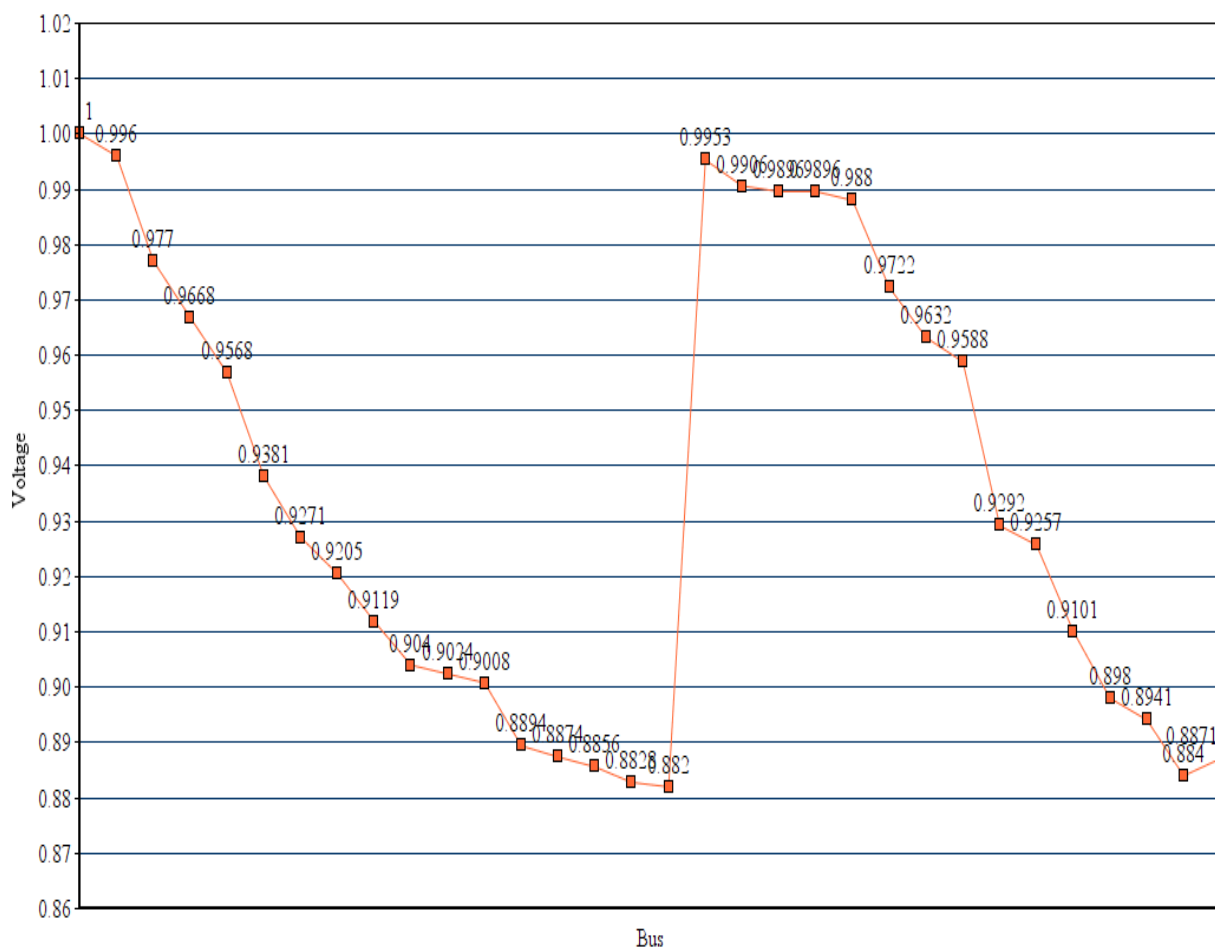


Figure 6.3 Voltage Vs Bus Graph without Prim's algorithm

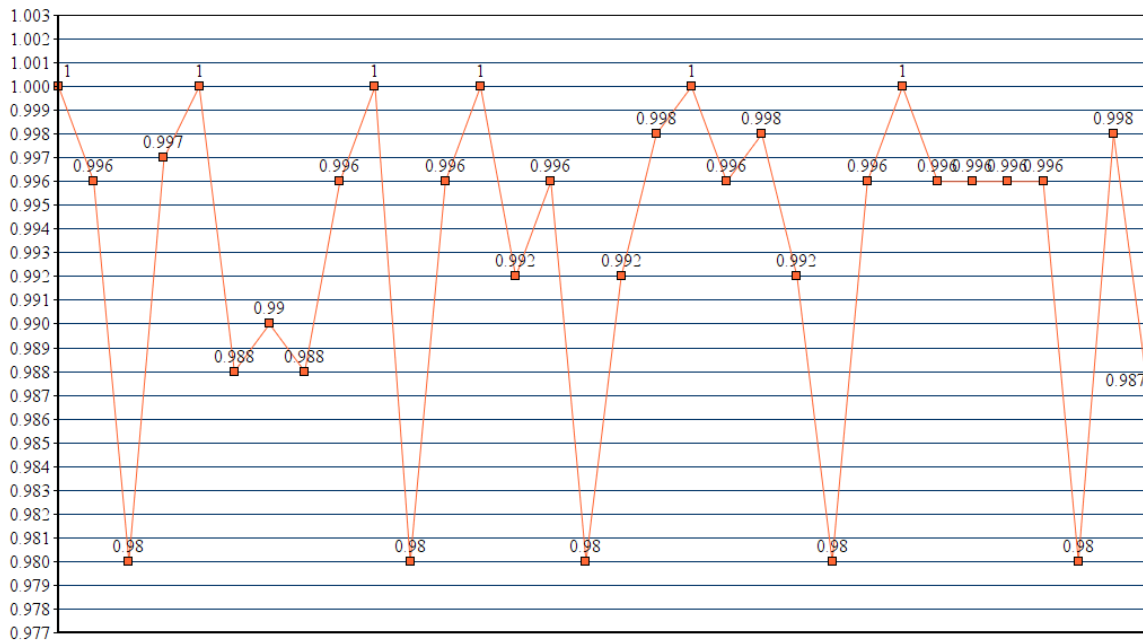


Figure 6.4 VoltageVs Bus Graph with Prim's Algorithm

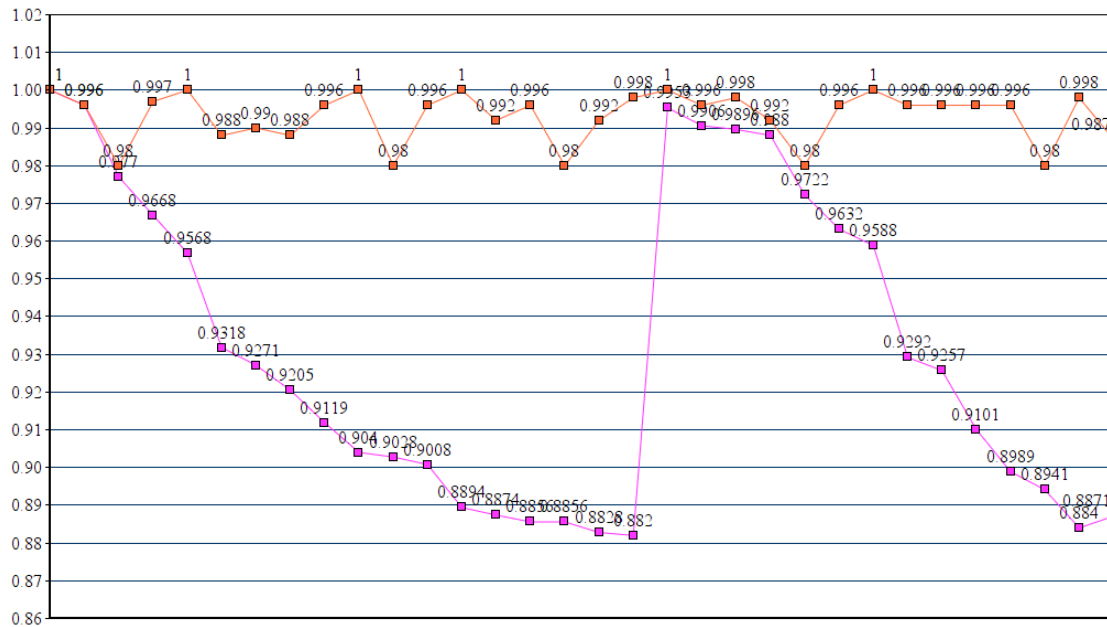


Figure 6.5 Voltage Vs Bus comparisons in both the cases

6.2 Reconfiguration Results

Load Connectivity Matrix

Table 6.2 LCM Matrix for 33 Bus systems without any tie switches:

S1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S3	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S4	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S5	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S6	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S7	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S8	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S9	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S10	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S11	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S12	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S13	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S15	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S17	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S18	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
S19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
S20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
S21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
S22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
S23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
S24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
S25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
S26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
S27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
S28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
S29	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
S30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
S31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
S32	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 6.3 LCM matrix of 33 Bus systems with all the tie switches

[illegible]

After applying the proposed methodology

Results of most optimal path when fault at certain line occurs

Fault occurring at	Weight of the edge	MST Weight	New nodes in MST	MST after penalty
(2,3)	0.4930	20.407	(21,8)	19.86
(3,4)	0.3661	19.5928	(22,21)	20.8839
(4,5)	0.381	19.57782	(22,21) (28,27)	21.5217
(5,6)	0.8190	19.1399	(28,27)	18.9884
(6,7)	0.1872	19.833	(21,22) (8,7)	23.715
(7,8)	1.7117	18.9	(22,21)	20.1911
(8,9)	1.0299	19.5818	(21,22) (8,7)	21.1612

Fault occurring at	Weight of the edge	MST Weight	New nodes in MST	MST after penalty
(9,10)	1.0440	19.316	(13,12)	19.856
(10,11)	0.1967	20.7033	(22,21) (22,12)	23.1633
(11,12)	0.3744	19.9856	(22,21) (13,21)	22.52
(12,13)	1.4600	20.0184	(22,21)	20.1891
(13,14)	0.5416	21.1095	(21,22) (13,12)	21.8118
(14,15)	0.5909	19.7675	(22,21)	21.058
(15,16)	0.7462	19.8655	(21,22)	21.156

Fault occurring at	Weight of the edge	MST Weight	New nodes in MST	MST after penalty
(16,17)	1.2889	19.3217	(21,22) (8,7)	20.9022
(17,18)	0.7320	19.8797	(22,21) (8,7)	21.4573
(2,19)	0.1649	17.693	(22,21) (8,21)	22.736
(19,20)	1.5642	19.336	(22,21) (21,8)	21.3294
(20,21)	0.4095	20.4914	(21,22) (21,12)	21.4474
(21,22)	0.7089	18.5597	(22,12)	21.3011
(3,23)	0.4512	19.50862	(21,22)	20.7997

Fault occurring at	Weight of the edge	MST Weight	New nodes in MST	MST after penalty
(30,31)	0.3105	19.8021	(21,22)	21.5933
(31,32)	0.3105	19.8021	(22,21)	21.5933
(32,33)	0.3411	20.2675	(22,21)	21.5627

Table 6.4 Reconfiguration result

6.3 Line Losses Reduction Results:

Without Prim's algorithm

PI = 0.36 MW QI = 0.241

With Prim's Algorithm

PI = 0.27MW QI = 0.194

CHAPTER 7.

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The major results that we have achieved in this research are implementation of graph theory for loss minimization and distribution system configuration. This research is motivated by recent development of the “smart grid” vision to modernize the grid

With the vision to modernize the infrastructure of grid and support “Self-healing” feature in distribution network this work is motivated by the recent development of the “Smart Grid”.

The proposed reconfiguration algorithm minimizes both the real and reactive line losses.

Load flow analysis Simulation on IEEE 33 Bus system verify the voltage profile Vs each bus in both the case without Prim’s algorithm and with with prim’s algorithm. We achieved minimization of losses because of the voltage profile we achieved after application of prim’s algorithm.

We proposed Load Connectivity Matrix which gives us information about the connectivity of a line connected between buses in time of reconfiguration. In this research we developed an algorithm to achieve this matrix and it is been presented in a form of a flow chart for better understanding.

After proposing the methodology for achieving the distribution system configuration we have achieved to get most optimal path in IEEE 33 bus system when fault at certain line occurs. We are able to achieve two different reconfiguration patterns when fault at certain line occurs which helps us to more reconfiguration option which we can use according to our necessity.

7.2 FUTURE WORK

Self-healing smart grid is one of the most desirable technologies with the increasing needs of renewable energy. Such a smart grid should definitely have the capabilities of wide-area awareness, grid monitoring on real-time basis and early warning that recognize accessible operating conditions in timely manner. In order to take preventive actions and corrective control actions smart grid allows the system to operate automatically in accessible conditions. PMU's and high bandwidth communication development enhance the technical feasibility of a smart grid.

Even though we achieved some significant progress in this dissertation, the following important topics (issues) remain to be discussed in future:

1. The proposed spanning search algorithm is intended to find the distribution network topology with a radial structure. Improvements should be made for the proposed algorithm to apply it on a weakly meshed distribution network.
2. The proposed load connectivity matrix algorithm is useful to see connectivity of Bus systems with fewer buses. If we use this complex algorithm on bigger networks then it

becomes difficult to access the information of certain line at certain fault. So this algorithm should be developed more.

3. In this proposed methodology of system reconfiguration considered load models are constant current, power and impedance loads. It is important to study after considering load variations in the distribution reconfiguration planning. It is necessary to build proper load model that could represent the load variations.

References

1. T. D. Sudhakar and K. N. Srinivas, "Power system restoration based on Kruskal's algorithm," *2011 1st International Conference on Electrical Energy Systems*, Newport Beach, CA, 2011, pp. 281-287.
2. P. R. Babu, C. P. Rakesh, M. N. Kumar, G. Srikanth and D. P. Reddy, "A Novel Approach for Solving Distribution Networks," *2009 Annual IEEE India Conference*, Gujarat, 2009, pp. 1-5.
3. Lin Ming Jin & Shu Park Chan, "An electrical method for finding suboptimal routes", *ISCAS'89 1989 IEEE* pp 935 – 938
4. Hiroyuki Mork, Senji Tsuzuki, "A fast method for topological observability analysis using minimum spanning tree technique", *IEEE Transaction on power system* vol. 6, no. 2, May 1991, pp 491 – 500
5. Cavellucci&Lyra, "Minimization of energy losses in electric power distribution system by intelligent search strategies", *International transaction in operational research*, vol. 4, no. 1, 1997, pp 23 – 33
6. M. Eriksson, M. Armendariz, O. O. Vasilenko, A. Saleem and L. Nordström, "Multiagent-Based Distribution Automation Solution for Self-Healing Grids," in *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2620-2628, April 2015

7. T. D. Sudhakar and K. N. Srinivas, "Prim's algorithm for loss Minimization and service Restoration in distribution Networks," *International Journal of Electrical and Computer Engineering*, ISSN 0974-2190 Volume 2, Number 1 , pp. 43—62, 2010
8. R. Taleski and D. Rajicic, 'Distribution Network Reconfiguration for Energy Loss Reduction', *IEEE Trans. on Power Systems*, Vol.12, No.1, pp.398-406, February 1997.

Appendix A: Program code for LCM matrix

```
l=load('linedata33bus.m');

s = l(:,2);

t = l(:,3);

dim = size(s);

num_bus = max(max(s),max(t));

num_edges = dim(1,1);

lcm = ones(num_edges,num_bus);

flag = 1;

count = 1;

s_new = s;

t_new = t;

while flag

[val_s,ind_s,j_s] = unique(s);

rep_s = histc(s,val_s);

[val_t,ind_t,j_t] = unique(t);

rep_t = histc(t,val_t);


[val_sn,ind_sn,j_sn] = unique(s_new);

rep_sn = histc(s_new,val_sn);

[val_tn,ind_tn,j_tn] = unique(t_new);

rep_tn = histc(t_new,val_tn);

common = intersect(val_sn(rep_sn>=1),val_tn(rep_tn>=1));
```

```

leaf = [setdiff(val_sn(rep_sn==1),common);setdiff(val_tn(rep_tn==1),common)];

leaf = leaf(2:length(leaf));

if length(leaf)==0

flag=0

break

end

if count == 1

leaf_neigh = zeros(size(leaf));

leaf_his = zeros(size(leaf));

else

leaf_his = [leaf_his; zeros(1,size(leaf_his,2))];

end

count_node = 1;

row_ind_arr = [];

for node = leaf

%[val_s,ind_s,j_s] = unique(s_new);

%[val_t,ind_t,j_t] = unique(t_new);

if find(val_sn == node)

leaf_node_ind = find(val_sn == node);

row_ind = find(j_sn == leaf_node_ind);

row_ind_arr = [row_ind_arrow_ind];

if all(leaf_neigh)

neigh_ind = find(leaf_neigh == node);

for ind=1:size(leaf_his,1)-1

```

```

lcm(row_ind,leaf_his(ind,neigh_ind)) = 0;

end

lcm(row_ind,node) = 0;

leaf_neigh(count_node) = t_new(row_ind);

leaf_his(ind+1,neigh_ind) = node;

else

leaf_neigh(count_node) = t_new(row_ind);

leaf_his(count_node) = node;

lcm(row_ind,node) = 0;

end

%s_new = [s_new(1:row_ind-1); s_new(row_ind+1:length(s_new))];

%t_new = [t_new(1:row_ind-1);t_new(row_ind+1:length(t_new))];

count_node = count_node + 1;

else

leaf_node_ind = find(val_tn == node);

row_ind = find(j_tn == leaf_node_ind);

row_ind_arr = [row_ind_arrow_ind];

if all(leaf_neigh)

neigh_ind = find(leaf_neigh == node);

for ind=1:size(leaf_his,1)-1

lcm(row_ind,leaf_his(ind,neigh_ind)) = 0;

end

lcm(row_ind,node) = 0;

leaf_neigh(count_node) = s_new(row_ind);

```

```

leaf_his(ind+1,neigh_ind) = node;

else

leaf_neigh(count_node) = s_new(row_ind);

leaf_his(count_node) = node

lcm(row_ind,node) = 0;

end

%t_new = [t_new(1:row_ind-1);t_new(row_ind+1:length(t_new))];

% s_new = [s_new(1:row_ind-1); s_new(row_ind+1:length(s_new))];

count_node = count_node + 1;

end

end

leaf_neigh = leaf_neigh(1:count_node-1);

if find(leaf_his==0)

if count==1

break

else

[r,c] = find(leaf_his==0)

leaf_his(:,c)=[];

end

end

count = count + 1;

s_new(row_ind_arr) = zeros(size(row_ind_arr));

t_new(row_ind_arr) = zeros(size(row_ind_arr));

```

end

Appendix B: Program code for Load Flow Analysis

```
clc;

clearall;

formatshort;

tic

m=load('loaddata33bus.m');

l=load('linedata33bus.m');


br=length(l);

no=length(m);

MVA=100;

KV=11;

Zb=(KV^2)/MVA;

% Per unit Values

for i=1:br

    R(i,1)=(l(i,4))/Zb;

    X(i,1)=(l(i,5))/Zb;

end

for i=1:no

    P(i,1)=(m(i,2))/(1000*MVA);

    Q(i,1)=(m(i,3))/(1000*MVA);

end
```

R

X

P

Q

C=zeros(br,no);

for i=1:br

a=l(i,2);

b=l(i,3);

for j=1:no

if a==j

C(i,j)=-1;

end

if b==j

C(i,j)=1;

end

end

end

C

e=1;

for i=1:no

d=0;

for j=1:br

if C(j,i)==-1

d=1;

end

```

end

if d==0

endnode(e,1)=i;

e=e+1;

end

end

endnode

h=length(endnode);

for j=1:h

e=2;


f=endnode(j,1);

% while (f~=1)

for s=1:no

if (f~=1)

k=1;

for i=1:br

if ((C(i,f)==1)&&(k==1))

f=i;

k=2;

end

end

k=1;

for i=1:no

if ((C(f,i)==-1)&&(k==1));

```

```

f=i;

g(j,e)=i;

e=e+1;

k=3;

end

end

end

end

end

end

for i=1:h

g(i,1)=endnode(i,1);

end

g;

w=length(g(1,:))

for i=1:h

j=1;

for k=1:no

for t=1:w

if g(i,t)==k

g(i,t)=g(i,j);

g(i,j)=k;

j=j+1;

end

end

end

```



```

end

g;

for k=1:br

e=1;

for i=1:h

for j=1:w-1

if (g(i,j)==k)

if g(i,j+1)~=0

adjb(k,e)=g(i,j+1);

e=e+1;

else

adjb(k,1)=0;

end

end

end

end

end

adjb;

for i=1:br-1

for j=h:-1:1

for k=j:-1:2

if adjb(i,j)==adjb(i,k-1)

adjb(i,j)=0;

end

end

```

```

end

end

adjb;

x=length(adjb(:,1));

ab=length(adjb(1,:));

for i=1:x

for j=1:ab

ifadjb(i,j)==0 && j~=ab

ifadjb(i,j+1)~=0

adjb(i,j)=adjb(i,j+1);

adjb(i,j+1)=0;

end

end

ifadjb(i,j)~=0

adjb(i,j)=adjb(i,j)-1;

end

end

end

adjb;

for i=1:x-1

for j=1:ab

adjcb(i,j)=adjb(i+1,j);

end

end

b=length(adjcb);

```

```

% voltage current program

for i=1:no
    vb(i,1)=1;
end

for s=1:10
    for i=1:no
        nlc(i,1)=conj(complex(P(i,1),Q(i,1)))/(vb(i,1));
    end

    nlc;

    for i=1:br
        Ibr(i,1)=nlc(i+1,1);
    end

    Ibr;

    xy=length(adjcb(1,:));

    for i=br-1:-1:1
        for k=1:xy
            if adjcb(i,k)~=0
                u=adjcb(i,k);
                % Ibr(i,1)=nlc(i+1,1)+Ibr(k,1);
                Ibr(i,1)=Ibr(i,1)+Ibr(u,1);
            end
        end
    end
end
end
end

```

```

Ibr;

for i=2:no

g=0;

for a=1:b

ifxy>1

ifadjcb(a,2)==i-1

u=adjcb(a,1);

vb(i,1)=((vb(u,1))-((Ibr(i-1,1))*(complex((R(i-1,1)),X(i-1,1))))));

g=1;

end

ifadjcb(a,3)==i-1

u=adjcb(a,1);

vb(i,1)=((vb(u,1))-((Ibr(i-1,1))*(complex((R(i-1,1)),X(i-1,1))))));

g=1;

end

end

end

if g==0

vb(i,1)=((vb(i-1,1))-((Ibr(i-1,1))*(complex((R(i-1,1)),X(i-1,1))))));

end

end

s=s+1;

end

nlc;

Ibr;

```

```

vb

vbp=[abs(vb) angle(vb)*180/pi]

toc;

for i=1:no

va(i,2:3)=vbp(i,1:2);

end

for i=1:no

va(i,1)=i;

end

va;


Ibrp=[abs(Ibr) angle(Ibr)*180/pi];

PL(1,1)=0;

QL(1,1)=0;

% losses

for f=1:br

Pl(f,1)=(Ibrp(f,1)^2)*R(f,1);

Ql(f,1)=X(f,1)*(Ibrp(f,1)^2);

PL(1,1)=PL(1,1)+Pl(f,1);

QL(1,1)=QL(1,1)+Ql(f,1);

end

```

```

Plosskw=(PI)*100000
Qlosskw=(Ql)*100000
PL=(PL)*100000
QL=(QL)*100000
voltage = vbp(:,1)
angle = vbp(:,2)*(pi/180)

```

Appendix C: Program code for comparing new Edges in Graph

```

l1 = load('linedata33bus.m');
l2 = load('data2.m');
l1_r = [l1(:,2) l1(:,3)];
l2_r = [l2(:,2) l2(:,3)];
penalty = 0.3
final = [l2(ismember(l1_r,l2_r,'rows'),[2:3]) l2(ismember(l1_r,l2_r,'rows'),4)+penalty]

```

Appendix D: Program code for Geneerating Minimum Sapnning for IEEE 33 Bus system

```
W = [0.0922 0.4930 0.3661 0.3811 0.8190 0.1872 1.7117 1.0299 1.0440 0.1967 0.3744 1.460  
0.5416 0.5909 0.7462 1.2889 0.7320 0.1649 1.5642 0.4095 2.7089 0.4512 0.8980 0.8959 0.2031  
0.284 1.0589 0.8043 0.50774 0.9745 0.3105 0.3411 2 0.5 0.5 0.5 2];
```

```
DG = sparse([1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 2 19 20 21 3 23 24 6 26 27 28 29 30 31  
32 8 9 33 25 12],[2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29  
30 31 32 33 21 15 18 29 22],W);
```

```
UG = tril(DG + DG')
```

```
view(biograph(UG,[],'ShowArrows','off','ShowWeights','on'))
```

```
[ST,pred] = graphminspantree(UG)
```

```
view(biograph(ST,[],'ShowArrows','off','ShowWeights','on'))
```