Optimal Micro Phasor Measurement Unit Placement for Complete Observability of the Distribution System

by

Brittney Kerns

BASc, University of British Columbia Okanagan, 2017

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE COLLEGE OF GRADUATE STUDIES

(Electrical Engineering)

The University of British Columbia

(Okanagan)

September 2018

The following individuals certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis/dissertation entitled:

Optimal Micro Phasor Measurement Unit Placement for Complete Observability of the Distribution System

submitted by Brittney Kerns in partial fulfillment of the requirements of the degree of Master of Applied Science.

Dr. Wilson Eberle, School of Engineering

Supervisor

Dr. Liwei Wang, School of Engineering

Co-Supervisor

Dr. Christine Chen, Department of Electrical and Computer Engineering

Supervisory Committee Member

Dr. Zheng Liu, School of Engineering

Supervisory Committee Member

Dr. Sina Kheirkhah, School of Engineering

University Examiner

Abstract

The electrical power distribution system was originally designed for one way power flow. With the addition of renewable energy sources on the electrical grid, bidirectional power flow is now occurring and is causing previously unseen fluctuations in voltage. This, in addition to an increasing power demand from utility users, is decreasing grid stability and increasing the chance of cascading blackouts. The distribution system needs to be monitored in real time so that minor issues relating to grid stability can be noted and fixed before they cascade into system failure.

A micro phasor measurement unit (μPMU) is a device that is able to monitor the distribution system in real time. When placed at a node, it is capable of measuring the voltage phasor at the node and all incident current phasors to the node. However, the high cost of these devices, in addition to the communication infrastructure that would be needed, makes it unfeasible to place them at every node on a feeder. However, it is also not necessary to place them on every node since, if the line impedance is known, the surrounding node voltage values can be accurately calculated.

Therefore, this thesis proposes an algorithm that optimally places μPMUs on distribution networks for complete observability. This algorithm is based on a greedy algorithm which has many benefits such as fast computation time and high reliability. High reliability occurs when most of the distribution system is being monitored even in the event of a μPMU failure. The algorithm was tested on standard IEEE distribution feeders: 13-node, 33-node, 34-node, 37-node and 123-node. The results presented include the number of μPMUs needed for complete observability, computation time, and a measure of the reliability.

Lay Summary

The number of renewable energy sources connected to the electrical grid are limited since too many can result in frequent power outages. Real time monitoring of the distribution system, the part of the power system that brings power into homes and businesses, is needed in order to see and fix issues before they cascade into a blackout. The key goal of this research was to enable a way to monitor the distribution system utilizing existing technology and be as economical as possible. A micro phasor measurement unit (μPMU) is a device capable of monitoring the distribution system in real time, however it is expensive to implement. Therefore, the contribution of this thesis is the development of an algorithm that places μPMUs in a way that minimizes how many is needed while still achieving a high reliability, which occurs if most of the system is still monitored when a μPMU fails.

Table of Contents

List of Tables

List of Figures

Glossary

BOI Bus Observability Index

μPMU Micro Phasor Measurement Unit

NDC Nationally Determined Contribution

NREL National Renewable Energy Laboratory

PMU Phasor Measurement Unit

PSL Power Standards Lab

SORI System Observability Redundancy Index

TPES Total Primary Energy Supply

UNFCCC United Nations Framework Convention on Climate Change

Acknowledgments

I would like to express my extreme gratitude to my supervisors, Dr. Liwei Wang and Dr. Wilson Eberle for their constant support throughout this endeavor. Their advice and guidance were instrumental to the success of my research.

Additionally, I am grateful for the funding I received through Mitacs Accelerate, a collaboration with Enbala Power Networks, for without which would have made it difficult for me pursue this research. Furthermore, I would like to specifically thank Malcolm Metcalfe, who's ideas and knowledge inspired me to do this research and will continue to inspire me throughout my career.

And last, but not least, I would like to thank my dad, John Kerns, for his unending support.

Dedication

To Sean, for his love and support

Chapter 1

Introduction

1.1 Motivation

Climate change is an increasingly concerning topic that has been gaining attention worldwide due to the adverse effects on public health and the environment. This attention has culminated in a yearly conference held by the United Nations Framework Convention on Climate Change (UNFCCC). The most recent notable conference took place in 2015 and was held in Paris, France. The outcome of this conference was the Paris Agreement. This agreement has the cooperation of over 190 countries who's planned contributions aims for a goal of keeping global temperature rise below two degrees Celsius [1]. Each country determined which contributions, called Nationally Determined Contributions (NDCs), they would make. Canada's submitted NDC has a target of 30% emission reduction below 2005 levels by 2030 [2].

In 2015, fossil fuels made up 73% of Canada's Total Primary Energy Supply (TPES) [3]. TPES is calculated by the following: Production + Imports - Exports + Stock changes. Canada's proposed NDC consists of two main parts: pricing carbon pollution and expanding the use of clean electricity and low-carbon fuels [2]. In order to make these changes, traditional coal units will be phased-out and an upgrade to the existing electricity grid will occur. This includes smart grid technologies and increasing the sources of clean power.

1.1.1 Power System Overview

Power systems are comprised of four main parts: generation, transmission system, distribution system, homes and businesses. Firstly, power is generated and the most common way to generate power is by burning fossil fuels [4]. Next, the generated power flows along the transmission system to the distribution system. The transmission and distribution systems compose the delivery part of the power system. Lastly, power flows from the distribution system to customer sites, which are homes and businesses. This is illustrated in Figure 1.1 below.

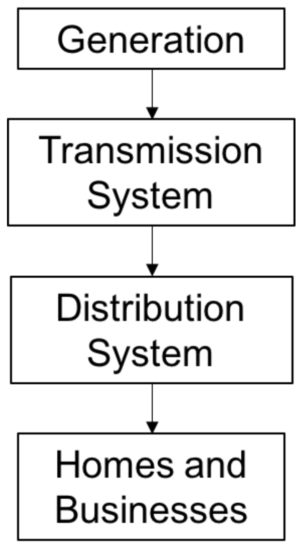

Figure 1.1: A typical power system.

1.1.2 The Problem

The distribution system was originally designed to carry power **one way** from transmission systems to homes and businesses. The addition of renewable energy sources in the distribution system is now causing **bidirectional** power flow to occur. For example, solar panels on top of homes will generate power when the sun is shining. If this generated power exceeds what is needed for the home at that time, it will flow to the distribution system. When this occurs, generator speed will decrease as there is more capacity than demand and therefore it will not need to produce as much power. When a system contains a high number of renewable energy devices, fewer generators would be needed in terms of power demand. However, generators play an important role.

Generators can store a lot of energy in their rotating generator rotor, therefore they have a lot of inertia. Essentially, if the generator's input ceased to produce any power, the generator would continue producing some electricity until their stored energy depletes. Inertia slows the rate of frequency decline when a fault or failure on the system occurs and is a very important component on the power system. This slowing of frequency decline allows operators to find a solution before other components, ones that cannot operate at a lower frequency, trip offline. Solar and wind power contain virtually no inertia. Therefore, if a power system contains a high penetration of renewable energy sources, frequency would decline rapidly during these events which would lead to a higher blackout occurrence.

The second issue is that power demand from utility users has been increasing due to the increasing number of electronic devices being used and operated. These changes have been causing loads to change very quickly [5], leading to an increasingly unstable grid [6].

Due to these challenges, there is currently a **limit to the number of renewable energy sources** that can be connected to the grid. A study was done by the National Renewable Energy Laboratory (NREL) on the Eastern Interconnection power system. They found that this power system could accommodate a maximum of 30% renewable supply capacity (of the total supply) on the grid as it is currently operated [7]. This value needs to be increased in order to reduce the amount of greenhouse gases being generated and, in turn, combat climate change.

1.1.3 The Solution

In order to allow a higher penetration of renewable energy sources, **real-time monitoring of the distribution system** needs to be implemented. With real-time measurements, issues that arise can be rectified immediately before it cascades into system failure. As an example, having real-time information could have prevented the 2003 great north eastern blackout in Canada and the United States [8].

Furthermore, a better understanding of the distribution system could be formulated and highly advanced control strategies could be put into place. This would allow the power system to accommodate a much higher number of renewable energy sources.

1.2 Phasor Measurement Units

Phasor Measurement Units (PMUs) are highly accurate measurement devices that produce GPS time stamped measurements of both voltage phasors and current phasors. This allows them to synchronize measurements from distant locations, giving a real-time picture of the complete power system [9]. Due to this synchronization ability, they are one of the most important devices in power system monitoring and control [10]. PMUs were invented in the mid 1980s [11] and have been used in transmission systems since the mid 1990s [12]. Recently, there has been increased interest in placing these devices at the distribution level. However, the voltage angle differences between locations on distribution systems are up to two orders of magnitude smaller than those on transmission networks [13]. Therefore, a much more accurate PMU is required.

A Micro Phasor Measurement Unit (μPMU) is such a device that is able to discern between the small voltage angle differences on distribution networks. Currently, there is only one μPMU commercially available and it is made by a company called Power Standards Lab (PSL) [14]. Their μPMUs have an amplitude resolution of 0.0002% of full scale and an angle resolution of 0.001 degrees, which is sufficient for the distribution system.

Placing μPMUs on the distribution system can provide multiple benefits such as improved monitoring, protection and control [15]. Some of the benefits are highlighted below [16]:

1. It can provide a real-time snapshot of the distribution system which would enable a faster response if any issues arise.

2. If a fault were to occur, the data from the μPMUs could be analyzed to potentially prevent faults from occurring again in the future.

3. If a power outage were to occur, the location of where the power outage will be known immediately. This would enable utilities to restore power faster.

Note that in literature, the term PMU is used for either the transmission system or the distribution system whereas μPMU refers to **just** the distribution system. Therefore, throughout this thesis, PMU will be used generally to mean transmission level or distribution level PMU while μPMU will refer solely to distribution level μPMUs.

1.3 Necessity of Placement Algorithm

μPMUs are expensive devices. One unit from PSL costs approximately \$8000 CAD. In addition, the communication infrastructure that would need to be implemented could have a cost significantly greater than the μPMU cost [17], making it unfeasible to place them on every node in a network. Fortunately, it is not necessary to place μPMUs at every node. If the line impedance is known, the voltages of the surrounding nodes can be calculated. Therefore, an optimization algorithm can be used to optimally place a minimum number of μPMUs on distribution networks. Note that the terms node and bus are used interchangeably throughout this thesis.

1.3.1 Distribution vs. Transmission Systems

Placement algorithms should be optimized for either the transmission system or the distribution system [18]. The key difference between them is their topologies: distribution systems are radial while transmission systems are meshed. This can be seen when comparing a distribution feeder, such as the IEEE 13-node as seen in Figure 1.2, and a transmission feeder, the IEEE 14-node as seen in Figure 1.3.

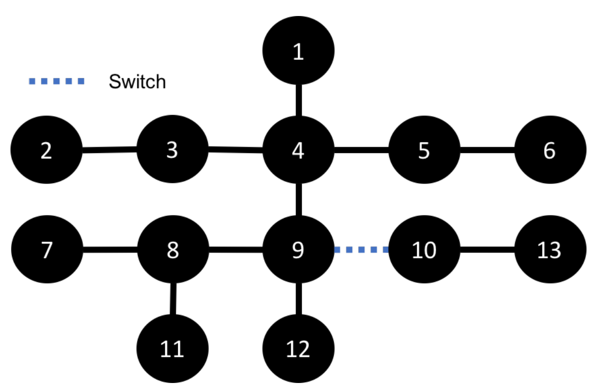

Figure 1.2: 13-node radial IEEE distribution test feeder.

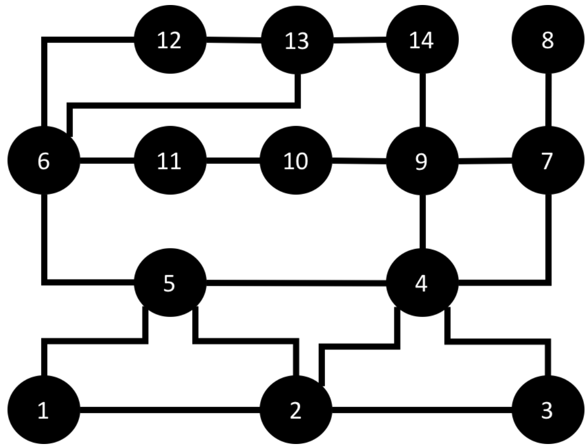

Figure 1.3: 14-node meshed IEEE transmission test feeder.

The other noticeable difference is the number of nodes for each network. Real world distribution networks are comprised of tens to hundreds of thousands of buses while transmission systems range between a few hundred to thousands of buses [19]. Due to this factor, computation time is a much more important factor to consider for distribution systems. Many placement algorithms previously used for transmission systems will be far too computationally expensive for distribution systems. Additionally, specific placement goals may be different for transmission systems due to their topology. For example, on transmission systems one goal could be to place PMUs in such a way that complete observability occurs even in the event of a PMU failure, such as in [20]. Due to the radial nature of distribution systems, this goal is not feasible as the number of PMUs that would be needed would increase drastically, making the cost unfeasible. Complete observability is defined in this thesis in Section 2.2. Lastly, distribution systems typically contain switches, which need to be accounted for in the placement algorithm.

To summarize, PMU placement strategies should be optimized for the distribution grid due to the three following reasons:

- Computation time: distribution systems have tens of thousands to hundreds of thousands more nodes than transmission systems.

- Topology: distributions systems are radial and this factor can be used to decrease computation time.

- Switches: distribution systems contain switches which need to be accounted for in the placement algorithm.

1.4 Thesis Contributions

This thesis proposes a novel μPMU placement algorithm that outperforms others in similar works while considering four factors in order to make it robust and reliable. These factors are listed below and explained in more detail in Section 2.2:

1. Computation time: The aim was to keep this as low as possible in order to be feasible for real world distribution networks with many nodes.

2. Redundancy: The aim was to have at least 90% of a network observed in the event of one μPMU failure.

3. Simplicity: The aim was to have a simple, single input that would make it easy to implement in the real world.

4. Network Reconfiguration: The distribution system typically contains switches which can change its topology if the status of the switches change. The aim was to place the μPMUs in such a way that complete observability is achieved for all configurations.

This combination of factors has not been considered together for the placement problem on the distribution system. Previous works have only focused on one or two of these items. Therefore, the contributions of this thesis are as follows:

- A fast, easy to implement μPMU placement algorithm that optimally places a minimum or near minimum number of these devices on distribution systems

- This algorithm will include the possibility of network reconfiguration and will keep its full observability in all cases

- This algorithm will be highly reliable - i.e. if a μPMU fails, most buses will still be observed

1.5 Thesis Outline

This thesis consists of five chapters and is briefly described below.

- Chapter 1 introduces the thesis. It explains the motivation, introduces PMUs/μPMUs, explains the necessity of optimized placement algorithms for distribution systems and states the contribution of the thesis.

- Chapter 2 explains some background information and overviews the literature that solved the μPMU placement problem for complete observability of the distribution system. The background topics are: connectivity matrix, observability, redundancy, network reconfiguration and optimization algorithms. For the review of previous works, it highlights their algorithm's advantages, disadvantages and limitations.

- Chapter 3 presents the proposed μPMU placement algorithm. It states what assumptions were used, how network reconfiguration was considered, and explains the proposed algorithm step by step.

- Chapter 4 shows where the μPMUs are placed for different IEEE distribution test feeders: 13-node, 33-node, 34-node, 37-node and 123-node. It compares the results of the proposed algorithm with the results of the works outlined in the literature review. It also presents a redundancy analysis which shows the reliability of the system if one or more μPMUs were to fail.

- Chapter 5 concludes the thesis and discusses future work.

Chapter 2

Literature Review

2.1 Overview

This chapter is comprised of two main sections. The first section, Section 2.2, explains the background concepts: connectivity matrix, observability, redundancy, network reconfiguration and optimization algorithms. In the second section, Section 2.3, a review is completed for existing optimization algorithms that were used to place μPMUs for complete observability of the distribution system. These optimization algorithms are as follows: integer programming, exhaustive search method, simulated annealing, graph theory, greedy algorithm.

2.2 Background

This section describes what a connectivity matrix is, the topics of observability and redundancy, what network reconfiguration is and why it is important to consider, as well as introduce optimization algorithms. These important topics are imperative for understanding certain aspects of the literature review as well as the proposed solution later in the thesis.

2.2.1 Connectivity Matrix

The connectivity matrix is the binary admittance matrix of a power system network and essentially shows which buses are connected to one another. Usually denoted as matrix A, its formal definition is defined in (2.1), where i and j describe buses:

$$A(i,j) = \begin{cases} 1, & \text{if } i = j \\ 1, & \text{if bus i and bus j are connected} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

Example 2.1

A simple 4-node network, as shown below in Figure 2.1, would have a connectivity matrix as shown in (2.2). The rows and columns are numbered as 1, 2, 3 and 4 which correspond to the numbered nodes of the network. Therefore, since node 1 is connected to node 2, there is a "1" in the A(1,2) and A(2,1) spot. Note that there are always "1's" on the diagonal of the A matrix, this is described in its formal definition above in (2.1).

It can be seen that the number of nodes in a system determines the size of the matrix. For example, a 4-node network would correspond to a 4x4 connectivity matrix and a 13-node network would correspond to a 13x13 connectivity matrix.

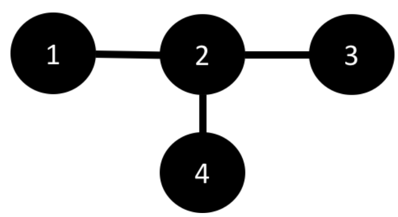

Figure 2.1: Example 4-node network.

$$A = \begin{array}{c c} & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array} \tag{2.2}$$

2.2.2 Observability

With regards to μPMU placement, observability refers to the number of measurements known on a network. For example, complete observability occurs when every node voltage phasor and line current phasor is either measured by a μPMU or can be directly calculated using neighboring measured values. Incomplete observability occurs when these conditions are not met. The focus in this thesis is placing μPMUs such that the complete observability condition is met.

Observability can be determined using either numerical methods or topology based algorithms. Numerical methods use the information or gain matrix and the network is completely observed if the matrix has full rank [21]. If a matrix has full rank, its rows/columns are linearly independent from each other. Topology based algorithms use graph theory and it is the more popular method to check for observability. That is because the only information required is network connectivity, measurement type, and location. The system is said to be observable if a full rank spanning tree can be achieved [22].

Example 2.2

Referring back to the 4-node example in Figure 2.1, if a μPMU were to be placed at node 2, complete observability would be achieved. By measuring the voltage at node 2 and all currents flowing into and out of node 2, and assuming the line impedance is known, the voltages at nodes 1, 3, and 4 can be accurately calculated.

2.2.3 Redundancy

Redundancy of a system is related to the reliability of the system. It can be calculated using two indices as proposed in [23]: Bus Observability Index (BOI) and System Observability Redundancy Index (SORI). The BOI (β_i) for each node is the number of μPMUs measuring each node. SORI (γ) adds all BOI values as seen in (2.3). For this equation, i corresponds to the node number and n is the total number of nodes for the given distribution network.

$$\gamma = \sum_{i=1}^{n} \beta_i \tag{2.3}$$

If a μPMU placement set has a high SORI value, it will be measuring more nodes in the event of a μPMU failure. Therefore, the μPMU placement set is said to be reliable. This will be explained more in the following example.

Example 2.3

An example illustrating the importance of having high redundancy is shown below. In Figure 2.2 below, there is a completely observed 7-node network. If the μPMU located on bus 2 or bus 6 failed, only one node would be unobserved (nodes 1 and 7, respectively). If the μPMU on bus 4 failed, then two nodes would be unobserved (nodes 4 and 5). Therefore, in the worst-case scenario for this particular μPMU placement set, two nodes would be unobserved if a μPMU were to fail. The best-case scenario is only one node would be unobserved.

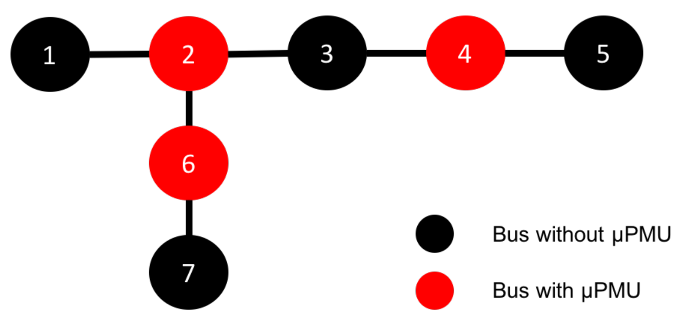

Figure 2.2: Example μPMU placement for a 7-node system.

Now, view Figure 2.3 below. The μPMUs are placed differently, however, complete observability is still achieved and the number of μPMUs used are the same. If a μPMU were to fail at bus 1 or bus 7, two nodes would be unobserved (nodes 1 & 2 and nodes 6 & 7, respectively). If the μPMU located at bus 4 were to fail, three nodes would be unobserved (nodes 3, 4 and 5). For this placement set, the worse-case scenario is three unobserved nodes in the case of a μPMU failure. Alternatively, the best-case scenario would see two nodes unobserved. Clearly, the placement set in Figure 2.2 is preferable to the one in Figure 2.3.

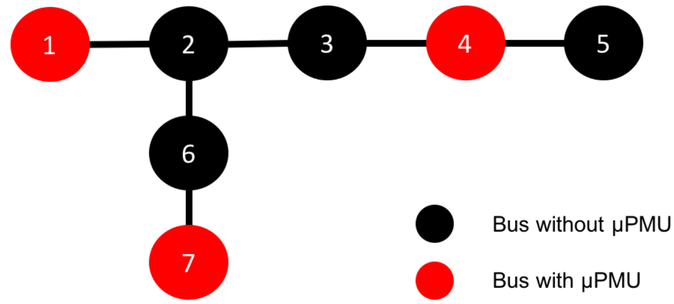

Figure 2.3: Example μPMU placement for a 7-node system.

This preference can also be determined by comparing the SORI values. The SORI value for the placement set in Figure 2.2 is 10 and the SORI value for Figure 2.3's placement is 7. Since a higher SORI value corresponds to a higher system reliability, it can again be concluded that the μPMU placement depicted in Figure 2.2 is preferable.

2.2.4 Network Reconfiguration

Switches are common on distribution systems. They are there to reduce power losses and improve power quality [24] by opening and closing them at key times. Therefore, network reconfiguration refers to the changing configurations that occur when switches change their status from open to closed, or vice versa. If a system had two switches for example, it would have four configurations:

- Configuration 1: switch 1 - open, switch 2 - open
- Configuration 2: switch 1 - open, switch 2 - closed
- Configuration 3: switch 1 - closed, switch 2 - open
- Configuration 4: switch 1 - closed, switch 2 - closed

As the number of switches increases, so does the complexity of the number of configurations.

This factor is important to consider because a μPMU placement set may achieve complete observability for one configuration, but not another. Therefore, μPMUs need to be placed in such a way that complete observability is achieved regardless of the switch(es) status.

Example 2.4

In order to visualize this, a simple example is shown below in Figure 2.4. It is the same placement from Figure 2.3, but this time there is a switch. When the switch is closed, complete observability is achieved. However, if the switch is open, there is no way to observe node 3.

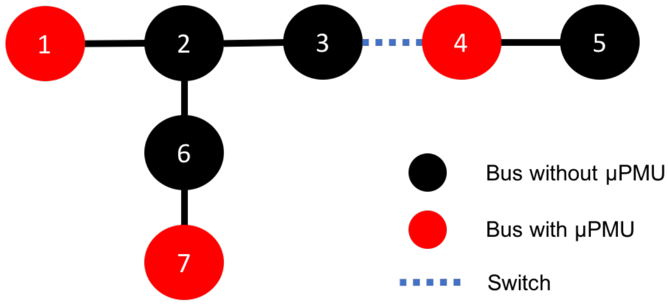

Figure 2.4: Example μPMU placement for a 7-node system with a switch.

If the switch had been considered, the μPMUs would likely have been placed as in Figure 2.5, which is the same placement from Figure 2.2 but now includes the switch. It can be seen that regardless of the status of the switch, complete observability is always achieved.

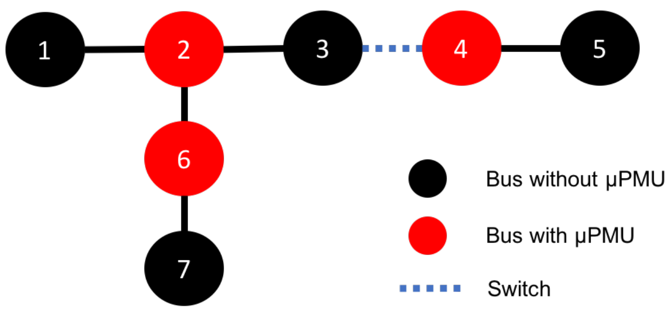

Figure 2.5: Example μPMU placement for a 7-node system with a switch.

2.2.5 Optimization Algorithms

Optimization algorithms can be used to place a minimum number of PMUs optimally on the network. The algorithms can be classified as deterministic or stochastic. Deterministic algorithms follow a strict path. In essence, each time the program is run, the algorithm will follow the exact same steps each time. On the other hand, stochastic algorithms tend to have some randomness and may take a slightly different path when the program is run.

The PMU placement problem typically uses stochastic algorithms, which can further be broken down into heuristic and metaheuristic [25]. Heuristic and metaheuristic algorithms were proposed in the early 1970's to overcome problems such as risk of divergence and difficulties passing over local optimal solutions [26]. Heuristic algorithms can either produce a good approximation or an exact solution, depending on the algorithm itself [21]. Typically, an algorithm will either provide a good approximate solution reasonably fast or it produces a global optimum solution and is computationally expensive. Metaheuristic algorithms can be defined the same as heuristics, however, they generally perform better [25].

2.3 Previous μPMU Placement Methods

This review focuses on papers that placed μPMUs for complete observability of distribution networks. The optimization algorithms used in these papers were: integer programming, exhaustive search method, simulated annealing, graph theory, and greedy algorithm. These algorithms will be explained and the papers will be analyzed to highlight the advantages, disadvantages, and limitations of the methods where present.

2.3.1 Integer Programming

Integer programming is the most popular method to place PMUs on a network. This is due to the fact it guarantees finding the minimum number of PMUs needed for any network. It is defined as a mathematical programming method that finds the maximum or minimum of a linear objective under linear constraints [25]. The disadvantages of this method are its high computation time and there is no way to include preferred places to place the μPMUs. For example, if a μPMU is placed at an end node, it is only able to observe that node and one other. Therefore, it is better to place them elsewhere in order to optimize the use of μPMUs and achieve a higher redundancy.

This problem can be formulated using (2.4) below.

$$
\begin{aligned}
&\min \sum_{i}^{n} w_i x_i \\
&\text{subject to:} \quad f(x) = A.x >= \mathbf{b}
\end{aligned}
\tag{2.4}
$$

The variable w_i is the cost of a μPMU installed at bus i and the total number of buses is n. Typically, it is assumed that the cost is the same at any location. **b** is a vector and it relates to the redundancy of the system. Essentially, it would be a desired BOI vector as described earlier. This can be used to meet a certain redundancy requirement for the system, however it is usually assumed to be a vector of ones. x is a binary decision variable vector and is defined in (2.5) below:

$$
x_i = \begin{cases} 1, & \text{if a } \mu\text{PMU is located at bus } i \\ 0, & \text{otherwise} \end{cases}
\tag{2.5}
$$

Three papers used integer programming to place μPMUs: [19], [27], and [28].

In [19], they formulated the μPMU placement problem based on a few assumptions:

1. μPMUs have unlimited channels
2. Each node has communication available
3. The cost to install a μPMU is the same for any node, i.e. w_i is just a vector of ones
4. **b** was also chosen to be a vector of ones

The goal of their paper was to reduce the computation time. In order to do this, they separated the test feeders into primary and secondary networks. The primary network is chosen first and in their example, they chose the path containing the largest number of consecutively connected buses. The rest of the buses form the secondary network. They would solve the primary network first to be completely observable while ignoring the secondary network. Then, the secondary network can be solved. There is a reduced number of buses to consider when solving the secondary network as some will be observed from the primary network μPMU placement, decreasing computation time.

They tested their method on IEEE feeders: 13-node, 34-node and 123-node. In their results, they were able to achieve a 19% faster computation time for the 13-node network, 51% for the 34-node network and 60% for the 123-node network when compared to the classic integer programming method. However, although they achieved a faster computation time, they did not achieve the minimal μPMU case. Another issue with this method is the way they described how to choose a primary network. There can be multiple choices for the primary network and a different choice could lead to even more μPMUs needed for complete observability of the system. Finally, as it is not possible to include preferred nodes to place μPMUs on for integer programming, many μPMUs were placed on end nodes. When μPMUs are placed on end nodes, it leads to a much lower SORI value and therefore a less robust and less reliable system. Their μPMU placement for the IEEE 34-node feeder can be seen in Figure 2.6 below.

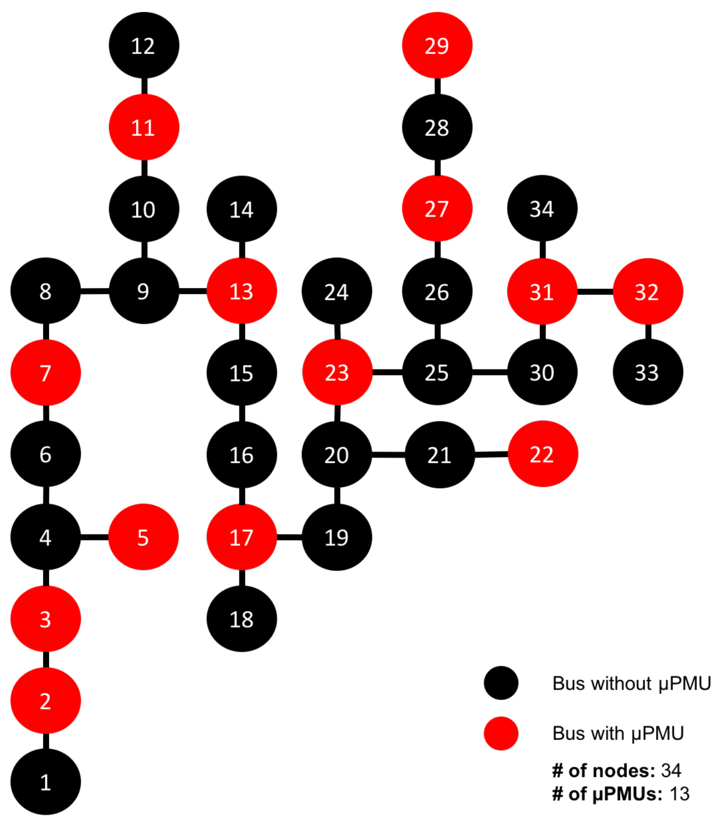

Figure 2.6: IEEE 34-node μPMU placement results from [19].

In [27], they used integer programming in order to place μPMUs on distribution systems.

However, the focus on this paper was network reconfiguration. They explained that a change in network configuration can have a large impact on optimal PMU placement.

Similar to [19], they decided to keep **b** from (2.4) as a vector of ones. After some analysis, they came to an inequality which they applied and closed all switches in order to achieve complete observability for all configurations. Then, (2.4) will become (2.6) below.

$$\begin{aligned} &\min \sum_{k=1}^{n} w_k x_k \\ &\text{subject to:} \quad f(x) = A.x \geq b \end{aligned} \tag{2.6}$$

Where A is the connectivity matrix and is described in (2.7) below.

$$A_{kl} = \begin{cases} r, & \text{if k = l} \\ 1, & \text{if bus k is connected to bus j } i \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

They essentially transform the constraints of all configurations into one general form they solve. The detriment of this method is that it overcomplicates the problem and does not lead to a minimal μPMU case. They tested their method on IEEE networks: RBTS-2 and 33-node. Their system and placement method for the 33-node feeder can be seen in Figure 2.7 below. Note that μPMUs were again placed on end nodes. For this network, if a μPMU were placed on node 2, the μPMUs from nodes 1, 3 and 19 could all be removed while still achieving complete observability.

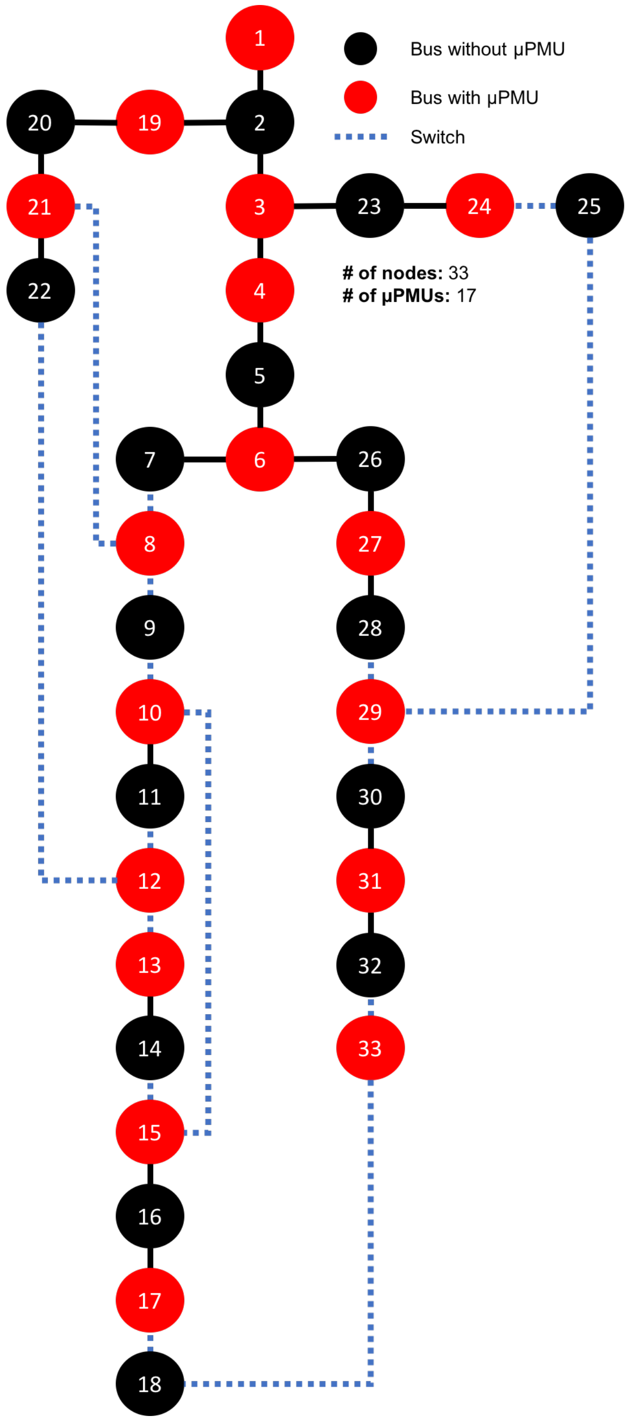

Figure 2.7: IEEE 33-node μPMU placement results from [27].

In [28], their focus was mainly on μPMU placement for incomplete observability, using state estimation to estimate the remaining values. However, they did include placement for complete observability in one of their cases therefore it was included in this literature review. As the topic of model order reduction or state estimation is not relevant to this thesis, it will not be described. A standard integer programming method was used to place the μPMUs. They tested their method on IEEE 34-node and 123-node test feeders. Their placement results for the IEEE 34-node feeder can be seen below in Figure 2.8. They did not achieve the minimal case for this feeder, but did achieve minimal results for the 123-node feeder. Again, it can be seen that μPMUs were placed on end nodes, corresponding to a lower SORI value.

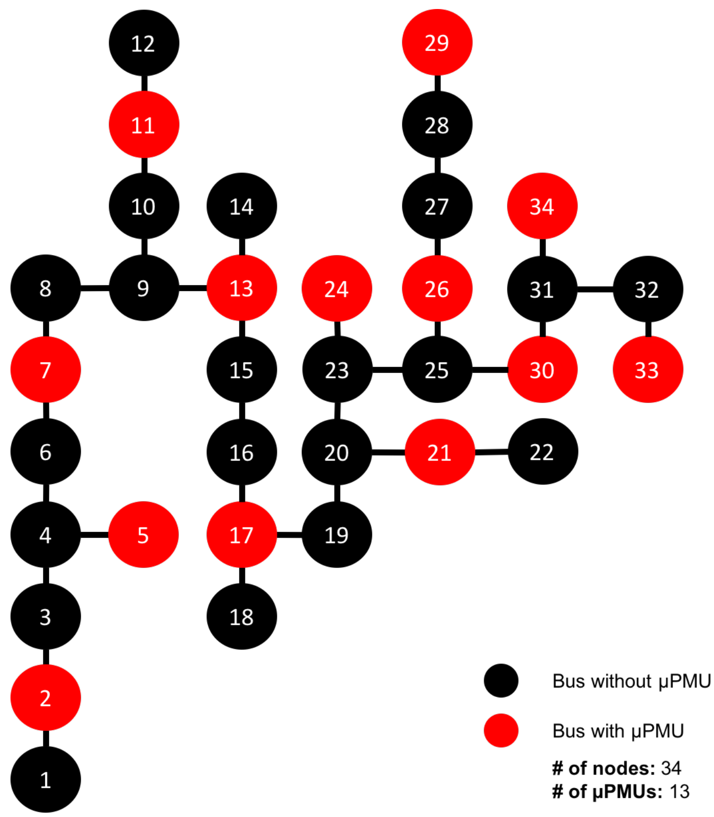

Figure 2.8: IEEE 34-node μPMU placement results from [28].

To summarize, the integer programming method of solving the μPMU placement problem has the disadvantage of typically placing μPMUs on end nodes. This corresponds to a lower SORI value and therefore lower system reliability. Furthermore, none of the papers described were able to achieve consistently minimal results.

2.3.2 Exhaustive Search Method

The exhaustive search method evaluates all possible options and combinations in order to find the global optimum [29]. Similar to integer programming, it finds the minimal case for μPMU placement. Unlike integer programming, because all possible μPMU placement configurations are found, it is able to find a placement set with a high redundancy. However, due to its exhaustive nature, it is too computationally expensive for large networks [30].

Two papers used an exhaustive type search method and both were written by the same authors: [29] and [13].

In [29], they tried to reduce the computation time of the exhaustive search method. To do this, they first reduced the search space by removing certain nodes as possible μPMU placement locations. The first type of nodes removed were end nodes. As stated in the integer programming section, this is not a desirable location for μPMU placement because it is only able to observe itself and one other node. The second type of nodes they removed from the search space were the nodes connected to end nodes. The reason for this is such: in order to observe the end nodes without placing a μPMU there, there would have to be a μPMU placed at the node connected to it.

Next, they attempted to narrow the search range for how many μPMUs would be needed in order to achieve complete observability of a distribution network. They used (2.8) to do this. The variable N is the number of buses and N_P^{UB} is the theoretical "upper bound" or maximum number of μPMUs that would be needed.

$$N_P^{UB} = \frac{N}{2} \tag{2.8}$$

Due to their preprogramming steps, some μPMUs would already be assigned to the nodes that are connected to end nodes. Therefore, N_P^{UB} can be further broken down into (2.9) as seen below.

$$N_P^{UB} = N_{pre} + N_{unobs} \tag{2.9}$$

Once these preprogramming steps have been completed, their exhaustive algorithm can be run.

The algorithm can be summarized in the following steps:

1. Start searching with the number of μPMUs equal to the midpoint of a lower search bound and the upper search bound described above. This midpoint value is the test point. The lower search bound starts at 0.

2. For the test point, all combinations are generated and tested.

3. If one of the combinations can make the system completely observed, the test point becomes the upper limit. If it doesn't, the test point becomes the lower limit. This continues until the difference between the upper and lower limit is one.

4. The final result, number of μPMUs needed, is the last upper limit.

Occasionally, multiple solutions with the same minimal number of μPMUs are found. In this case, the optimum solution would be the one with the highest SORI value.

They tested their algorithm on three IEEE feeders: 13-node, 34-node and 37-node. Their placement result for the 34-node feeder can be observed in Figure 2.9. They were able to achieve the minimal μPMU case for all networks they tested. Although they did achieve a faster computation time than the standard exhaustive algorithm, this method is still not feasible for even a slightly larger network such as the IEEE 123-node network.

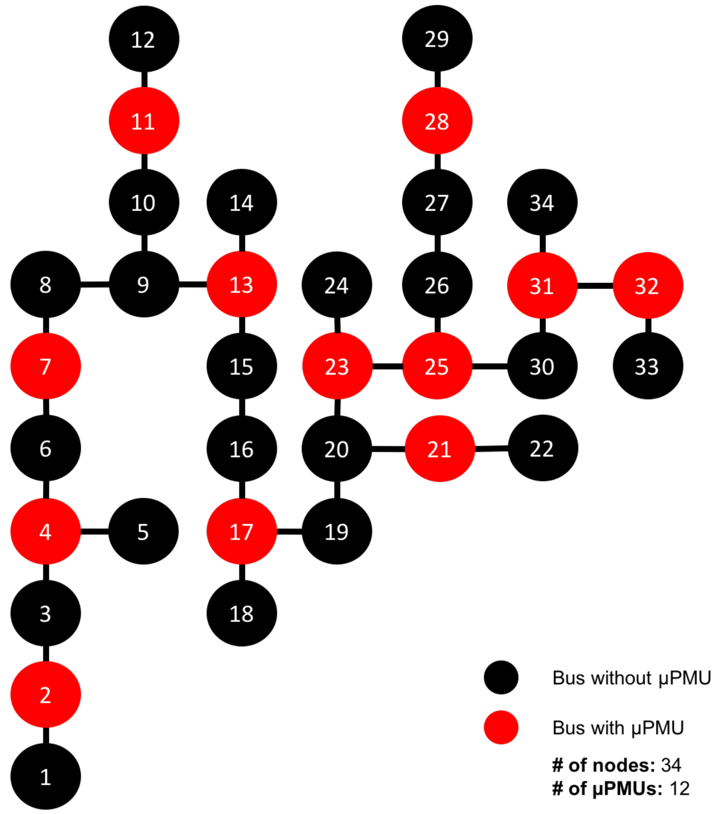

Figure 2.9: IEEE 34-node μPMU placement results from [29].

Example 2.5

This example will work through the exhaustive algorithm steps for the IEEE 123-node network in order to highlight the extensive computation time that would be needed.

This network would have an original theoretical upper-bound of $123/2 = 61.5$ which would be rounded up to 62. The number of end nodes is 46, therefore 46 μPMUs would be placed on the nodes connected to those end nodes. The lower search bound will start with 0 and the upper bound is $62 - 46 = 16$. The test point will be the middle of the bound which is 8.

To summarize, at this point 46 μPMUs have been placed, and a further 46 are end nodes. Therefore, the remaining nodes in the search space has been reduced to $123 - 46 - 46 = 31$ possible places.

The test point, 8, will find every combination of places in those 31 spaces. Using (2.10) below, which is a standard formula to calculate the number of possible combinations without repetition, the number of combinations can be found.

$$C_{n,k} = \frac{n!}{k!(n-k)!} \tag{2.10}$$

In our case, n is 31 and k is 8. This calculation comes out to be 7,888,725. Therefore, over 7 million different placement configurations would be tried and tested to see if complete observability occurs. This is a staggering number and it is just the first iteration. After testing these combinations, the number 8 would either become the new upper limit or the new lower limit, and another iteration would be done. Testing these possible combinations would require extensive computation time, which would only increase as the network size increases. Therefore, this method is not feasible to use for μPMU placement.

In [13], which is written by the same authors as [29], they supply another exhaustive search algorithm that has the same preprogramming steps. The only difference this algorithm has is in steps 1 and 3. In step 1, the test point is the upper bound, instead of it being the middle number. In step 3, after each iteration, the upper bound is decreased by one. The end result is the same for both papers.

2.3.3 Simulated Annealing

This method is inspired from annealing in metallurgy and is operated by trying random variations of the current solution [30]. It is faster than exhaustive search as all combinations are not tried [29]. The disadvantage of this method is its aim is to find a good approximation, not an optimal solution [6]. Therefore, the minimal μPMU case may not be achieved and it would be possible for μPMUs to be placed on end nodes (unless customization were done to prevent this).

The same paper that used the exhaustive search method also explored simulated annealing [29]. They applied the same pre-programming steps in order to reduce computation time. Afterwards, they applied their simulated annealing algorithm. The steps to the simulated annealing algorithm are summarized below:

1. The test point is chosen as the midpoint between a lower bound and an upper bound. The

lower bound starts at 0 and the upper bound starts as half the total number of buses as described in (2.8).

2. Then, generate a random placement set with the number of μPMUs equal to the test point.

3. Move one of the μPMUs to a space not occupied by a μPMU. These locations exclude end buses and those connected to end buses.

4. If complete observability is achieved, the lower bound is changed to equal the test point. If complete observability is not achieved, go to step 3. If incomplete observability occurs after several μPMU moves, then the upper bound is changed to equal the test point.

5. This continues until the difference between the upper and lower limits is equal to 1.

Note that this method is a little more complex than the summarized steps listed and the full method can be read in [29].

Again, minimal results were achieved using this customized simulated annealing algorithm. The μPMU placement set result was identical to that of their customized exhaustive one. However, although this method should yield faster results than exhaustive, the results section in [29] show that this algorithm was much slower than their customized exhaustive method. In fact, their exhaustive algorithm yielded results about 91% faster than the simulated annealing algorithm.

2.3.4 Graph Theory

Graph theory converts the network into a graph where the buses become vertices and the distribution lines become edges [31].

In [31], they approached the optimal μPMU placement problem in a graph theoretical way. Some terminology is described below:

- Pendant vertex: end node
- Cut vertex: all other nodes

Their proposed method to place μPMUs occurs in two stages. Stage 1 can be summarized below in one step:

1. Highlight pendant vertices and place μPMUs at the cut vertices connected to the pendant vertices.

Stage 2 is summarized in two steps below.

1. Find and group the unobserved vertices into groups of no more than 3.

2. Install one μPMU per group which makes that group observable.

They tested their algorithm on standard IEEE test feeders: 13-node, 34-node, and 37-node. At first glance, it appears this method is extremely fast, provides minimal results, and provides a high redundancy. However, they did not consider network reconfiguration and, upon further inspection, this method may not be able to consistently provide the minimal μPMU case. If there is a larger and more complex network, such as the 123-node, depending on how the unobserved nodes are grouped, different non-optimal results may occur. Their placement result for the 34-node can be seen below in Figure 2.10.

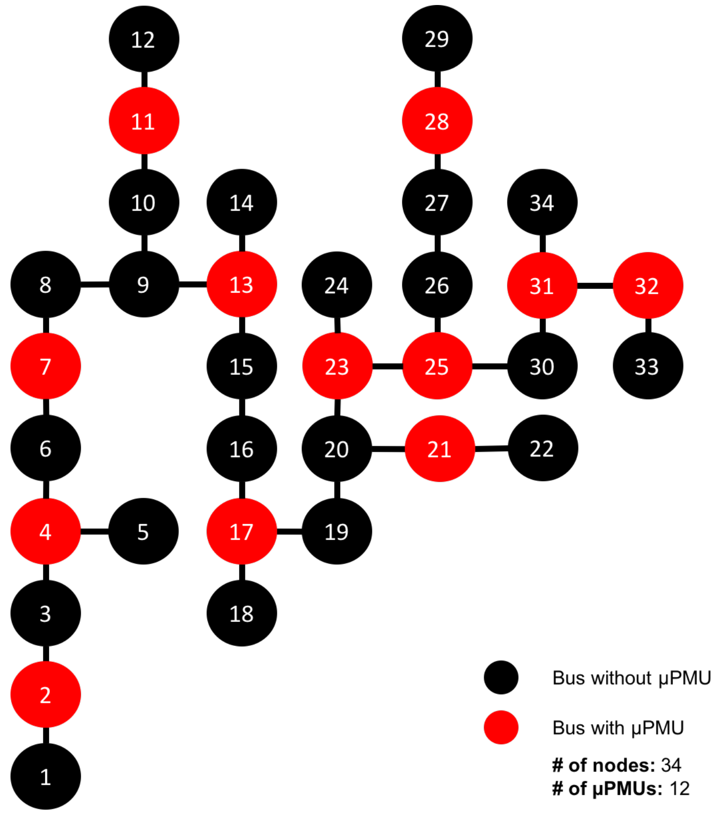

Figure 2.10: IEEE 34-node μPMU placement results from [31].

2.3.5 Greedy Algorithm

This algorithm always takes the best immediate solution [15], hence the term "greedy." It is generally fast, easy to implement and is adaptable [21]. The disadvantage to this method is it provides a good approximation, and not a minimal μPMU placement case. The advantage to this method, other than fast computation time, is it places μPMUs in such a way that high redundancy is achieved.

The two papers that used the greedy algorithm to place μPMUs were [32] and [24] and they were written by the same authors.

Additionally, these two papers and [27] were the only papers that considered network reconfiguration. They used ant colony optimization to solve the network reconfiguration problem and a greedy algorithm to optimally place μPMUs. Their goal was to see the effect of network reconfiguration on μPMU placement. They tested their algorithm on the base case, which assumed no switches were on the network, as well as different network configurations.

Their greedy algorithm can be described in the following steps:

1. Place a μPMU at a bus that can observe the highest number of buses that do not contain a μPMU.

2. Is complete observability achieved? If complete observability is not achieved, go to step 1. If complete observability is achieved, end program.

They tested their method on the IEEE 33-node feeder. Unfortunately, they did not achieve a minimal μPMU placement result for even the base configuration. Furthermore, the number of μPMUs needed increased for each different network reconfiguration. The placement result for the base case can be seen below in Figure 2.11. Their result would still be completely observable even if the μPMU at node 23 were removed. Note that due to the inherent nature of the greedy algorithm, no μPMUs were placed on end nodes.

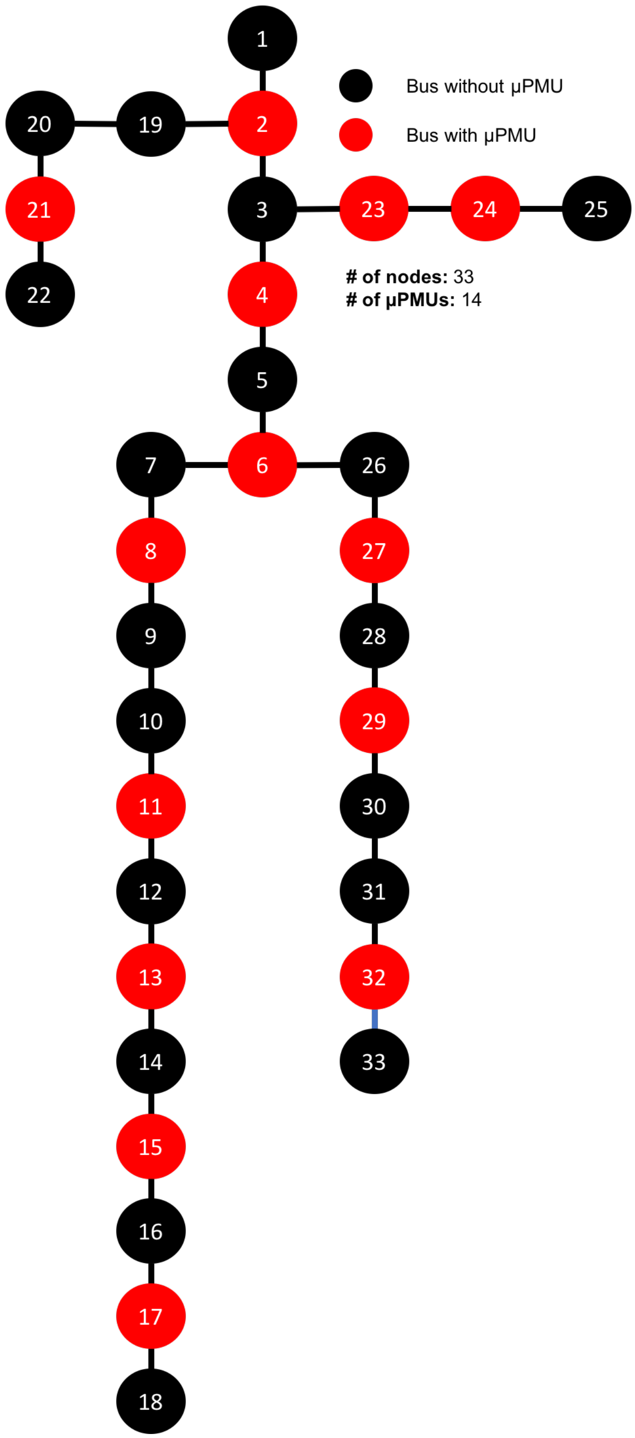

Figure 2.11: IEEE 33-node μPMU placement results from [32].

2.4 Summary

Integer programming methods have the advantage of being able to find the global optimum, which in this case refers to the minimal μPMU case. However, their high computation time and typically low redundancy solutions make them a less feasible choice to solve the optimal μPMU placement problem for distribution systems. On the other hand, exhaustive algorithms provide a high redundancy solution while still achieving a minimal μPMU case. However, their extensive computation time is simply not feasible for larger networks, even with some preprogramming steps.

Graph theory and greedy algorithms are both good candidates for solving the μPMU problem. However, they would need to be customized in order to provide or consistently provide the minimal μPMU solution.

Lastly, network reconfiguration was not considered in most papers, even though this is an important factor to consider since distribution networks can be reconfigured. The ones that did, vastly overcomplicated the problem and the results were solutions that did not provide the minimal μPMU case.

These findings are summarized in Table 2.1 below. The bolded cells depict a favorable characteristic.

Table 2.1: Binary Comparison Summary of Review Algorithms

Method	Minimum μPMU case?	μPMUs on end nodes?	High computation?	Consider network reconfiguration?
Integer Programming [19]	no	yes	yes	no
Integer Programming [27]	no	yes	yes	**yes**
Integer Programming [28]	no	yes	yes	no
Exhaustive Search [29], [13]	**yes**	**no**	yes	no
Simulated Annealing [29]	**yes**	**no**	yes	no
Graph Theory [31]	**yes**	**no**	**no**	no
Greedy Algorithm [32], [24]	no	**no**	**no**	**yes**

In conclusion, there has yet to be an algorithm that is simple to implement, have a computation time low enough that is feasible to use in the real world, is highly redundant, considers network reconfiguration and produces consistently minimal results.

Chapter 3

Proposed Hybrid Greedy μPMU Placement Algorithm

3.1 Overview

This chapter will explain the proposed placement algorithm, which is a hybrid greedy algorithm. In Section 3.2, the assumptions that were used in the placement algorithm are reviewed. In Section 3.3, an explanation of how network reconfiguration was considered is provided. Following this, Section 3.4 provides detail for the proposed algorithm. Finally, Section 3.5 summarizes the chapter.

3.2 Assumptions

The following assumptions were used to create the proposed algorithm. They are listed as follows:

1. Distribution networks are balanced. Therefore, only one phase was considered.

2. μPMUs have unlimited channels. That means if a μPMU were placed at a node, it would be able to measure the voltage phasor of that node and an unlimited number of current phasors incident to the node.

3. Line impedances are known. Knowing the line impedances allows for the calculation of surrounding node voltage values.

4. It costs the same to place a μPMU anywhere on the network.

5. Switches are ideal components. Therefore, if a switch was closed, the two nodes immediately before and after it would "merge" into one node.

These assumptions are consistent with the other works discussed in Chapter 2.

3.3 Network Reconfiguration

As explained in Section 2.2, network reconfiguration is a very important factor to consider. In order to take this factor into account, a straightforward solution is proposed. It was assumed that the worst-case scenario was in effect: all switches were "open". This way, even when the switches were closed, complete observability is still achieved. This idea is best illustrated with an example, which will be an expansion from the Network Reconfiguration section in Section 2.2.

Example 3.1

Figure 3.1 below depicts a 7-node feeder with one switch. Taking the worst-case scenario, this switch is assumed to be open. In order to achieve complete observability of this network, three μPMUs are needed. Since the switch is open, the only way to observe nodes 4 and 5 would be to place a μPMU at either node 4 or node 5. The best placement would be to place the μPMU at node 4 as a higher SORI value would be achieved when the switch is closed. The other two μPMUs can be placed at nodes 2 and 6 in order to achieve the minimal case with the highest redundancy. There are other placement options for those two μPMUs, however they would achieve a lower SORI value.

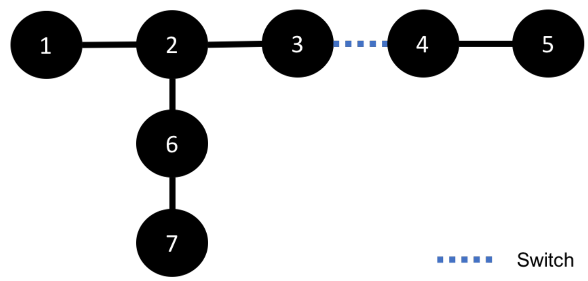

Figure 3.1: Simple 7-node network with a switch.

This placement can be seen below in Figure 3.2. Now, if the switch is open or closed, complete observability is achieved.

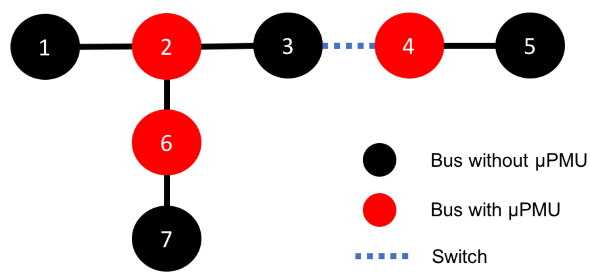

Figure 3.2: Simple 7-node network with a switch.

3.4 Hybrid Greedy μPMU Placement Algorithm

The goal of the proposed algorithm was to achieve the combined benefits of the previous literature which can be summarized as follows:

- Completely observed network with minimum or near minimum number of μPMUs
- Computationally inexpensive
- High SORI value
- Achieve complete observability under any configuration
- Simple input: connectivity matrix

Note that if the minimum number of μPMUs is not achieved for a particular network using the proposed algorithm, it is only off by a difference of one μPMU and the trade-off is a much higher reliability. This will be explained in Section 4.4.

The proposed algorithm is a hybrid greedy algorithm with one input: the connectivity matrix. This connectivity matrix is built with all switches set to the "open" position in order to take into account network reconfiguration. The greedy algorithm was chosen as it is computationally inexpensive, easily adaptable, and it inherently places μPMUs in such a way that a high SORI value is achieved. The algorithm outputs the number of μPMUs needed for a given network and on which nodes to place them.

This hybrid greedy algorithm is comprised of five parts: pre-greedy algorithm, greedy algorithm, configuration 1, configuration 2, and final solution.

3.4.1 Pre-greedy Algorithm

In general, greedy algorithms are a computationally inexpensive method to place μPMUs. However, due to the fact that distribution networks can have hundreds of thousands of buses, it is important to decrease the computation time as much as possible in order to find a solution in a reasonable amount of time for real-world networks. Therefore, a pre-greedy algorithm step was taken.

This pre-greedy algorithm step simply identifies end nodes and places μPMUs on the nodes connected to end nodes, similar to what was done in [29] and [31]. By placing μPMUs at the nodes connected to end nodes, a higher SORI value is achieved and the system is more reliable.

As the connectivity matrix is the sole input to the customized greedy algorithm, end nodes and nodes connected to end nodes must be identified from that matrix. This can be done in the following steps:

1. Find the columns in the connectivity matrix that have only two 1's in it.

2. Look at the rows of the columns from step 1.

3. Find the row number that is not the same as the column number and has a "1" located there. This row number will correspond to the node that is connected to the end node.

To illustrate these steps, an example is used.

Example 3.2

This example is based on a 7-node network as seen in Figure 3.3. It's corresponding connectivity matrix can be seen in (3.1).

In the first step, we need to identify the end nodes. If we look at the connectivity matrix, we can see the only columns that contain two 1's are columns 1, 5, and 7. This directly corresponds to the end nodes as they are numbered in Figure 3.3.

Next, the nodes connected to end nodes can be identified. Let us focus on the first end node, node 1. We will look at column 1 of (3.1) and isolate the rows that contain 1's. Now, we find the row number that is **not** the same as the column number, which is column 1, and has a "1" located there. This corresponds to row 2. Therefore, node 2 is connected to node 1 and node 1 is an end node. Now that we have identified this, we can place a μPMU on node 2.

These same steps can be done for end nodes 5 and 7 which would correspond to placing μPMUs at nodes 4 and 6.

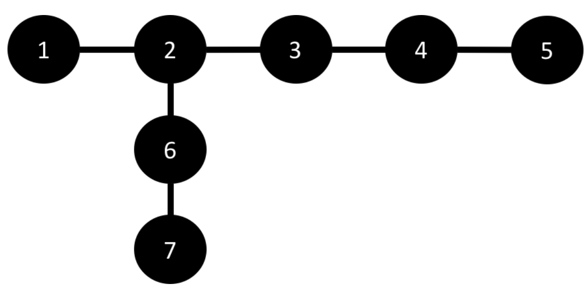

Figure 3.3: 7-node radial network.

$$
A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
\end{array}
\tag{3.1}
$$

The μPMUs that are placed during the pre-greedy algorithm portion will be **fixed** μPMUs and will not move during configuration 1 and 2 of the proposed algorithm. This is because the μPMUs need to be here in order to observe the end nodes.

3.4.2 Greedy Algorithm

Once the μPMUs have been placed on the nodes connected to the end nodes, the greedy algorithm portion runs. The steps to this part of the algorithm are summarized below and an example follows to further explain the steps.

1. Determine the number of nodes a μPMU could observe if it was placed there. If that node already has a μPMU there, or it is an end node, a "0" goes here.

2. Place the μPMU at the node that will observe the highest number of nodes.

3. Continue step 2 until complete observability has been achieved.

To understand these steps, a 9-node network has been created as seen in Figure 3.4. This network has no switches and contains three μPMUs that were placed from the pre-greedy algorithm part.

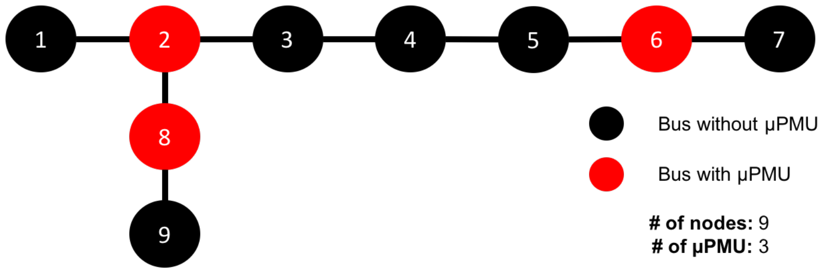

Figure 3.4: Simple 9-node radial network.

In step 1, the algorithm determines how many nodes a μPMU could observe if it were placed there. The result is a vector of values as given by (3.2). Note that the column numbers refer to node numbers, similar to the connectivity matrix. For example, if a μPMU were to be placed at node 3, it would be able to observe three nodes: node 2, node 3, and node 4.

$$O = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 3 & 3 & 3 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \tag{3.2}$$

In step 2, the μPMU is placed at the node that can observe the highest number of nodes and does not currently have a μPMU on it. According to this vector, a μPMU should be placed at either node 3, node 4 or node 5. Due to the way the algorithm runs, a μPMU will be placed at the first node it “encounters” which would be node 3.

In step 3, the algorithm will check to see if complete observability has been achieved. To do this, another matrix is used. This matrix can be defined in (3.3) below. Essentially, if a μPMU is located on a node, that column of the B matrix will be identical to the column in the connectivity matrix. Otherwise, the column will only have “0’s” in it.

$$B(:,j) = \begin{cases} A(:,j), & \text{if } \mu\text{PMU is placed at node j} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The specific “B” matrix for this example, after the μPMU has been placed at node 3, can be seen in (3.4).

$$
B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}
\tag{3.4}
$$

In order to see if complete observability has been achieved, the rows are checked one by one. If there is at least one "1" in every row, the system is completely observed. In this example, complete observability is achieved after placing a μPMU at node 3

Note that the same result would have been achieved if a μPMU were placed at nodes 4 or 5 instead of node 3. Complete observability would have been achieved for all three cases and the SORI value would have also been the same, due to each node being able to observe 3 nodes if a μPMU were placed there.

Once complete observability has been achieved, the greedy algorithm part of the proposed algorithm finishes. The μPMUs placed during the greedy algorithm stage are **unfixed**. This means that they may be shifted during the configuration 1 and 2 portions of this algorithm.

After the greedy algorithm step, there can be more μPMUs placed than are needed. The table below compares how many μPMUs are placed after the greedy algorithm part finishes with the minimal number case. The minimal number case was found using an integer programming method.

Table 3.1: Greedy Algorithm μPMU Placement

IEEE test feeder	# of μPMUs (greedy)	# of μPMUs (integer)
13-node	5	5
33-node	12	11
34-node	13	12
37-node	12	12
123-node	47	47

These values are very close. However, some customization was done in order to closer match the minimum case for all networks as well as make the algorithm more robust.

3.4.3 Configuration 1

This part of the algorithm, called configuration 1, starts with the placement results from the pre-greedy and greedy algorithm parts. The next steps can be described as follows:

1. Remove an unfixed μPMU
2. If complete observability is still achieved, save this configuration. If not, place the μPMU back.
3. Continue steps 1 and 2 until all unfixed μPMUs have been systematically removed and tested for complete observability.

Referring back to the example in Figure 3.4, the only unfixed μPMU is the one that was placed at node 3 during the greedy algorithm part. In this case, when removed, complete observability would not occur. Therefore, it would be placed back and this part of the algorithm would end.

3.4.4 Configuration 2

This part of the algorithm is called configuration 2 and it also takes the results from the pre-greedy and greedy algorithm parts. The steps for this part of the algorithm can be outlined as follows:

1. Place a μPMU on a node that would observe the largest number of nodes, is not an end node, and does not already contain a μPMU. The μPMU placed here will be referred to as "μPMU A".

2. Remove the μPMU that was observing the node that now has "μPMU A" on it. Note that the μPMU must be unfixed in order to remove it. If the only μPMU(s) observing that node is/are fixed, none will be removed.

3. Remove a different unfixed μPMU and check for observability.

4. If this configuration achieves complete observability, then it is saved and the algorithm goes back to step 1.

5. If this configuration does not achieve complete observability, place the unfixed μPMU back and go to step 3. This will continue until all μPMUs have been systematically removed and the system checked for complete observability.

6. If all free nodes (that are not end nodes) have had a μPMU placed on it (from step 1) and other unfixed μPMUs systematically removed and checked for observability, the algorithm ends.

Note that configuration 2 is very similar to configuration 1, the only difference is it tries shifting an unfixed μPMU before systematically trying to remove them.

To explain these steps further, an example is provided below.

Example 3.3

The IEEE 34-node network is explored in this example. After the pre-greedy and greedy algorithm part of the algorithm runs, the placement result can be seen in Figure 3.5 below. Thirteen μPMUs were placed, however, we know from the integer programming result shown in Table 3.1 that only twelve μPMUs are needed for complete observability.

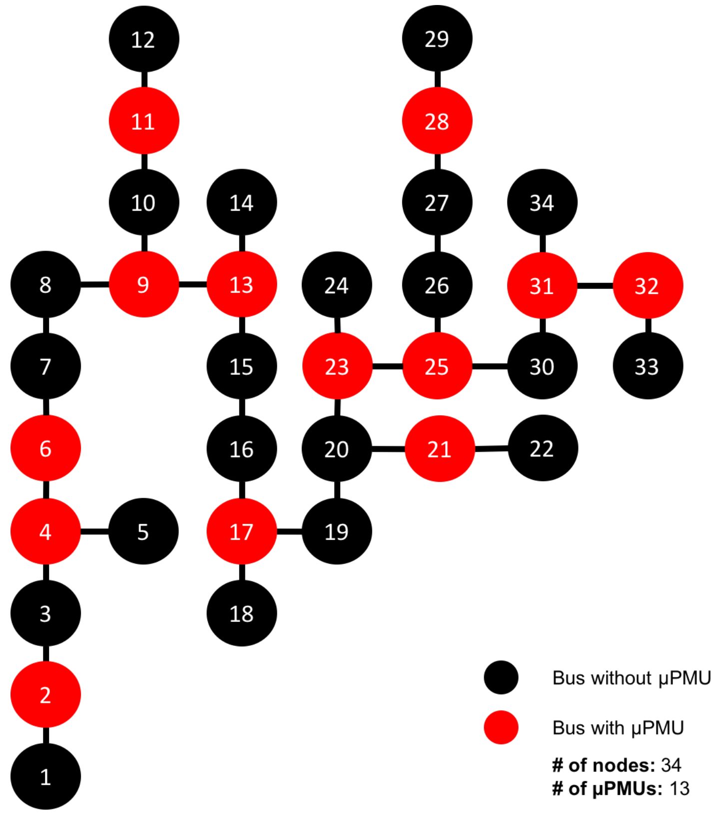

Figure 3.5: IEEE 34-node μPMU placement after greedy algorithm part runs.

For step 1 to occur, it would look at the "O" vector for this network as seen in (3.5) and (3.6). Note that the vector was split into two equations due to its length for the purposes of this thesis. Again, the column numbers refer to the node numbers. Then, during step 1, it would place a μPMU at a free node that would be able to observe the highest number of nodes **and** the μPMU(s) observing that node is/are not fixed μPMU(s).

$$O_1 = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 3 & 0 & 0 & 0 & 3 & 3 & 0 & 3 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 \end{pmatrix} \end{array} \quad (3.5)$$

$$O_2 = \begin{array}{c} \begin{matrix} 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 & 33 & 34 \end{matrix} \\ \begin{pmatrix} 3 & 4 & 0 & 0 & 0 & 0 & 0 & 3 & 3 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \quad (3.6)$$

In this case, node 20 has the highest number of nodes it could observe if a μPMU were placed there. However, from Figure 3.5, it can be seen that the two μPMUs observing node 20 are fixed μPMUs (because they are connected to end nodes). Therefore, a μPMU will not be placed here. Next, it would look at the second-best case, a node that could observe three nodes if a μPMU were placed there. The first node it would check is node 3. However, similar to node 20, it is only being observed by fixed μPMUs. Therefore, it would look to the next node which is node 7. In this case, a μPMU will be placed here as the μPMU observing node 7 is not fixed.

In step 2, the μPMU that was observing node 7 will be removed. Referring to Figure 3.5, it can be seen that the μPMU that was observing node 7 is located at node 6. Note that this is effectively shifting the μPMU from node 6 to node 7.

Now, starting in step 3, an unfixed μPMU will be removed and observability will be tested. The first unfixed μPMU (that is not the one we just shifted), is the μPMU located at node 9. Therefore, this μPMU will be removed and the observability of this placement will be tested. After checking the observability, it is found that complete observability is still achieved when the μPMU from node 9 is removed.

Therefore, in step 4, this configuration will be saved, and the algorithm will try to repeat to see if even more μPMUs can be removed. In this test case, it is not possible as this is the minimum μPMU case.

In step 5, if complete observability was not achieved, the μPMU would be placed back onto node 9, and the next unfixed μPMU would be removed.

In step 6, the algorithm would look to see if every unfixed μPMU has been shifted. So far, only one μPMU has been shifted. Therefore, the algorithm will be repeated from step 1 until every unfixed μPMU has been effectively shifted.

3.4.5 Final Solution

Now, there are two configurations which have been saved from the configuration 1 and configuration 2 parts of this algorithm. Therefore, the last part of this algorithm will choose the configuration that has the smaller number of μPMUs.

The algorithm can be summarized in Figure 3.5 below.

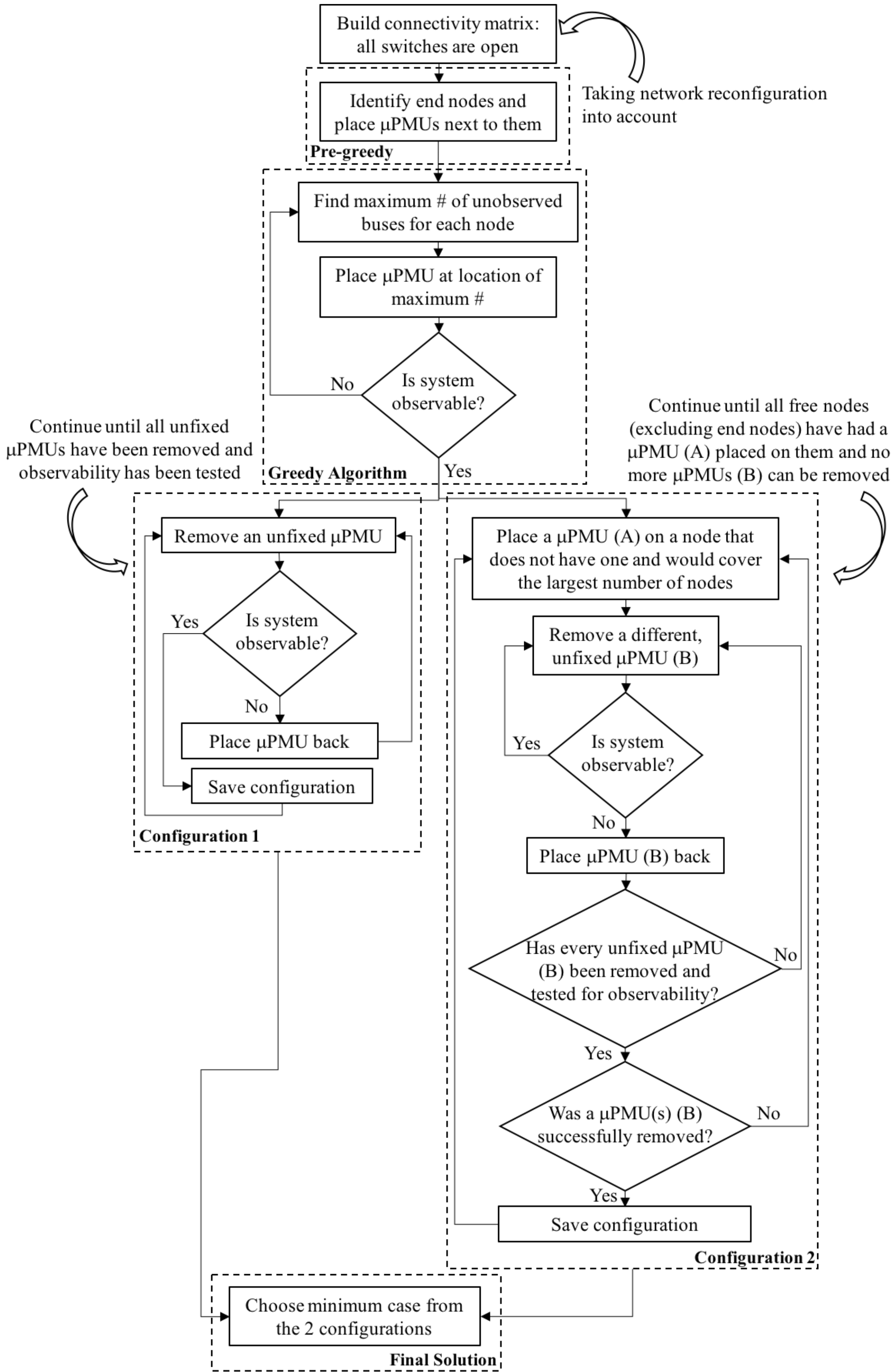

Figure 3.6: Flow diagram of the hybrid greedy algorithm.

3.5 Summary

The proposed hybrid greedy algorithm optimally places μPMUs on a given distribution network while considering network reconfiguration. It is computationally fast and it is easy to implement as the only input needed is the connectivity matrix. This algorithm inherently places μPMUs in such a way that a high SORI value is achieved because it focuses on placing μPMUs at the buses that can observe the highest number of nodes.

Chapter 4

Results and Analysis

4.1 Overview

In this chapter, the results of the proposed algorithm will be reviewed. First, in Section 4.2, the placement results from the proposed algorithm are shown and explained. Next, in Section 4.3, the results of the proposed algorithm are compared with the results from the previous works from Section 2.3. In Section 4.4, the proposed placement's reliability was tested in the event that 1, 2, and 3 μPMUs failed. Finally, Section 4.5 summarizes the chapter and highlights the key findings.

4.2 Placement Results

This section reviews the μPMU placement from the proposed algorithm for the following IEEE distribution test feeders: 13-node, 33-node, 34-node, 37-node and 123-node.

4.2.1 IEEE 13-node network

The IEEE 13-node feeder was the smallest feeder tested. It is comprised of thirteen buses and one switch. Since this algorithm takes into account network reconfiguration, the switch was assumed to be open. In order to have complete observability of the system under any configuration, six μPMUs would be needed, as seen below in Figure 4.1.

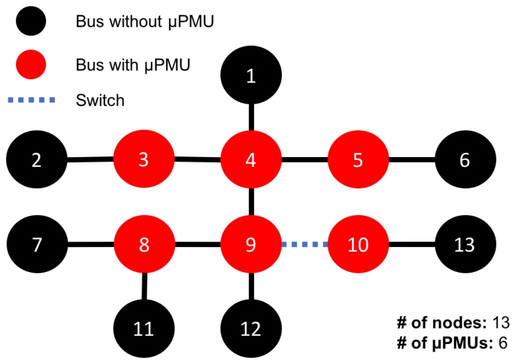

Figure 4.1: Proposed hybrid greedy μPMU placement on IEEE 13-node feeder with switch open.

Previous literary works, explained in Section 2.3, assumed the switch was closed, which is considered normal operation. If the switch is closed, only five μPMUs are needed which can be seen in Figure 4.2 below.

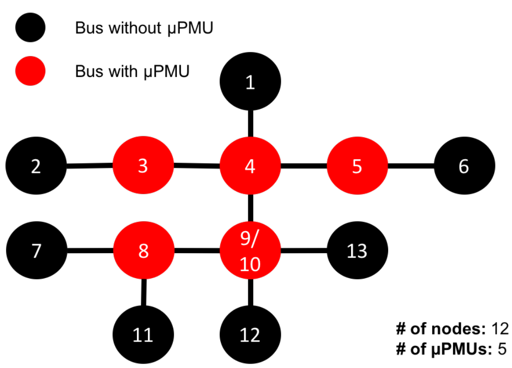

Figure 4.2: Proposed hybrid greedy μPMU placement on IEEE 13-node feeder with switch closed.

4.2.2 IEEE 33-node network

The IEEE 33-node feeder is comprised of 33 nodes and no switches. The hybrid greedy algorithm placed twelve μPMUs in order to achieve complete observability, as seen below in Figure 4.3.

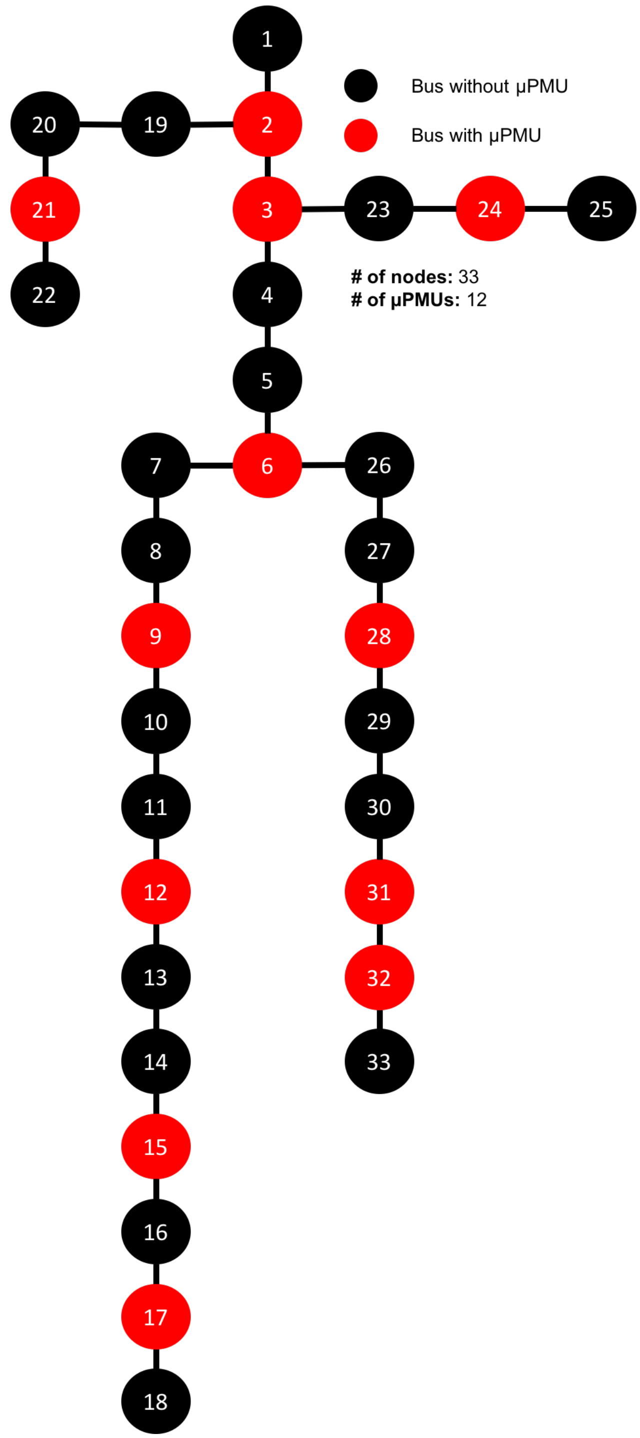

Figure 4.3: Proposed hybrid greedy μPMU placement for IEEE 33-node feeder.

However, according to integer programming, only 11 μPMUs are needed for the minimal case. Integer programming would place the μPMUs as seen in Figure 4.4 below.

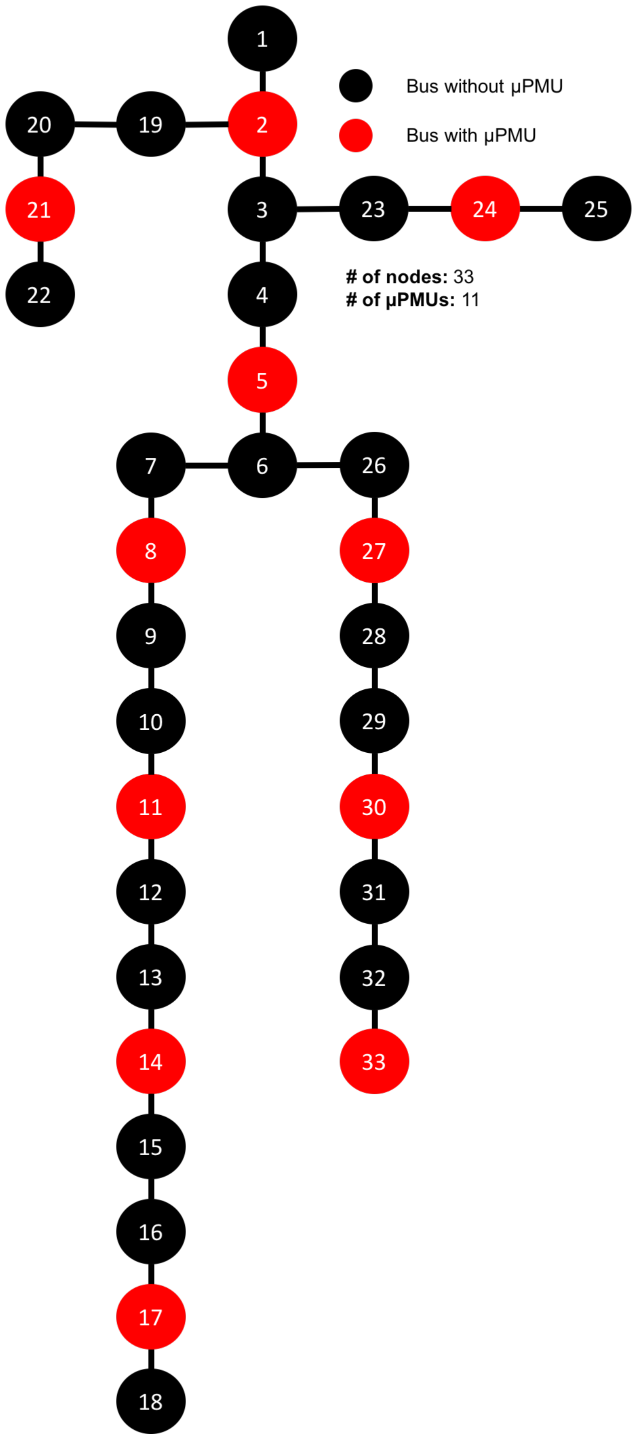

Figure 4.4: Integer programming μPMU placement for IEEE 33-node feeder.

The proposed algorithm was unable to achieve this minimal μPMU placement configuration due to its greedy aspect. During the greedy algorithm part of the proposed algorithm, the first μPMU will be placed on node 6 as it will be able to observe four nodes. After the μPMU on node

6 is placed, the rest of the μPMUs are placed accordingly in order to achieve complete observability. Now, in order to achieve the minimal case seen in Figure 4.4, multiple μPMUs would have to be shifted simultaneously before the excess μPMU can be removed. Currently, the proposed algorithm only shifts one μPMU at a time during configuration 2. However, although the minimum case is not achieved, there is a tradeoff which makes the proposed placement configuration a more desirable option. This is explained in Section 4.4.

4.2.3 IEEE 34-node network

This IEEE network has 34 nodes and no switches. The placement for this feeder can be seen below in Figure 4.5. Only twelve μPMUs were placed for complete observability, which is the minimal case.

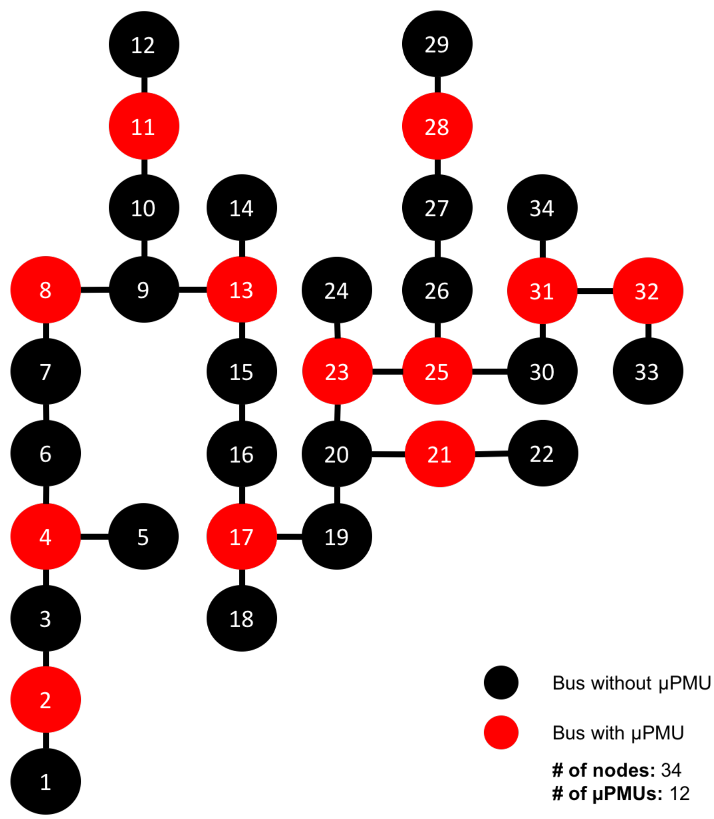

Figure 4.5: Proposed hybrid greedy μPMU placement for IEEE 34-node feeder.

4.2.4 IEEE 37-node network

The IEEE 37-node network is composed of 37 nodes and no switches. The μPMU placement from the hybrid greedy algorithm can be seen in Figure 4.6 below. Again, it was found that only twelve μPMUs were needed for complete observability of the system, which is consistent with the results from the integer programming method.

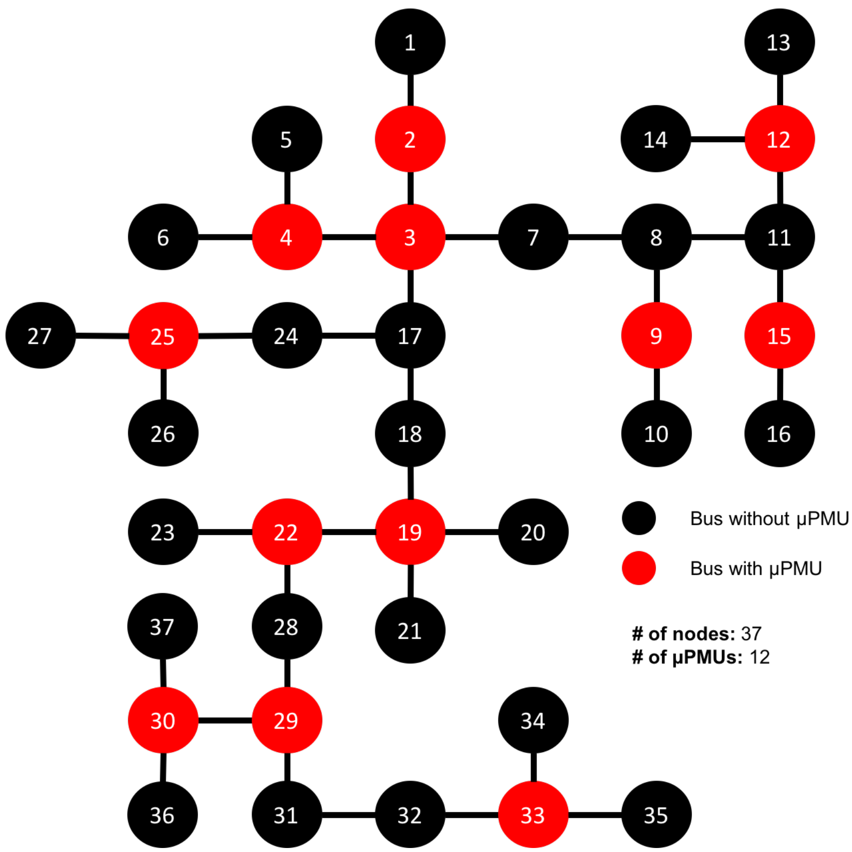

Figure 4.6: Proposed hybrid greedy μPMU placement for IEEE 37-node feeder.

4.2.5 IEEE 123-node network

This network was both the largest and most complex network tested. It is comprised of 123 nodes and twelve switches. It takes 47 μPMUs to achieve complete observability when taking into account network reconfiguration, as seen in Figure 4.5. Note that some nodes located on the "outside" of the network, nodes 124 to 129, would not be observed if the switches were open which was assumed to be okay. Again, this assumption is consistent with the previous works. When the switches are closed, nodes 124 to 129 are observed.

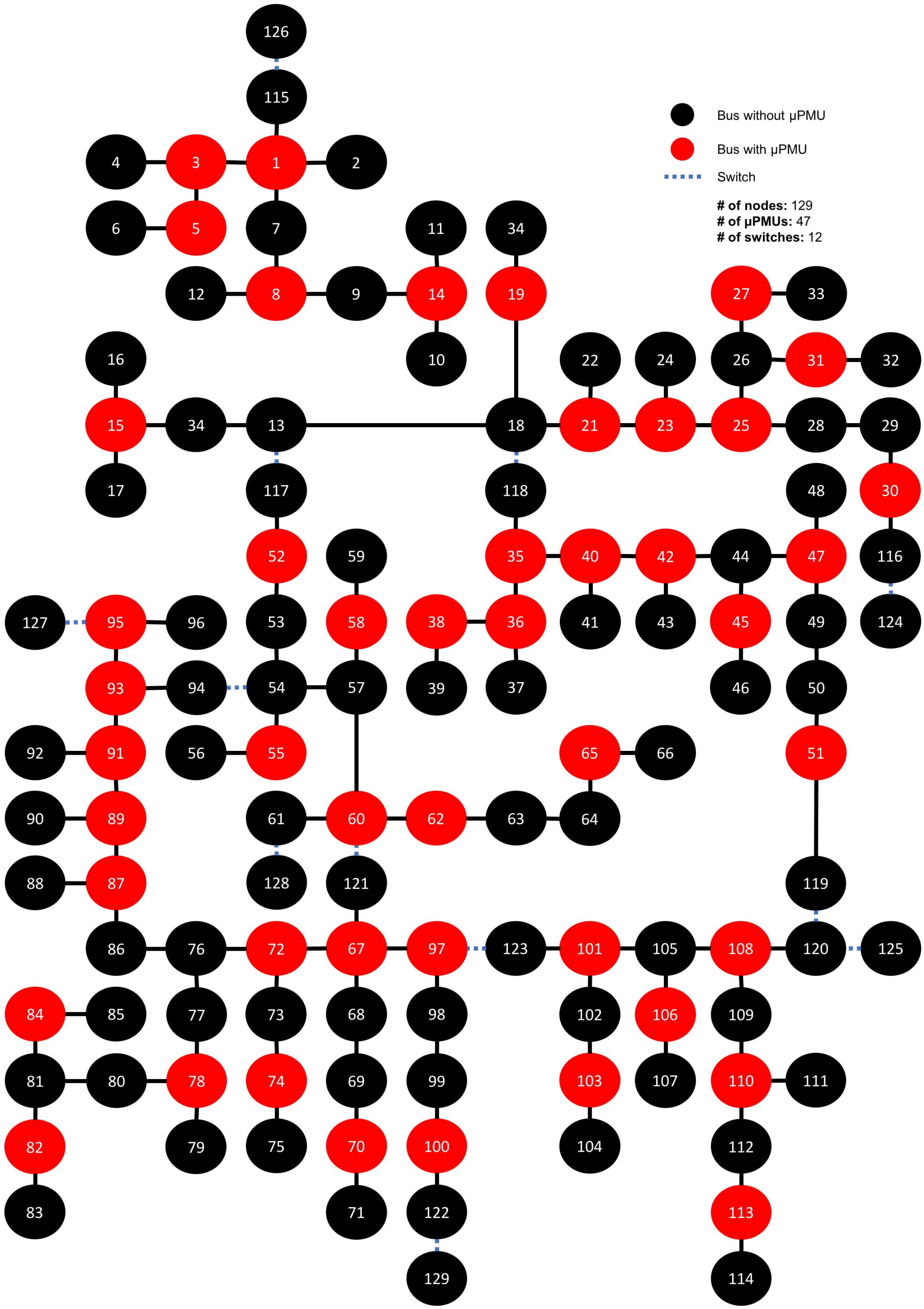

Figure 4.7: Proposed hybrid greedy μPMU placement for IEEE 123-node feeder.

4.2.6 Results Summary

The results from the proposed hybrid greedy μPMU placement algorithm are summarized in Table 4.1 below. Note that this algorithm is extremely fast, even for the larger 123-node network. The algorithm was run with MATLAB version R2017a on a MacBook Pro which has an Intel Core i5 processor with a speed of 2.3 GHz.

Table 4.1: Results of the Proposed Hybrid Greedy μPMU Placement Algorithm

Test Feeder	Number of μPMUs	Computation Time (s)	SORI
13-node	5	0.0003	20
33-node	12	0.009	39
34-node	12	0.002	42
37-node	12	0.002	47
123-node	47	0.04	168

4.3 Comparisons to Past Works

The proposed algorithm and previous works were compared on three factors. The first factor was the number of μPMUs needed for complete observability of the distribution networks tested. This can be seen in Table 4.2 below. The values on the table were taken from the results section in the sources listed. Each paper did not test for every network, therefore there is not a value in every space. The proposed algorithm placed either the same, or lower, number of μPMUs for the same networks.

Table 4.2: Number of μPMUs placed

Method	13-node	33-node	34-node	37-node	123-node
Integer Programming [19]	6		13		49
Integer Programming [27]		17			
Integer Programming [28]			13		47
Exhaustive Search Method [29], [13]	5		12	12	
Simulated Annealing [29]	5		12	12	
Graph Theory [31]	5		12	12	
Greedy Algorithm [32], [24]		14			
Proposed Hybrid Greedy Algorithm	**5**	**12**	**12**	**12**	**47**

The second factor that was compared was the computation time, which can be seen in Table 4.3 below. Again, these values were taken from the results section of the respective papers. Some of the previous works did not include the computation time, therefore it is not listed on the table. Note that this is simply for interest, it is hard to directly compare computation times as different computers were used as well as different MATLAB versions. However, it can be seen that the proposed algorithm was much faster than the other works listed.

Table 4.3: Computation Time (s) for μPMU Placement

Method	13-node	33-node	34-node	37-node	123-node
Integer Programming [19]	0.0354		0.0413		0.208
Exhaustive Search Method [29]	0.016		0.295	0.303	
Exhaustive Search Method [13]	0.09		0.35	0.34	
Simulated Annealing [29]	0.035		3.543	4.418	
Proposed Hybrid Greedy Algorithm	**0.0003**	**0.009**	**0.002**	**0.002**	**0.04**

The last factor compared was the SORI value, seen in table Table 4.4 below. This value is very important as a higher SORI value correlates to a more reliable system. Only two papers included the SORI calculation in their results, [29] and [13]. The rest of the values were calculated using a MATLAB program with their recorded placement results. Again, it can be seen that the SORI values from the proposed algorithm either match or exceed the ones from other works, excluding the ones for the 33-node feeder. The previous works who tested their placement method on the IEEE 33-node feeder, placed a much higher number of μPMUs. Therefore, they would be able to achieve a higher SORI value.

Table 4.4: SORI Values

Method	13-node	33-node	34-node	37-node	123-node
Integer Programming [19]	16		40		147
Integer Programming [27]		51			
Integer Programming [28]			36		135
Exhaustive Search Method [29], [13]	20		42	47	
Simulated Annealing [29]	20		42	47	
Graph Theory [31]	20		42	47	
Greedy Algorithm [32], [24]		44			
Proposed Hybrid Greedy Algorithm	**20**	**39**	**42**	**47**	**168**

4.4 Redundancy Analysis

Two different analyses were done to see the effects of one, two, and three μPMU failures.

4.4.1 Percent Coverage

The first analysis was done to view how much of the network would be covered, as a percentage, in three different cases: one μPMU fails, two μPMUs fail, three μPMUs fail. The steps of this calculation are summarized below:

1. Find every possible combination of μPMU or μPMUs failure(s).

2. Calculate how many nodes would be unobserved for each scenario.

3. Calculate the percent coverage for the best and worst-case scenarios.

To further explain this, a walkthrough will be done for the IEEE 13-node network. For a visual, we will reuse Figure 4.2, which can now be viewed in Figure 4.8 below.

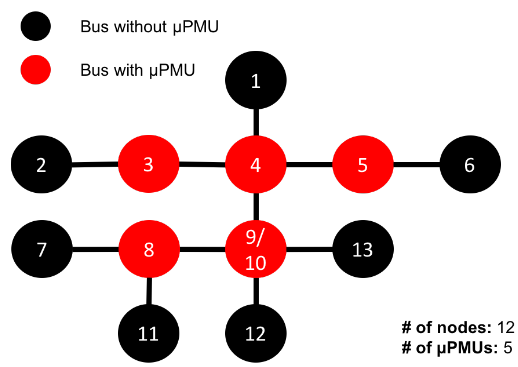

Figure 4.8: Proposed hybrid greedy μPMU placement on IEEE 13-node feeder with switch closed.

In step one, we will find every possible combination for one μPMU failure. The total number of combinations can be calculated using (2.10). In the case of one μPMU failure, there are only five possible scenarios:

1. μPMU at node 3 fails

2. μPMU at node 4 fails

3. μPMU at node 5 fails

4. μPMU at node 8 fails

5. μPMU at node 9 fails

In step two, we will see how many nodes become unobserved if each scenario occurs. For example, if the μPMU at node 3 were to fail, only node 2 becomes unobserved. These results can be seen in (4.1) below. Note the column numbers refer to the node that the μPMU failed on. To summarize, only one node becomes unobserved if the μPMU at node 3, 4, or 5 were to fail and two nodes become unobserved if the μPMU at node 8 or 9 were to fail.

$$\begin{array}{cc} & \begin{array}{ccccc} 3 & 4 & 5 & 8 & 9 \end{array} \\ U = & \begin{pmatrix} 1 & 1 & 1 & 2 & 2 \end{pmatrix} \end{array} \tag{4.1}$$

In step three, the percent coverage is calculated. Referring to (4.1), we can see that the best-case scenario occurs when only one node is unobserved while the worst-case scenario occurs when two nodes become unobserved. The percent coverage can be calculated using (4.2) below, where n represents the total number of nodes in the network and U represents the number of unobserved nodes.

$$Coverage(\%) = \frac{n-U}{n} * 100\% \tag{4.2}$$

The results for all networks and cases are summarized in Table 4.5 below. Generally speaking, as the number of nodes in the system increases, so does the percent coverage. This makes sense because of the radial structure of distribution systems and the hybrid greedy process: if any μPMU were to fail, the total coverage would be much higher on a higher node system. Therefore, if a large distribution network was considered, it would be assumed that these results would be further improved.

Table 4.5: Percent observability coverage in the event of μPMU failure

	1 μPMU fail		2 μPMU fail		3 μPMU fail	
	Best Case	Worst Case	Best Case	Worst Case	Best Case	Worst Case
13-node	91.76	83.33	83.33	58.33	67.67	41.67
33-node	96.67	87.88	93.94	78.79	87.88	69.70
34-node	97.06	88.24	94.12	79.41	88.24	70.59
37-node	97.30	91.89	97.30	81.08	94.59	72.97
123-node	99.19	95.93	98.37	92.68	97.56	90.24

4.4.2 Percent Unobserved Nodes

The second analysis done is related to the first. Essentially, if one μPMU were to fail, what is the percent chance that only one node is unobserved? Multiple nodes? This was tested for the same three cases as the first analysis: one μPMU fails, two μPMUs fail, three μPMUs fail. This calculation can be summarized in one step below:

1. From the U vector, calculate the percent chance that 1, 2, 3, and 4+ nodes will be unobserved for each case: one μPMU fails, two μPMUs fail, three μPMUs fail.

To illustrate this, we will again look at the 13-node network. The U vector was already found for the case of one μPMU failure, seen in (4.1). In that U vector, there is a 3/5 chance that only one node will be unobserved while there is a 2/5 chance that two nodes will be unobserved. Converting this value to a percent, that translates to 60% and 40% respectively. The same process was used for each network and the three different cases.

The three tables below summarize these results.

Table 4.6: Percent probability of unobserved nodes in the case of 1 μPMU failure

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
13-node	60	40	0	0
33-node	25	33.33	33.33	8.33
34-node	33.33	25	33.33	8.33
37-node	25	33.33	25	16.67
123-node	38.29	40.43	17.02	4.26

Table 4.7: Percent probability of unobserved nodes in the case of 2 μPMU failures

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
13-node	0	10	80	10
33-node	0	3.03	15.15	81.82
34-node	0	4.55	18.18	77.27
37-node	4.55	9.09	21.21	65.15
123-node	0	13.6	31.36	55.04

Table 4.8: Percent probability of unobserved nodes in the case of 3 μPMU failures

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
13-node	0	0	0	100
33-node	0	0	0	100
34-node	0	0	0	100
37-node	0	0.91	5	94.09
123-node	0	0	4.43	95.57

It can be seen that these numbers are not predictable because it is highly dependent on the specific topology of each distribution network as well as where the μPMUs are placed. This further illustrates why a high SORI value is extremely important.

4.4.3 Special IEEE 33-node Case

In Section 4.2, the placement results for the IEEE 33-node feeder were proven to be non-minimal. One extra μPMU was placed using the proposed algorithm when compared to the minimum case. However, as previously stated, there is a tradeoff that makes this placement configuration preferable. These benefits will be illustrated by comparing the redundancy analysis between the proposed algorithm and the minimum case. Note the SORI value for the proposed algorithm is 39 and the minimum case is 34.

Percent Coverage

First, the percent coverage was analyzed and compared between the proposed algorithm and the minimum case. This can be seen in Table 4.9 below.

Table 4.9: Percent observability coverage for the 33-node feeder in the event of μPMU failure

	1 μPMU fail		2 μPMU fail		3 μPMU fail	
	Best Case	Worst Case	Best Case	Worst Case	Best Case	Worst Case
Proposed	96.97	87.88	93.94	78.79	87.88	69.70
Minimum	93.94	87.88	84.85	78.79	75.76	69.70

The worst-case percent coverage is identical between the two different placement configurations. However, the best-case scenario is worse for the minimum case and decreases rapidly from the first μPMU fail to three μPMU failures as compared to the proposed case.

Percent Unobserved Nodes

To further illustrate the benefits of the proposed algorithm over the minimal case, an analysis was done to look at the percentage of unobserved nodes. Firstly, the case of one μPMU failure was tested and the results of this analysis can be seen in Table 4.10 below.

Table 4.10: Percent probability of unobserved nodes in the case of 1 μPMU failure

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
Proposed	25	33.33	33.33	8.33
Minimum	0	18.18	72.73	9.09

The different distribution of these percentages clearly highlights how the proposed algorithm is much more reliable than the minimum case. In the event of one μPMU failure, there is a 0% chance of only one node being unobserved for the minimum case while the proposed algorithm has a 25% chance of that occurring. To summarize the minimum case, in the event that one μPMU fails, there is a high probability of 3 nodes being unobserved. Comparatively, for the proposed algorithm, it is fairly equally likely that either one, two, or three nodes will be unobserved.

The case of two μPMU failures can be seen in Table 4.11.

Table 4.11: Percent probability of unobserved nodes in the case of 1 μPMU failure

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
Proposed	0	3.03	15.15	81.82
Minimum	0	0	0	100

To summarize the minimum placement configuration, in the event that two μPMUs fail, at least four nodes will be unobserved. Comparatively for the proposed algorithm, there is a high probability of at least four nodes being unobserved. However, there is still a chance that only two or three nodes will be unobserved.

The case for three μPMU failures can be seen in Table 4.12, however it will not be explained as it is redundant.

Table 4.12: Percent probability of unobserved nodes in the case of 1 μPMU failure

	Number of Unobserved Nodes			
	1 node	2 nodes	3 nodes	4+ nodes
Proposed	0	0	0	100
Minimum	0	0	0	100

4.5 Summary

The proposed hybrid greedy algorithm places a minimum or near minimum number of μPMUs for a given distribution network with a low computation time. It provides either the same or better results than those in previous literary works in terms of computation time, SORI value and number of μPMUs needed.

Due to the inherent greedy nature of the algorithm, it automatically places μPMUs in such a way that a high SORI value is achieved, which leads to a highly reliable system. This was analyzed by testing the percent coverage of the system and how many nodes would be unobserved if one, two, or three μPMUs failed.

Chapter 5

Conclusion

5.1 Overview

This chapter will summarize the thesis in Section 5.2 and talk about future work in this area in Section 5.3.

5.2 Summary

The proposed greedy algorithm is a very computationally inexpensive method to place μPMUs on distribution systems. Computation speed is extremely important as real-world distribution networks can be comprised of hundreds of thousands of buses. The greedy aspects of the algorithm inherently place μPMUs in such a way that a high SORI value is achieved, meaning the system is able to observe most nodes even in the event of a μPMU failure. Lastly, the proposed algorithm was able to achieve the minimal case for all networks except one, the IEEE 33-node network. For this particular network, only one extra μPMU was place when compared to the minimum case and the trade-off was a much higher reliability. Lastly, the customized greedy algorithm outperformed the proposed algorithms in previous literature.

5.3 Future Work

There are two main areas that can be focused on for future work on the distribution system. The first would continue research in the area of μPMU placement for complete observability and the second research area would be in μPMU placement for incomplete observability.

5.3.1 Complete Observability

To continue research in this area, a more complex μPMU failure analysis could be done. A goal for this analysis would determine a target SORI value for any given distribution system. Additionally, a cost analysis could be explored. This analysis would compare the cost of placing μPMUs in different locations as well as the total cost for different placement configurations. Lastly, future work in this area should consider unbalanced networks and limited μPMU channels.

5.3.2 Incomplete Observability

Future work in this area would research μPMU placement for incomplete observability. For incomplete observability, state estimation would need to be implemented in order to estimate the values that are not being observed. Therefore, work could be done to develop a highly accurate state estimation technique. Then, one could see what happens to the accuracy if there is a μPMU failure or other kind of failure. One could also determine how many μPMUs one would need in order to achieve a specific accuracy.

Bibliography

[1] "Paris agreement," 2015. [Online]. Available: http://unfccc.int/files/home/application/pdf/paris_agreement.pdf

[2] "Canada's 2017 nationally determined contribution submission to the united nations framework convention on climate change," May 2017. [Online]. Available: https://www.canada.ca/en/envhttp://climateactiontracker.org/countries/canada.html

[3] (2018) Energy and greenhouse gas emissions (ghgs). [Online]. Available: https://www.nrcan.gc.ca/energy/facts/energy-ghgs/20063

[4] J. Casazza, *Understanding electric power systems : an overview of the technology and the marketplac*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2003.

[5] D. Macii, G. Barchi, and L. Schenato, "On the role of phasor measurement units for distribution system state estimation," in *2014 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems Proceedings*, Sept 2014, pp. 1–6.

[6] K. K. More and H. T. Jadhav, "A literature review on optimal placement of phasor measurement units," in *2013 International Conference on Power, Energy and Control (ICPEC)*, Feb 2013, pp. 220–224.

[7] K. TWEED, "The eastern us grid can handle more renewables than you think," August 2016.

[8] K. Zhu, L. Nordstrom, and L. Ekstam, "Application and analysis of optimum pmu placement methods with application to state estimation accuracy," in *2009 IEEE Power Energy Society General Meeting*, July 2009, pp. 1–7.

[9] X. Tai, D. Marelli, E. Rohr, and M. Fu, "Optimal pmu placement for power system state estimation with random component outages," *International Journal of Electrical Power & Energy Systems*, vol. 51, pp. 35 – 42, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061513000677

[10] X. Dongjie, H. Renmu, W. Peng, and X. Tao, "Comparison of several pmu placement algorithms for state estimation," in *2004 Eighth IEE International Conference on Developments in Power System Protection*, vol. 1, April 2004, pp. 32–35 Vol.1.

[11] M. Zhou, V. Centeno, A. G. Phadke, Y. Hu, D. Novosel, and H. A. R. Volskis, "A preprocessing method for effective pmu placement studies," pp. 2862 – 2867, 05 2008.

[12] A. G. Phadke, "Modern wide area measurement systems, an overview," in *8th International Conference on Advances in Power System Control, Operation and Management (APSCOM 2009)*, Nov 2009, pp. 1–1.

[13] X. Chen, T. Chen, K. J. Tseng, Y. Sun, and G. Amaratunga, "Hybrid approach based on global search algorithm for optimal placement of μpmu in distribution networks," in *2016 IEEE Innovative Smart Grid Technologies - Asia (ISGT-Asia)*, Nov 2016, pp. 559–563.

[14] (2018, April). [Online]. Available: https://www.powerstandards.com/product/micropmu/highlights/

[15] N. M. Manousakis, G. N. Korres, and P. S. Georgilakis, "Optimal placement of phasor measurement units: A literature review," in *2011 16th International Conference on Intelligent System Applications to Power Systems*, Sept 2011, pp. 1–6.

[16] M. Wache and D. C. Murray, "Application of synchrophasor measurements for distribution networks," in *2011 IEEE Power and Energy Society General Meeting*, July 2011, pp. 1–4.

[17] M. B. Mohammadi, R. A. Hooshmand, and F. H. Fesharaki, "A new approach for optimal placement of pmus and their required communication infrastructure in order to minimize the cost of the wams," *IEEE Transactions on Smart Grid*, vol. 7, no. 1, pp. 84–93, Jan 2016.

[18] H. G. Abood and V. Sreeram, "A review on phasor measurement units placement for state estimation studies," in *2014 Australasian Universities Power Engineering Conference (AUPEC)*, Sept 2014, pp. 1–6.

[19] E. Jamil, M. Rihan, and M. A. Anees, "Towards optimal placement of phasor measurement units for smart distribution systems," in *2014 6th IEEE Power India International Conference (PIICON)*, Dec 2014, pp. 1–6.

[20] R. Sodhi, S. C. Srivastava, and S. N. Singh, "Multi-criteria decision-making approach for multistage optimal placement of phasor measurement units," *IET Generation, Transmission Distribution*, vol. 5, no. 2, pp. 181–190, February 2011.

[21] A. Abdelaziz, A. M Ibrahim, and R. Salem, "Optimal pmu placement for complete observability using heuristic methods," in *the Fifteenth International Middle East Power Systems Conference, MEPCON'2012*, 12 2012.

[22] B. Mohammadi-ivatloo, "Optimal placement of pmus for power system observability using topology based formulated algorithms," vol. 9, 12 2009.

[23] D. Dua, S. Dambhare, R. K. Gajbhiye, and S. A. Soman, "Optimal multistage scheduling of pmu placement: An ilp approach," *IEEE Transactions on Power Delivery*, vol. 23, no. 4, pp. 1812–1820, Oct 2008.

[24] H. A. Abdelsalam, A. Y. Abdelaziz, R. A. Osama, and R. H. Salem, "Impact of distribution system reconfiguration on optimal placement of phasor measurement units," in *2014 Clemson University Power Systems Conference*, March 2014, pp. 1–6.

[25] X.-S. Yang, *Engineering Optimization*. John Wiley & Sons, Inc., 2010. [Online]. Available: http://dx.doi.org/10.1002/9780470640425

[26] M. Gavrilas, “Heuristic and metaheuristic optimization techniques with application to power systems,” in *Proceedings of the 12th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering*, ser. MMACTEE’10. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 95–103. [Online]. Available: http://dl.acm.org/citation.cfm?id=1948923.1948939

[27] V.-k. Tran, H.-s. Zhang, and V.-n. Nguyen, “Optimal pmu placement in multi-configuration power distribution networks,” *Proceedings of the 2016 4th International Conference on Electrical & Electronics Engineering and Computer Science (ICEEECS 2016)*, 2016. [Online]. Available: http://dx.doi.org/10.2991/iceeecs-16.2016.104

[28] D. Carta, A. Benigni, and C. Muscas, “Model order reduction for pmu-based state estimation in distribution grids,” *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–10, 2017.

[29] X. Chen, T. Chen, K. J. Tseng, Y. Sun, and G. Amaratunga, “Customized optimal upmu placement method for distribution networks,” in *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, Oct 2016, pp. 135–140.

[30] N. M. Manousakis, G. N. Korres, and P. S. Georgilakis, “Taxonomy of pmu placement methodologies,” *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 1070–1077, May 2012.

[31] A. Tahabilder, P. K. Ghosh, S. Chatterjee, and N. Rahman, “Distribution system monitoring by using micro-pmu in graph-theoretic way,” in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, Sept 2017, pp. 159–163.

[32] H. A. Abdelsalam, A. Y. Abdelaziz, and V. Mukherjee, “Optimal pmu placement in a distribution network considering network reconfiguration,” in *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, March 2014, pp. 191–196.

Appendix A

MATLAB Code for Hybrid Greedy Placement Algorithm

```
clc

M = xlsread('/Users/brittneykerns/Dropbox/From FD1/IEEE Test Feeders/test.xlsx',5);
n = max(M(:)); % Number of buses in the system
A = zeros(n); % Initializing binary admittance matrix

% Creating the binary admittance matrix (A)
for i = 1:size(M,1)
    for j = 1:(size(M,2)-2)
        r = 1;
        while j+r < size(M,2)
            if M(i,j) > 0 && (M(i,j+r) < 0 || isnan(M(i,j+r)) == 1) && M(i,j+r+1) > 0
                A(M(i,j), M(i,j+r+1)) = 1;
                A(M(i,j+r+1), M(i,j)) = 1;
                break
            elseif M(i,j) > 0 && M(i,j+r) == 0
                break
            end
            r = r+1;
        end
    end
end

```

```
for j = 1:size(M,2)
    for i = 1:(size(M,1)-2)
        r = 1;
        while i+r < size(M,1)
            if M(i,j) > 0 && (M(i+r,j) < 0 || isnan(M(i+r,j)) == 1) && M(i+r+1,j) > 0
                A(M(i,j), M(i+r+1,j)) = 1;
                A(M(i+r+1,j), M(i,j)) = 1;
                break
            elseif M(i,j) > 0 && M(i+r,j) == 0
                break
            end
            r = r+1;
        end
    end
end

for i = 1:n
    for j = 1:n
        if i == j
            A(i,j) = 1;
        end
    end
end

% Beginning the Hybrid Greedy Placement Algorithm
tic

% Finding how many buses are observable at each node (if you place a PMU there)
D = zeros(1,n);
for j = 1:n
    D(1,j) = sum(A(:,j));
end

% Finding the maximum # of observable buses
Dmax = max(D);

% Initializing observabitlity matrix (B) and PMU placement vector (E)
E = zeros(1,n);
B = zeros(n);
r = ones(n);

```

```
for s = 0:(Dmax−2)
    while 1
        flag = 0;
        for q = 1:n
            r = zeros(1,n);
            for j = 1:n
                if D(1,j) == (Dmax−s)
                    for i = 1:n
                       if A(i,j) == 1 && sum(B(i,:)) == 0
                          r(1,j) = r(1,j) + 1;
                       end
                    end
                end
            end
            if sum(r) == 0
                flag = 1;
                break
            end
            if r(1,q) == max(r)
                E(1,q) = 1;
            end
            for j = 1:n
                for i = 1:n
                    % Placing PMU at according nodes
                    if E(1,j) == 1 && A(i,j) == 1
                        B(i,j) = 1;
                    end
                end
            end
        end
        if flag == 1
            break
        end
    end
end

G = B;
F = E;

for i = 1:size(M,1)
    for j = 1:size(M,2)
```

```
        if M(i,j) > 0 && E(1,M(i,j)) == 1 % Finding where a PMU is placed in order to move it
            % Moving PMU to the right
            if j <= (size(M,2)-2) && M(i,j+2) > 0 && E(1,M(i,j+2)) == 0 % Move PMU right one space if its
                free
                F(1,M(i,j+2)) = 1;
                F(1,M(i,j)) = 0;
                G(:,M(i,j)) = 0;
                G(:,M(i,j+2)) = A(:,M(i,j+2));
                for k = j:(size(M,2)-2)
                    r = 1;
                    while k+r < size(M,2)
                        if M(i,k+r) < 0 && M(i,k+r+1) > 0 && E(1,M(i,k+r+1)) == 1
                            % Removing PMU from system
                            G(:,M(i,k+r+1)) = 0; % Removes next right horizontally spaced PMU
                            F(1,M(i,k+r+1)) = 0;
                            % Checking for observability
                            C = any(G,2);
                            if any(C == 0) == 0 % it is observable
                                B = G;
                                E = F;
                                flag = 1;
                                break
                            else
                                flag = 1;
                                break
                            end
                        end
                        if flag == 1
                            break
                        end
                        r = r+1;
                    end
                    F = E;
                    G = B;
                end
            end
            if j <= (size(M,2)-2) && M(i,j+2) > 0 && E(1,M(i,j+2)) == 0 % Move PMU right one space if its
                free
                F(1,M(i,j+2)) = 1;
                F(1,M(i,j)) = 0;
                G(:,M(i,j)) = 0;
```

```
            G(:,M(i,j+2)) = A(:,M(i,j+2));
            for k = j:-1:3
                r = 1;
                while k-r > 1
                    if M(i,k-r) < 0 && M(i,k-r-1) > 0 && E(1,M(i,k-r-1)) == 1
                        % Removing PMU from system
                        G(:,M(i,k-r-1)) = 0; % Removes next left horizontally spaced PMU
                        F(1,M(i,k-r-1)) = 0;
                        % Checking for observability
                        C = any(G,2);
                        if any(C == 0) == 0 % it is observable
                            B = G;
                            E = F;
                            flag = 1;
                            break
                        else
                            flag = 1;
                            break
                        end
                    end
                    if flag == 1
                        break
                    end
                    r = r+1;
                end
                F = E;
                G = B;
            end
        end
        % Moving PMU to the left
        if j >= 3 && j <= size(M,2) && M(i,j-2) > 0 && E(1,M(i,j-2)) == 0 % Move PMU left one space if its free
            F(1,M(i,j-2)) = 1;
            F(1,M(i,j)) = 0;
            G(:,M(i,j)) = 0;
            G(:,M(i,j-2)) = A(:,M(i,j-2));
            for k = j:(size(M,2)-2)
                r = 1;
                while k+r < size(M,2)
                    if M(i,k+r) < 0 && M(i,k+r+1) > 0 && E(1,M(i,k+r+1)) == 1
                        % Removing PMU from system
```

```
                G(:,M(i,k+r+1)) = 0; % Removes next right horizontally spaced PMU
                F(1,M(i,k+r+1)) = 0;
                % Checking for observability
                C = any(G,2);
                if any(C == 0) == 0 % it is observable
                    B = G;
                    E = F;
                    flag = 1;
                    break
                else
                    flag = 1;
                    break
                end
            end
            if flag == 1
                break
            end
            r = r+1;
        end
        F = E;
        G = B;
    end
end
if j >= 3 && j <= size(M,2) && M(i,j-2) > 0 && E(1,M(i,j-2)) == 0 % Move PMU left one space if its free
    F(1,M(i,j-2)) = 1;
    F(1,M(i,j)) = 0;
    G(:,M(i,j)) = 0;
    G(:,M(i,j-2)) = A(:,M(i,j-2));
    for k = j:-1:3
        r = 1;
        while k-r > 1
            if M(i,k-r) < 0 && M(i,k-r-1) > 0 && E(1,M(i,k-r-1)) == 1
                % Removing PMU from system
                G(:,M(i,k-r-1)) = 0; % Removes next left horizontally spaced PMU
                F(1,M(i,k-r-1)) = 0;
                % Checking for observability
                C = any(G,2);
                if any(C == 0) == 0 % it is observable
                    B = G;
                    E = F;
```

```
                            flag = 1;
                            break
                        else
                            flag = 1;
                            break
                        end
                    end
                    if flag == 1
                        break
                    end
                    r = r+1;
                end
                F = E;
                G = B;
            end
        end
        % Moving PMU down
        if i <= (size(M,1)-2) && M(i+2,j) > 0 && E(1,M(i+2,j)) == 0 % Move PMU down one space if it's
            free
            F(1,M(i+2,j)) = 1;
            F(1,M(i,j)) = 0;
            G(:,M(i,j)) = 0;
            G(:,M(i+2,j)) = A(:,M(i+2,j));
            flag = 0;
            for k = i:-1:3
                r = 1;
                while k-r > 1
                    if M(k-r,j) < 0 && M(k-r-1,j) > 0 && E(1,M(k-r-1,j)) == 1
                        % Removing PMU from system
                        G(:,M(k-r-1,j)) = 0; % Removes next PMU located upwards
                        F(:,M(k-r-1,j)) = 0;
                        % Checking for observability
                        C = any(G,2);
                        if any(C == 0) == 0 % it is observable
                            B = G;
                            E = F;
                            flag = 1;
                            break
                        else
                            flag = 1;
                            break
```

```
                    end
                end
                if flag == 1
                    break
                end
                r = r+1;
            end
            F = E;
            G = B;
        end
    end
    if i <= (size(M,1)-2) && M(i+2,j) > 0 && E(1,M(i+2,j)) == 0 % Move PMU down one space if it's
        free
        F(1,M(i+2,j)) = 1;
        F(1,M(i,j)) = 0;
        G(:,M(i,j)) = 0;
        G(:,M(i+2,j)) = A(:,M(i+2,j));
        flag = 0;
        for k = i:(size(M,1)-2)
            r = 1;
            while k+r < size(M,1)
                if M(k+r,j) < 0 && M(k+r+1,j) > 0 && E(1,M(k+r+1,j)) == 1
                    % Removing PMU from system
                    G(:,M(k+r+1,j)) = 0; % Removes next PMU located downwards
                    F(:,M(k+r+1,j)) = 0;
                    % Checking for observability
                    C = any(G,2);
                    if any(C == 0) == 0 % it is observable
                        B = G;
                        E = F;
                        flag = 1;
                        break
                    else
                        flag = 1;
                        break
                    end
                end
                if flag == 1
                    break
                end
                r = r+1;
```

```
                end
                F = E;
                G = B;
            end
        end
        % Moving PMU up
        if i > 2 && M(i-2,j) > 0 && E(1,M(i-2,j)) == 0 % Move PMU up one space if it's free
            F(1,M(i-2,j)) = 1;
            F(1,M(i,j)) = 0;
            G(:,M(i,j)) = 0;
            G(:,M(i-2,j)) = A(:,M(i-2,j));
            flag = 0;
            for k = i:-1:3
                r = 1;
                while k-r > 1
                    if M(k-r,j) < 0 && M(k-r-1,j) > 0 && E(1,M(k-r-1,j)) == 1
                        % Removing PMU from system
                        G(:,M(k-r-1,j)) = 0; % Removes next PMU located upwards
                        F(:,M(k-r-1,j)) = 0;
                        % Checking for observability
                        C = any(G,2);
                        if any(C == 0) == 0 % it is observable
                            B = G;
                            E = F;
                            flag = 1;
                            break
                        else
                            flag = 1;
                            break
                        end
                    end
                    if flag == 1
                        break
                    end
                    r = r+1;
                end
                F = E;
                G = B;
            end
        end
        if i > 2 && M(i-2,j) > 0 && E(1,M(i-2,j)) == 0 % Move PMU up one space if it's free
```

```
                F(1,M(i-2,j)) = 1;
                F(1,M(i,j)) = 0;
                G(:,M(i,j)) = 0;
                G(:,M(i-2,j)) = A(:,M(i-2,j));
                flag = 0;
                for k = i:(size(M,1)-2)
                    r = 1;
                    while k+r < size(M,1)
                        if M(k+r,j) < 0 && M(k+r+1,j) > 0 && E(1,M(k+r+1,j)) == 1
                            % Removing PMU from system
                            G(:,M(k+r+1,j)) = 0; % Removes next PMU located downwards
                            F(:,M(k+r+1,j)) = 0;
                            % Checking for observability
                            C = any(G,2);
                            if any(C == 0) == 0 % it is observable
                                B = G;
                                E = F;
                                flag = 1;
                                break
                            else
                                flag = 1;
                                break
                            end
                        end
                        if flag == 1
                            break
                        end
                        r = r+1;
                    end
                    F = E;
                    G = B;
                end
            end
        end
    end
end

toc

F = zeros(1,sum(E));
j = 1;
```

```
for i = 1:n
    if E(1,i) == 1
        F(1,j) = i;
        j = j + 1;
    end
end

C = any(B,2);
if any(C == 0) == 0
    disp('This system is observable.')
else
    disp('This system is not observable.')
end

disp(sum(E))
disp(F)
```