Power System Observability and Optimal Phasor Measurement Unit Placement

A PROJECT REPORT
SUBMITTED FOR COURSE EE8725
ADVANCED POWER SYSTEM ANALYSIS AND ECONOMICS
BY

Pei Xu

Advisor:
Bruce F. Wollenberg

Department of Electrical and Computer Engineering
College of Science and Engineering
University of Minnesota - Twin Cities
December, 2015

Acknowledgements

I would like to express my heartfelt thanks to Prof. Wollenberg, who offered me precious instructions and suggestions in and after class. Without his instruction, this project report could not be what it is. I hope Prof. Wollenberg has a good health.

Dedication

For my wife. Thank her for her support.

To my unborn child or maybe children. I love you.

Abstract

Power systems' secure operation demands a comprehensive and detailed monitoring system to guarantee systems' observability. However, redundant measurements would cause problems in control, management and cost. The optimal Phasor Measurement Unit (PMU) placement problem is referred to as to minimize the amount of Phasor Measurement Units installed in a power system and, meanwhile, to maintain the full system's observability. This report introduces how to analyze power system observability based on PMUs, and analyzes four algorithms for optimal PMU placement problems. Besides, a modified Simulated Annealing Method is proposed. Simulation based on the four algorithms, including Depth-First Search, Graph Theoretic Procedure, Simulated Annealing Method and Recursive Security N Algorithm, is conducted in Matlab for the sake of comparing these algorithms' performances.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter mainly introduces the concept of power system observability and the basic properties of Phasor Measurement Units (PMUs), and describes the proposal of this project. The structure of this project report is shown at the end of this chapter.

1.1 Introduction to Power System Observability

There has been the demand on precise and immediate monitoring for power systems since the birth of power systems. We need to know the concrete operating state of power systems in order to ensure the security and optimum operation of the systems. The idea of state estimation of power systems is proposed based on this demand.

Basically, state estimation is the process to evaluate a power system's operating state according to the information provided by a set of measurements installed in this system. In state estimation, power system observability is an essential concept. A power system is considered observable, if it is possible to specify the bus voltage magnitude and angle at every bus in the whole network from measurements [1].

The importance of power system observability is very much in evidence. Only when a power system is guaranteed to be observable can it be possible that the state estimation to this system is comprehensive and accurate. In some cases, the power system observability is even the prerequisite for state estimation.

1.2 Introduction to Phasor Measurement Units

A PMU is a device that measures electrical waves employing synchronization signals, which usually are from global positioning systems (GPS). It can directly provide the voltage phasor of the bus at which it is installed and the current phasors of all incident branches. The introduction of PMUs is a significant innovation in power systems.

PMUs are faster than traditional measurements. Via synchronized signals, PMUs and other synchronized measurements have a much lower time skew. In theory, the accuracy of synchronization provided by GPS time stamping can be better than 1 microsecond [2]. This remarkably improves the accuracy of measurements, and also is beneficial to many other applications like system protection and control assessment.

Most importantly, the introduction of PMUs makes it possible to directly measure phase angles between the phasors at different locations. Traditionally, state estimation is formulated as a weighted least squares problem [3], due to the absence of measurements who can measure phasors. This problem only can be solved via non-linear iterations. The introduction of PMUs makes state estimation able to be achieved via linear estimators, which significantly increase the efficiency of solving state estimation problem. In [4], the authors proved that if a power system's observability is achieved through placing PMUs in this system, the system state can be obtained by running a linear state estimator in a single iteration [4].

1.3 Proposal of this Project

From the brief introduction to PMUs, it can be recognized that it is possible to directly measure all a system's states via simply placing PMUs at all buses without running any state estimator. Nevertheless, it is obviously not economical. Besides the consideration of economy, the increasing scale of power systems also makes it tough to control and manage so many PMUs if they are installed at all buses. Hence, it is inadvisable to simply install a PMU at every bus. Our goal here is to main a power system's observability through as few PMUs as possible. That is the optimal PMU placement (OPP) problem.

This project mainly focuses on four algorithms for OPP problem: Depth-First Search (DFS) Method, Graph Theoretic (GTh) Procedure, Simulated Annealing (SA) Method

and Recursive Security N (RSN) Algorithm. The performances of these algorithms are analyzed and compared in theory and via simulation. Their merits and demerits are summarized.

1.4 Structure of this Project Report

The structure of the following parts of this report is listed below.

- Chapter 2 introduces observability analysis using PMUs, and further expound the formulation of OPP problems.
- Chapter 3 describes four algorithms for the OPP problem, the algorithms which are mentioned above.
- Chapter 4 shows the simulation results of solving OPP via the four algorithms, and provides an analysis to the results.
- Chapter 5 lists all the programs coded and used for the simulation.
- Chapter 6 summaries the four algorithms' performance, and proposes the author's future work on the OPP problem.

Chapter 2

Observability Analysis using Phasor Measurement Units

This chapter elaborates the method for analyzing power system observability in theory and sets forth the optimal PMU Placement (OPP) problem.

2.1 Observability Analysis

The basis of OPP is to guarantee the full observability of a given power system. That is to say, in the OPP problem, it is necessary to check the system's observability before the final solution to the problem or rather the final placement of PMUs is determined. A power system's observability can be decided by this way: if the measurements installed in a power system can build a spanning tree of full rank of this system, the system is fully observable [1]. Basically, two analysis methods – numerical or algebra, and topological observability analysis – can be employed for observability analysis.

2.1.1 Numerical or Algebra Observability Analysis

In state estimation, the measurement model can be described as

$$\bar{z} = h(\bar{x}) + \bar{e} \tag{2.1}$$

where $\bar{z}$ is the measurement vector, $\bar{x}$ is the system state vector that contains all buses' voltage phasor, $h(\bar{x})$ is the nonlinear function related to the measurement vector and

state vector, and $\bar{e}$ is the measurement error or noise vector.

Considering the accuracy of data provided by PMUs, $\bar{e}$ is very small and thus is neglected in general. In consideration of that $h(\bar{x})$ relates measurements to bus voltage magnitudes and angles [1], the state estimator for this system is linear, if only PMUs are used as measurements in a power system. Therefore, exclusive usage of PMUs leads to a linear state estimator that can be described as

$$\bar{z} = \mathbf{H}\bar{x} \tag{2.2}$$

where $\mathbf{H}$ is the measurement function matrix and also a coefficient matrix related to the system state vector $\bar{x}$.

For a power system with n buses and m PMUs , $\bar{x}$ is a vector with $2n-1$ dimensions and $\mathbf{H}$ is a matrix with $m \times (2n-1)$ dimensions. If the system is fully observable, $\bar{z}$ provided by Equation 2.2 should have $2n-1$ valid elements. This means that a power system with n buses is observable if

$$Rank(\mathbf{H}) = 2n - 1 \tag{2.3}$$

where n the number of buses in the system.

A power system is considered numerically or algebraically observable if it satisfies Equation 2.3.

2.1.2 Topological Observability Analysis

When topology is used to represent a power system, the system can be taken into account as a graph with n apexes, which represent n buses, and b edges, which represents b branches each of which connects two buses.

This graph can be described as

$$G = (V, E) \tag{2.4}$$

where V is the set of the apexes in this graph, and E is the set of the edges in this graph.

A subgraph can be described as

$$G' = (V', E') \tag{2.5}$$

where $V' \subseteq V$ and $E' \subseteq E$.

Usually, a power system is considered as topologically observable, if in a subgraph there is a G' that contains all the apexes of G, namely, whose $V \subseteq V'$.

2.2 PMU Placement Rules

2.2.1 For Numerical Observability

If only numerical observability is employed, the rank of the matrix $\mathbf{H}$ must be checked whenever a PMU is installed at or removed from a bus in order to evaluate the system's observability.

Basically, in the OPP problem, two methods are able to be implemented for numerical observability:

- For a n-bus system, simulatively install PMUs one by one at any possible bus to improve the rank of $\mathbf{H}$ until Equation 2.3 is satisfied, namely, until $rank(\mathbf{H}) = 2n - 1$.

- For a n-bus system, assume that there is a PMU at each bus, then simulatively remove PMUs one by one from any possible bus until Equation 2.3 is unable to be satisfied, namely, until $rank(\mathbf{H}) < 2n - 1$, and finally withdraw the last 'remove'.

In consideration of the scale of power systems in real life, both of the two methods have to attempt a large number of combinations before the optimal placement is found. And, every attempt will incur an examination of the rank of $\mathbf{H}$, which significantly increase the time needed by the two methods. Besides, the measurement matrix generated through these two methods may be ill-conditioned. Therefore, in practice, numerical observability is rarely employed for solving the OPP problem.

2.2.2 For Topological Observability

Topological observability is a more practically employable method dealing with power system observability. Our analysis and discussion about power system observability in this report are all based topological observability if there is no special instruction.

When topological observability is employed, the OPP problem can be turned into the problem of finding a minimal subgraph which satisfies $V \subseteq V'$ if the subgraph is described via Equation 2.5. Therefore, in order to prune unnecessary branches in the graph, a set of observability rules must be determined for the sake of determining the observable areas in a system after a PMU is installed at a bus in this system.

With respect to topological observability, basically, we have:

1. for a bus at which a PMU is installed, voltage and current phasors of all lines connected to this bus are known;

2. if voltage and current phasors at a bus are known, voltage phasor at the other buses connected to this bus can be calculated via Ohm's Law; and

3. if voltage phasors at two connected buses are known, the current phasor at the line that connects the two buses can be calculated.

The 1st rule is called direct measurements, and the 2nd and 3rd rules are called pseudo measurements. The prototype of these three rules was originally proposed via Baldwin et al. in [5]. The above three rules are the transliteration of the original rules. In practice or simulation, these three rules can be further summarized as that if there is a PMU installed at a bus, that bus and all its adjacent buses are observable.

Formulation for OPP Problem Based on Topological Observability

The objective of the OPP problem is to minimize the installation cost of PMUs. For a n-bus system, the OPP problem can be formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} w_i x_i \\ \text{s.t.} \quad & F(\bar{X}) \geq \bar{I} \end{aligned} \tag{2.6}$$

where w_i is the cost of installing a PMU at the Bus i,

$x_i = 1$ if a PMU is installed at Bus i and otherwise $x_i = 0$, $\bar{X}$ is a vector composed of x_i,

$F(X)$ is a vector function that represents the observability constraint functions, and

$\bar{I}$ is a vector who has n entries and whose element are all 1 if our goal is only to make sure the system fully observable [1] .

According to the above-mentioned three rules, the constraint vector function is obtained as[2]

$$F(\bar{X}) = \mathbf{A}\bar{X} \tag{2.7}$$

where $\mathbf{A}$ is the binary adjacency matrix. Namely, the element of matrix A is defined as

$$a_{mn} = \begin{cases} 1 & \text{if } m = n, \\ 1 & \text{if Bus } m \text{ and Bus } n \text{ are connected,} \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

where a_{mn} represents the element located at the m^{th} row and the n^{th} column in $\mathbf{A}$.

For the Bus i in a n-bus system, the corresponding constraint function f_i can be represented as

$$f_i(\bar{X}) = \sum_{j=1}^{n} a_{ij}x_j \tag{2.9}$$

If $f_i(\bar{X}) \neq 0$, namely, if any $a_{ij} = 1$ and $x_j = 1$ $(j = 1, 2, ..., n)$, the Bus i is observable. If all buses are observable, namely, if all $f_i(\bar{X})$ in $F(\bar{X})$ are non-zero, which also must be grater than 1, the power system is fully observable.

Assume that $w_i = 1$, Equation 2.6 can be converted into

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{s.t.} \quad & F(\bar{X}) \geq \bar{I} \end{aligned} \tag{2.10}$$

In general, Equation 2.10 is used when there is no significant difference in the costs of installing a PMU at any bus.

Redundancy in Observability

According to Rule 1 to 3, if a bus j can make Bus i observable $(i \neq j)$, that bus j must be adjacent to Bus i (i.e. $a_{ij} = 1$) and that a PMU is installed at bus j (i.e. $x_j = 1$).

[1] A detailed discussion about how the right part of the constraint inequality should be constructed is provided in the Subsection 2.2.2 Redundancy in Observability.

[2] Equations in this subsection are only based on Rule 1 to 3 and without the consideration of ZI buses. Discussion aimed at ZI buses in OPP problems is provided in the next sections.

Hence, if bus j can make Bus i observable ($i \neq j$), $a_{ij}x_j = 1$ must be satisfied. That is to say, for a n-bus system, the equation

$$\sum a_{ij}x_j \quad \forall j \in \mathbb{N}; 1 \leq j \leq n; j \neq i \tag{2.11}$$

reflects how many such bus j exist to support Bus i observable.

Meanwhile, according to Rule 1 to 3, Bus i is also observable if a PMU is installed at that bus. Based on Equation 2.8, $a_{ii} = 1$. Therefore,

$$a_{ii}x_i \tag{2.12}$$

reflects if a Bus i is observable due to that there is a PMU installed at that bus.

Equation 2.9 is obtained via combining the above two equations. It can be recognized that what Equation 2.9 reflects, in fact, is how many sources there are to support the Bus i to be observable.

Therefore, if all buses in a system are observable (namely, for each bus in that system there is at least one source to make that bus observable), the inequality $F(\bar{X}) \geq \bar{I}$ must be satisfied. This is how we obtain the constraint function in Equation 2.6 and 2.10.

In practice, we sometimes hope there is some redundancy in a system's observability. In other words, we may hope there are more than one source to support some or even all buses' observability so that these buses are guaranteed to be observable when one source is lost. The author here calls it the requirement of redundancy in observability. In this case, we need to modify $\bar{I}$ in constraint inequality $F(\bar{X}) \geq \bar{I}$. Then, we have

$$F(\bar{X}) \geq \bar{b} \tag{2.13}$$

where $\bar{b}$ is n by 1 vector whose elements $b_i \geq 1$.

If, for example, we hope that for any bus in a system, there are at least two sources who can make that bus observable, the constraint inequality in Equation 2.6 and 2.10 can be written as Equation 2.13 where $\bar{b}$ is a n by 1 vector whose entries are all 2 and n is the number of buses in that system.

2.3 Zero Injection Buses in OPP Problem

For zero injection (ZI) buses, we have:

4. for a ZI bus without a PMU, if its outgoing currents are all known except for one, then the unknown outgoing current can be calculated via Kirchhoff's Current Law (KCL);

5. for a ZI bus, if the voltage phasors of all buses connected to this bus are all known, the voltage phasor of this bus can be calculated via Kirchhoff's Voltage Law (KVL); and

6. for a group of connected ZI buses, if the voltage phasors of all buses connected to this group of buses are known, the voltage phasors of this group of buses can be calculated via KVL.

These three rules are called extended measurements [6], which are with respect to ZI buses. These rules can be further summarized as that if all but one of a ZI bus and its adjacent buses are observable, than the ZI bus and all its adjacent buses are observable. Employing these three rules, a more optimal OPP solution may be obtained compared with that only employing the Rule 1 to 3.

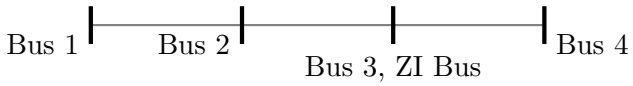

Figure 2.1: Example of Employing ZI Bus in OPP Problems

Now considering the situation shown in Fig. 2.1.

If we only employ Rule 1 to 3, (namely, not consider the ZI bus's property,) at least 2 PMUs must be installed into the system for the sake of full observability, since at most three buses will be considered observable if a PMU is installed into the system.

If we employ Rule 4 to 6 additionally, a PMU installed at Bus 2 is enough to make the system fully observable, since if a PMU is installed at Bus 2, then Bus 1 and Bus 3 are considered observable according to Rule 1 to 3, and because of that Bus 2 and 3 are both observable and that Bus 3 is a ZI bus, Bus 4 is observable via calculation using KCL and KVL.

2.4 Formulation for OPP Problem Considering ZI Buses

The introduction of ZI buses makes OPP problems become complex. Basically, we have two methods to deal with ZI buses. The first one is about to modify the adjacency matrix **A** of a given system. The second one is about to directly modify Equation 2.7 ($F(\bar{X}) = \mathbf{A}\bar{X}$).

2.4.1 Merging Method

The basic idea of this method is quite simple. That is to merge each ZI bus into one of its adjacent buses. According to Rule 4 to 6, if all of a ZI bus's adjacent buses are observable, that ZI bus is also observable. Merging method's goal is to make all of each ZI bus's adjacent buses observable, and then each ZI bus guaranteed to be observable.

Merging ZI Buses on Graph

In some paper, e.g.[6], [7] and [8], the authors argued that a system's observability can be obtained via employing Equation 2.7 to analyze the modified adjacency matrix of the system in which each ZI bus is merged into one of its adjacent bus. Some of them mentioned the drawback of this kind of method but some of them not. The author here argues that this method sometimes is employable but may lead to unideal results.

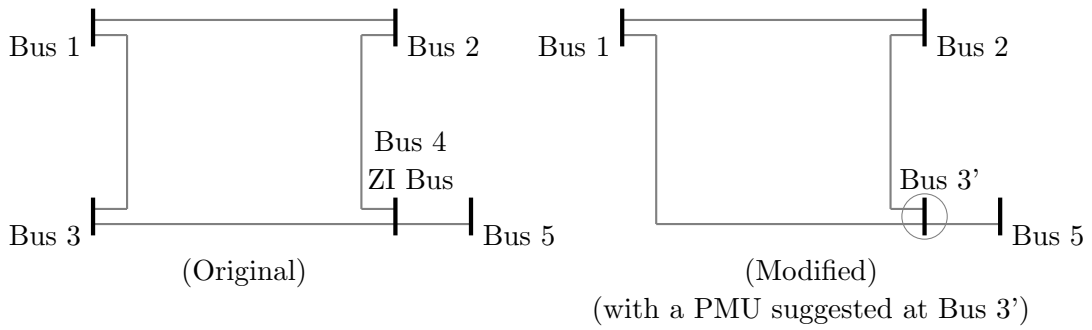

Figure 2.2: ZI Bus Merging Method From [6]

Fig. 2.2 is an example from [6]. The modified diagram is obtained via merging Bus 4 into Bus 3. In [6], the authors argue that the adjacency matrix constructed according to the modified diagram can be used to analyze the original system's observability.

However, according to Equation 2.7, the modified system can be fully observable through simply installing a PMU at Bus 3', which in fact is a nonexistent bus in the original system. We have two direct means to deal with such a situation: (1) install a PMU at either of Bus 3 or 4 in the original system; or (2) install a PMU at each of Bus 3 and 4 in the original system.

The (1) method is useless in the case shown in Fig. 2.2, since it is obvious that installing a PMU at either of Bus 3 or 4 is unable to make the system fully observable. In [7], the authors argued that we can consider ZI buses are nonexistent after each of them is merged into one of its adjacent buses. Their idea is, in the case shown in Fig. 2.2, to install a PMU at Bus 3 in order to make the original system fully observable since a PMU installed at Bus 3' is able to ensure the modified system's full observability. Their idea obviously does not work in this case.

The (2) method works well and can guarantee the system's observability. Nevertheless, this method could result in redundancy in the situation like that shown in Fig. 2.3. The system shown in Fig. 2.3 can be fully observable through simply installing a PMU at any Bus. However, if we use merging method and employ the (2) method, at least two buses should be installed a PMU at.

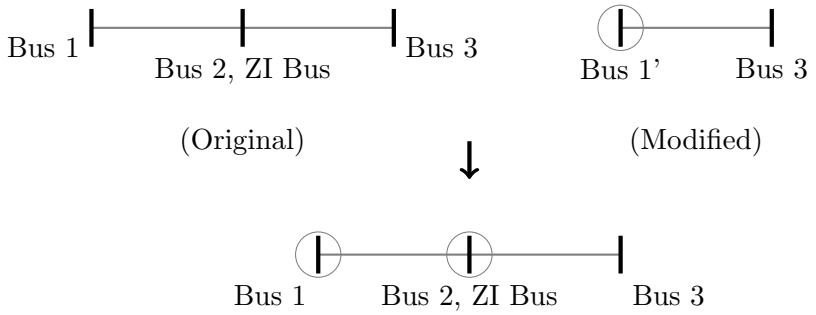

Figure 2.3: Redundancy Caused by ZI Bus Merging Method (2)

As a matter of fact, there is still a (3) method to deal with the situation shown in Fig. 2.2 or 2.3. That is, we enforce a rule that it is illegal to install a PMU at any bus generated via merger. However, this method, although solves the problems in Fig. 2.2 and 2.3, still could cause redundancy in some certain cases like that shown in Fig. 2.4.

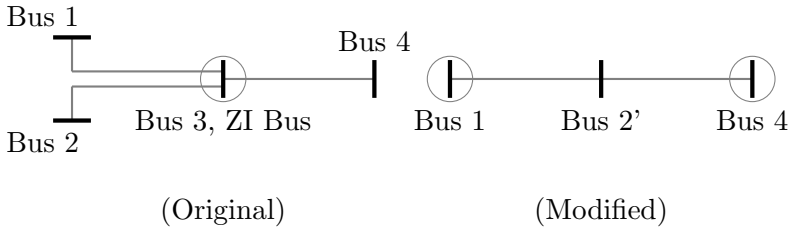

Figure 2.4: Redundancy Caused by ZI Bus Merging Method (3)

In Fig. 2.4, the modified diagram is obtained via merging Bus 3 into Bus 2. If Bus 2' is not allowed to be installed a PMU at, two PMUs installed respectively at Bus 1 and 4 are needed to ensure the whole system's observability. However, it is obvious that a PMU installed at Bus 3 is enough to ensure the original system's full observability.

In summary, the author here argues that it is not a very reliable method to analyze original system's observability through analyzing the modified system where each ZI bus is merged into one of its adjacent buses.

Merging ZI Buses' Constraint Equations

The strategy of this method is to consider a pseudo flow measurement installed on the branch between a ZI bus and one of its adjacent buses, and then to deal with ZI buses as normal buses.

According to Rule 4 to 5, we have the conclusion that a ZI bus is observable if all its adjacent buses are observable, and the conclusion that a bus connected to a ZI bus is observable if all other buses connected to that ZI bus and that ZI bus are observable. Hence if we can ensure all buses but one connected to a ZI bus are observable, then this group of buses are observable if either of that one bus or the ZI bus is observable. Take the example shown in Fig. 2.4. The two conclusions can be expressed as:

- if Bus 1 and 2 are observable, then the system is fully observable if at least one of the ZI bus (i.e. Bus 3) and Bus 4 is observable; or
- if Bus 1 and 4 are observable, then the system is fully observable if at least one of the ZI bus (i.e. Bus 3) and Bus 2 is observable; or

- if Bus 2 and 4 are observable, then the system is fully observable if at least one of the ZI bus (i.e. Bus 3) and Bus 1 is observable.

From the above example, when a bus connected to the ZI bus is chosen to pair up with the ZI bus, the system will become fully observable if at least one of these two buses is observable and the other but not these two buses are observable. This means that this pair of ZI bus and one of its adjacent buses can be considered as a group and have the same constraint function about their observability; and the constraint function is obtained via merging the two buses' constraint functions together. Based on this conclusion, in Fig. 2.4, the adjacency matrix of the original system

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{2.14}$$

can be modified as

$$\mathbf{A}' = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{2.15}$$

if Bus 2 is chosen to pair up with the ZI Bus 3. Then, we use a modified vector constraint equation

$$F'(\bar{X}) = \mathbf{A}'\bar{X} \geq \bar{b} \tag{2.16}$$

to identify the whole original system's observability. In Equation 2.16

We could recognized that there are always repeated constraints equations among f_i of which $F'(\bar{X})$ is composed, since each ZI bus and one of its adjacent buses have the same constraint equation. Therefore, in practice, F' can be further simplified through removing one from each pair of two repeated equations for the sake of reducing the amount of computation.

This method, in fact, is regarded each ZI bus and the bus into whose equation the ZI bus's constraint equation is merged as a same node in the system's diagram. However, since no bus is really eliminated, this method will not lead to the embarrassment caused by the above-mentioned (1) method.

Nevertheless, Equation 2.16 is just a sufficient not a necessary condition to a system's full observability, since a ZI bus always could be considered to have the same constraint equation with any rather than just one of its adjacent buses. That is to say, a system's full observability cannot necessarily ensure that the inequality in Equation 2.16 must be satisfied, because ZI buses' observability may be guaranteed via other but not the selected one bus. Take the above example. If two PMUs are installed at Bus 2 and 4 respectively and no other PMU is installed in this sytem, such that the modified vector constraint function is

$$F'(\bar{X}) = \mathbf{A}' \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 1 \end{pmatrix} \tag{2.17}$$

According to Equation 2.16, this system is considered not fully observable since Bus 1 is thought unobservable. However, we can find that Bus 1 is, in fact, observable, since Bus 2, 3 and 4 are all observable and Bus 3 is a ZI bus and thus Bus 1 is observable via employing KCL and KVL according to Rule 4 to 5. Hence, if we employ this merging method and put a PMU at bus 1 in order to make the entire system observable, the redundancy appears. [3]

Summary

The constraint inequality for a system's observability (i.e. Equation 2.13 $F(\bar{X}) \geq \bar{b}$) used in merging method is still linear, since this merge still uses $F(\bar{X}) = A\bar{X}$ to construct the inequality constraint, where A may be a reduced or modified but not necessarily an accurate adjacency matrix for the original system. Hence, programming like integer programming can be employed to solve OPP problems if merging methods are employed to deal with ZI buses.

However, from the above analysis, it can be recognized that neither of merging ZI buses on graph nor merging ZI buses' constraint equations can provide an accurate constraint vector function $F(\bar{X})$ like that in Equation 2.7 aimed at the system without ZI bus, the Equation 2.7 who can accurately reflect how many sources there are to

[3] In [8], the authors mention that this merging method could lead to redundancy and give an example based on IEEE 9-bus system, but do not provide any theoretical explanation. The author here gives an explanation in theory.

support a bus's observability. Therefore, when using merging method to solve OPP problems, redundancy may appear.

2.4.2 Nonlinear Constraint Function Method

The idea of this method is to revise Equation 2.7 in consideration of ZI buses' properties. And, because we finally may obtain a new and nonlinear constraint function vector $F(\bar{X})$, this method is called nonlinear constraint function method.

According to Rule 4 to 6, for each bus in the group composed of a ZI bus and its adjacent buses, a bus is observable if other buses in that group are observable. Therefore, we introduces an auxiliary binary variable y_{ij} into Equation 2.7 to represent if Bus i is observable supported by other buses but may not Bus i in the group of a ZI bus j and all of its adjacent buses. Through this way, each equation f_i in the constraint vector function Equation 2.7, (i.e. Equation 2.9) can be revised as

$$f_i = \sum_{j=1}^{n} a_{ij}x_j + \sum_{j=1}^{n} a_{ij}z_j y_{ij} \tag{2.18}$$

where z_j is a binary parameter who is 1 if bus j is a ZI bus or 0 otherwise; and y_{ij} is a auxiliary binary variable who is 1 if all of bus j and its adjacent buses but not including Bus i are observable or 0 otherwise.

In the $a_{ij}z_j y_{ij}$ part of Equation 2.18, a_{ij} ensures Bus i and j must be connected; and z_j ensures bus j must be a ZI bus. The two parameters determine that Rule 4 to 6 only can be applied to ZI buses. Through modifying Equation 2.9 into 2.18, a revised constraint vector function $F(\bar{X})$ is obtained finally.

In practice, y_{ij} have to be calculated via logical operation. If, for example, there is a ZI bus m to which k normal buses composed of bus 1 to k where $m \notin [1, k]$ are connected, then the constraint function for bus i to k and for bus m can be obtained as

$$
\begin{aligned}
f_1 &= \sum_{j=1}^{n} a_{1j}x_j + (f_2 \wedge f_3 \wedge ... \wedge f_k \wedge f_m) \\
f_2 &= \sum_{j=1}^{n} a_{2j}x_j + (f_1 \wedge f_3 \wedge ... \wedge f_k \wedge f_m) \\
&... \\
f_k &= \sum_{j=1}^{n} a_{kj}x_j + (f_1 \wedge f_2 \wedge ... \wedge f_{k-1} \wedge f_m) \\
f_m &= \sum_{j=1}^{n} a_{mj}x_j + (f_1 \wedge f_2 \wedge ... \wedge f_k)
\end{aligned}
\tag{2.19}
$$

where $\wedge$ is the logical operation AND.

Considering the situation shown in Fig. 2.5 where four PMUs are respectively installed at Bus 1, 2, 6 and 7, there are two sources to support the observability of Bus 3, and it is the same to Bus 5.

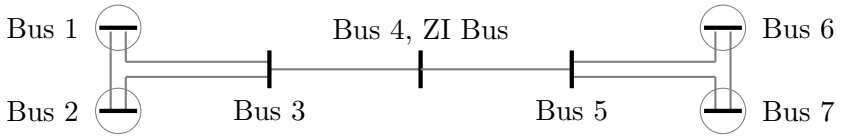

Figure 2.5: Observability Supported via Multiple Sources Applying ZI Buses' Properties

Employing Equation 2.18, we have

$$
\begin{aligned}
&x_1 = x_2 = x_6 = x_7 = 1 \\
&x_3 = x_4 = x_5 = 0 \\
&f_1 = x_1 + x_2 + x_3 = 2 \\
&f_2 = x_1 + x_2 + x_3 = 2 \\
&f_3 = x_1 + x_2 + x_3 + x_4 + (f_4 \wedge f_5) = 3 \\
&f_4 = x_3 + x_4 + x_5 + (f_3 \wedge f_5) = 1 \\
&f_5 = x_4 + x_5 + x_6 + x_7 + (f_3 \wedge f_4) = 3 \\
&f_6 = x_5 + x_6 + x_7 = 2 \\
&f_7 = x_5 + x_6 + x_7 = 2
\end{aligned}
\tag{2.20}
$$

Since all f here are greater than 1, the system is considered observable.

From the above equations, we can recognize that the constraint equations of which the constraint function is composed are coupled. Take the above example, any change in the value of f_4 or of f_5 may result in a change in the value of f_3. If we only calculate f_1 to f_n (where n is the total number of buses in a given system) once in a certain order. The final result may be inaccurate. Therefore, we need to repeatedly calculate f_1 to f_n and take into account their values in the last calculation, until the values of f_1 to f_n does not change after an iteration.

Besides, we can notice, from the above equations, that $f_4 = 1$, $f_3 = 3$ and $f_5 = 3$. If we consider that what the constraint function f_i reflects is how many sets of branches there are through which Bus i be observable. The above equations are accurate for Bus 4 but inaccurate for Bus 3 and 5, since for Bus 4, the branch from Bus 3 to 4 and the branch from Bus 4 to 5 work together to make bus 4 observable. The outage of anyone of the two branches could make bus 4 unobservable, but since for each of Bus 3 and 5, its observability is supported via two but not three sets of branches.

On the other hand, if the constraint function f_i is considered to reflect how many sources there are to support the observability of f_i, the equations are inaccurate for all of Bus 3, 4 and 5, since the observability of each of the three buses is supported by two sources[4] .

The author here proposes a method to avoiding this inaccuracy.

(1) In the first case, namely, when hoping the constraint function can reflect how many sets of branches there are through which a bus is observable, we need to do once more calculation at the end of calculating the constraint function. In this calculation, for buses connected to a ZI Bus i, the term of $\wedge$ operation should time x_i. This revision is applicable when we consider the situation where branches may outage.

Take the example shown in Fig. 2.5. After calculating out Equation 2.20, we need

[4] It is easy to spot that for each of Bus 3 and 5 there are two sources that are supporting its observability. For Bus 4, its observability is achieved via the cooperation of Bus 3 and 5. It will not become unobservable if only one PMU loses in the system. That is to say, there are two sources that are supporting the observability of Bus 4

to revise the result via doing the below calculation.

$$\begin{aligned} f_3^{'} &= x_1 + x_2 + x_3 + x_4 + (f_4 \wedge f_5) \times x_4 = 2 + 0 = 2 \\ f_4^{'} &= x_3 + x_4 + x_5 + (f_3 \wedge f_5) = 0 + 1 = 1 \\ f_5^{'} &= x_4 + x_5 + x_6 + x_7 + (f_3 \wedge f_4) \times x_4 = 2 + 0 = 2 \end{aligned} \tag{2.21}$$

(2) In the second case, namely, when hoping the constraint function can reflect how many sources there are are through which a bus is observable, in the calculation we need to do at the end of calculating the constraint function, we use the operation min to replace $\wedge$ firstly, and then for the buses connected to a ZI Bus i, its term of min operation also needs to time x_i. This revision is applicable when we consider the situation where PMUs may lose.

Take the example shown in Fig. 2.5. After calculating out Equation 2.20, we need to revise the result via doing the below calculation.

$$\begin{aligned} f_3^{'} &= x_1 + x_2 + x_3 + x_4 + min\,(f_4, f_5) \times x_4 = 2 + 0 = 2 \\ f_4^{'} &= x_3 + x_4 + x_5 + min\,(f_3, f_5) = 0 + 2 = 2 \\ f_5^{'} &= x_4 + x_5 + x_6 + x_7 + min\,(f_3, f_4) \times x_4 = 2 + 0 = 2 \end{aligned} \tag{2.22}$$

If our goal is just to ensure a system's full observability without the consideration of redundancy in observability, Equation 2.18 is employable. In this case, the logical operation $\wedge$ can be replaced with multiplication operation $\times$, or the addition operation $+$ can be replaced with the logical operation $\vee$, since we do not care about the concrete value of the constraint function but only care about if the value is greater than zero.

Summary

No matter which operations from $\wedge$, $\vee$, $+$, $\times$ and $min()$ are chosen, the revised constraint function is not linear once the system has any ZI bus. The possible existence of logical operations makes it quite difficult to solve OPP problems via programming methods. And, even $\times$ and $+$ operations are chosen to construct equations, the equation is likely to have quite high degree as the scale of the system increases[5] . Furthermore, the

[5] Several equations with 14 degree will appear when solving the OPP problem for IEEE 30-bus system if multiplication operation is chosen to construct constraint function $F(\bar{X})$.

coupling among constraint equations of which the constraint function is composed makes the function hard to be calculated unless using loop in program or using recursive.

In [9], OPP problems are proven as the NP-complete problem even without consideration of special application scenarios like branch outage, measurement loss, redundant observability and the existence of conventional measurements. That is the reason why most algorithms aimed at OPP problems are based on search algorithms, rather than programming methods. In next chapter, the author briefly introduces common approaches to OPP problems and mainly introduce four kinds of search algorithms for solving OPP problems.

Chapter 3

Algorithms for OPP Problem

In last chapter, the author introduces the formulation for OPP problems based on topological observability analysis, and points out OPP problems are NP-complete. In this chapter, a brief introduction to various approaches to the OPP problem is given firstly. Then four kinds of search algorithms employed to solve OPP problems are introduced in detail.

3.1 Brief Introduction to Common Approaches to OPP Problem

Since the OPP problem was proposed, many feasible algorithms or optimization methods have been formulated, like graph theoretic procedure and Dual search (a modified bisecting search and simulated annealing-based method) [5], Branch and Bound optimization method [10], Non-dominated sorting genetic algorithm [11], Tabu search [12], non-linear integer programming [13], linear integer programming [14], and heuristics-based algorithms [16]. These approaches are numerical and topological observability.

Approaches only based on topological observability were proposed as well, such as Depth First Search [15] and Minimum Spanning Tree Method. These approaches can ensure full topological observability but not ensure satisfying Equation 2.3.

Furthermore, some approaches in respect to certain special cases or situations are proposed, for instance, contingency constrained OPP techniques [17] [18] and OPP problems in the system with conventional measurements [19].

Many of those proposed optimization methods are not guaranteed to always be able to find the really optimal solution to OPP problems. The reasons leading to those unideal results are various as well. For integer programming and some local search algorithms, besides employing inaccurate constraint functions, which is mentioned in last chapter, they also may be stuck in local minima [21] [22] or local maxmima in practice, and thus miss the global optimal solution.

A more detailed summary about the common approaches to OPP problems can be found in [23].

In this project, four search algorithms, including Depth-First Search (DFS) Method, Graph Theoretic (GTh) Procedure, Simulated Annealing (SA) Method and Recursive Security N (RSN) Algorithm, are focused on. The details of these four algorithms are introduced respectively in Section 3.2, 3.3, 3.4 and 3.5. The simulation employing these algorithms are provided in Chapter 4.

3.2 Depth First Search Method

Depth First Search (DFS) Method is a kind of tree search technique. In general, DFS searches a tree repeatedly from an ancestor node directly to one of its child node until no child node exists and then go back and search from another child node of the current node's parent node. Fig. 3.1 shows the order in which nodes are visited when DFS is employed to search a tree.

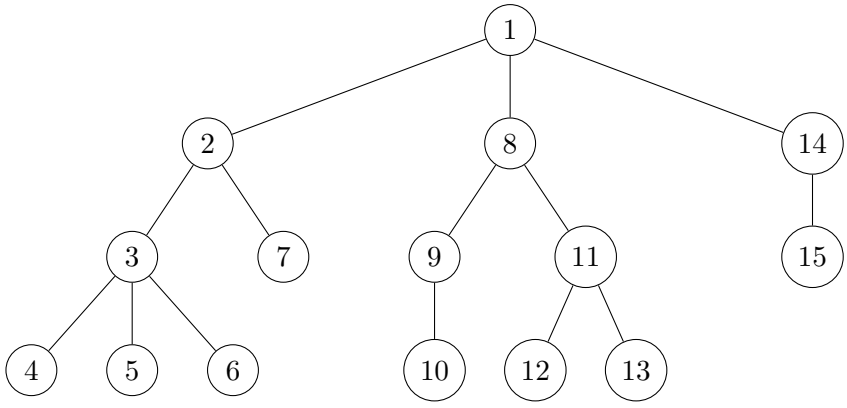

Figure 3.1: Order in Which Nodes Are Visited When DFS Is Employed

When applying to OPP problems, any node in the search tree of DFS represent a state of the given power system, the state which describes which bus is or buses are installed PMU(s) at like the vector $\bar{X}$ in Equation 2.7. A node's child node represents the state in which a new PMU is installed at a bus who has not been installed in the state the child node's pareent node represents.

DFS is complete. That is to say, DFS could traverse the whole search tree if the depth and the amount of branches of the search tree are finite and if we do not terminate search during the process. This means, in OPP problems, DFS could find out all possible schemes for PMUs placement, since any given power system always has finite buses and branches and thus has finite state space when applying DFS. Hence DFS has the potential to find the optimal solutions to OPP problems. Nevertheless, the efficiency of such strategy is quite low. For a 118-bus system, the amount of states can reach k where $k = \sum_{n=1}^{118} C_{118}^{n} = 3.3231 \times 10^{35}$. This state space is too huge, and it is just a 118-bus system which may be much smaller compared to power systems in real life. The completeness of DFS also means if there is at least one solution to OPP problems, DFS must be able to find out a solution,

Therefore, when solving OPP problems, we usually do not hope DFS to traverse the whole search true for the sake of find the optimal solution; but hope DFS to find an optimal or suboptimal solution with limited search times. This goal is achieved via a heuristic algorithm employed when DFS chooses which child node should be visited next time. The idea of this heuristic algorithm is to always choose the node who represents the state where as many buses will become observable from unobservable conditions as possible. Fig. 3.2 is the flow chart of DFS applied in OPP problems. Every time when a bus is placed a PMU at, the id of this bus should be recorded; and the final solution is a list composed of the ids of buses each of whom is placed a PMU at. As for how to judge the given system's observability, DFS only employs Equation 2.7 to construct constraint function for judging a system's observability and does not consider ZI buses.

From the flowchart, we can spot that DFS in OPP problems performs in the way of greedy algorithms. Namely, DFS here always trends to find local optimal solutions instead of global optimal solutions. And, greedy algorithms usually performs faster but is not optimal[22].

DFS is a quite basic algorithm for OPP problems. It does not consider ZI buses at

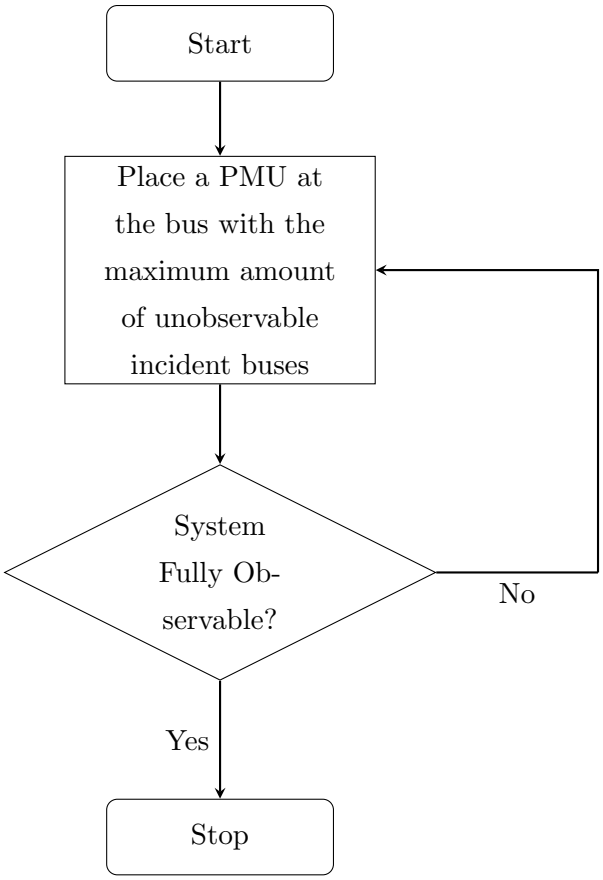

Figure 3.2: Flowchart for Applying DFS in OPP Problems

all and only employs Rule 1 to 3 or rather only employs Equation 2.7 when determining the observability of a bus. Such a simple, linear equation plus the greedy heuristic algorithm employed by DFS makes DFS able to find solutions much fast, although the price is the optimality of the solution it finds.

3.3 Graph Theoretic Procedure

Graph Theoretic (GTh) procedure is similar to DFS except that GTh takes ZI buses into account. In other words, the only difference between DFS and GTh is how to judge if the system is fully observable. GTh will adopt some technique to consider ZI buses when judging a given system's observability.

According to the introduction in Chapter 2.4, we can deal with ZI buses through merging method or nonlinear constraint function method. And, each of the two methods has some different sub-methods.

If merging method is employed, we first need to reconstruct the adjacency matrix A. The new adjacency matrix can be obtained via either of merging ZI buses on graph or merging ZI buses' constraint equations. Nevertheless, according to the analysis in Chapter 2.4.1, both of these two methods may cause redundancy and thus be unable to find the optimal solution.

Many examples have been provided in Chapter 2.4.1. From these examples, it can be recognized that a worse solution will be found if we take into account ZI buses but adopt an improper method to deal with ZI buses in some certain cases.[1] And, the worse solutions are more likely to be obtained when we adopt merging method, since this method uses a quite inaccurate constraint vector function $F(\bar{X}) = \mathbf{A}'\bar{X}$. Hence, GTh will not necessarily find a better solution compared with DFS in some cases, especially when using merging method.

Besides, since the nature of DFS here is a greedy algorithm who only cares about local optimal solutions, GTh, who has the similar pattern in how to expand child nodes, is very likely to be unable to find optimal solutions, even if it employs revising method. Nevertheless, due to that revising method itself may cause fewer redundancy, GTh, comparing to DFS, may be able to find better solutions when employing revising method. However, the price is that more resources will be used in constructing and computing the revised constraint vector function $F(\bar{X})$.

3.4 Simulated Annealing Method

Simulated Annealing (SA) search is a kind of stochastic, local search. The basic idea of local search is that the algorithm will keep searching if there is an adjacent solution, which is better than the current solution, and accept the new found solution if it is better than the current solution. Such a basic algorithm is called hill-climbing (HC)

[1] 'A Worse Solution' here means a solution worse than that found by DFS who does not take into account of ZI buses. Take Fig. 2.2 for example. When DFS is employed, Bus 3 will be chosen firstly; then the algorithm will stop due to the full observability of the system shown in Fig. 2.2; and we finally obtain an optimal solution where only one PMU is installed in the system. However, if GTh is employed and we use (3) method introduced in Chapter 2.4.1, the solution we obtain will have two PMUs.

search. The problem of HC search is that it will be stuck at the local optimal solution and thus may miss the global optimal solution. As shown in Fig. 3.3, HC search starts searching at Point A and will stop at Point B if we want it to find a maximum value at the given curve line, since all of B's adjacent points have lower value than B has. However, Point B is just the local optimal point but not the global optimal point, which should be Point C.

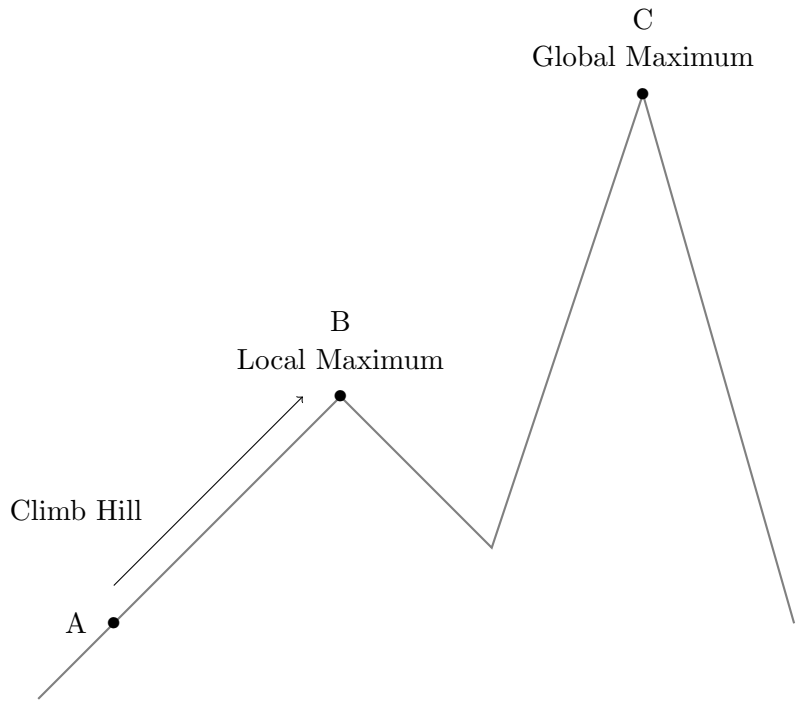

Figure 3.3: Local Search Algorithm Stuck at the Local Optimal Point

SA search simulates the motivation of atoms during the process of annealing in metallurgy. During searching, SA search will accept a worse solution in probability if there is no better solution found. SA search is expected to be able to find the global optimal solution in some probability through leaping out from the corral caused by the local optimal solution.

The application of SA search in OPP problems was originally proposed via in [5]. Its basic process is described in Fig. 3.4, and the process of simulated annealing in this method is described in Fig. 3.5. In this original version, an upper bound and

```
Input: placement                                          // obtained via GTh
Output: placement_best                          // i.e.  the best placement found
1  placement_best ← placement
2  upper_bound ← the amount of PMUs used in placement
3  lower_bound ← 0; while upper_bound − lower_bound > 1 do
4      v_test ← CalculateVtest(upper_bound, lower_bound)
5      M ← GenerateM(v_test); placement_new ← SimulatedAnnealing(v_test, M)
6      if placement_new = None then
                    // Adjust lower_bound if no better placement is found
7          lower_bound ← v_test)
8      else
                     // Adjust upper_bound if a better placement is found
9          upper_bound ← v_test
10         placement_best ← placement_new
11     end
12 end
13 return placement_best
```

Figure 3.4: Pseudo-Code for the Basic Process of Original SA Method

lower bound are set to constrain the search space. The upper bound means the possible maximum amount of PMUs presumed to make the system fully observable, and the lower bound means the possible minimum amount of PMUs presumed. The algorithm will always try to find the solution in which the amount of PMUs installed is equal to v_{test}, i.e. the half or 85% of the sum of the upper and lower bound. It is defined as the function **GenerateVtest()** in Fig 3.6. In each iteration, the search space is limited via M who is calculated out via the function **GenerateM()** shown in Fig 3.7.

During searching, the new found placement that can make the entire system observable must be the best placement found so far, since the amount of PMUs used in this new placement is equal to v_{test}, which is smaller than the *upper_bound* who is the amount of PMUs used in the best placement found before. The algorithm will accept such a new placement unconditionally, and then adjust the upper bound. If the algorithm cannot

```
Function  SimulatedAnnealing()
Input: v_test, M
Output: placement_new                          // i.e. the new placement found
Output: None                                   // if no acceptable placement found
T ← 15                                                    // Initial Temperature
Randomly generate a placement where the amount of PMUs installed is v_test
E ← the amount of unobservable buses when placement is applied
for i ← 1 to 40 do
    for j ← 1 to M do
        placement_new ← Perturb(placement)    // Generate a new placement
        E_new ← the amount of unobservable buses if placement_new is applied
        if E_new = 0 then
            /* Find an effective placement who can support the entire
               system's observability with fewer PMUs                  */
            return placement_new
        end
        ΔE ← E_new − E
        if ΔE > 0 and exp(−ΔE/T) < a random number between 0 and 1 then
            break          // Reject a futility placement in probability
        end
        placement ← placement_new
        E ← E_new
    end
    T ← 0.879 × T                                                 // Annealing
end
return None
```

Figure 3.5: Pseudo-Code for the Process of Simulated Annealing in Original SA Method

find any better placement during the iteration from 1 to M, it considers that v_{test} may be under-estimated and thus increase v_{test} via increasing the lower bound. As for how

Function $GreateVtest()$
Input: $upper\_bound$, $lower\_bound$
Output: v_{test}
if $lower\_bound < 1$ **then**
  $coefficient \leftarrow 0.85$
else
  $coefficient \leftarrow 0.5$
end
$v_{test} \leftarrow coefficient \times (upper\_bound - lower\_bound)$
/* v_{test} should be adjusted to be an integer before return */
return v_{test}

Figure 3.6: Pseudo-Code for Determining v_{test} in the Original SA Method

Function $GreateM()$
Input: v_{test}
Input: N // the total number of buses in the system
Output: M
$M \leftarrow 0.002 * C_N^{v_{test}}$
/* M should be adjusted to be an integer before return */
return M

Figure 3.7: Pseudo-Code for Determining M in the Original SA Method

to randomly generate new placements via the function $Perturb()$, this algorithm will generate a new placment via randomly moving a PMU from one bus to anther bus without PMU in the given *placement*.

This original version of SA method proposed in [5] has a lower probability to find the optimal solution, since it may miss the optimal solution via misjudging the possible amount of PMUs in the optimal placement. The author here proposes a modified SA method in which a new perturb function is applied in order to increase randomness and thus increase the probability of finding optimal solutions.

The process of the modified SA method is described in Fig. 3.8.

Input: *placement* // Initial *placement* is obtained via GTh
Output: $placement_{best}$ // i.e. the best placement found
$T \leftarrow 15$ // Initial Temperature
$placement_{best} \leftarrow placement$
$T_{diminished} \leftarrow 0$
while $T_{diminished} < N$ **do**
  // N is the total number of buses
  for $j \leftarrow 1$ ***to*** M **do**
    // M here can use the total number of buses as well
    $placement_{new} \leftarrow$ **NewPerturb**$(placement)$
    $E_{new} \leftarrow$ the amount of unobservable buses when $placement_{new}$ is applied
    if $E_{new} = 0$ ***and*** *no more PMUs in* $placement_{new}$ *compared to* $placement_{best}$ **then**
      /* Find a better placement */
      $T_{diminished} \leftarrow -1$
      $placement_{best} \leftarrow placement_{new}$
      break
    end
    $\Delta E \leftarrow E_{new} - E$
    if $\Delta E > 0$ ***and*** $exp(-\Delta E/T) <$ *a random number between 0 and 1* **then**
      break // Reject a futility placement in probability
    end
  end
  $T_{diminished} \leftarrow T_{diminished} + 1$
  $T \leftarrow 0.879 \times T$ // Annealing
end
return $placement_{best}$

Figure 3.8: Pseudo-Code for Modified SA Method Applied in OPP Problems

In the modified SA method[2] , in order to increase the randomness, in the function **NewPerturb()**, random solutions are generated via two methods: (1) to randomly move a PMU to a bus without PMU, and (2) to randomly remove or add a PMU; whereas the original SA method only applies the first method and uses the upper and lower bound to limit the amount of PMUs expected to be used. The (2) method can help the algorithm quickly approach the optimal solution without spending too much time checking if v_test is reasonable through searching. Therefore, the modified SA method is faster than the original version if a worse placement is given as the initially placement (for example, use the placement where each bus is installed a PMU at as the initial placement). Nevertheless, it may not perform better if the initial placement is close to the optimal placement (for example, the placement obtained via GTh), since the search space of this modified SA method is larger when it approaches the optimal solution. In summary, this modified SA method, in general, could have a higher probability to find optimal solution.[3]

As a matter of facts, many modifications to SA method, like applying genetic algorithm and changing the parameters, have been proposed in order to increase the probability of SA method's finding the global optimal solution and, meanwhile, to limit the search time used. Nevertheless, Since, the nature of SA search is the combination of stochastic and local search, any modified SA method still may miss the global optimal solution sometime, although it may have a higher probability to find the global optimal solution for OPP problems compared to the original SA method.

3.5 Recursive Security N Algorithm

According to the analysis in Chapter 2.1.2, the process of finding the optimal solution to OPP problems is also the process of finding out the minimum subgraph of $G = (V, E)$.

Prim algorithm is a common algorithm to find a given graph's minimum spanning tree. In Prim algorithm, all branches in a graph are removed firstly, then select any node as the initially introduced node, and then a minimum spanning tree is generated via repeatedly introducing an 'unintroduced' node by restoring a branch with minimum

[2] The initial temperature T_0 here still adopts 15. More experiment needed to demonstrate if a better initial temperature exists. See [5] for more details about why 15 is selected as the initial temperature.

[3] This conclusion is demonstrated in Chapter 4.3.

weight between the 'unintroduced' node and an introduced node who is connected by only one branch at present.

When employing minimum spanning tree algorithm in OPP problems, the most important third process is changed into introducing a PMU at the bus who, if a PMU is installed at it, will maximize the observable area with the existing PMUs. Nevertheless, this process is quite difficult to achieve, since the constraint function Equ. 2.13 is nonlinear if we consider ZI buses. Using the term in Prim algorithm, that is to say, we cannot directly, accurately determine a branch's weight.

Furthermore, sine we have no idea which bus must be able to lead to a better solution if it is selected as the initial bus at whom a PMU is installed firstly. When Prim algorithm is employed in a system with N buses, at least N minimum spanning trees need to be generated using each bus as the initial location where the fist PMU is installed. This means that the amount of spanning trees increases with the system's scale as well, which further improves the time complexity of Prim algorithm.

As a matter of fact, due to the excessive time complexity, OPP problems is almost unsolvable if we solve the problems strictly based on the algorithm for searching minimum spanning trees.

In [24], the authors proposed a modified minimum spanning tree algorithm named Recursive Security N (RSN) Algorithm. This algorithm does not consider ZI buses when generating minimum spanning trees. However, dissimilarly to DFS, this algorithm maximizes the observable area only through installing PMUs at the buses who are not observable with existing PMUs. In order to further constraint the amount of the generated minimum spanning trees, this algorithm only does the modified DFS N times for a system with N buses, and , every time uses a different bus as the initial bus at which the first PMU is install. Besides, it uses the symmetric reverse Cuthill-McKee permutation of the adjacency matrix in order to improve the quality of generated minimum spanning trees. However, since the trees generated in this step are not necessarily the real minimum spanning tree, all generated spanning trees needs to be stored for the following verification and pruning.

As minimum spanning trees are generated, several PMU placements are obtained. RSN needs to check if any alternative pattern exists for each placement. An alternative pattern is a placement that can make the system still fully observable and the placement

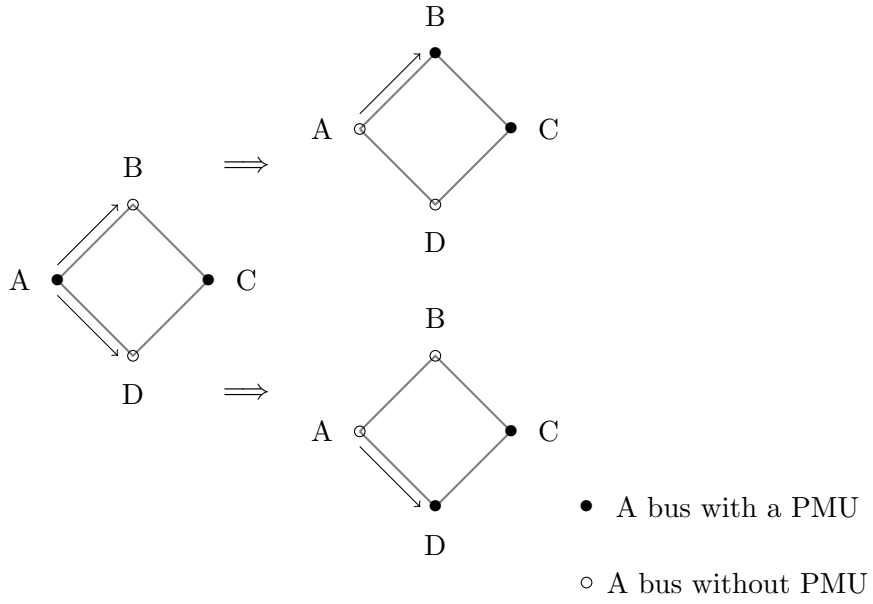

Figure 3.9: Search for Alternative Patterns in RSN Method

that is generated via moving a PMU to its adjacent buses. As what is shown in Fig. 3.9, the two sets of placement on the right side are two alternative patterns of the set of placement on the left.

The existence of any alternative pattern of a placement means that there may be any pure transit node in the placement, namely, the node who represents a bus at which a redundant PMU is installed (shown in Fig. 3.10). Due to that there is no way to directly identify which bus is a pure transit node, we have to find pure transit nodes through

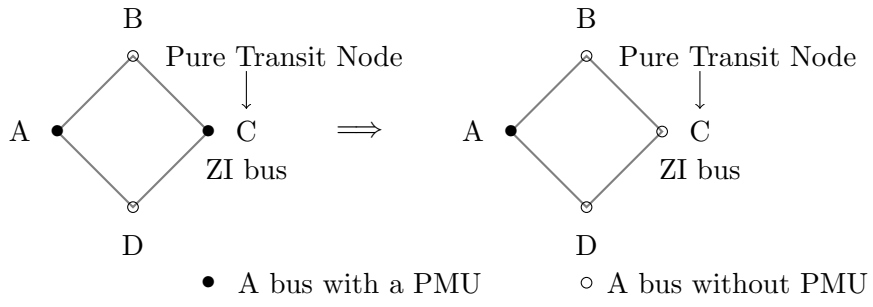

Figure 3.10: Remove Pure Transit Node in RSN Method

checking if the whole system is still fully observable after a PMU is removed from a bus. If a pure transit node is found, the PMU installed at the bus represented by the pure transit node should be removed. This process needs to be processed repeatedly until no pure transit exists for any obtained placements. In the process, more placements may generated due to the different order of removing pure transit nodes. That means that this process needs a lot of computation resources as well with a higher time complexity.

RSN Algorithm as a modified minimum spanning tree algorithm for OPP problems is practicable. It effectively limits the time and space complexity of minimum spanning tree algorithm when it is applied in OPP problems, and is able to obtain several sets of optimal or very suboptimal placements. However, the time complexity of this algorithm still could increase exponentially as the system's scale increases.

Chapter 4

Case Studies

In this chapter, IEEE 14-bus, 30-bus, 57-bus and 118-bus systems and New England 39-bus system will be employed for observability analysis using DFS, GTh, SA and RSN methods. All simulations are run at Matlab (R2014b) on a computer with MacOS, 2.7 GHz i7 CPU and 16GB 1600MHz DDR3 Memory.

All simulations are conducted without consideration of redundant observability; and for any bus i, w_i is assumed to be 1, namely, it is considered that the cost of installing a PMU at any bus is the same.

A list of used programs are provided in Chapter 5. The code of these programs are submitted with this report separately.

4.1 Basic Information of Test Cases

The basic information of IEEE 14-bus, 30-bus, 57-bus and 118-bus systems and New England 39-bus system is listed in Table 4.1. They are called 14-bus, 30-bus, 57-bus, 118-bus and 39-bus system in the following part of this chapter.

4.2 Simulation Result for DFS and GTh

The optimal placements found by DFS, GTh using the method of merging ZI buses' constraint equations and GTh using nonlinear constraint function method are respectively shown in Table 4.2, 4.3, 4.4.

Table 4.1: Basic Information of Test Cases

System	No. of Branches	ZI buses
14-bus	20	7
30-bus	41	6, 9, 22, 25, 27, 28
57-bus	78	4, 7, 11, 21, 22, 24, 26, 34, 36, 37, 39, 40, 45, 46, 48
118-bus	179	5, 9, 30, 37, 38, 63, 64, 68, 71, 81
39-bus	46	2, 5, 6, 10, 11, 13, 14, 17, 19, 22

From the three tables, it can be recognized that when GTh using merging method is adopted, the quality of the solution is quite bad. It is even worse than DFS who does not care about ZI buses at all. This is consistent with our analysis in Chapter 3.3. This result further demonstrates that merging method is very likely to cause redundancy. And this property makes the solution found by GTh using merging method is not better and even worse (in the IEEE 57-bus test case) than that found by DFS. Merging method could not effectively employ the feature of ZI buses. The author here argues that merging method, which is mentioned as a useful method in many papers, in fact is a futile mean to deal with ZI buses. As for the solutions found by GTh using nonlinear constraint function method, most of them are better than those found by DFS, and

Table 4.2: Optimal Solutions Found by DFS

System	No. of PMUs	Location of PMUs	Time Taken
14-bus	6	1, 4, 6, 8, 10, 14	0.001791s
30-bus	10	1, 5, 6, 10, 11, 12, 18, 24, 26, 27	0.006418s
57-bus	19	1, 4, 7, 9, 15, 19, 21, 24, 27, 30, 32, 36, 38, 39, 41, 46, 50, 52, 54	0.019477s
118-bus	42	1, 5, 9, 12, 13, 17, 19, 21, 25, 28, 30, 32, 36, 37, 41, 43, 46, 49, 52, 57, 58, 59, 62, 65, 70, 72, 73, 77, 80, 83, 86, 89, 91, 93, 95, 100, 102, 105, 110, 115, 116, 118	0.096274s
39-bus	16	2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 26, 33, 36, 37, 38, 39	0.010599s

Table 4.3: Optimal Solutions Found by GTh using the Method of Merging ZI Buses' Constraint Equations

System	No. of PMUs	Location of PMUs	Time Taken
14-bus	6	1, 4, 6, 8, 10, 14	0.001685s
30-bus	10	1, 5, 6, 10, 11, 12, 18, 24, 26, 27	0.006093s
57-bus	21	2, 4, 7, 9, 15, 16, 17, 19, 21, 24, 27, 30, 32, 36, 38, 39, 41, 46, 50, 52, 54	0.021701s
118-bus	42	1, 5, 9, 12, 13, 17, 19, 21, 25, 28, 30, 32, 36, 37, 41, 43, 46, 49, 52, 57, 58, 59, 62, 65, 70, 72, 73, 77, 80, 83, 86, 89, 91, 93, 95, 100, 102, 105, 110, 115, 116, 118	0.095118s
39-bus	16	2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 26, 33, 36, 37, 38, 39	0.010599s

Table 4.4: Optimal Solutions Found by GTh using Nonlinear Constraint Function Method

System	No. of PMUs	Location of PMUs	Time Taken
14-bus	5	1, 4, 6, 10, 14	0.024160s
30-bus	8	1, 5, 6, 10, 12, 18, 24, 27	0.112373s
57-bus	19	1, 4, 7, 9, 15, 19, 21, 24, 27, 30, 32, 36, 38, 39, 41, 46, 50, 52, 54	0.472040s
118-bus	40	1, 5, 9, 12, 13, 17, 19, 21, 25, 28, 30, 32, 36, 37, 41, 43, 46, 49, 52, 57, 58, 59, 62, 65, 70, 72, 77, 80, 83, 86, 89, 91, 93, 95, 100, 102, 105, 110, 115, 118	1.296038s
39-bus	16	2, 4, 6, 8, 10, 12, 16, 18, 20, 22, 26, 33, 36, 37, 38, 39	0.228232s

none of them is worse.

Besides, all the three methods are very fast. Their nature as greedy algorithm is able to guarantee that their time complexity is very limited. Nevertheless, in most situations,

we hope OPP problems can be solved via a very optimal solution within acceptable time, rather than within as little time as possible. That is to say, the optimality of the solution found by an algorithm is a more important criterion to evaluate that algorithm. Taking into account of this criterion, both of DFS and GTh are not a valuable algorithm for OPP problems, since the optimality of solutions found by them is not good, at least compared to SA and RSN, which are analyzed in the next two sections.

4.3 Simulation Result for SA

Two version of SA methods will be analyzed in this section. One is the original version proposed in [5]. The other one is the modified version proposed in Chapter 3.4. They both use nonlinear constraint function method to deal with ZI buses. In order to analyze the probabilities of the two methods' finding optimal solutions, simulation employing these two methods was conduct 20 times for each test system. Table 4.5 records the most optimal solutions found by the the two methods respectively. Table 4.6 records the distribution of the solutions found by the two algorithms.

From Table 4.5 and 4.6, it can be recognized that the original SA method performed not very well. For the IEEE 118-bus test system, the original SA method even did not find any placement better than the initial placement obtained via GTh. In contrast, the modified SA method proposed in Chapter 3.4 performed not worse both in the optimality of the best placements found and in the probability of finding suboptimal placements. Although the modified SA method needs more time to finish a search, the time taken via this algorithm is still acceptable (in the consideration of that it needs about fifteen minutes to find a placement for the 118-bus system).

Besides, it can be spot that SA method is not a very stable method. It may find the optimal solution and, meanwhile, may only find the suboptimal solution. However, in general, compared to either of DFS or GTh, SA method is able to find better solutions, at least not worse.

Table 4.5: Optimal Solutions Found by SA Method

System	Method	Mini. No. of PMUs[a]	Times[b](t)	Prob. ($t/20$)	Time Taken[c]
14-bus	Original SA	3	1	0.05	0.3061s
	Modified SA	3	2	0.1	0.9093s
30-bus	Ori SA	7	1	0.05	4.3420s
	Modified SA	7	12	0.60	16.5214s
57-bus	Original SA	16	1	-	15.3289s
	Modified SA	13	2	0.10	157.5678s
118-bus	Original SA	40	20	-	34.5018s
	Modified SA	33	1	0.05	920.9419s
39-bus	Original SA	13	3	-	9.0349s
	Modified SA	10	1	0.05	46.4288s

[a] 'Mini. No. of PMUs' here means the amount of PMUs needed in the best placement found by an algorithm.

[b] 'Times' here means how many times an algorithm found a placement where the amount of PMUs installed is the minimum it found.

[c] 'Time Taken' here means the time taken for an algorithm to finish a search in average.

4.4 Simulation Result for RSN

Recursive Security N (RSN) Algorithm here is employed to solve OPP problems for the five test systems based on the idea of minimum spanning tree. Differing from the above-mentioned three algorithms, RSN, if possible, could return multiple sets of placement in general, the sets all of whom are the optimal among those it finds.[1] Table 4.7 records the simulation for RSN algorithm.

From Table 4.7, RSN performed quite well among these four algorithms. It is a stable algorithm who will always return the same result for a certain system within acceptable time.

The RSN algorithm implemented here uses the symmetric reverse Cuthill-McKee permutation to reverse the adjacency matrix firstly, and only tries to obtain optimal

[1] This property can be achieved in the above-mentioned algorithms as well. However, not too many papers mentioned that.

Table 4.6: Placement Sets Found by SA Method

System	No. of PMUs Needed in the set	Times Found[a] by Original SA	Times Found[a] by Modified SA
14-bus	3	1	2
	4	15	16
	5	4	2
30-bus	7	1	12
	8	19	8
57-bus	13	-	2
	14	-	11
	15	-	6
	16	1	1
	17	1	-
	18	6	-
	19	12	-
118-bus	33	-	1
	36	-	3
	37	-	14
	38	-	2
	40	20	-
39-bus	10	-	1
	11	-	10
	12	-	9
	13	4	-
	14	8	-
	15	8	-

[a] 'Times Found' here means how many times an algorithm found a placement in which the corresponding number of PMUs are installed.

placements through removing pure transit nodes from those placements who have alternative patterns and who use the minimum amount of PMUs among all placements

Table 4.7: Placement Sets Found by RSN Algorithm

System	No. of PMUs	No. of Sets Found[a]	Time Taken
14-bus	3	1	0.067584s
30-bus	7	4	10.954884s
57-bus	12	2	644.239306s
118-bus	36	4	73.284955s
39-bus	9	1	52.229655s

[a] 'No. of Sets Found' here means how many sets of placements, who had the same amount of PMUs, the algorithm found.

generated via minimum spanning tree algorithm. The difference in such details, like how to deal with alternative patterns and how to generate minimum spanning trees, may lead to the difference in the final result returned by the algorithm. In [24], the authors argued only 31 PMUs are needed in the optimal placement found by RSN for IEEE 118-bus system. However, the author here did not obtain such a good placement via RSN. The author thinks the reason may be some difference in the details of how to implement the algorithm.

Chapter 5

Matlab Programs

In this project, 11 files of programs were coded. They includes 5 files who record some basic data of the five test systems, and 6 files who are the implements for 6 kinds of algorithms for OPP problems, and 1 file who is the main entrance file of this set of programs. All these programs run well in Matlab R2014b. A detailed list of these files is provided in this chapter.

5.1 Programs about Studied Systems

- $case14.m$ Data of IEEE 14-bus system.
- $case30.m$ Data of IEEE 30-bus system.
- $case39.m$ Data of New England 39-bus system.
- $case57.m$ Data of IEEE 57-bus system.
- $case118.m$ Data of IEEE 118-bus system.

All these file are from MATPOWER. In this project, the systems' branch data provided via these files are used to construct the adjacent matrices, and the bus data and generator data are used to identify zero injection buses.

5.2 Algorithm Programs

In the experiment part of this project, six algorithms for OPP problems are implemented. They include DFS, GTh using merging method, GTh using revising constraint function method, original SA method, modified SA method proposed in this report, and RSN algorithm. The six algorithms are packaged in the form of function contained in six program files. All the six programs are originated and follow the MIT license.

5.2.1 *OPP_DFS.m*

This file is an implement for Depth First Search Method in OPP problems. It contains a function **OPP_DFS**. See Chapter 3.2 for details of DFS algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

Output:

- **placement** Type: a $1 \times m$ array.

  The optimal PMU placement found by DFS. m is the total number of PMUs used in this placement.

- **msg** Some console information.
  - **msg.time** Type: float.

    The time taken by the algorithm to find the placement. Unit: second.
  - **msg.method** Type: string.

    The name of the search method used.

5.2.2 *OPP_GThM.m*

This file is an implement for Graph Theoretic Procedure using Merging ZI Buses' Constraint Equation Method in OPP problems. It contains a function **OPP_GThM**. See Chapter 3.3 and 2.4.1 for details of this algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

- **ZI_buses** Type: a $1 \times z$ array.

  The array composed by the id of zero injection buses. z is the total number of zero injection buses in the system.

Output:

- **placement** Type: a $1 \times m$ array.

  The optimal PMU placement found by this algorithm. m is the total number of PMUs used in this placement.

- **msg** Some console information.

  - **msg.time** Type: float.

    The time taken by the algorithm to find the placement. Unit: second.

  - **msg.method** Type: string.

    The name of the search method used.

5.2.3 *OPP_GThN.m*

This file is an implement for Graph Theoretic Procedure using Nonlinear Constraint Function Method in OPP problems. It contains a function **OPP_GThM**. See Chapter 3.3 and 2.4.2 for details of this algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

- **ZI_buses** Type: a $1 \times z$ array.

  The array composed by the id of zero injection buses. z is the total number of zero injection buses in the system.

Output:

- **placement** Type: a $1 \times m$ array.

  The optimal PMU placement found by this algorithm. m is the total number of PMUs used in this placement.

- **msg** Some console information.

  - **msg.time** Type: float.

    The time taken by the algorithm to find the placement. Unit: second.

  - **msg.method** Type: string.

    The name of the search method used.

5.2.4 *OPP_SA.m*

This file is an implement for the original Simulated Annealing method in OPP problems. It contains a function **OPP_SA**. See Chapter 3.4 for details of this algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

- **ZI_buses** Type: a $1 \times z$ array.

  The array composed by the id of zero injection buses. z is the total number of zero injection buses in the system.

Output:

- **placement** Type: a $1 \times m$ array.

  The optimal PMU placement found by this algorithm. m is the total number of PMUs used in this placement.

- **msg** Some console information.

  - **msg.time** Type: float.

    The time taken by the algorithm to find the placement. Unit: second.

  - **msg.method** Type: string.

    The name of the search method used.

5.2.5 *OPP_SAB.m*

This file is an implement for the modified Simulated Annealing Method proposed by the author in this report. It contains a function **OPP_SAB**. See Chapter 3.4 for details of this algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

- **ZI_buses** Type: a $1 \times z$ array.

  The array composed by the id of zero injection buses. z is the total number of zero injection buses in the system.

Output:

- **placement** Type: a $1 \times m$ array.

  The optimal PMU placement found by this algorithm. m is the total number of PMUs used in this placement.

- **msg** Some console information.

- **msg.time** Type: float.

  The time taken by the algorithm to find the placement. Unit: second.

- **msg.method** Type: string.

  The name of the search method used.

5.2.6 *OPP_RSN.m*

This file is an implement for Recursive Security N Algorithm in OPP problems. It contains a function **OPP_RSN**. See Chapter 3.5 for details of this algorithm.

Input:

- **A** Type: a $N \times N$ matrix.

  The adjacency matrix of the given system. N is the total number of buses in the given system.

- **ZI_buses** Type: a $1 \times z$ array.

  The array composed by the id of zero injection buses. z is the total number of zero injection buses in the system.

Output:

- **placements** Type: a $n \times m$ matrix.

  The optimal PMU placements found by RSN. n. is the number of optimal placements found by the algorithm. m is the total number of PMUs used in each optimal placement.

- **msg** Some console information.

  - **msg.time** Type: float.

    The time taken by the algorithm to find the placement. Unit: second.

  - **msg.method** Type: string.

    The name of the search method used.

5.3 The *main* Program

5.3.1 *main.m*

This is the entrance file of this set of programs. It could load the relative *case.m* according to user's choice, generate the adjacency matrix, identify the zero injection buses, and run simulation using the chosen algorithm.

5.3.2 *auto_test.m*

This file could automatically load the case files one by one, and use the six algorithms to solve OPP problems for each test system for 20 times.

Chapter 6

Conclusion and Future Work

PMUs as a kind of advanced measurement devices in power systems at present provides us a more accurate and convenient way to evaluate power systems' state. However, its relatively high price make it a problem to optimize the placement of PMUs.

In this project, the rules of PMU placement and the formulation of OPP problems based on topological observability analysis are introduced in details. A revision to the nonlinear constraint function used in the formula of OPP problems is proposed in order to make the function accurately reflect each bus's observability.

In the algorithm chapter, i.e. Chapter 3, a brief introduction to common approaches to OPP problems are provided, and four kinds of algorithms for OPP problems, including Depth First Search (DFS), Graph Theoretic Procedure (GTh) and Simulated Annealing (SA) Method, are introduced minutely. The author here proposed a modified SA method who has a higher probability find optimal or suboptimal placement compared to the original SA method proposed in [5].

Simulation for IEEE 14-bus, 30-bus, 57-bus and 118-bus and New England 39-bus system using the above-mentioned four algorithms were conducted. The simulation results are shown in the case studies chapter, i.e. Chapter 4.

From the simulation, DFS and GTh are demonstrated to be the most fast algorithm among the four algorithms, as the analysis in Chapter 3.2 and 3.3. Through the comparison of the simulation result for GTh using merging method and that for GTh using nonlinear constraint function method, it is proven that merging method is not a very useful method to deal with zero injection buses. It is in the accordance the analysis

in Chapter 2.4.1, who analyzes how and why the merging method cannot work as we expected.

Besides, in the simulation, the proposed modified SA method is shown performing better than the original SA method. Although the modified one needs relatively more time to finish a search, the time taken by this algorithm is still acceptable.

Recursive Security N Algorithm as an implement for the idea of minimum spanning tree in OPP problems performs pretty well in the simulation. While the time taken by this algorithm is similar to the proposed modified SA method, it is much more stable than SA method.

In consideration of that all simulation was conducted at Matlab, the time taken by all algorithms is expected to reduce through compiling the code in C or C++. The author may try it later.

On the other hands, none of these four algorithms can guarantee that it must be able to always find the really optimal placements.[1] It is frustrated that you find none of your conscientious work can achieve the best result.

In the next phase, most of the author's attention may be paid into developing a new algorithm for OPP problems, not just modifying the existent algorithms. The author may try to introduce the concept of machine learning and/or use some ways of data mining to organize the new algorithm.

[1] For IEEE 57-bus system, the optimal placement needs 11 PMUs. For IEEE 118-bus system, the optimal placement needs 28 PMUs. For New England 39-bus system, the optimal placement needs 8 PMUs. [4]

References

[1] G. R. Krumpholz, K. A. Clements and P. W. Davis, *Power System Observability: A Practical Algorithm Using Network Topology*, 1980.

[2] KEMA, Inc. *Substation Communications: Enabler of Automation / An Assessment of Communications Technologies*, 2006.

[3] F. Schweppe, .J. Wildes, and D. Rom, *Power System Static State Estimation: Parts I, II, and III*, 1969.

[4] B.K. Saha Roy , A.K. Sinha and A.K. Pradhan, *An Optimal PMU Placement Technique for Power System Observability*, 2012.

[5] T.L. Baldwin, L. Milli, M.B. Boisen and R. Adapa, *Power System Observability with Minimal Phasor Measurement Placement*, 1993.

[6] B. Mohammadi-Ivatloo, *Optimal Placement of PMUs for Power System Observability Using Topology Based Formulated Algorithms*, 2009.

[7] K. Mazlumi, M. Azari and S. Beheshti, *Optimal Multistage Scheduling of PMU Placement for Power System Observability*, 2012.

[8] G. R. Tankasala, S. Sanisetty and V. Vala, *Optimal Placement of Phasor Measurement Units for State Estimation using Artificial Intelligence Techniques*, 2012.

[9] T.W. Haynes, S.M. Hedetniemi, S.T. Hedetniemi and M.A. Henning, *Domination in Graphs Applied to Electric Power Networks*, 2002.

[10] B. Mohammadi-Ivatloo, *Optimal PMU Placement for Power System Observability Considering Secondary Voltage Control*, 2003.

[11] B. Milosevic and M. Begovic *Nondominated Sorting Genetic Algorithm for Optimal Phasor Measurement Placement*, 2003.

[12] J. Peng,Y. H. Sun and H.F. Wang, *Optimal PMU Placement for Full Network Observability Using Tabu Search Algorithm*, 2003.

[13] B. Xu and A. Abur, *Observability Analysis and Measurement Placement for System with PMUs*, 2004.

[14] B. Gou, *Generalized Integer Linear Programming Formulation for Optimal PMU Placement*, 2008.

[15] R.F. Nuqui and A.G. Phadke, *Phasor Measurement Unit Placement Techniques for Complete and Incomplete Observability*, 2005.

[16] B.K. SahaRoy, A.K. Sinha and A.K. Pradhan, *Optimal Phasor Measurement Unit Placement for Power System Observability A Heuristic Approach*, 2011.

[17] C. Rakpenthai, S. Premrudeepreechachacharan, S. Uatrongjit and N. Watson, *An Optimal PMU Placement Method Against Measurement Loss and Branch Outage*, 2007.

[18] F. Aminifar, A. Khodaei, M.F. Firuzabad and M. Shahidehpour, *Contingency Constrained PMU Placement in Power Networks*, 2010.

[19] R. Kavasseri and S.K. Srinivasan, *Joint Optimal Placement of PMU and Conventional Measurements in Power Systems*, 2010.

[20] G. Venugopal, R. Veilumuthu and P. A. Theresa, *Optimal PMU Placement and Observability of Power System using PSAT*.

[21] S. Chakrabarti and E. Kyriakides, *Optimal Placement of Phasor Measurement Units for Power System Observability*, 2008.

[22] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2009.

[23] N. M. Manousakis, G. N. Korres and P. S. Georgilakis, *Taxonomy of PMU Placement Methodologies*, 2012.

[24] G.B. Denegri, M. Invernizzi, F. Milano, M. Fiorina, and P. Scarpellini, *A Security Oriented Approach to PMU Positioning for Advanced Monitoring of a Transmission Grid*, 2002.

[25] Y. Nguegan and N. Tchokonte, *Real-time Identification and Monitoring of the Voltage Stability Margin in Electric Power Transmission Systems using Synchronized Phasor Measurements*, 2009.

Appendix A

Acronyms

Table A.1: Acronyms

Acronym	Meaning
PMU	Phasor Measurement Unit
GPS	Global Positioning System
OPP	Optimal PMU Placement
ZI Buses	Zero Injection Buses
DFS	Depth First Search
GTh	Graph Theoretic (Procedure)
SA	Simulated Annealing (Algorithm)
MST	Minimum Spanning Tree (Algorithm)
RSN	Recursive Security N (Algorithm)

Appendix B

Output of Matlab Program

This appendix shows the output of Matlab program. This program solved OPP problems for IEEE 14-bus, 30-bus, 57-bus and 118-bus system and New England 39-bus system employing DFS, GTh using merger method, GTh using nonlinear constraint function method, original SA method, modified SA method proposed in this report, and RSN algorithm. The program solved OPP problems via the two SA methods for 20 times in order to compare their probabilities of finding better solutions.

DFS: Depth First Search
GThM: Graphic Theoretic Procedure using Merger Method
GThN: Graphic Theoretic Procedure using Nonlinear Constraint Function Method
SA: Simulated Annealing Method (Original Version proposed in 1993)
SAB: Simulated Annealing Method (Modified Version proposed in the report)
RSN : Recursive Security N Algorithm

Test System	Method Name	No. of PMUs	Time Taken (second)	Placement (Bus No.)
14-bus	DFS	6	0.001976	1,4,6,8,10,14
14-bus	GThM	6	0.001751	1,4,6,8,10,14
14-bus	GThN	5	0.023269	1,4,6,10,14
14-bus	SA	4	0.400628	2,5,6,9
14-bus	SA	4	0.210347	4,5,6,9
14-bus	SA	5	0.368545	1,4,6,10,14
14-bus	SA	4	0.214343	2,7,10,13
14-bus	SA	5	0.373459	1,4,6,10,14
14-bus	SA	4	0.334748	1,4,6,9
14-bus	SA	5	0.391334	1,4,6,10,14
14-bus	SA	5	0.358548	1,4,6,10,14
14-bus	SA	4	0.191457	2,7,10,13
14-bus	SA	4	0.238645	4,5,6,9
14-bus	SA	4	0.161075	2,9,11,13
14-bus	SA	4	0.361106	2,9,10,13
14-bus	SA	4	0.213695	1,4,6,9
14-bus	SA	4	0.141350	2,9,11,13
14-bus	SA	3	0.414616	2,6,9
14-bus	SA	4	0.435659	4,5,10,13
14-bus	SA	4	0.370105	2,9,11,12
14-bus	SA	4	0.330718	2,7,11,13
14-bus	SA	4	0.361561	3,5,6,9
14-bus	SA	4	0.251029	2,8,10,13
14-bus	SAB	4	1.116902	2,4,10,13
14-bus	SAB	4	1.404378	1,4,6,9
14-bus	SAB	4	0.700514	1,4,10,13
14-bus	SAB	4	0.698070	1,3,6,9
14-bus	SAB	3	0.914807	2,6,9
14-bus	SAB	5	0.713897	1,4,6,10,14
14-bus	SAB	4	1.126273	1,3,6,9
14-bus	SAB	4	0.985165	2,4,10,13
14-bus	SAB	4	0.695731	1,4,10,13
14-bus	SAB	4	0.738061	1,4,6,9
14-bus	SAB	3	1.047284	2,6,9
14-bus	SAB	4	0.714231	2,4,10,13
14-bus	SAB	4	0.697082	1,3,6,9
14-bus	SAB	4	1.228053	1,4,10,13
14-bus	SAB	4	0.855026	2,8,11,13
14-bus	SAB	4	0.708633	1,4,10,13
14-bus	SAB	4	1.267305	1,4,6,9
14-bus	SAB	4	1.177024	1,4,10,13
14-bus	SAB	4	0.696330	1,4,10,13
14-bus	SAB	5	0.701509	1,4,6,10,14
14-bus	RSN	3	0.069142	2,9,6

Test System	Method Name	No. of PMUs	Time Taken (second)	Placement (Bus No.)
30-bus	DFS	10	0.006018	1,5,6,10,11,12,18,24,26,27
30-bus	GThM	10	0.006258	1,5,6,10,11,12,18,24,26,27
30-bus	GThN	8	0.113341	1,5,6,10,12,18,24,27
30-bus	SA	8	4.985187	1,5,6,10,12,18,24,27
30-bus	SA	8	4.891549	1,5,6,10,12,18,24,27
30-bus	SA	8	3.306658	1,5,6,10,12,18,24,27
30-bus	SA	8	5.141082	1,5,6,10,12,18,24,27
30-bus	SA	8	4.814823	1,5,6,10,12,18,24,27
30-bus	SA	8	5.109725	1,5,6,10,12,18,24,27
30-bus	SA	8	3.462940	1,5,6,10,12,18,24,27
30-bus	SA	8	5.126343	1,5,6,10,12,18,24,27
30-bus	SA	8	5.712235	1,5,6,10,12,18,24,27
30-bus	SA	8	3.806914	1,5,6,10,12,18,24,27
30-bus	SA	8	4.709019	1,5,6,10,12,18,24,27
30-bus	SA	8	3.786448	1,5,6,10,12,18,24,27
30-bus	SA	8	3.895860	1,5,6,10,12,18,24,27
30-bus	SA	7	4.206429	2,4,10,12,19,24,29
30-bus	SA	8	3.983722	1,5,6,10,12,18,24,27
30-bus	SA	8	3.711355	1,5,6,10,12,18,24,27
30-bus	SA	8	3.664384	1,5,6,10,12,18,24,27
30-bus	SA	8	4.029376	1,5,6,10,12,18,24,27
30-bus	SA	8	4.131098	1,5,6,10,12,18,24,27
30-bus	SA	8	4.366501	1,5,6,10,12,18,24,27
30-bus	SAB	8	12.421121	1,5,6,10,12,18,24,27
30-bus	SAB	7	16.193801	1,5,10,12,18,24,27
30-bus	SAB	7	16.947689	1,5,10,12,15,18,27
30-bus	SAB	7	13.366350	1,5,10,12,18,24,27
30-bus	SAB	7	24.050341	1,5,10,12,18,24,27
30-bus	SAB	7	25.893899	1,5,10,12,18,24,27
30-bus	SAB	7	21.453570	1,5,10,12,18,24,27
30-bus	SAB	7	21.021144	1,5,10,12,18,24,27
30-bus	SAB	7	13.835493	1,5,10,12,18,24,27
30-bus	SAB	8	13.693924	1,5,6,10,12,18,24,27
30-bus	SAB	7	21.584478	1,5,10,12,18,24,27
30-bus	SAB	8	12.521783	1,5,6,10,12,18,24,27
30-bus	SAB	8	12.497287	1,5,6,10,12,18,24,27
30-bus	SAB	8	12.261112	1,5,6,10,12,18,24,27
30-bus	SAB	8	12.722759	1,5,6,10,12,18,24,27
30-bus	SAB	7	15.930417	1,5,10,12,18,24,27
30-bus	SAB	7	23.394058	1,5,10,12,18,24,27
30-bus	SAB	8	12.643671	1,5,6,10,12,18,24,27
30-bus	SAB	8	12.827971	1,5,6,10,12,18,24,27
30-bus	SAB	7	15.168059	1,5,10,12,18,24,27
30-bus	RSN	7	11.037119	3,19,7,12,10,27,24 - 3,19,7,12,10,30,24 - 3,19,2,12,10,27,24 - 3,19,2,12,10,30,24

Test System	Method Name	No. of PMUs	Time Taken (second)	Placement (Bus No.)
39-bus	DFS	16	0.011258	2,4,6,8,10,12,16,18,20,22,26,33,36,37,38,39
39-bus	GThM	16	0.011139	2,4,6,8,10,12,16,18,20,22,26,33,36,37,38,39
39-bus	GThN	16	0.227507	2,4,6,8,10,12,16,18,20,22,26,33,36,37,38,39
39-bus	SA	15	11.052120	2,4,6,9,12,13,18,20,23,25,26,29,35,37,39
39-bus	SA	15	8.437277	4,6,9,10,13,17,18,19,20,22,23,25,28,29,30
39-bus	SA	14	7.981179	3,6,9,13,16,22,24,29,30,32,34,36,37,38
39-bus	SA	15	11.046832	3,6,7,11,16,19,20,23,24,25,29,32,33,37,39
39-bus	SA	15	9.350800	2,3,8,9,13,16,21,23,25,26,27,33,34,35,38
39-bus	SA	15	11.333231	2,6,7,10,15,16,17,22,24,25,26,29,34,36,39
39-bus	SA	13	13.354932	4,7,8,11,17,20,22,23,25,26,32,38,39
39-bus	SA	14	8.322369	1,5,8,11,18,19,20,22,23,25,26,32,35,38
39-bus	SA	15	7.777543	5,7,9,10,13,18,20,22,23,27,29,30,31,35,37
39-bus	SA	13	10.491265	1,3,6,10,15,17,19,20,22,23,29,37,39
39-bus	SA	14	5.424760	1,4,6,9,17,18,21,24,29,32,33,34,36,37
39-bus	SA	15	12.833846	1,3,8,10,12,13,14,16,19,20,23,25,29,32,33
39-bus	SA	13	10.825685	4,6,11,16,17,23,24,26,29,30,34,37,39
39-bus	SA	14	6.938016	4,6,15,16,18,21,25,26,30,32,34,36,38,39
39-bus	SA	14	8.458553	1,7,9,10,16,17,18,19,22,29,34,36,37,38
39-bus	SA	15	7.882252	3,7,10,14,15,17,21,23,27,28,33,34,37,38,39
39-bus	SA	13	10.561736	1,2,6,7,9,16,18,23,25,28,29,32,34
39-bus	SA	14	6.101397	1,2,3,6,14,18,20,23,26,29,32,35,37,39
39-bus	SA	14	7.756777	2,8,9,11,19,21,23,27,28,29,31,32,34,37
39-bus	SA	14	4.767571	2,3,8,10,16,21,25,26,28,34,36,37,38,39
39-bus	SAB	11	61.747274	2,7,12,16,20,22,26,36,37,38,39
39-bus	SAB	12	44.577056	2,4,8,12,16,20,22,26,36,37,38,39
39-bus	SAB	12	39.641055	2,4,8,16,20,22,26,32,36,37,38,39
39-bus	SAB	11	46.117356	6,10,16,18,20,21,29,30,36,37,39
39-bus	SAB	10	46.557233	2,8,13,16,20,23,26,37,38,39
39-bus	SAB	11	66.316458	2,6,12,16,20,22,23,26,37,38,39
39-bus	SAB	12	59.737376	6,9,12,18,20,22,24,25,26,36,38,39
39-bus	SAB	12	37.029808	1,7,8,10,16,18,20,25,26,35,36,38
39-bus	SAB	11	48.257757	2,8,9,13,16,17,20,23,26,37,38
39-bus	SAB	12	34.701709	2,6,8,12,16,20,22,26,36,37,38,39
39-bus	SAB	12	34.029254	1,8,11,14,16,18,22,26,34,36,37,38
39-bus	SAB	12	46.901605	2,4,8,12,16,20,22,26,36,37,38,39
39-bus	SAB	11	39.250096	2,6,9,12,16,20,22,26,29,36,37
39-bus	SAB	11	48.899197	2,6,10,16,20,22,26,36,37,38,39
39-bus	SAB	12	48.786337	2,8,10,16,20,22,26,31,36,37,38,39
39-bus	SAB	11	54.758912	2,6,9,10,16,20,22,26,29,36,37
39-bus	SAB	11	47.315122	2,8,13,16,26,34,35,36,37,38,39
39-bus	SAB	11	43.573262	2,6,10,16,18,20,22,29,36,37,39
39-bus	SAB	12	35.596871	2,4,6,12,16,18,20,21,29,36,37,39
39-bus	SAB	11	44.782997	2,6,13,16,20,22,25,26,36,38,39
39-bus	RSN	9	52.803874	20,23,29,16,25,1,13,3,8

Test System	Method Name	No. of PMUs	Time Taken (second)	Placement (Bus No.)
57-bus	DFS	19	0.018877	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	GThM	21	0.022881	2,4,7,9,15,16,17,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	GThN	19	0.484104	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	13.801445	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	14.985099	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	18	15.303917	1,4,9,13,17,19,23,27,28,30,32,39,42,44,47,50,52,54
57-bus	SA	18	19.086622	1,6,12,15,16,19,23,29,30,33,34,43,44,49,50,53,55,56
57-bus	SA	17	8.751567	1,6,10,13,14,15,20,25,28,30,32,38,45,51,53,54,56
57-bus	SA	19	17.832585	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	20.537740	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	18	14.956422	1,2,7,12,13,17,18,22,26,29,30,32,38,47,50,52,54,56
57-bus	SA	19	15.162944	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	16	18.317980	1,4,9,15,20,24,29,31,33,38,43,45,46,51,54,56
57-bus	SA	19	17.169125	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	18	9.998793	1,7,8,9,15,16,18,22,24,29,30,33,36,38,42,51,54,56
57-bus	SA	19	13.018819	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	15.598523	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	14.076289	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	18	15.598689	1,3,4,8,9,18,20,24,27,31,32,38,43,47,50,53,56,57
57-bus	SA	18	12.763367	1,2,6,9,12,18,20,25,29,32,39,42,44,46,49,50,53,54
57-bus	SA	19	15.605927	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	18.064713	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SA	19	15.948079	1,4,7,9,15,19,21,24,27,30,32,36,38,39,41,46,50,52,54
57-bus	SAB	14	131.329460	1,4,9,19,27,30,32,38,40,41,46,50,52,54
57-bus	SAB	14	200.985312	1,5,9,15,20,27,31,32,36,41,48,50,52,54
57-bus	SAB	14	170.444095	1,4,9,14,19,27,30,32,36,38,41,50,52,54
57-bus	SAB	15	176.397004	1,4,7,9,15,19,24,27,30,32,38,39,41,50,53
57-bus	SAB	14	156.606591	1,4,9,15,19,27,31,32,38,39,41,50,52,54
57-bus	SAB	14	133.914800	1,6,9,19,27,30,32,36,38,41,46,50,52,54
57-bus	SAB	15	138.425343	1,4,9,15,19,27,30,32,36,38,39,41,50,52,54
57-bus	SAB	14	180.271737	1,4,9,19,26,29,30,32,38,39,41,46,50,54
57-bus	SAB	15	185.139500	1,4,9,15,19,21,27,30,32,39,41,46,50,52,54
57-bus	SAB	14	254.308700	1,4,9,14,19,27,30,32,36,38,41,50,52,54
57-bus	SAB	13	125.716019	1,4,9,19,27,30,32,38,46,50,52,54,56
57-bus	SAB	13	160.070840	1,4,9,15,19,26,29,31,32,38,50,54,56
57-bus	SAB	15	135.288300	1,4,9,19,24,27,30,32,38,39,41,46,50,52,54
57-bus	SAB	14	156.089477	1,4,9,15,19,27,30,32,38,39,41,50,52,54
57-bus	SAB	16	126.630613	1,4,7,9,15,19,21,27,30,32,39,41,46,50,52,54
57-bus	SAB	14	151.618363	1,4,9,19,27,30,32,36,38,41,46,50,52,54
57-bus	SAB	14	150.709926	1,4,9,19,27,30,32,37,38,41,46,50,52,54
57-bus	SAB	15	158.976931	1,6,9,15,19,21,27,30,32,36,41,46,50,52,54
57-bus	SAB	15	118.637822	1,4,9,19,24,27,30,32,38,39,41,46,50,52,54
57-bus	SAB	14	139.796393	1,4,9,14,19,27,30,32,36,38,41,50,52,54
57-bus	RSN	12	655.032995	32,30,56,38,51,54,27,13,1,29,19,4 -32,30,24,56,38,51,54,13,1,29,19,4

Test System	Method Name	No. of PMUs	Time Taken (second)	Placement (Bus No.)
118-bus	DFS	42	0.091308	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,73,77,80,83,86,89,91,93,95,100,102,105,110,115,116,118
118-bus	GThM	42	0.092146	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,73,77,80,83,86,89,91,93,95,100,102,105,110,115,116,118
118-bus	GThN	40	1.281339	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	32.056067	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	36.785193	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	35.388628	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	33.833047	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	39.198110	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	29.618475	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	31.464577	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	34.515021	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	35.404875	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	29.679522	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	32.141739	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	40.705551	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	35.033709	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	34.784021	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	34.278415	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	32.299174	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	38.005304	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	32.482412	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	33.683827	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SA	40	38.678469	1,5,9,12,13,17,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,65,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SAB	36	790.365671	1,5,9,12,13,19,21,25,28,32,36,37,42,43,46,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus	SAB	37	773.830434	

```
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             38              912.560608
    1,5,9,12,13,19,21,25,28,30,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              739.255199
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              1291.208705
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              689.868552
    1,5,9,12,13,19,21,25,28,32,36,37,42,43,46,49,52,57,58,59,62,70,72,77,80,84,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             33              1635.133311
    1,9,11,12,19,21,25,28,32,36,37,41,43,46,50,52,58,59,62,70,72,77,80,83,86,89,91,94,102,105,110,115,118
118-bus         SAB             37              746.768716
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              905.858084
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              728.406867
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              859.739850
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              1118.080978
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,78,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             38              789.065863
    1,5,9,12,13,17,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              753.022380
    1,5,9,12,13,19,21,25,28,32,36,37,40,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             36              731.227349
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,72,75,77,80,83,86,89,91,93,95,100,102,105,110,115
118-bus         SAB             37              991.594118
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              770.145096
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             37              711.270507
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,96,100,102,105,110,115,118
118-bus         SAB             37              873.010368
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,83,86,89,91,93,95,100,102,105,110,115,118
118-bus         SAB             36              1608.426650
    1,5,9,12,13,19,21,25,28,32,36,37,41,43,46,49,52,57,58,59,62,70,72,77,80,85,86,91,93,95,100,102,105,110,115,118
118-bus         RSN             36              74.920507
    1,13,12,19,36,5,9,29,115,21,43,37,32,25,59,53,62,58,41,57,72,46,110,105,49,70,118,101,78,80,94,91,82,89,84,86 -
1,13,12,19,36,5,9,29,115,21,43,37,32,25,59,53,62,58,41,57,72,46,110,105,49,70,118,101,78,80,95,92,90,82,85,87 -
1,13,12,19,36,5,9,29,115,21,43,37,32,25,59,53,62,58,41,57,73,46,110,105,49,70,118,101,78,80,94,91,82,89,84,86 -
1,13,12,19,36,5,9,29,115,21,43,37,32,25,59,53,62,58,41,57,73,46,110,105,49,70,118,101,78,80,95,92,90,82,85,87
```