MULTI-AREA STATE ESTIMATION IN ADVERSARIAL ENVIRONMENT

A Thesis

by

YUQI ZHOU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Le Xie
Committee Members,	Daniel Ragsdale
	Garng M. Huang
	I-Hong Hou
Head of Department,	Miroslav M. Begovic

May 2018

Major Subject: Electrical Engineering

ABSTRACT

Electric power systems have changed rapidly these years with the integration of smart grid technologies as well as the development of control and computing techniques. After the deregulation of modern power systems, operation and control over a large-scale power system are distributed to regional transmission organizations (RTOs). However, under the cyber-physical environment, power systems undertake a lot of challenges and may also be vulnerable to malicious cyber attacks. These changes and challenges in the wide-area monitoring systems (WAMSs) suggest the need for the development on distributed multi-area monitoring, control and computing algorithms.

Power system state estimation (SE) is a data processing algorithm that converts meter readings and other information into an estimate of a static state. SE serves as central function of Energy Management System (EMS) which communicates with Supervisory Control and Data Acquisition (SCADA) front end after data is obtained from remote terminal units (RTUs) and intelligent electronic devices (IEDs). The performance of downstream applications such as contingency analysis and economic dispatch will heavily depend on SE.

In terms of system monitoring, we propose a class of false analog data injection attack that can misguide the system as if topology errors had occurred. Since calculating measurements given the state value is an underdetermined problem, an optimization method is proposed to conduct a reverse estimation based on the target topology and state to achieve the topology attack. Then we investigate the bad data detection algorithm in a multi-area environment and a detection algorithm based on area sensitivity is proposed to help locate bad data and possible false date injection attacks. In order to improve the computing efficiency of SE so that the

computing time can catch up with SCADA rate, we propose a graph-based parallel computing algorithm for static SE. The proposed algorithm can help control center to achieve SCADA-rate SE and further help with the computing time of downstream applications.

ACKNOWLEDGEMENTS

First of all, I would like to thank my research advisor, Dr. Le Xie, for his generous support and guidance throughout my master study. It is always an honor and privilege working with Dr. Xie, who is erudite, inspiring and supportive. He guides me on how to conduct high quality research as a researcher and his insights on research also cultivate my research taste during these years. What's more, his attitude towards research and life influences me consistently and profoundly. He is a role model who inspires and motivates me a lot in my research and career. This thesis would not be possible without him.

Thanks also go to my committee members Dr. Daniel Ragsdale, Dr. I-Hong Hou, Dr. Garng M. Huang and Dr. Katherine Davis for their help and suggestions during completion of my master thesis.

I would like to thank my internship mentor Dr. Guangyi Liu from Global Energy Interconnection Research Institute North America. It is also a precious experience meeting researchers and other research interns in the research institute.

I would also like to thank my friends in the group, Xinbo Geng, Mohammad Sadegh Modarresi, Tong Huang, Hao Ming, Yuanyuan Li, Benjamin Wiseman, Xiangtian Zheng, Rayan EI Helou, Dr. Meng Wu, Dr. Hung-Ming Chou, Dr. Bin Wang. It is such a pleasure to work in a lab with all these smart and talented researchers.

Last but not least, I would like to thank my family and friends for their love and support during my graduate study.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a thesis committee consisting of Dr. Le Xie, Dr. Garng M. Huang, Dr. Katherine Davis, Dr. I-Hong Hou of Department of Electrical & Computer Engineering and Dr. Daniel Ragsdale of the Department of Computer Science & Engineering. The work and analysis in Chapter IV was conducted during internship in Global Energy Interconnection Research Institute North America and was supervised by Dr. Guangyi Liu. All other work conducted for the thesis was completed by the student independently.

The work in Chapter II and Chapter III was supported by Texas A&M cyber security seed grant and Energy Institute seed grant. The work in Chapter IV was supported by the State Grid Corporation technology project SGSHXT00JFJS1700138.

NOMENCLATURE

SE	State Estimation
SCADA	Supervisory Control and Data Acquisition
WLS	Weighted Least Squares
CA	Contingency Analysis
AGC	Automatic Generation Control
ED	Economic Dispatch
EMS	Energy Management System
FDIA	False Data Injection Attack
TP	Topology Processor
CB	Circuit Breaker
WAMS	Wide-Area Monitoring Systems
ISO	Independent System Operator
RTO	Regional Transmission Organizations
IED	Intelligent Electronic Devices
DOE	Department of Energy
PNNL	Pacific Northwest National Laboratory
HPC	High Performance Computing

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

CHAPTER I

INTRODUCTION

1.1 Power System State Estimation

Power system state estimation (SE) is defined as a data processing algorithm that converts meter readings and other available information into an estimate of the static state of an electric power system. The idea of SE was brought into power system by Fred Schweppe [1], [2]. State estimation utilizes the real-time data provided by Supervisory Control and Data Acquisition (SCADA) to give an estimate of the status of the power system. Weighted Least Squares (WLS) method is preferred and widely used in the existing state estimators. The performance of downstream EMS applications such as contingency analysis (CA), automatic generation control (AGC) and economic dispatch (ED) will heavily depend on the computing time and the results of SE. State estimators which serves as the central function of Energy Management Systems (EMS) usually include functions such as topology processing, bad data detection, observability analysis, etc.

As one of the most important computing algorithms in EMS, it processes the raw measurements from meters and converts it into reliable state for the system and its downstream applications. SE is widely used in power system monitoring. Since cyber side and physical side of the modern power systems may not consistent with each other all the time, SE also serves as a key part in between that can filter incorrect data and information generated by either side.

State estimation has been investigated broadly since last century. The framework of SE has been firmly established and its key applications such as topology processing, bad data detection, parameter estimation, observability analysis have been widely investigated by

researchers after the introduction of SE. Relevant research has also been conducted when SE is associated with its applications, electricity market and cyber attacks. Over the next few decades, research topics and focuses on SE may include but not limited to: distributed multi-area state estimation, high performance computing of state estimation, distribution state estimation, robust state estimation, cyber intrusion and its detection, big data application on state estimation, etc.

1.2 Bad Data Detection

In power system state estimation, bad data usually exists due to meter biases, telecommunication failure, false data injection, etc. Failure to detect bad data may cause unsatisfactory estimation results. Furthermore, a state estimation may also be deceived by erroneous topology information.

In WLS SE, bad data detection and identification are usually conducted after the estimation with processing of measurement residuals. Detection refers to the procedure where a system is analyzed to determine whether the measurements contain bad data while identification involves figuring out which measurements contain bad data. The analysis of bad data is usually finished with residual sensitivity matrix that is obtained through linearization [3]. In practice Chi-squared Test of bad data is commonly used for bad data detection.

1.3 Cyber Security in Power Grid

Power grid is a large, interconnected and complex system made up of numerous components. In terms of physical side, power system is usually designed to withstand some problems such as contingency. However, as data technology improves nowadays people rely more on cyber side of the system, and also since the data is accessible through the internet

(although there are firewalls among different layers in power system control), cyber security is becoming more and more important and needs more attention in the design of smart grid to reduce the risk of possible cyber intrusion.

In 2015, a well-known cyber attack in Ukraine penetrated the control center with malware and the power was shut off for several hours before it was recovered. To improve the security of the grid and prevent it from potential cyber intrusions, both theoretical research and hardware update will contribute. Since the existing standards and protections aim mainly at large element of the system, distribution systems with lower voltage are more vulnerable to cyber attacks and therefore calls for more attention in the future cyber security research. Also, for future research work, both pre-analysis research on cyber attacks and countermeasures after the attack should be focused and investigated.

1.4 Contributions

The thesis is mainly made up of three parts: high performance parallel computing of SE, false data injection attacks towards topology error and bad data detection algorithm in multi-area SE. The main contributions of the thesis can be summarized as:

1. A graph based parallel computing algorithm is developed for WLS state estimation. The measurements obtained before the SE are partitioned based on nodes in the graph and then SE algorithm is split accordingly. During the graph-based parallel computing, not only the load is split to provide parallelism for computing but at the same time the meaningless computations on zero elements are abandoned. In the proposed algorithm, matrices are stored in compact form so that the computing efficiency is maximized while giving an accurate result. The graph-based parallel

algorithm can help control centers achieve sub-second real state estimation for large scale power systems.

2. A class of false analog data injection attack that can misguide the system as if topology errors had occurred is proposed. By utilizing the measurement redundancy with respect to the state variables, the adversary who knows the system configuration is shown to be capable of computing the corresponding measurement value with the intentionally misguided topology. The attack is designed such that the state as well as residue distribution after state estimation will converge to those in the system with a topology error. It is shown that the attack can be launched even if the attacker is constrained to some specific meters. The attack is detrimental to the system since manipulation of analog data will lead to a forged digital topology status, and the state after the error is identified and modified will be significantly biased with the intended wrong topology. This work investigates the cyber and physical sides of a system and an optimization model based on DC SE is used to connect analog measurement data with digital status in the system topology. The idea is feasible because calculating measurement value when given state value is an underdetermined system. The parallelism in determining measurements provides possibilities of false data injection attacks in state estimation
3. A bad data detection algorithm that can help quickly detect whether a sub-area contains a bad data in multi-area state estimation is proposed [4]. The detection algorithm utilizes area sensitivity to help locate the bad data in a multi-area system and can also prevent the system from potential false data injection attacks. The proposed method focuses on the changes and sensitivity of each area instead of

dealing with bad data from statistical point of view. The algorithm can help detect false data injection attacks which sometimes cannot be detected by traditional chi-squared test.

CHAPTER II

FALSE ANALOG DATA INJECTION ATTACK TOWARDS TOPOLOGY ERRORS

2.1 Introduction

In modern power systems, topology processor (TP) performs analysis on the status of circuit breakers (CBs) to determine the system model, which can be further used in state estimation (SE). The statuses of CBs are correct and known most of the time, but in some cases the assumed statuses may turned out to be erroneous [24]. This happens when isolation switches are not telemetered or operated, other reasons may include unreported breaker manipulation, communication failure, cyber attacks, etc. In these cases, the topology model given by TP will be an incorrect one, which will entail a topology error. Topology errors usually cause the state estimation result to be significantly biased, mainly because of the mismatch between measurements and system nodal admittance matrix. Apart from inaccurate results and convergence problems in state estimation, this type of error may also cause bad data detection process to malfunction.

False data injection attack (FDIA) was first introduced in [25] and this class of cyber attack can mislead state estimation process by adding false data to measurement. Related work on FDIA can be roughly classified into the following categories. Unmitigated FDIA against SE [26]-[29], economic attacks on electricity market [30]-[32], detection and protection methods against FDIA [33],[34].

In this chapter, we propose an attack strategy against system topology. Without having to compromise the circuit breaker status, the adversary is able to forge a topology error by manipulating analog measurements in the system. The attack is designed such that measurement

residues will comply with the distribution of residues in the system with target topology error. Meanwhile, the difference between actual state values and theoretical state values is also minimized to match with topology error and avoid further detection based on state variables. The attack can mislead the system operators if the status of compromised branch is unknown or hacked. And after detection and identification of the designed topology error, adjusting system topology accordingly will lead the system to a biased SE result.

2.2 Topology Errors

In this section we start with the formulation of DC state estimation, although the proposed approach is generalizable towards AC state estimation as well. In DC state estimation, the relationship between measurements and state variables is linearized. Measurement model can be written as:

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{e} \tag{1}$$

where $\boldsymbol{z}$ is real measurements, $\boldsymbol{x}$ is vector of bus angles and $\boldsymbol{e}$ is measurement error vector. $\boldsymbol{H}$ is the measurement Jacobian matrix.

The estimated state can be derived using Weighted Least Square (WLS) method:

$$\hat{\boldsymbol{x}} = (\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{H})^{-\boldsymbol{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{z} \tag{2}$$

where $\boldsymbol{R}^{-\boldsymbol{1}}$ is diagonal weighting matrix.

And the residue can be given:

$$\boldsymbol{r} = \boldsymbol{z} - \boldsymbol{H}\hat{\boldsymbol{x}} \tag{3}$$

Based on (1) - (3), the following equations can be derived:

$$\boldsymbol{M} = \boldsymbol{H}(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{H})^{-\boldsymbol{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}} \tag{4}$$

$$\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{M})\boldsymbol{e} \tag{5}$$

Expected value of the residue after WLS state estimation:

$$E(\boldsymbol{r}) = \boldsymbol{0} \tag{6}$$

Errors in the status of breakers and switches will lead to incorrect information about network topology [35]. Topology errors are known to have more influence on measurement residues and the accuracy of state estimation result than parameter errors mainly because of the mismatch on $\boldsymbol{H}$ matrix [36]. Let $\boldsymbol{H_t}$ be the system true measurement Jacobian matrix and $\boldsymbol{H_e}$ be the erroneous Jacobian matrix, then the error of the above Jacobian matrices which is caused by topology errors can be given as:

$$\boldsymbol{D} = \boldsymbol{H_t} - \boldsymbol{H_e} \tag{7}$$

For a single branch status error, the non-zero elements in error Jacobian matrix $\boldsymbol{D}$ are the first derivative of related measurements (injections on both buses and power flows on the branch) with respect to state variables (phase angles) of both buses. With known topology errors in the system, the measurement residue then becomes:

$$\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{M_e})(\boldsymbol{Dx} + \boldsymbol{e}) \tag{8}$$

$$\boldsymbol{M_e} = \boldsymbol{H_e}(\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R^{-1}}\boldsymbol{H_e})^{\boldsymbol{-1}}\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R^{-1}} \tag{9}$$

And the expected value of the residue with topology error:

$$(\boldsymbol{r}) = (\boldsymbol{I} - \boldsymbol{M_e})\boldsymbol{Dx} \tag{10}$$

There exist certain types of topology errors that are non-detectable. Existing detection algorithms [4], [35], [37] commonly use measurement residue to detect and identify topology errors. A topology error is assumed to be detectable if $(\boldsymbol{I} - \boldsymbol{M_e})\boldsymbol{Dx} \neq \boldsymbol{0}$ for any state $\boldsymbol{x}$. When a detectable topology error is present in the system, the bias vector can be represented as:

$$\boldsymbol{Dx = Lf} \tag{11}$$

where $\boldsymbol{L}$ is measurement to branch incidence matrix and $\boldsymbol{f}$ is branch flow error vector.

The detection for a detectable topology error in the system can be dealt with using normalized residues [35]. From (11), it can be seen that the distribution of measurement residue of a system with topology error is related to both system topology and branch flow errors.

2.3 Attack Model

2.3.1 Preliminary

State estimation is usually solved as an overdetermined system, where there are more measurements than state variables and WLS method is commonly used to estimate the system state. Given a set of measurements $\boldsymbol{z}$, the overdetermined system is able to obtain a unique estimated state $\boldsymbol{x}$. Inversely, if given a fixed state $\boldsymbol{x}$, the following linear system can be derived:

$$\boldsymbol{H^T R^{-1} z = (H^T R^{-1} H)\hat{x} = b} \tag{12}$$

During state estimation, once $\hat{\boldsymbol{x}}$ is determined and system topology is known then the right-hand side of the above equation can be calculated. Consider $\mathbf{z}$ as unknown variables, then the coefficient matrix for the system is $\boldsymbol{H^T R^{-1}}$, which normally has more columns than rows. Furthermore, since the row vectors of system Jacobian matrix are linearly independent, $\boldsymbol{H^T R^{-1}}$ is a full rank matrix. Suppose we have m measurements and n state variables, then for the above system:

$$rank(\boldsymbol{H^T R^{-1}}) = rank(\boldsymbol{H^T R^{-1}}|\boldsymbol{b}) = n < m \tag{13}$$

The system becomes underdetermined and according to Rouché-Capelli theorem, there will be infinite number of solutions.

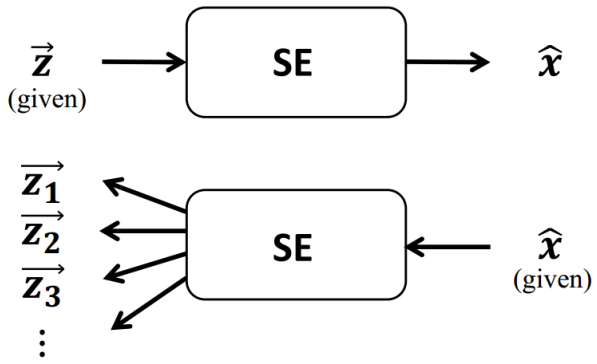

Figure 2.1: Overdetermined and underdetermined systems

The underdetermined system as well as the parallelism in determining measurements provides possibilities of false data injection attacks in state estimation. The overdetermined and underdetermined systems are described in Fig. 2.1.

2.3.2 False Data Injection Attacks

The goal of the proposed attack is to generate a topology error in the system by adding false data to system measurements. The attack will not compromise topology processor or circuit breakers, instead the attacker will utilize the redundancy in state estimation and manipulate measurement data to compensate the influence caused by the mismatch of Jacobian matrix through the topology error so that the system will appear to have a topology error.

Suppose the attacker knows the measurement data as well as system topology information a short time before the false data injection attack. Measurement data and Jacobian matrix at this stage can be denoted as $\boldsymbol{z}$ and $\boldsymbol{H_t}$. The real time measurement data taken right on/before the attack can be denoted as $\boldsymbol{z_t}$. If the attacker is able to get the information about

system right before the attack or can manipulate measurement data before it is obtained from the SCADA system, then $\boldsymbol{z}$ can be treated as real time measurement data.

With measurement data $\boldsymbol{z_t}$ and the topology information. The attacker is able to conduct state estimation based on the wrong topology. Given an intended topology, the measurement Jacobian matrix $\boldsymbol{H_e = H_t - D}$ will be used in the state estimation:

$$\tilde{\boldsymbol{x}} = (\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{H_e})^{-\boldsymbol{1}}\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{z_t} \tag{14}$$

The state above is the one when the system actually has a topology error and the attacker will expect that the state after the attack can be as close as possible to this state.

Assume the system topology does not change during the short time before the attack, the real time measurement Jacobian matrix can be known instantly since the non-zero elements in *H* matrix are the line susceptance in the system. During real time state estimation, $\boldsymbol{H_t}$ will be used as Jacobian matrix because the attacker only manipulates analog data but does not actually attack topology processor.

Therefore, for the real time state estimation the only unknown parameters are measurements since we already know *H* matrix based on system topology and estimated state based on (14). The real time measurement data, which is the attacked one, can be represented as:

$$\boldsymbol{z_a = z_t + a} \tag{15}$$

where $\boldsymbol{a}$ is the false data vector on the measurements.

In most ideal situation, the adversary would expect the following equation to hold after the attack:

$$(\boldsymbol{H_t}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{H_t})\tilde{\boldsymbol{x}} = \boldsymbol{H_t}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{z_a} \tag{16}$$

After the false data injection, the attacker would want the system to identify it as a topology error rather than a malicious attack. The measurement residue distribution will change when the power flow values are compromised, but the attacker can easily find a legitimate measurement residue that is close to the residue in the ideal situation:

$$\|(\boldsymbol{z_a} - \boldsymbol{H_t}\tilde{\boldsymbol{x}}) - (\boldsymbol{I} - \boldsymbol{M_e})\boldsymbol{D}\tilde{\boldsymbol{x}}\|_\infty < \varepsilon \tag{17}$$

where $\boldsymbol{M_e} = \boldsymbol{H_e}(\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R^{-1}}\boldsymbol{H_e})^{\boldsymbol{-1}}\boldsymbol{H_e}^{\boldsymbol{T}}\boldsymbol{R^{-1}}$, $\boldsymbol{D} = \boldsymbol{H_t} - \boldsymbol{H_e}$ and ε is the threshold.

The state in (17) should be $(\boldsymbol{H_t}^{\boldsymbol{T}}\boldsymbol{R^{-1}}\boldsymbol{H_t})^{\boldsymbol{-1}}\boldsymbol{H_t}^{\boldsymbol{T}}\boldsymbol{R^{-1}}\boldsymbol{z_a}$ if (16) does not hold. Under the assumption that (16) holds or the difference between both sides is small enough, $\tilde{\boldsymbol{x}}$ can be used as state vector in (17). The threshold can be adjusted and allow some violations for large systems.

The attacked measurement as well as the false data vector is bounded due to the constraint in (17). The determination of the entire set of measurement data not only needs to approach the estimated state in (14) after the state estimation but also needs to comply with the restrictions on the residue.

Suppose the system is free of attacks, then the real time state can be forecasted. State forecasting has been discussed in [33], [38], [39]. Autoregressive (AR) model can be used to forecast the state. The ith state variable on time t can be expressed as:

$$x_t{}^i = \sum_{j=1}^{p} \left(\varphi_j{}^i x_{t-j}{}^i\right) + v_t{}^i \tag{18}$$

where p is AR process order, φ is AR parameters and v_t is modeling uncertainties.

Parameter φ can be represented with autocorrelations and solved using Yule-Walker equations. For each state variable, its AR parameter:

$$\begin{bmatrix}\varphi_1\\ \varphi_2\\ \vdots\\ \varphi_p\end{bmatrix}=\begin{bmatrix}1 & \rho_1 & \cdots & \rho_{p-1}\\ \rho_1 & 1 & \cdots & \rho_{p-2}\\ \vdots & \vdots & \ddots & \vdots\\ \rho_{p-1} & \rho_{p-2} & \cdots & 1\end{bmatrix}^{-1}\begin{bmatrix}\rho_1\\ \rho_2\\ \vdots\\ \rho_p\end{bmatrix} \tag{19}$$

where ρ_k is the kth lag autocorrelation.

During the state forecasting, the dynamic model of system state is usually given as:

$$\boldsymbol{x_t = F_t x_{t-1} + v_t} \tag{20}$$

where $\boldsymbol{F_t}$ is the state transition matrix.

Note that $\boldsymbol{x_t}$ above is different from the estimated state in (14) which is the state with topology error in the system. After the attack with system topology errors, the attacker would also want to see an obvious difference between the state after attack and the forecasted state, otherwise the attack itself is meaningless. The restriction on target state and the forecasted state is provided below, where δ is the threshold set before the attack:

$$\|\tilde{\boldsymbol{x}} - \boldsymbol{x_t}\|_2 > \delta \tag{21}$$

Additionally, in real attack the adversary may only have access to limited number of measurements and the attack on each measurement may be restrained as well:

$$\boldsymbol{z_{a_{min}} \le z_a \le z_{a_{max}}} \tag{22}$$

Therefore, based on all the discussion above, the attack can be formulated as an optimization problem:

$$\begin{aligned}
\underset{\boldsymbol{z_a}}{\text{minimize}} \quad & \left\|\left(\boldsymbol{H_t^T R^{-1} H_t}\right)\tilde{\boldsymbol{x}} - \boldsymbol{H_t^T R^{-1} z_a}\right\|_2 \\
\text{subject to} \quad & \boldsymbol{z_{a_{min}} \le z_a \le z_{a_{max}}} \\
& \|(\boldsymbol{z_a - H_t}\tilde{\boldsymbol{x}}) - (\boldsymbol{I - M_e})\boldsymbol{D}\tilde{\boldsymbol{x}}\|_\infty < \varepsilon \\
& \|\tilde{\boldsymbol{x}} - \boldsymbol{x_t}\|_2 > \delta
\end{aligned}$$

where $\boldsymbol{z_{a_{min}}}$ and $\boldsymbol{z_{a_{max}}}$ are lower and upper bound of attacked measurements, the setup of which can be determined before the attack. For the measurement that the attacker is unable to compromise, fix the value of $\boldsymbol{z_{a_{min}}}$ and $\boldsymbol{z_{a_{max}}}$, after which the constraint of that measurement will turn into an equality constraint.

The attacker is able to obtain an optimal measurement data set for the intended attack on system topology with the above formulation. After the above optimization problem is solved, one can also easily get the attack vector $\boldsymbol{a = z_a - z_t}$, which is also the false data to be injected to the measurement data.

2.4 Illustrative Example

In this section, we illustrate the attack on the IEEE-14 bus system. A false data injection attack is performed with a forged topology error and an inclusion error is considered on branch 3-4. The branch is believed to be in service when it is actually open. The attacked 14-bus system is shown in Fig. 2.2.

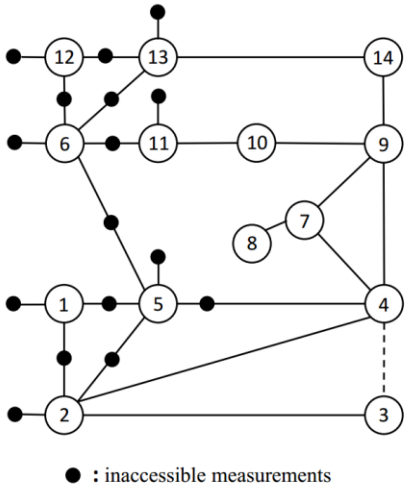

Figure 2.2: IEEE 14-bus system

Suppose the attacker is only able to compromise limited number of meters in the system during the attack. The measurements that cannot be attacked including injections and power flows are also labelled.

Assume the attacker can manage to obtain the measurement data a short time before the attack, the threshold ε in (17) is set to be 0.8 in this example. Measurements considered are real injection and real power flows (including reverse power flows), and measurement set can be denoted as $z = [P_i, P_{ij}, P_{ji}]^T$. There are 54 measurement data in the system and after solving the proposed optimization problem, the attacker is able to get an optimal solution for $\boldsymbol{z_a}$. The measurements before and after the attack are shown in Fig. 2.3.

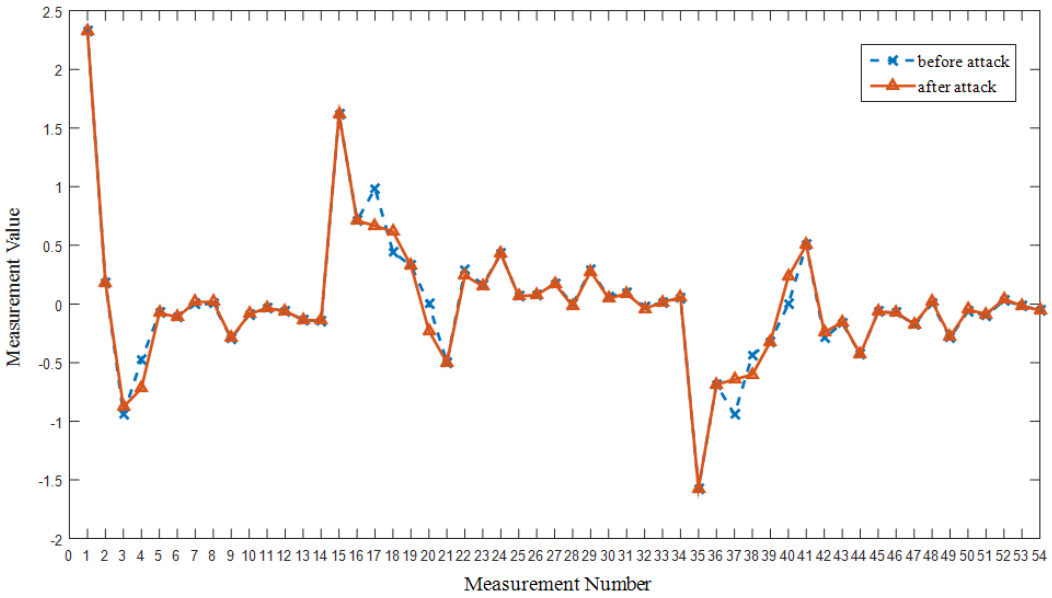

Figure 2.3: Measurements before and after the attack

The measurements before and after the attack are shown. The measurements that cannot be hacked include power injections $P_i(z_k, k \in \{1,2,5,6,11,12,13\})$, real power flows P_{ij} $(z_k, k \in \{15,16,19,21,24,25,26,27,33\})$ and power flows P_{ji} $(z_k, k \in \{35,36,39,41,44,45,46,47,53\})$.

These measurements are treated as inaccessible to the attacker, thus the value cannot be changed during the attack. The attack vector $\boldsymbol{a = z_a - z_t}$ is shown in Fig. 2.4.

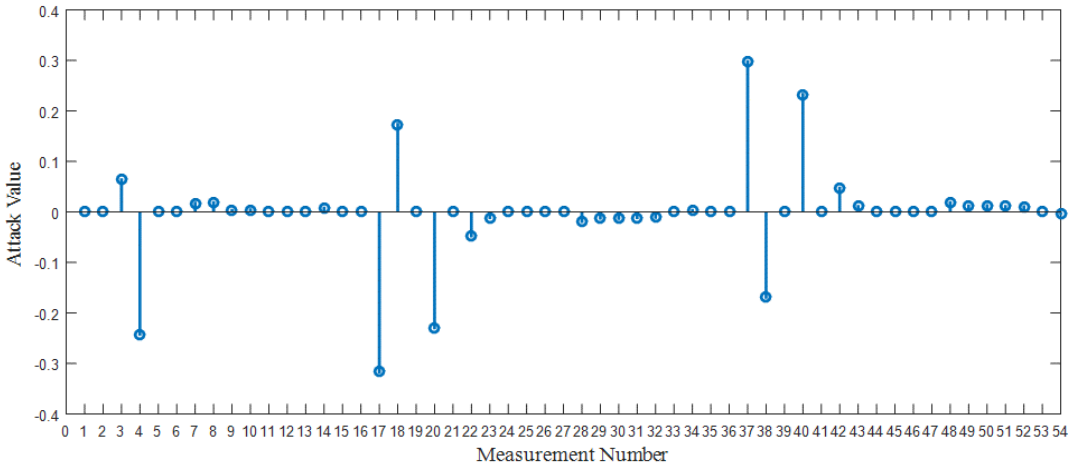

Figure 2.4: Measurement attack vector

Since some measurements are inaccessible to the attacker, the corresponding value of the attack vector should be zero, as shown above.

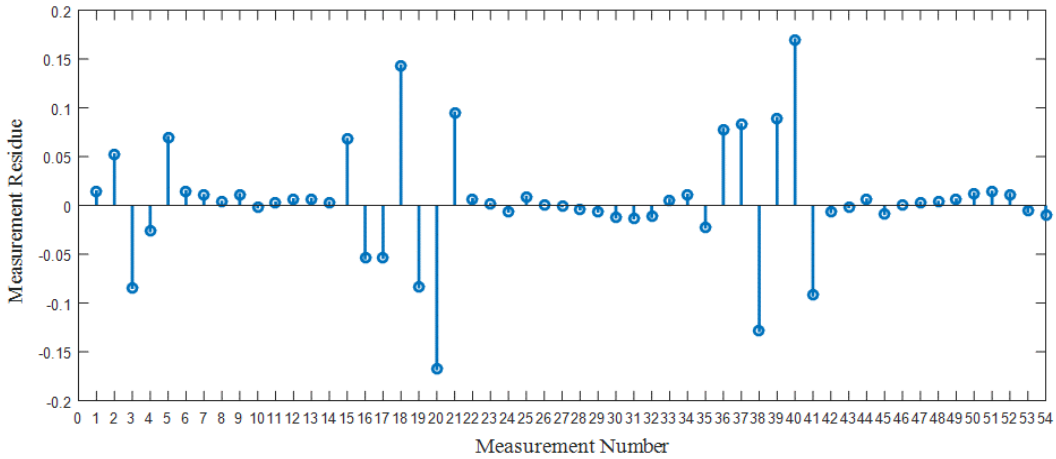

Figure 2.5: Measurement residue after the attack

After introducing the attack bias above, the system will have a residue that corresponds to a topology error on branch 3-4 as shown in Fig. 2.5, where the incident measurement residues of branch 3-4 are usually evident.

The objective function value of the attack model at solution is 0.0212. Basically the system state after the attack is equivalent to the theoretical state value $\tilde{\boldsymbol{x}}$ in (14). The comparison of the state after attack and the theoretical one is shown in Fig. 2.6.

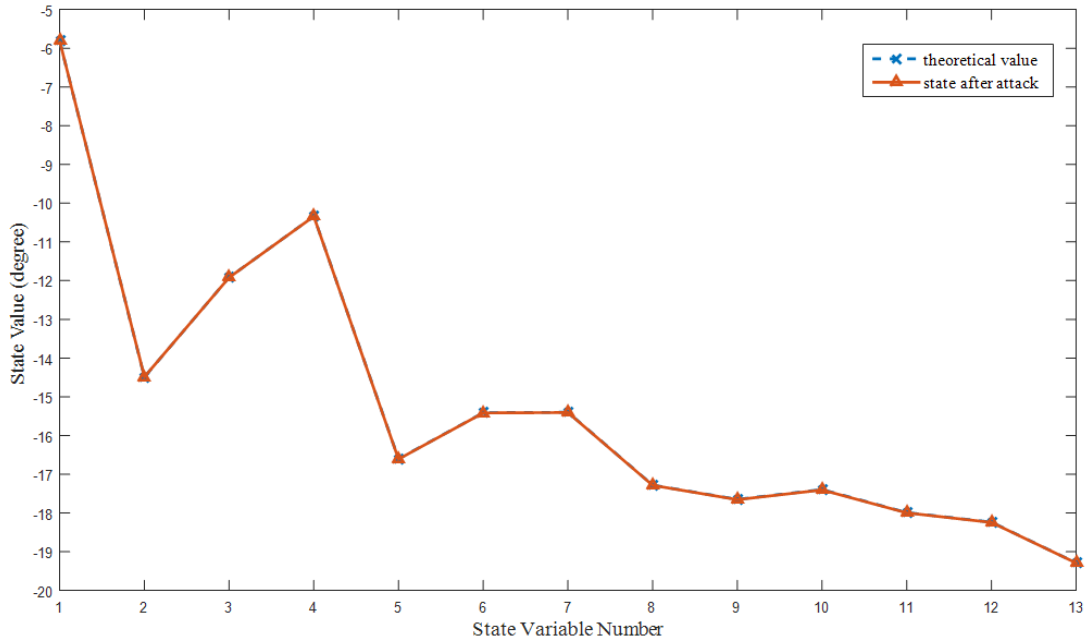

Figure 2.6: Theoretical state and state after the attack

The difference of theoretical state value and actual state value at each bus (apart from slack bus) is small, as indicated in Fig. 2.7. In this attack example, the absolute value of difference at each bus is below 0.01 degree.

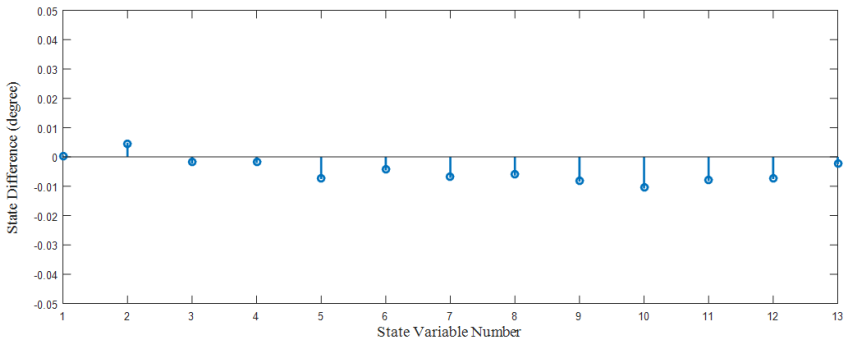

Figure. 2.7: Difference of state value at each bus

After the attack, it can be seen that the state values match perfectly with the theoretical values. Even with limited number of compromised measurements, the attacker is still able to forge a topology error in the system with a target state.

2.5 Conclusion

In this chapter, we propose a type of false analog data injection attack that can forge a topology error in the system. The attacker only needs to falsify analog measurement data to attack the system topology. Calculating measurement value given state values is an underdetermined problem, therefore the adversary is able to generate multiple attack bias as a result of system redundancy.

The work is different from existing topology attack research where most assume that the attacker is able to manipulate the topology processor. The attack vector can be obtained by solving an optimization problem, after which the state value should converge to the theoretical state value. Even with limited number of target measurements, the attacker is still able to finish an attack towards system topology errors.

CHAPTER III

BAD DATA DETECTION IN MULTI-AREA STATE ESTIMATION[*]

3.1 Introduction

In power system state estimation, bad data usually exists due to meter biases, topology errors, communication failure, false data injection, etc. Failure to detect bad data may cause unsatisfactory estimation results. With large-scale deployment of new sensors in the smart grid, this challenge is further compounded by potential cyber intrusion such as false data injection attacks (FDIA) [30].

In practice Chi-squared χ^2 Test of bad data is commonly used. In multi-area state estimation, [40] introduces a bad data detection method for subsystem using chi-squared test where the local threshold is determined by local degrees of freedom. However, given the connectivity of the system, the error as well as the residual $r = z_i - h_i(\hat{x})$ does not necessarily follow normal distribution. What's more, the commonly used chi-squared test has its vulnerability against false data injection attacks in multi-area state estimation if the attacker can manipulate the data such that they fall under the null space of the measurement mapping matrix. Therefore, it is desirable to come up with an efficient algorithm that can quickly pinpoint which area contains bad data.

For centralized bad data identification and elimination, there are two common approaches: (1) Post-processing of measurement residuals after the state estimation. (2)

---

[*] © 2017 IEEE. Reprinted, with permission, from Yuqi Zhou, Detection of bad data in multi-area state estimation, Power and Energy Conference (TPEC), IEEE Texas, Feb. 2017

Adjusting measurements weights $\frac{1}{{\sigma_i}^2}$ during iteration in Weighted Least Square (WLS) state estimation and update the weights before next iteration. It has been tested that a change in part of the system will normally cause big changes to nearby nodes and small changes to remote nodes. Therefore, we propose a method which is based on a sensitivity index called area sensitivity. The weighted overall change of measurement residual is calculated and used to identify the area in which the bad data exists. It offers a possible way to detect bad data and can also be extendable to detect false data injection attacks in multi-area state estimation. Given the assumption that the entire system is observable, our approach will detect which area contains a single bad data.

3.2 AC State Estimation

In AC state estimation, the relationship between measurements and state variables is nonlinear. The nonlinear measurement model is:

$$z = h(x) + e \tag{1}$$

where z is vector of measurements, x is vector of state variables and e is measurement error vector. $h(x)$ is the nonlinear function that relates measurements to state variables.

In nonlinear WLS state estimation, we can formulate the problem as the following optimization problem:

$$\text{minimize:} \quad J(x) = [z - h(x)]^T R^{-1} [z - h(x)]$$

$$\text{subject to:} \quad z = h(x) + e$$

$J(x) = \sum_{i=1}^{m} \frac{(z_i - h_i(x))^2}{R_{ii}}$ is the objective function, where m is the number of measurements and $R_{ii} = {\sigma_i}^2$ is the variance of the error in measurement i.

According to first-order optimality, we have the following equations at the minimum:

$$\frac{\partial J(x)}{\partial x} = -H^T(x)R^{-1}[z - h(x)] = 0 \tag{2}$$

where $H(x) = \frac{\partial h(x)}{\partial x}$ is the Jacobian matrix of $h(x)$. Iterative method can be used to solve this nonlinear problem [24].

The proposed algorithm on bad data detection as well as the illustrative example on the IEEE 14-bus system is based on AC state estimation.

3.3 Chi-squared Test

One of the common methods used for bad data detection is the Chi-squared χ^2 Test. Chi-squared distribution is a special case of gamma distribution. When $\alpha = \frac{m-n}{2}$, where m and n are integers and $m > n$ and $\beta = 2$, the gamma probability density function (pdf) becomes:

$$f(x) = \frac{1}{\Gamma(\frac{m-n}{2})2^{\frac{m-n}{2}}} x^{(\frac{m-n}{2})-1} e^{-\frac{x}{2}} \tag{3}$$

Which is the chi-squared pdf with $(m - n)$ degrees of freedom. Chi-squared distribution plays an important part in statistical inference, especially when sampling from a normal distribution [41].

If ${\chi_p}^2$ is denoted as a chi-squared random variable with p degrees of freedom, then we can easily prove the following two facts:

(1) If Y is a $N(0,1)$ random variable, then $Y^2 \sim {\chi_1}^2$.

(2) If $X_1, \cdots, X_n$ are independent and $X_i \sim {\chi_{p_i}}^2$, then $X_1 + \cdots + X_n \sim {\chi_{p_1+\cdots+p_n}}^2$.

Based on these two facts. Suppose we have a set of n independent random variables $X_1, \cdots, X_n$, where $X_i \sim N(0,1)$, Then $Y = \sum_{i=1}^{n} {X_i}^2 \sim {\chi_n}^2$, where degrees of freedom equal n.

When Chi-squared χ^2 Test is used for detecting bad data in WLS state estimation, we compute the objective function $J(\hat{x}) = \sum_{i=1}^{m} \frac{(z_i - h_i(\hat{x}))^2}{\sigma_i^2}$ after the state estimation, where m is number of measurement, z_i is measured value of measurement i, $h_i(\hat{x})$ is estimated value of measurement i and σ_i^2 is variance of the error in measurement i. $J(\hat{x})$ is then compared with $\chi^2_{(m-n),p}$, where degrees of freedom is $(m-n)$ and detection confidence probability is p. If $J(\hat{x}) \geq \chi^2_{(m-n),p}$, then it is assumed that there is bad data in the measurements under the confidence of p. Otherwise, the measurements can be assumed to have no bad data.

3.4 Bad Data Detection Based on Area Sensitivity

3.4.1 Area Sensitivity

In a multi-area state estimation, wide-area monitoring systems (WAMS) are involved in information gathering and processing [11]. Due to the large size of the power system, the determination of the states over the whole system becomes challenging. A good way is to let the ISO (Independent System Operators) keep their state estimators and also have a central entity to coordinate their results to determine the system-wide state [10]. For a multi-area state estimation, the basic structure of the system is composed of a central control center and several sub-areas which can be regarded as local control centers in the system. Fig. 3.1 shows the typical architecture of multi-area state estimation.

The system has k interconnected areas and each sub-area is considered to be connected with Central Control Center. Information is exchanged between Central Center and each area.

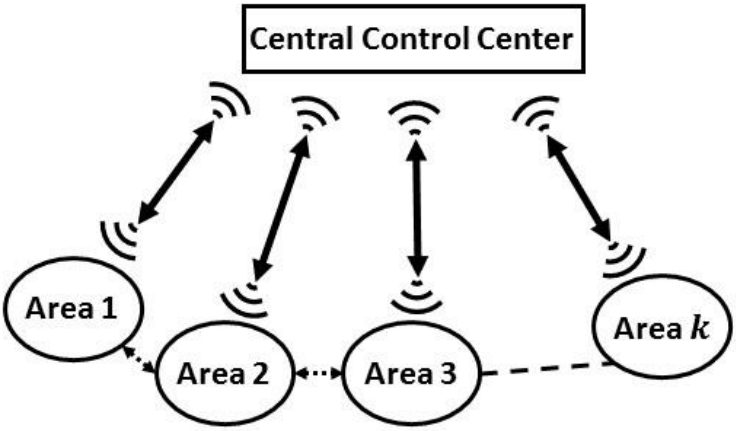

Figure 3.1: Multi-area state estimation

Suppose a state estimation is running in the control center every 5 minutes (actual frequency may be higher). Then states as well as measurement residuals for each area can be derived after every state estimation. In reality, the operation of security monitoring and control in a control center is based on wide-area measurements every 2 seconds or so from the Supervisory Control and Data Acquisition (SCADA) system [42].

We denote m_i as the number of measurements in area i and $\overrightarrow{r_i(t)}$ as the vector of residuals of area i at time t. $\overrightarrow{r_i(t-1)}$ represents the vector of residuals of area i at time $t-1$, or our benchmark for residue vector (residues when the system is free of bad data). The area sensitivity in area i is defined as:

$$\xi^i(t) = \frac{\left\| [\overrightarrow{r_i(t)} - \overrightarrow{r_i(t-1)}] \right\|_2}{m_i} \tag{4}$$

Therefore, suppose a state estimation is finished on the system in above, k area sensitivity $\xi^1, \xi^2, \ldots, \xi^k$ will also be obtained for each area in the system. Area sensitivity shows the overall changes within each area between two consecutive time snapshots.

3.4.2 Bad Data Detection Algorithm

We presume that the branches among areas are trustworthy and there is no bad data between every two areas. If there is a bad data occurring in a sub-area, then normally it will also affect the nearby nodes. The measurement residuals of each area will then change accordingly due to the existence of bad data. The area where the bad data exists is usually more sensitive and responsive to such changes. In WLS state estimation, after the bad data occurs, the bad data area will usually have a larger change in measurement residuals than remote areas. We then use a sensitivity index called area sensitivity ξ^i to reflect such changes for each separate area i.

After a bad data is introduced into the system, based on our formulation above, we can generate k area sensitivity values for these areas after state estimation. The compromised area normally would have a larger ξ than other areas in the system. On the other hand, if there is no bad data between two consecutive state estimation, each ξ^i should be small and close to 0.

Area j is suspected to have a bad data if:

$$p = \frac{\xi_j}{\sum_{i=1}^{k} \xi_i} > f(k) \tag{5}$$

If ξ_j is much larger than any other ξ_i, then the above problem can also be simplified to:

$$j = argmax(\xi_i) \tag{6}$$

The above assumption holds if after computing the area sensitivity there is only one ξ_j that satisfies (5). However, if the bad data exists in a measurement which is close to the boundary of the area. Then the bad data in such measurement may also cause changes in the connected area because of the structure of the system. The bad data may also cause significant changes in the residues in nearby areas too. In this possible situation, there may be more than one

area sensitivity value ξ^i that satisfies (5). Then a good way to solve this problem is that: we run state estimation again but with different sets of variance ${\sigma_i}^2$ on different areas accordingly.

In order to make the results more precise in the state estimation, we put more weight on the better meters and trust more on these measurements. For good device, we assign small ${\sigma_i}^2$ while for bad device we assign large ${\sigma_i}^2$. In WLS state estimation problem, if there is a bad data shows up in a high-variance area, then there will be small changes in the estimated results, thus leads to a smaller area sensitivity. If it is in a low-variance area, then it will cause a larger area sensitivity. In our case, however, small ${\sigma_i}^2$ should be assigned and all the measurements in the suspected areas are treated as good measurements. In this way the existence of bad data in the suspected areas will lead to larger residues than usual. By applying this method, the effect of bad data on the area is amplified. After following the procedures above, we will have k' new updated area sensitivity ${\xi_i}'$ for k' suspected areas. Then the area with the largest ${\xi_i}'$ is candidate for locating the bad data. This method will be explained in the illustrative example on the IEEE 14-bus system.

Based on the algorithm above, the detailed steps to do the multi-area state estimation bad data detection can be stated as follows:

(1) Compute area sensitivity

(a) On time $(t-1)$, get residue vector $\overrightarrow{r_i(t-1)}$ for each area after the state estimation.

(b) On time t, get residue vector $\overrightarrow{r_i(t)}$ for each area after the state estimation.

(c) Compute area sensitivity for each area:

$$\xi^i = \frac{\left\| [\overrightarrow{r_i(t)} - \overrightarrow{r_i(t-1)}] \right\|_2}{m_i}$$

(2) Bad data detection

(a) Compare area sensitivity value ξ^i.

(b) If $\xi^1 \approx \xi^2 \approx \cdots \approx \xi^k \approx 0$, then the system is said to be free of bad data for each sub-area.

(c) If $\xi^i \neq 0$, and there is only one area that satisfies:

$$p = \frac{\xi_j}{\sum_{i=1}^{k} \xi_i} > f(k)$$

then the bad data comes from area j. If $\xi^i \neq 0$, and there is more than one area (k' areas) that satisfies the above constraint, then go to step d.

(d) For k' areas, run state estimation again. During state estimation, we assign smaller ${\sigma_i}^2$ uniformly and respectively for each measurement within each suspected area as well as the power flow measurement between the suspected areas. Then return to step 1. The corresponding area with the largest updated area sensitivity ${\xi_i}'$ is said to have bad data.

3.5 Illustrative Example

The bad data algorithm is tested on the IEEE 14-bus system in this section. Suppose we have control over the central control center. In Fig. 3.2, the 14-bus system is divided into 4 sub-areas which represent local control centers.

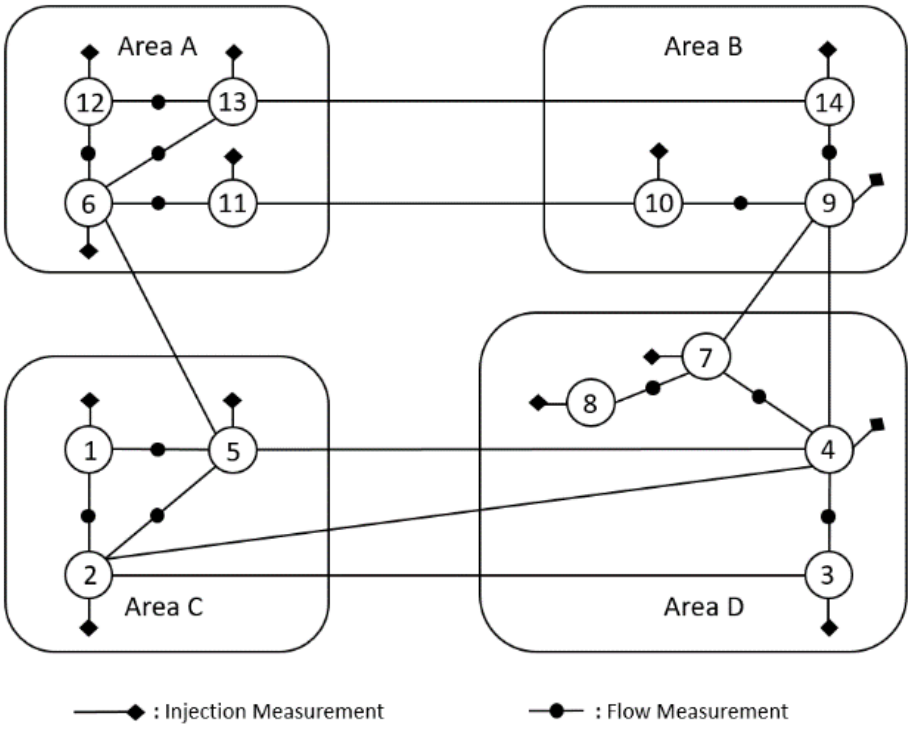

Figure 3.2: IEEE 14-bus system

The measurements that are considered are those within each area, including internal injections and internal power flows, which are labelled above. In the example, measurements between areas are assumed to be trustworthy and there is only bad data within each area. The algorithm will be tested on both measurement error and topology error.

3.5.1 Measurement Error

For the traditional chi-squared test, it focuses on the whole system. So even if there is a truly suspicious data for a specific measurement, the detection may still indicate no bad data. WLS state estimation is an unbiased estimator if and only if the measurement error is statistically distributed and the model is accurate, both of which may not exist in a practical power system

[43]. In the proposed algorithm, however, it focuses on the sub-areas as well as the measurements within each area rather than dealing with the whole system statistically.

Suppose there is a data of certain value to be injected to a measurement in the system, there is no guarantee that it will be detected. To simplify the problem and also focus more on each area and internal measurements, we double each measurement respectively in the test to serve as the new measurement after the error is introduced. In this way, we can think of the problem as a false data injection problem where the injected value of the measurement is equal to the measurement value itself. The test results of bad data on injection measurements are in Table 3.1 and Table 3.2 and bad data on power flow measurements are in Table 3.3 and Table 3.4.

Table 3.1. Test Results for Real Injection Bad Data

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
1	0.0047	0.0041	0.1297	0.0141	C	C
2	0.0005	0.0005	0.0074	0.0023	C	C
3	0.0028	0.0032	0.0120	0.0411	D	D
4	0.0015	0.0038	0.0056	0.0150	D	D
5	0.0006	0.0003	0.0029	0.0007	C	C
6	0.0035	0.0006	0.0009	0.0003	A	A
9	0.0011	0.0145	0.0008	0.0033	B	B
10	0.0010	0.0055	0.0001	0.0004	B	B
11	0.0013	0.0006	0.0001	0.0001	A	A
12	0.0026	0.0003	0.0002	0.0001	A	A
13	0.0050	0.0018	0.0004	0.0002	A	A
14	0.0014	0.0091	0.0002	0.0005	B	B

Table 3.2. Test Results for Reactive Injection Bad Data

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
1	0.0001	0.0001	0.0092	0.0008	C	C
2	0.0013	0.0006	0.0103	0.0024	C	C
3	0.0003	0.0001	0.0005	0.0023	D	D
4	0.0001	0.0002	0.0004	0.0012	D	D
5	0.0001	0.0001	0.0006	0.0001	C	C
6	0.0013	0.0002	0.0004	0.0003	A	A
8	0.0003	0.0007	0.0003	0.0093	D	D
9	0.0006	0.0080	0.0004	0.0016	B	B
10	0.0006	0.0035	0.0001	0.0002	B	B
11	0.0007	0.0003	0.0000	0.0000	A	A
12	0.0007	0.0001	0.0001	0.0000	A	A
13	0.0021	0.0007	0.0002	0.0001	A	A
14	0.0005	0.0030	0.0001	0.0001	B	B

Table 3.3. Test Results for Real Power Flow Bad Data

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
1-2	0.0011	0.0011	0.1109	0.0108	C	C
1-5	0.0019	0.0016	0.0598	0.0049	C	C
2-5	0.0012	0.0011	0.0326	0.0035	C	C
3-4	0.0005	0.0009	0.0023	0.0146	D	D
4-7	0.0020	0.0040	0.0038	0.0157	D	D
6-11	0.0038	0.0012	0.0002	0.0006	A	A
6-12	0.0044	0.0003	0.0003	0.0001	A	A
6-13	0.0095	0.0027	0.0009	0.0007	A	A

Table 3.3. Continued

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
9-10	0.0006	0.0040	0.0002	0.0004	B	B
9-14	0.0009	0.0079	0.0003	0.0005	B	B
12-13	0.0009	0.0001	0.0000	0.0000	A	A

Table 3.4. Test Results for Reactive Power Flow Bad Data

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
1-2	0.0005	0.0003	0.0143	0.0011	C	C
1-5	0.0001	0.0001	0.0031	0.0003	C	C
2-5	0.0001	0.0001	0.0009	0.0001	C	C
3-4	0.0002	0.0002	0.0005	0.0028	D	D
4-7	0.0007	0.0013	0.0014	0.0054	D	D
6-11	0.0018	0.0006	0.0001	0.0003	A	A
6-12	0.0014	0.0001	0.0001	0.0001	A	A
6-13	0.0037	0.0012	0.0004	0.0003	A	A
7-8	0.0003	0.0010	0.0003	0.0096	D	D
9-10	0.0005	0.0032	0.0001	0.0003	B	B
9-14	0.0004	0.0030	0.0001	0.0002	B	B
12-13	0.0004	0.0000	0.0000	0.0000	A	A

Predicted area stands for the candidate area for locating the bad data after applying the proposed algorithm and actual result represents the area which the bad data actually originates from. Real injections at bus 7 and bus 8 as well as reactive injection at bus 7 are missing in the

table because the values of these measurements are 0. Real power flow of branch 7-8 is missing due to the similar reason.

From the results, it can be concluded that the algorithm is working perfectly in detecting measurement error or false data injection in the test and every prediction based on area sensitivity is in accordance with the actual result.

3.5.2 Topology Error

Errors in the status data of breaker will result in erroneous assertion of network topology in terms of branch outage, bus split or shunt capacitor/reactor switching [35]. Such error in the power system is called topology error. In our test, we only focus on the branch outage in topology error. If there is a branch outage in the system, usually there will be more than one measurement that is affected as well due to the features of power system. We can also think of it as a special case where there are multiple measurement errors in the system.

The purpose of the proposed algorithm is to find where the bad data possibly comes from rather than detect the specific position of the bad data. If there is no branch outage in the system, then we would expect the area sensitivity ξ^i of each area to be small and close to 0. However, if there is a branch outage in the system, each area sensitivity value will change accordingly. The results are shown in Table 3.5.

The result for branch 7-8 is not shown here because there is only one branch between bus 7 and bus 8. The state estimation will not converge if this branch is open. In the topology error part, the step d of bad data detection is not implemented during the test because branch outage will cause multiple changes in the system.

Table 3.5. Test Results for Topology Error

	ξ^A	ξ^B	ξ^C	ξ^D	**Predicted Area**	**Actual Result**
1-2	0.0119	0.0067	0.2201	0.0278	C	C
1-5	0.0035	0.0014	0.0940	0.0107	C	C
2-5	0.0011	0.0013	0.0394	0.0079	C	C
3-4	0.0009	0.0006	0.0100	0.0173	D	D
4-7	0.0088	0.0124	0.0057	0.0180	D	D
6-11	0.0053	0.0066	0.0014	0.0027	A,B	A
6-12	0.0084	0.0009	0.0002	0.0002	A	A
6-13	0.0156	0.0091	0.0005	0.0018	A	A
9-10	0.0043	0.0053	0.0011	0.0022	A,B	B
9-14	0.0056	0.0111	0.0009	0.0023	B	B
12-13	0.0019	0.0002	0.0001	0.0000	A	A

Adjusting the value of weight in different areas does not apply in this case. Therefore, we simply list all the areas with $p_j = \frac{\xi^j}{\sum_{i=1}^{k} \xi^i}$ that is larger than the threshold and consider them equally as the candidate areas. It can be seen that the average value of area sensitivity is greater than the one in the measurement error test.

Since the topology error may cause more noticeable changes to the system than a single bad data, we can only roughly locate an area where the outage may happen during the test. For example, if there is an outage in branch 9-10, by applying the algorithm we can only tell that the outage may take place within A or B or between A and B. The results imply that the accuracy is not as good as the one in the measurement error part but the algorithm is still efficient enough against topology error caused by branch outages.

3.5.3 Analysis

Normally, if the injected data on a certain measurement is increased, it should be easier to detect it. Therefore, we change the value of the data that is added to a measurement and observe the change of $p_j = \frac{\xi^j}{\sum_{i=1}^{k} \xi^i}$. The results for a false data injection attack on real power flow of branch 1-2 is shown in Fig. 3.3.

The x-axis represents the data injected to the measurement (unit: kW), the y-axis represents parameter $p_j = \frac{\xi^j}{\sum_{i=1}^{k} \xi^i}$. For a fixed injected data, larger p means that we have more confidence to say that the bad data exists in the predicted area.

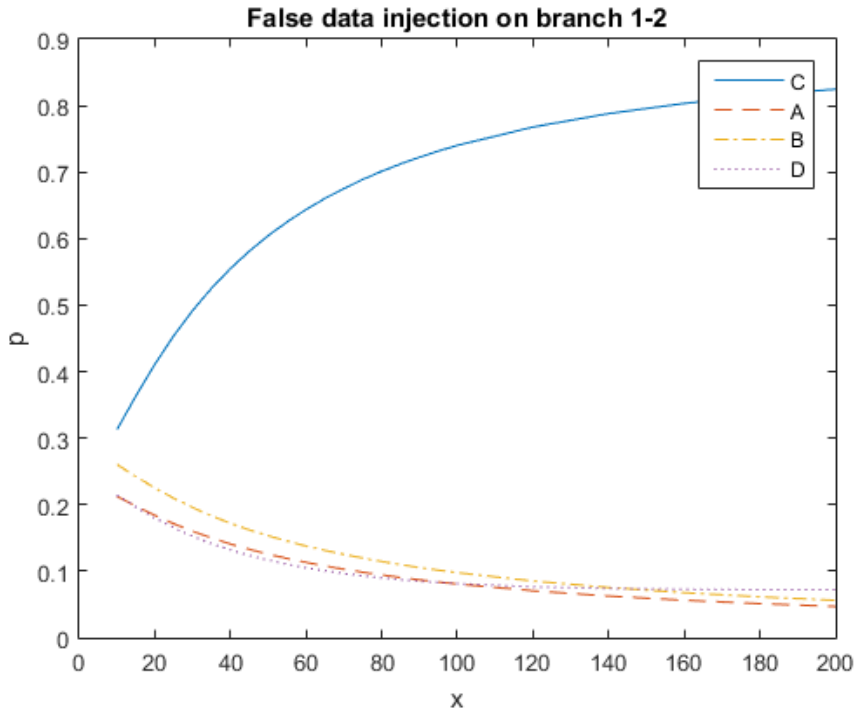

Figure 3.3: False data injection on branch 1-2

As the figure shows, the proposed algorithm itself is responsive to the changes on the system. When the system is free of bad data or the injected data is really small, p of different

areas remain at the same level. As the bad data becomes more and more obvious, p of the compromised area will become larger and larger. Meanwhile p of the other areas becomes smaller as the injected data increases.

Compared with chi-squared test, bad data detection based on area sensitivity can detect bad data in a wider range. Suppose it is certain that there is a bad data in the system, even if it cannot be detected by traditional detection algorithm, we can give a prediction on where the bad data comes from by adjusting the threshold lower. Also, for the chi-squared test, its objective function $J(\hat{x}) = \sum_{i=1}^{m} \frac{(z_i - h_i(\hat{x}))^2}{\sigma_i^2}$ has a wide range but without a certain upper bound. In our method, $p_j = \frac{\xi^j}{\sum_{i=1}^{k} \xi^i}$ will converge to a limit after the data injection is increased to a certain level.

Chi-squared test tells us whether a data belongs to bad data from view of statistics. A limit $\chi^2_{(m-n),p}$ is given based on degrees of freedom and possibility. It focuses on a given set of measurements and detect the bad data using statistics. In a false data injection attack on the system, an attacker could easily avoid such detection by manipulating the data. In DC state estimator, suppose $z = Hx + e$. In a potential false data injection attack [1], z^a is the bias introduced by the attacker. Then it can be proved that as long as we adjust z^a to make $(I - H(H^T R^{-1} H)^{-1} H^T R^{-1}) z^a = 0$, it can pass the detector. Suppose H is an $m \times n$ matrix, it was also stated in [25] that if an attacker can compromise k meters, where $k \geq m - n + 1$, then there always exists an attack vector that can bypass the detection.

In multi-area state estimation, central control center has control over each sub-area. If an attacker can manipulate the data within one of the areas, the proposed method can detect such attacks in the area even if it can pass the traditional chi-squared bad data detection. Suppose the attacker can manipulate some of the data in one local control center and adjust z^a so that the chi-

squared test cannot detect it. But after the measurements in the area are hacked, even if $\|(I - H(H^T R^{-1} H)^{-1} H^T R^{-1}) z^a\|_2$ is guaranteed to be small enough to remain undetected by the chi-squared test, it can be detected by the proposed algorithm because such attacks would cause the area sensitivity to change evidently.

3.6 Conclusion

In this chapter, we proposed a bad data detection algorithm based on area sensitivity in multi-area state estimation. The algorithm considers residue from two consecutive time snapshots, and can detect whether bad data exists in any particular area. Area sensitivity reflects the residual changes of each area that are caused by the existence of bad data.

The validity and efficacy of the algorithm are tested on the IEEE 14-bus system. Both measurement errors and topology errors are considered in the numerical simulation. The results indicate that the algorithm works better on measurement errors than topology errors. We also compared the proposed algorithm with the traditional chi-squared test. The proposed method based on area sensitivity focuses on the changes and sensitivity of each area instead of dealing with the bad data from statistical point of view. The algorithm can be further used to detect false data injection attacks which sometimes cannot be detected by traditional chi-squared test.

Since the area sensitivity takes into consideration the residue of two consecutive time snapshots, the noise will also be eliminated after calculating the area sensitivity ξ^i. If the central control center can have a higher frequency to run such algorithm on bad data detection, it can also serve as a possible early detection method for the system.

CHAPTER IV

GRAPH-BASED PARALLEL STATE ESTIMATION

4.1 Introduction

With the development of multi-core processors and parallel computing in recent years, investigation of faster state estimation becomes possible and is called for in the smart grid. An accurate and SCADA-rate SE will allow operators to monitor the system state and take actions in time to secure the system operation.

4.1.1 Background and Motivation

Power system state estimation (SE) is defined as a data processing algorithm that converts redundant meter readings and other available information into an estimate of the static state of an electric power system. Supervisory Control and Data Acquisition (SCADA) obtains data from remote terminal units (RTU) and intelligent electronic devices (IED). State estimators which serves as the central function of Energy Management Systems (EMS) will then collect data from the SCADA system and perform state estimation on the whole system. Weighted least squares (WLS) method is preferred and most commonly used in the existing state estimators. The performance of downstream applications such as contingency analysis, automatic generation control and economic dispatch will depend on the computing time and the results of SE to a great extent. U.S. Department of Energy (DOE) presented computational needs for next generation electric grid in [5], the future direction for state estimation is to achieve real-time state estimation by reducing solution time from minutes to seconds to even milliseconds to keep up with the SCADA measurement cycles. An accurate and much faster SE will allow operators to know the

states of the system and apply relevant applications with a higher frequency, which also provides opportunities to continuous monitoring of the system and bring on better power system reliability. Also, faster SE can also lead to faster bad data detection as well as other relevant analysis tools. High-performance computing makes the faster processing of state estimation possible. Parallel computing is effective especially when the data set to process is big enough which holds in modern power systems and the parallel algorithm itself should also be designed and implemented efficiently. Highly parallelized state estimation algorithm will allow real time update of grid status and provides more reliability in the grid control.

4.1.2 Literature Review

Many methods and algorithms have been proposed to address the problem to accelerate SE. The idea of multiprocessor state estimation was proposed in [6], [7]. Previous work in [8]-[11] discussed distributed state estimation. The system network is partitioned into several areas and, in each area, there is a local control center. Then the local SE results are obtained before they are coordinated to get the optimal result. Because of the independence of the local computation, it is conducted in parallel to improve the distributed SE performance on time consumption. It doesn't sacrifice the results accuracy. Parallel state estimation based on fast decoupled method was illustrated in [12], [13]. A decentralized robust state estimator was proposed in [14]. The matrix inversion lemma was used for parallel static state estimation in [15] and the block-partitioning algorithm was also presented. The algorithm is able to find the correct solution, but the partitioning approach has a vital effect on the success of the parallel algorithm. Approach in [16], [17] mainly uses matrix factorization to carry out the computation in parallel. The work was conducted by Pacific Northwest National Laboratory (PNNL). It employed high

performance computing and adapted algorithm to speed up state estimation computation. However, the focus is mainly on the problem-solving part, which is after the data processing and gain matrix formulation and decomposition. A matrix-splitting strategy for Gauss-Newton iteration was presented in [18], in which neighboring areas communicate with each other during each iteration. A computationally efficient linear model-based estimator was proposed in [19] to serve as an alternative for Gauss-Newton approach. A nonlinear state estimation using distributed semidefinite programming was proposed in [20]. High-performance computing (HPC) with GPUs were used in [21], in which a parallel relaxation-based joint state estimation algorithm was proposed for dynamic state estimation.

4.1.3 Contribution

A graph computing based high-performance parallel approach is proposed and applied to obtain a SCADA-rate power system state estimation by increasing the computation efficiency. The graph database is briefly introduced and then the feasibility of its applications into power system modeling is investigated. Then a graph computing based parallel state estimation is elaborated, including efficient node-based matrices formulation, and dense matrices processing. Based on system topology analysis and decomposition of the state estimation problem, the node-based matrices develop locally. For each node-based matrix, it only needs to acquire information from its 1-step neighbors, at most 2-step neighbors, and their connections. Compressed sparse row (CSR) is employed to store computation matrices for further improving calculation efficiency and relieving the burden introduced by matrix sparsity. After that, the gain matrix factorization is implemented by hierarchical parallel computing, and the whole SE problem is then solved with forward/backward substitution. Since the employed core algorithm is still the

commonly used one, the solution accuracy is not sacrificed, which can be supported by testing results, but the computation time is largely reduced when compared with conventional computing algorithm.

4.2 Graph Computing

4.2.1 Graph Database

Graph is usually used to model relations between objects. It is a collection of vertices, representing objects in a system, and edges, standing for relations between objects. In mathematics, a graph is expressed as $G = (V, E)$, in which V represents a set of vertices in the graph and the set of edges is denoted as E, indicating how these vertices relate to each other. Each edge is denoted by $e = (i, j)$ in E, where i and j are referred as head and tail of the edge respectively.

Graph database uses graph structures for semantic queries with vertices, edges and attributes to represent and store data in vertices and edges. Such database allows data to be linked together directly and retrieved with graph operation, also called graph traversal. The definition of the neighboring distance [22] in a graph is given below.

Definition 1: The distance $d(u, v)$ between vertices u and v in a graph is the shortest path between u and v.

Definition 2: Vertex v is called k-step neighbor of vertex u if $d(u, v) = k$.

Graphs are useful and intuitive in understanding and processing numerous diverse data and they have already been applied to model relations and processes in physical, biological, social and information systems.

4.2.2 Power System Modeling with Graph Database

Generally, a traditional power system is comprised of power generation, power delivery, power distribution, power conversion and power consumption. The five components are categorized into two types: (1) bus-attached, and (2) line-attached [44]. For example, generators and loads are considered as being directly connected to buses in the bus-branch model. Fig. 4.1 demonstrates a large-scale power system in the graph database. Each node represents a bus and the edges are with attributes which include measurements such as power flows.

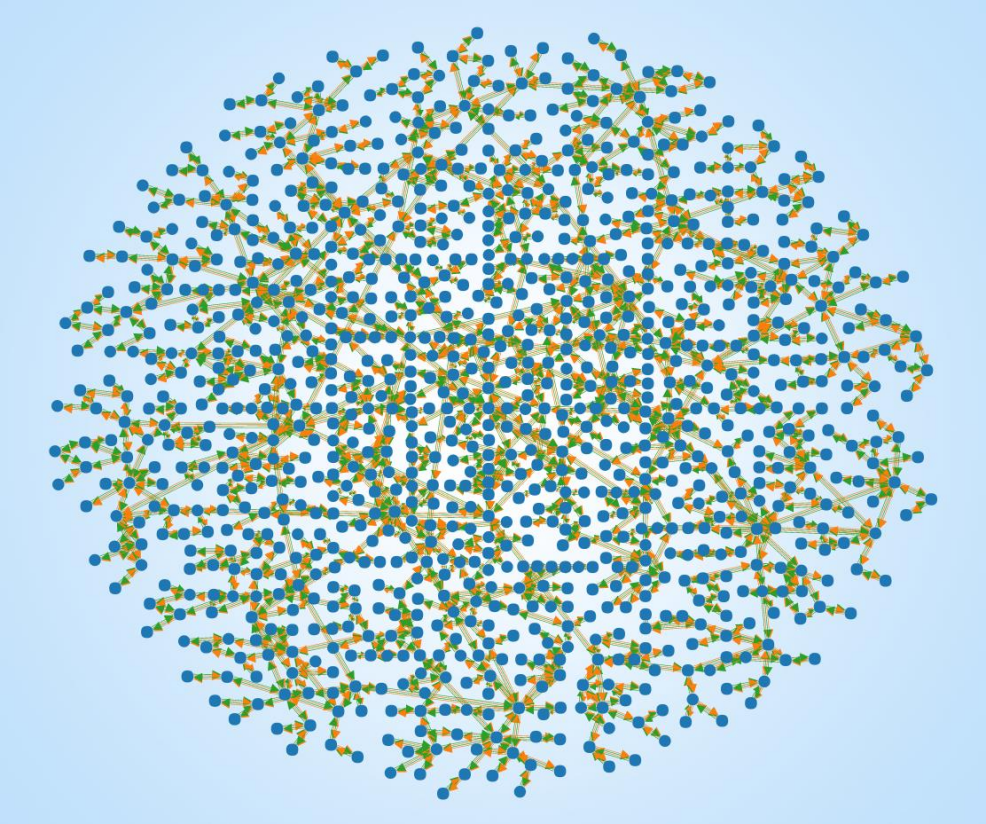

Figure 4.1: Graph database

In node-based parallel computing, computation at each node is independent from others, meaning local computations are performed simultaneously. In a graph, the connections between

nodes represent non-zeros in off-diagonal elements, and each node denotes a diagonal element in the corresponding matrix. Hierarchical parallel computing performs computation for nodes at the same level in parallel. The level next to it is performed after. The hierarchical parallel computing is applied to matrix factorization, using the Cholesky elimination algorithm. Three steps are involved: 1) fill-ins determination, 2) elimination tree formulation, and 3) elimination tree partition for hierarchical parallel computing.

4.3 Graph Computing Based Parallel State Estimation

4.3.1 Preliminary for Parallel State Estimation

For an *n*-bus power system with m measurements, its nonlinear WLS state estimation is formulated as:

$$\min \quad J(x) = [z - h(x)]^T R^{-1} [z - h(x)] \tag{1}$$

where z is the measurement vector of dimension m, x is the system state vector of dimension $2n - 1$, h is the nonlinear function of x, which relates the states to the error free measurements, and R^{-1} is the diagonal matrix consisting of the weight, $\frac{1}{\sigma_j{}^2}$, for each measurement j. To obtain the minimum of $J(x)$, the equation below is derived:

$$g(x) = \frac{\partial J(x)}{\partial x} = -H^T(x) R^{-1} r(x) = 0 \tag{2}$$

where $H(x) = \frac{\partial h(x)}{\partial x}$ is the Jacobian matrix, $r(x) = z - h(x)$.

Substituting the first-order Taylor's expansion of $g(x)$ in (2), the solution of the objective function can be found by iteratively solving (3).

$$G(x^k) \Delta x^k = H^T(x^k) R^{-1} r(x^k) \tag{3}$$

where $G(x^k) = \frac{\partial g(x^k)}{\partial x} = H^T(x^k)R^{-1}H(x^k)$, $x^{k+1} = x^k + \Delta x^k$, k is the iteration index, and x^k is a vector of system states at iteration k.

When the fast-decoupled model is applied, the measurement Jacobian matrix and gain matrix are constant. Subscripts A and R below denote the corresponding active and reactive components respectively.

$$H = \begin{bmatrix} H_{AA} & H_{AR} \\ H_{RA} & H_{RR} \end{bmatrix} = \begin{bmatrix} H_{AA} & 0 \\ 0 & H_{RR} \end{bmatrix} \tag{4}$$

$$G = \begin{bmatrix} G_{AA} & G_{AR} \\ G_{RA} & G_{RR} \end{bmatrix} = \begin{bmatrix} H_{AA}^T R_A^{-1} H_{AA} & 0 \\ 0 & {H_{RR}}^T R_R^{-1} H_{RR} \end{bmatrix} \tag{5}$$

where, $R^{-1} = diag(R_A^{-1}, R_R^{-1})$. In (5), $G_{AA} \in \mathcal{M}(n-1, n-1)$, while $G_{RR} \in \mathcal{M}(n, n)$. This is because the voltage angle at slack bus is the reference, so derivatives of measurements to the slack bus angle are not included in (4). And the equation (3) can be rewritten as:

$$\begin{cases} G_{AA}\Delta\theta^k = {H_{AA}}^T R_A^{-1} r_A(x^k) \\ G_{RR}\Delta|V|^k = {H_{RR}}^T R_R^{-1} r_R(x^k) \end{cases} \tag{6}$$

In the fast-decoupled state estimation, estimations of voltage angles and magnitudes are decoupled and the approach to solve the two equations in (6) is the same, e.g. LU decomposition and forward/backward substitution. Therefore, the rest of this paper only elaborates how to efficiently estimate voltage angles using graph computing technology, including matrices formulation and equation solving. The estimation of voltage magnitudes can be implemented in the similar way.

In conventional SE computation, measurements are usually random organized and the corresponding measurement Jacobian matrix will be a sparse matrix with a lot of zero elements. The sparsity of H matrix and G matrix in IEEE 118-bus system is shown in Fig. 4.2.

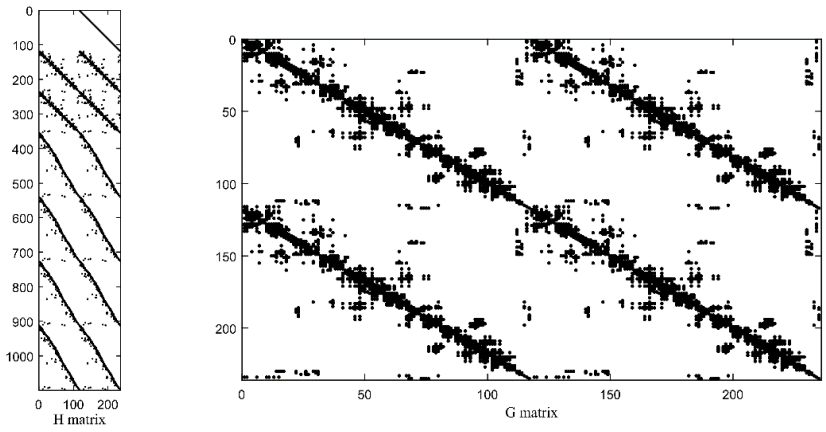

Figure 4.2: Sparsity of H matrix and G matrix

Graph computing technology is employed to first formulate the problem. The system measurements are reordered and partitioned into n parts based on the nodes they belong to. The partition standard of measurement model and the weight matrix are presented below. P_i is the power injection at bus i and P_{ij} is the vector of branch power flows from node i.

$$z_{A,i} = \begin{bmatrix} P_i & P_{ij} \end{bmatrix}^T = h_{A,i}(x) + e_{A,i}, \qquad i = 1, \cdots, n \tag{7}$$

$$z_A = \begin{bmatrix} {z_{A,1}}^T & {z_{A,2}}^T & \cdots & {z_{A,n}}^T \end{bmatrix}^T \tag{8}$$

$$h_A(x) = \begin{bmatrix} h_{A,1}(x)^T & h_{A,2}(x)^T & \cdots & h_{A,n}(x)^T \end{bmatrix}^T \tag{9}$$

The corresponding H matrix and G matrix of node i is:

$$H_{AA,i} = \frac{\partial h_{A,i}(x)}{\partial \theta} \tag{10}$$

$$G_{AA,i} = {H_{AA,i}}^T R_{A,i}^{-1} H_{AA,i} \tag{11}$$

Where ${R_{A,i}}^{-1}$ is the weight matrix for node i's measurements. Then $H_{AA,i}$ and $G_{AA,i}$ can be locally calculated with (10) and (11) in one-step graph traversal.

System level H matrix, H_{AA}, is formulated with aggregation of all node-based H matrices, $H_{AA,i}$, and the system-level gain matrix, G_{AA}, is obtained by summing up these node-based gain matrices, $G_{AA,i}$. The detailed derivations are presented in (12) and (13).

$$H_{AA} = \frac{\partial h_A(x)}{\partial \theta} = \left[{H_{AA,1}}^T \quad {H_{AA,2}}^T \quad \cdots \quad {H_{AA,n-1}}^T \right]^T \tag{12}$$

$$G_{AA} = {H_{AA}}^T {R_A}^{-1} H_{AA} = \sum_{i=1}^{n} {H_{AA,i}}^T \cdot R_{A,i}^{-1} \cdot H_{AA,i} = \sum_{i=1}^{n} G_{AA,i} \tag{13}$$

Where $R_A^{-1} = diag(R_{A,1}^{-1}, R_{A,2}^{-1}, \cdots, R_{A,n}^{-1})$

The RHS update of the active equation/part in (6) can be also implemented by adding each node's RHS update. Each node's RHS update is simply implemented within its 1-step neighboring nodes by local graph traversal.

$${H_{AA}}^T R_A^{-1} r_A(x^k) = \sum_{i=1}^{n} {H_{AA,i}}^T R_{A,i}^{-1} r_{A,i}(x^k) \tag{14}$$

The equations (7)-(14) indicate the feasibility of parallel state estimation. We derive these equations in hope of letting each node compute its own H matrix and G matrix in a parallel way before the results are coordinated to finish the centralized computation. The gain matrix is no longer calculated based on the whole system. Instead, since the gain matrix is proved to be additive, each node will compute its own H matrix and G matrix before they are coordinated to obtain the final gain matrix. Without further dealing with the matrix sparsity, the proposed graph computing method already divides each matrix/vector into n parts so that the computing time complexity is largely reduced.

4.3.2 Graph-Based Parallel State Estimation

With graph computing, non-zero elements in each node's RHS vector are updated independently and directly saved as attributes of corresponding nodes. Then the system-level RHS vector is updated in parallel with superposition of nodes' RHS attributes. This is because elements in $H_{AA,i}$, ${R_{A,i}}^{-1}$ and $r_{A,i}(x^k)$ are saved attributes of node *i*, node *i*'s 1-step neighboring nodes and connecting edges between them. With one-step graph traversal, all required information can be collected in the system graph.

Regarding the formulation of system-level gain matrix, G_{AA}, system topology analysis and theoretical derivation are first conducted to investigate locations and values of the non-zero entries. Then, a graph-based approach is developed to efficiently formulate and store system-level gain matrix. The system topology can be described with adjacency matrix in graph theory. Adjacency matrix A is an $n \times n$ matrix, the entries of which indicate whether vertices are adjacent in the graph.

$$A_{ij} = \begin{cases} 1, & if\ d(i,j) = 1 \\ 0, & if\ d(i,j) \neq 1 \end{cases} \tag{15}$$

The system topology matrix is defined as:

$$T = A + I_n \tag{16}$$

The topology matrix has the same structure with the system nodal admittance matrix Y.

$$T_{ij} = \begin{cases} 1, & if\ Y_{ij} \neq 0 \\ 0, & if\ Y_{ij} = 0 \end{cases} \tag{17}$$

Both the topology matrix and nodal admittance matrix have the same structure as Laplacian matrix, which is the matrix representation of the graph. For the convenience of following topology analysis, the node i and its 1-step neighbors are collected into a set, α_i.

$$\alpha_i := \{j:\ T_{ij} \neq 0\}, \quad j = 1,2,\cdots,n \tag{18}$$

β_i is a set that includes node i, its 1-step neighbors and its 2-step neighbors.

$$\beta_i := \{j:\ T_{ij} \neq 0\} \cup \{j: d(i,j) = 2\} = \alpha_i \cup \{j: d(i,j) = 2\} \tag{19}$$

The 1-step neighbors and 2-step neighbors of node i is:

$$\varphi_1(i) = \alpha_i - \{i\} = \{j: d(i,j) = 1\} \tag{20}$$

$$\varphi_2(i) = \beta_i - \alpha_i = \{j: d(i,j) = 2\} \tag{21}$$

For the H matrix of node i, $H_{AA,i}$, based on (12), column indices of its non-zero entries are the elements in the set of α_i. Besides, the non-zero elements in $G_{AA,i}$ are $\{G_{AA,i}(i,j)\}, i,j \in \beta_i$. In previous mentioned node-based H matrix and gain matrix formulation, the matrix sparsity is inevitably introduced because the derivatives of one node's measurements to its indirectly connected nodes' states are zeros. So, the proposed graph computing approach densifies these node-based matrices and efficiently stores them in graph database. Compared with $H_{AA,i}$, the dimension of the densified matrix $H_{AA,i}^D$, is $M_{A,i} \times n_i$, eliminating columns of indirectly connected nodes. n_i is the size of α_i. Then there is no column with all zero elements existing in $H_{AA,i}$. Regarding the dense node-based gain matrix, take node i as an example, its size reduces from $n \times n$ to $n_i \times n_i$, based on (11).

The IEEE 5-bus system is shown in Fig. 4.3, the node-based H matrix for node 1, presented in (22), is a dense matrix with non-zero column vectors. Its size is 3×3, which is reduced from 3×5. Besides, node 1's dense gain matrix's size is also 3×3, as displayed in (23), much smaller than the conventional one's 5×5.

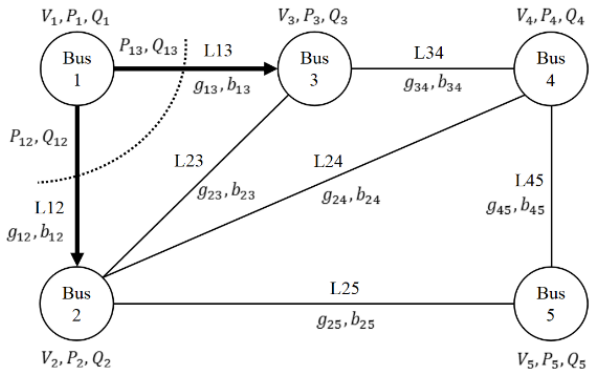

Figure 4.3: IEEE 5-bus system

IEEE 5-bus system is a small-scale system, whose system-level gain matrix and H matrix are very small and dense when compared with large systems. The reduction in matrix dimension would be more evident for large-scale systems.

$$H_{AA,1}^{D} = \begin{bmatrix} \frac{\partial P_1}{\partial \theta_1} & \frac{\partial P_1}{\partial \theta_2} & \frac{\partial P_1}{\partial \theta_3} \\ \frac{\partial P_{12}}{\partial \theta_1} & \frac{\partial P_{12}}{\partial \theta_2} & \frac{\partial P_{12}}{\partial \theta_3} \\ \frac{\partial P_{13}}{\partial \theta_1} & \frac{\partial P_{13}}{\partial \theta_2} & \frac{\partial P_{13}}{\partial \theta_3} \end{bmatrix} \in \mathcal{M}(3,3) \tag{22}$$

$$G_{AA,1}^{D} = {H_{AA,1}^{D}}^{T} \cdot R_{A,1}^{-1} \cdot H_{AA,1}^{D} \in \mathcal{M}(3,3) \tag{23}$$

However, there are still zero entries in $H_{AA,i}^{D}$ if saving the whole matrix. For example, $\frac{\partial P_{12}}{\partial \theta_3}$ is 0. Then the storage efficiency of $H_{AA,i}^{D}$ is further improved by using graph database characteristics. This is because a graph only represents objects and their relations, meaning only non-zero entries in $H_{AA,i}^{D}$ are stored as attributes of nodes/edges in the graph. Take P_{12} as an example, since P_{12} is the active power flow from node 1 to node 2, its derivation to node 3's state is 0, while its derivation to node 1 and node 2 are saved as H attributes in L_{12}. The node-based gain matrix is a dense matrix with all non-zero entries, which is mainly because the node-

based H matrix is full rank. In addition, since the gain matrix is a symmetric matrix, only upper/lower triangular elements need to be calculated and stored.

Regarding the dense node-based gain matrix, although the size is reduced, the system-level gain matrix formulation still has a high time complexity. To further store system gain matrix conveniently and efficiently in graph database, this paper divides the system gain matrix into row vectors, as shown in (24). $G_{AA}(i)$ represents the ith row vector in the system gain matrix G_{AA}, and it is stored in node i as an attribute. The column indices of its non-zero elements belong to β_i. In addition, for each row vector, it is stored in CSR format.

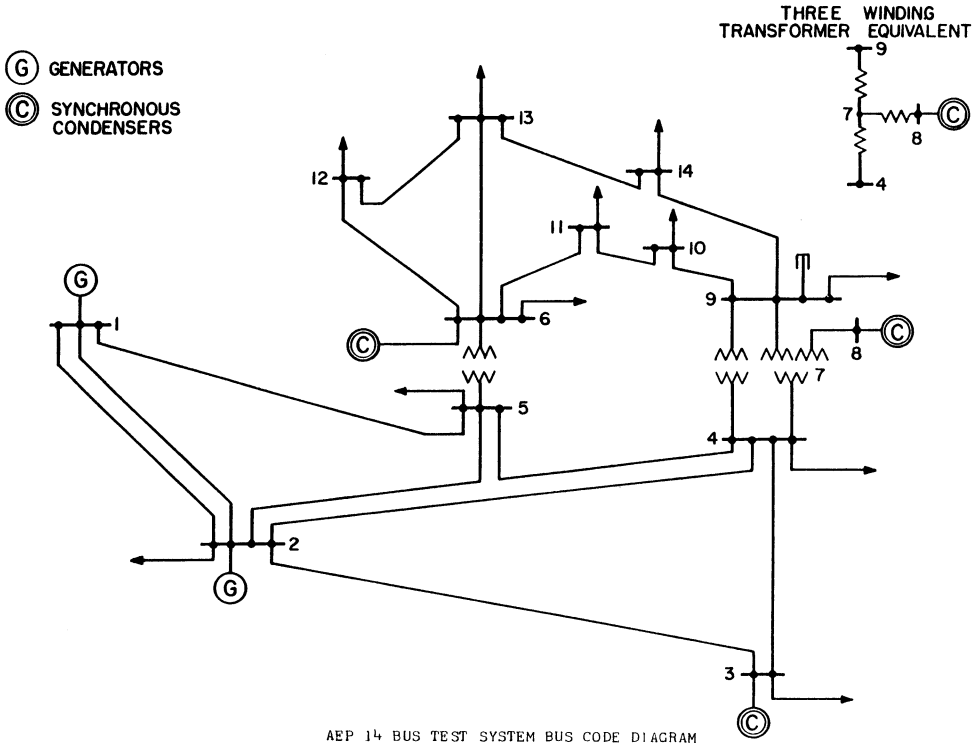

Figure 4.4: IEEE 14-bus system

For an IEEE 14-bus system shown in Fig. 4.4, applying the node-based measurement partition method will lead to a measurement Jacobian matrix with patterns rather than disorganized. The node H matrix of bus 6 as can be seen in Fig. 4.5.

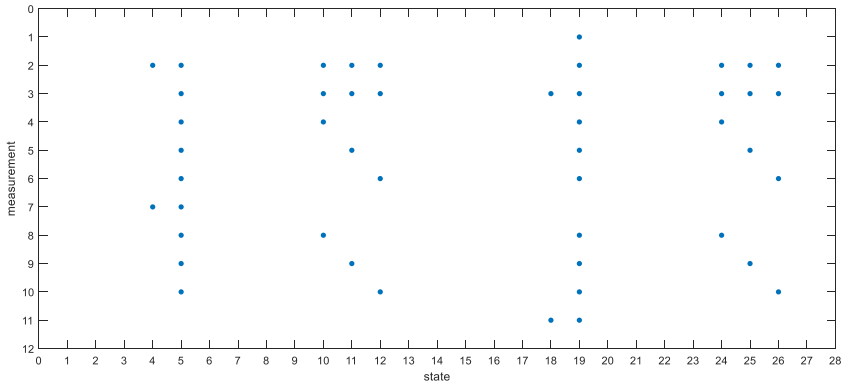

Figure 4.5: Node H matrix of bus 6

After the proposed partition, the node-based measurement Jacobian matrix looks organized as it consists of several column vectors which corresponding to the nodes that it connects with. With the measurement Jacobian matrices with the proposed partition method, the H matrix can be compress horizontally to eliminate irrelevant columns with all zero elements. Suppose the matrix can be compressed, a denser form of the node H matrix can be depicted in Fig. 4.6 below.

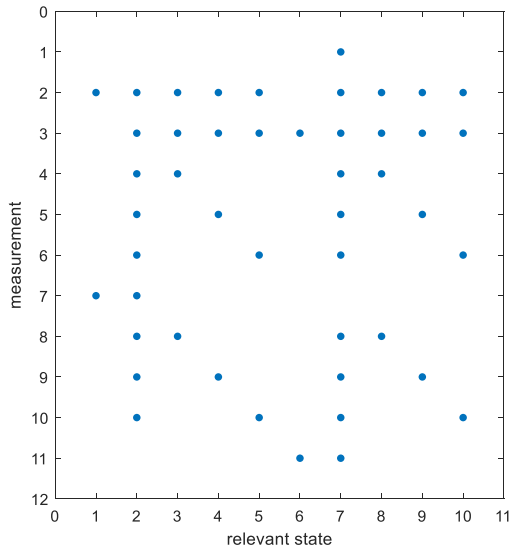

Figure 4.6: Dense node H matrix of bus 6

For the same 14 bus system, its H matrix before the partition and after the node-based partition can be illustrated with Fig. 4.7 below. Compared with the H matrix before the partition, which is disorganized, the H matrix after the node-based partition has certain patterns based on the nodes since the measurements are rearranged based on the nodes that they belong to. The non-zero columns in each small section in the right-hand matrix correspond to the states of the bus(node) that is geographically related to the specified node in the graph. Intuitively, the H matrix after the partition is easier to compress horizontally, which could help deal with the matrix sparsity afterwards.

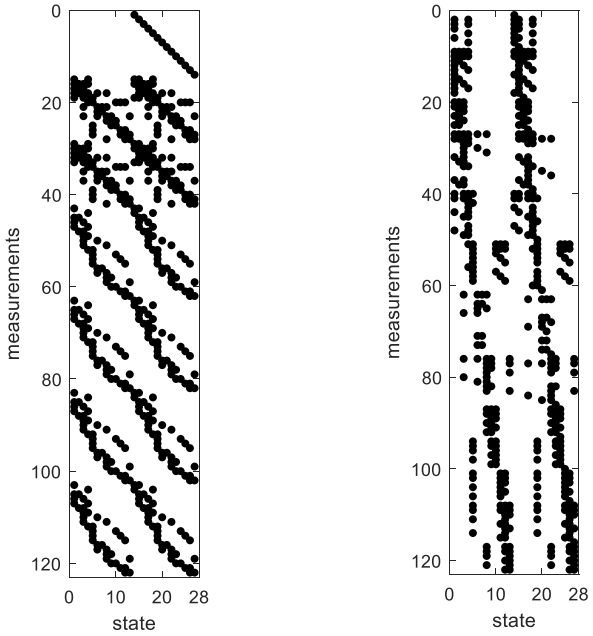

Figure 4.7: H matrix before and after the partition

In the following paragraphs, the location and calculation of the non-zero elements in each row vector are presented. Furthermore, fast decoupled state estimation can be used to further eliminate the zero elements in the dense node H matrix above during SE computation.

$$G_{AA} = [G_{AA}(1)^{T} \quad G_{AA}(2)^{T} \quad \cdots \quad G_{AA}(n)^{T}]^{T} \tag{24}$$

Node i is centered in this figure for the convenience of illustration. Its 1-step neighboring nodes are represented as $\{a^1, b^1, \cdots\}$, and its 2-step neighboring nodes are named as $\{a^2, b^2, \cdots\}$. The superscripts indicate the shortest distance, i.e. fewest steps, from the centered node to the corresponding node.

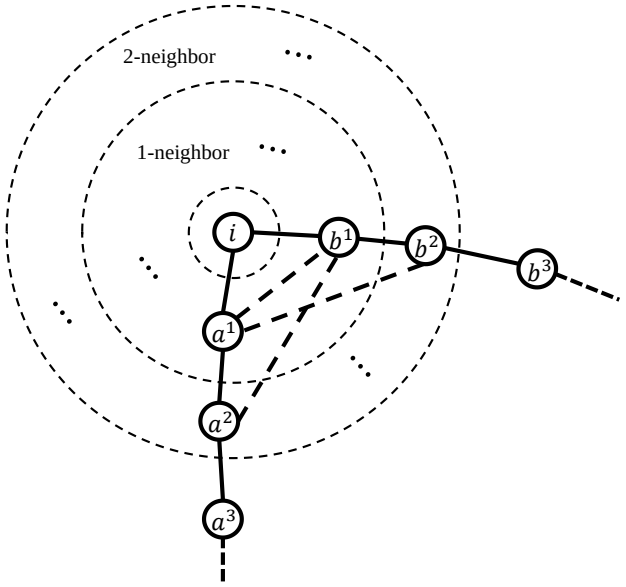

Figure 4.8: Generalized structure of node i centered graph

A generalized structure of one node in a graph is given in Fig. 4.8 above. According to the previous discussion, since node i is only directly connected to its 1-step neighboring nodes, the active power part of its dense H matrix is presented in (25), where $H_{AA,ii} = \frac{\partial h_{A,i}(x)}{\partial \theta_i}$, $H_{AA,ia^1} = \frac{\partial h_{A,i}(x)}{\partial \theta_{a^1}}$, $H_{AA,ib^1} = \frac{\partial h_{A,i}(x)}{\partial \theta_{b^1}}$ are column vectors. To simplify the notation, $H_{AA,ii}$, H_{AA,ia^1}, and H_{AA,ib^1} are denoted as $\mathcal{H}_{ii}$, $\mathcal{H}_{ia^1}$, and $\mathcal{H}_{ib^1}$, respectively.

$$H_{AA,i}^{D} = [H_{AA,ii} \quad H_{AA,ia^1} \quad H_{AA,ib^1}] = [\mathcal{H}_{ii} \quad \mathcal{H}_{ia^1} \quad \mathcal{H}_{ib^1}] \tag{25}$$

The corresponding dense node-based gain matrix will then become:

$$G_{AA,i}^{D} = \begin{matrix} i & a^1 & b^1 \end{matrix} \begin{bmatrix} \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ii} & \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ia^1} & \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ib^1} \\ \mathcal{H}_{ia^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ii} & \mathcal{H}_{ia^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ia^1} & \mathcal{H}_{ia^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ib^1} \\ \mathcal{H}_{ib^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ii} & \mathcal{H}_{ib^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ia^1} & \mathcal{H}_{ib^1}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ib^1} \end{bmatrix} \begin{matrix} i \\ a^1 \\ b^1 \end{matrix} \tag{26}$$

In (26), $\mathcal{R}_i^{-1}$ is a simplified notation for $R_{A,i}^{-1}$. The right-most column outside the matrix indicate the row indices in system-level gain matrix, and the row above the matrix denote the column indices in system-level gain matrix. For example, the first row of the matrix $G_{AA,i}^{D}$ is a part of the ith row vector in the system-level gain matrix, G_{AA}, and it only contributes non-zero elements to columns i, a^1, b^1 in the ith row vector.

In the system-level gain matrix, however, the ith row vector $G_{AA}(i)$ not only includes non-zero entries in the columns of node i and its 1-step neighbors, but also has non-zero elements contributed by node i's 2-step neighbors, as indicated in (27) below.

$$G_{AA}(i) = \begin{bmatrix} \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ii} + \mathcal{H}_{a^1i}{}^{T}\mathcal{R}_{a^1}^{-1}\mathcal{H}_{a^1i} + \mathcal{H}_{b^1i}{}^{T}\mathcal{R}_{b^1}^{-1}\mathcal{H}_{b^1i} \\ \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ia^1} + \mathcal{H}_{a^1i}{}^{T}\mathcal{R}_{a^1}^{-1}\mathcal{H}_{a^1a^1} + \mathcal{H}_{b^1i}{}^{T}\mathcal{R}_{b^1}^{-1}\mathcal{H}_{b^1a^1} \\ \mathcal{H}_{ii}{}^{T}\mathcal{R}_i^{-1}\mathcal{H}_{ib^1} + \mathcal{H}_{a^1i}{}^{T}\mathcal{R}_{a^1}^{-1}\mathcal{H}_{a^1b^1} + \mathcal{H}_{b^1i}{}^{T}\mathcal{R}_{b^1}^{-1}\mathcal{H}_{b^1b^1} \\ \mathcal{H}_{a^1i}{}^{T}\mathcal{R}_{a^1}^{-1}\mathcal{H}_{a^1a^2} + \mathcal{H}_{b^1i}{}^{T}\mathcal{R}_{b^1}^{-1}\mathcal{H}_{b^1a^2} \\ \mathcal{H}_{a^1i}{}^{T}\mathcal{R}_{a^1}^{-1}\mathcal{H}_{a^1b^2} + \mathcal{H}_{b^1i}{}^{T}\mathcal{R}_{b^1}^{-1}\mathcal{H}_{b^1b^2} \end{bmatrix}^{T} \begin{matrix} i \\ a^1 \\ b^1 \\ a^2 \\ b^2 \end{matrix} \tag{27}$$

The following lemmas are derived for entries in node H and node G matrices:

Lemma 1: For node i, only the node itself and its 1-step neighboring nodes provide non-zero entries in column *i* of the system H matrix.

Lemma 2: For node i, the node itself, its 1-step neighboring nodes and its 2-step neighboring nodes contribute to the non-zero entries in row *i* of the system gain matrix, G_{AA}.

The non-zero entries in $G_{AA}(i)$ can be categorized into three sets: diagonal entry= $\{G_{AA}(i,i)\}$, 1-step entry= $\{G_{AA}(i,a^1), G_{AA}(i,b^1), \cdots\}$, 2-step entry= $\{G_{AA}(i,a^2), G_{AA}(i,b^2), \cdots\}$.

The detailed illustrations on the diagonal entry, 1-step entry and 2-step entry using graph theory are presented in Fig. 4.9 − Fig. 4.11 below.

The value of the diagonal entry, e.g. $G_{AA}(i,i)$, is determined by node itself, its 1-step neighbors and connections between them.

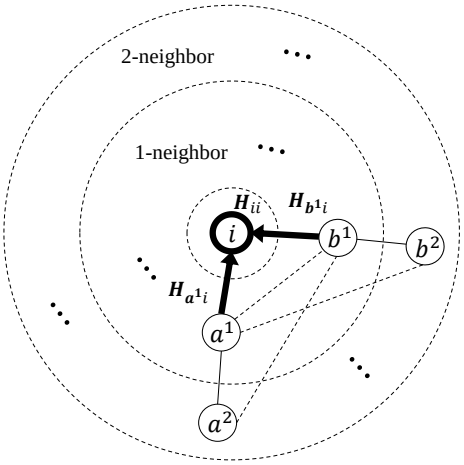

Figure 4.9: Diagonal entry

$\mathcal{H}_{ii}$, $\mathcal{H}_{a^1 i}$, $\mathcal{H}_{b^1 i}$ and corresponding weights at nodes *i*, a^1 and b^1 determine $G_{AA}(i,i)$. In addition, it can be found that all paths from the node *i* to its 1-step neighboring nodes have contributions to the corresponding elements in the 1-step entry.

Node a^1 is node *i*'s 1-step neighbor. The highlight lines are paths between node *i* and a^1 and all of them make contributions to $G_{AA}(i,a^1)$. Similarly, elements in 2-step entry are determined by the paths between node *i* and the corresponding 2-step neighbors. a^2 is the 2-step neighbor of node *i*. Node *i* can travel through the highlight lines to a^2.

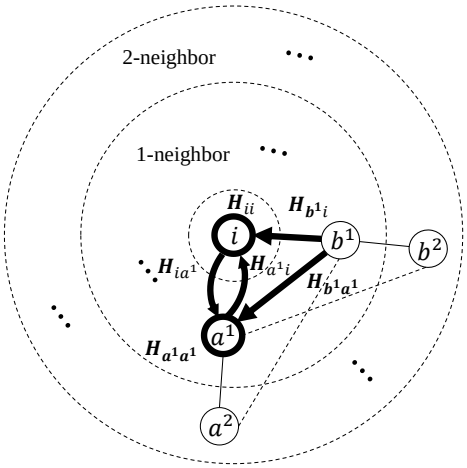

Figure 4.10: 1-step entry (node a^1 as an example)

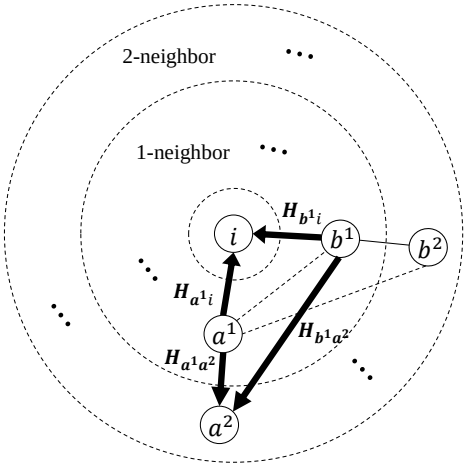

Figure 4.11: 2-step entry (node a^2 as an example)

Based on the above derivations and illustrations, when graph database is utilized for the computation, only 2-step graph traversal is needed. Each row vector in the system-level gain matrix is formulated and stored locally as an attribute for each node. Then the system-level gain matrix is built in parallel by obtaining these row vectors independently from nodes. Based on (27), the ith row of system-level gain matrix are sourced from the ith rows of node gain matrices of i, a^1 and b^1.

$$G^D_{AA,i}(i) = \begin{bmatrix} (\sum B^{(i)})^2 R_{P_i}{}^{-1} + B_{ia^1}{}^2 R_{P_{ia^1}}{}^{-1} + B_{ib^1}{}^2 R_{P_{ib^1}}{}^{-1} \\ -B_{ia^1}(\sum B^{(i)} R_{P_i}{}^{-1} + B_{ia^1} R_{P_{ia^1}}{}^{-1}) \\ -B_{ib^1}(\sum B^{(i)} R_{P_i}{}^{-1} + B_{ib^1} R_{P_{ib^1}}{}^{-1}) \end{bmatrix}^T \begin{matrix} i \\ a^1 \\ b^1 \end{matrix} \tag{28}$$

$$G^D_{AA,a^1}(i) = \begin{bmatrix} B_{a^1 i}{}^2 (R_{P_{a^1}}{}^{-1} + R_{P_{a^1 i}}{}^{-1}) \\ -B_{a^1 i}(\sum B^{(a^1)} R_{P_{a^1}}{}^{-1} + B_{a^1 i} R_{P_{a^1 i}}{}^{-1}) \\ B_{a^1 i} B_{a^1 b^1} R_{P_{a^1}}{}^{-1} \\ B_{a^1 i} B_{a^1 a^2} R_{P_{a^1}}{}^{-1} \\ B_{a^1 i} B_{a^1 b^1} R_{P_{a^1}}{}^{-1} \end{bmatrix}^T \begin{matrix} i \\ a^1 \\ b^1 \\ a^2 \\ b^2 \end{matrix} \tag{29}$$

$$G^D_{AA,b^1}(i) = \begin{bmatrix} B_{b^1 i}{}^2 (R_{P_{b^1}}{}^{-1} + R_{P_{b^1 i}}{}^{-1}) \\ B_{b^1 i} B_{b^1 a^1} R_{P_{b^1}}{}^{-1} \\ -B_{b^1 i}(\sum B^{(b^1)} R_{P_{b^1}}{}^{-1} + B_{b^1 i} R_{P_{b^1 i}}{}^{-1}) \\ B_{b^1 i} B_{b^1 a^2} R_{P_{b^1}}{}^{-1} \\ B_{b^1 i} B_{b^1 b^2} R_{P_{b^1}}{}^{-1} \end{bmatrix}^T \begin{matrix} i \\ a^1 \\ b^1 \\ a^2 \\ b^2 \end{matrix} \tag{30}$$

Where $\sum B^{(i)} = \sum B_{ia^1}$, $a^1 \in \varphi_1(i)$ is the sum of branch susceptance that is connected to node i.

Similarly, the RHS matrix which can be denoted as K. The formulation of node K matrix can be derived as:

$$K_i = \begin{bmatrix} -(\sum B^{(i)}) {R_{P_i}}^{-1} r_{P_i} + \sum B_{ia^1} {R_{P_{ia^1}}}^{-1} r_{P_{ia^1}} \\ -B_{ia^1}({R_{P_i}}^{-1} r_{P_i} + {R_{P_{ia^1}}}^{-1} r_{P_{ia^1}}) \\ -B_{ib^1}({R_{P_i}}^{-1} r_{P_i} + {R_{P_{ib^1}}}^{-1} r_{P_{ib^1}}) \end{bmatrix} \begin{matrix} i \\ a^1 \\ b^1 \end{matrix} \quad (31)$$

The procedures of the proposed graph-based state estimation approach can be summarized as:

Graph Computing based WLS Fast-Decoupled State Estimation Procedure:
1. Start Iterations, set iteration index $k = 0$;
2. Initialize the system state vector x^k, including θ^k and $\|V\|^k$ (flat start or not);
3. Formulate gain matrices, G_{AA} and G_{RR}, based on graph computing;
4. Decompose G_{AA} and G_{RR} using parallel LU solver;
5. Update right-hand-side vector ${H_{AA}}^T R_A^{-1}(z_A - h_A(x^k))$ based on graph computing, solve $\Delta\theta^k$, and update $\theta^{k+1} = \theta^k + \Delta\theta^k$;
6. Check convergence: max $\|\Delta x^k\| \le \epsilon$? If yes, output θ^{k+1} and $\|V\|^k$; If no, go to step 7;
7. Update right-hand-side vector ${H_{RR}}^T R_R^{-1}(z_R - h_R(x^k))$ based on graph computing, solve $\Delta\|V\|^k$, and update$\|V\|^{k+1} = \|V\|^k + \Delta\|V\|^k$;
8. Check convergence: max $\|\Delta x^k\| \le \epsilon$? If yes, output θ^{k+1} and $\|V\|^{k+1}$; If no, $k = k + 1$, go to step 5.

4.4 Illustrative Example

The proposed algorithm will be illustrated with IEEE-14 bus system as shown in Fig. 4.12. We assume that different types of measurements have different standard deviation, and ${\sigma_{V_i}}^2 = 1 \times 10^{-5}, {\sigma_{P_i}}^2 = {\sigma_{Q_i}}^2 = 1 \times 10^{-4}, {\sigma_{P_{ij}}}^2 = {\sigma_{Q_{ij}}}^2 = 6.4 \times 10^{-5}$. Instead of using fast decoupled SE, the example will use Gauss-Newton method which can help understand the graph-base algorithm better and the iteration stops when $|\Delta x^k| \le \varepsilon = 1 \times 10^{-5}$.

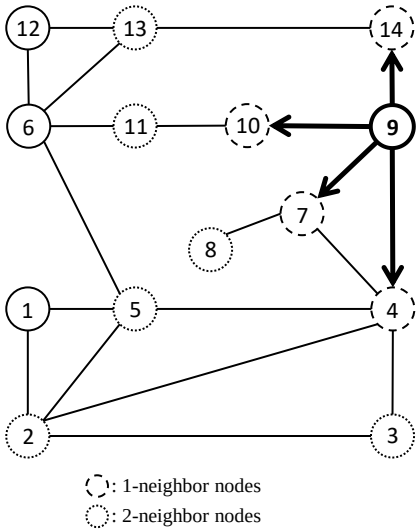

Figure 4.12: IEEE 14-bus system

The graph is partitioned into fourteen parts based on nodes. Bus 9 is chosen as our investigated node in the example. Its 1-neighbor nodes are $\varphi_1(9) = \{4,7,10,14\}$ and its 2-neighbor nodes are $\varphi_2(9) = \{2,3,5,8,11,13\}$. Node H matrix and node G matrix evaluated in the last iteration of bus 9 can be given by:

$$H_9 = \begin{matrix} \theta_4 & \theta_7 & \theta_9 & \theta_{10} & \theta_{14} \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -2.001 & -10.049 & 26.628 & -11.299 & -3.279 \\ 0.160 & 0.280 & -6.110 & 4.212 & 1.459 \\ 0 & 0 & 11.299 & -11.299 & 0 \\ 0 & 0 & 3.279 & 0 & -3.279 \\ -2.001 & 0 & 2.001 & 0 & 0 \\ 0 & -10.049 & 10.049 & 0 & 0 \\ 0 & 0 & -4.212 & 4.212 & 0 \\ 0 & 0 & -1.459 & 0 & 1.459 \\ -0.160 & 0 & -0.160 & 0 & 0 \\ 0 & 0.280 & -0.280 & 0 & 0 \end{bmatrix}$$

$$G_9 = 10^4 \times \begin{matrix} \theta_4 & \theta_7 & \theta_9 & \theta_{10} & \theta_{14} \end{matrix}$$
$$\begin{bmatrix} 10.329 & 20.156 & -60.566 & 23.286 & 6.795 \\ 20.156 & 258.95 & -427.18 & 114.72 & 33.356 \\ \mathbf{-60.566} & \mathbf{-427.18} & \mathbf{1157.9} & \mathbf{-553.80} & \mathbf{-116.35} \\ 23.286 & 114.72 & -553.80 & 372.60 & 43.193 \\ 6.7950 & 33.356 & -116.35 & 43.193 & 33.006 \end{bmatrix}$$

The H matrix and corresponding G matrix of its 1-neighbor nodes are as follow:

$$H_4 = \begin{matrix} \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_7 & \theta_9 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -5.452 & -5.490 & 41.241 & -22.954 & -5.344 & -2.001 \\ 2.348 & 1.889 & -11.737 & 7.936 & -0.277 & -0.160 \\ 0 & 0 & 22.954 & -22.954 & 0 & 0 \\ 0 & 0 & 5.344 & 0 & -5.344 & 0 \\ 0 & 0 & 2.001 & 0 & 0 & -2.001 \\ -5.452 & 0 & 5.452 & 0 & 0 & 0 \\ 0 & -5.490 & 5.490 & 0 & 0 & 0 \\ 0 & 0 & -7.936 & 7.936 & 0 & 0 \\ 0 & 0 & 0.277 & 0 & -0.277 & 0 \\ 0 & 0 & 0.160 & 0 & 0 & -0.160 \\ 2.348 & 0 & -2.348 & 0 & 0 & 0 \\ 0 & 1.889 & 1.889 & 0 & 0 & 0 \end{bmatrix}$$

$$G_4 = 10^4 \times \begin{matrix} \theta_2 & \theta_3 & \theta_4 & \theta_5 & \theta_7 & \theta_9 \end{matrix}$$
$$\begin{bmatrix} 90.276 & 34.363 & -307.42 & 143.76 & 28.484 & 10.536 \\ 34.363 & 86.374 & -301.24 & 141.01 & 28.815 & 10.686 \\ -307.42 & -301.24 & 2919.0 & -1961.4 & -261.89 & -86.965 \\ 143.76 & 141.01 & -1961.4 & 1511.5 & 120.47 & 44.674 \\ 28.484 & 28.815 & -261.89 & 120.47 & 73.378 & 10.740 \\ \mathbf{10.536} & \mathbf{10.686} & \mathbf{-86.965} & \mathbf{44.674} & \mathbf{10.740} & \mathbf{10.329} \end{bmatrix}$$

$$H_7 = \begin{matrix} \theta_4 & \theta_7 & \theta_8 & \theta_9 \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ -5.344 & 21.907 & -6.514 & -10.049 \\ 0.277 & 0.003 & 0.001 & -0.280 \\ 0 & 6.514 & -6.514 & 0 \\ 0 & 10.049 & 0 & -10.049 \\ -5.344 & 5.344 & 0 & 0 \\ 0 & 0.001 & 0.001 & 0 \\ 0 & 0.280 & 0 & -0.280 \\ 0.277 & -0.277 & 0 & 0 \end{bmatrix}$$

$$G_7 = 10^4 \times \begin{matrix} \theta_4 & \theta_7 & \theta_8 & \theta_9 \end{matrix}$$
$$\begin{bmatrix} 73.378 & -161.81 & 34.812 & 53.623 \\ -161.81 & 748.85 & -209.01 & -378.03 \\ 34.812 & -209.01 & 108.74 & 65.458 \\ \mathbf{53.623} & \mathbf{-378.03} & \mathbf{65.458} & \mathbf{258.95} \end{bmatrix}$$

$$H_{10} = \begin{matrix} \theta_9 & \theta_{10} & \theta_{11} \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ -11.271 & 16.042 & -4.770 \\ 4.285 & -6.353 & 2.067 \\ 0 & 4.770 & -4.770 \\ -11.271 & 11.271 & 0 \\ 0 & -2.067 & 2.067 \\ 4.285 & -4.285 & 0 \end{bmatrix}$$

$$G_{10} = 10^4 \times \begin{matrix} \theta_9 & \theta_{10} & \theta_{11} \end{matrix}$$
$$\begin{bmatrix} \mathbf{372.60} & \mathbf{-435.23} & \mathbf{62.625} \\ -435.23 & 567.11 & -131.89 \\ 62.625 & -131.89 & 69.261 \end{bmatrix}$$

$$H_{14} = \begin{matrix} \theta_9 & \theta_{13} & \theta_{14} \end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ -3.216 & -2.432 & 5.647 \\ 1.594 & 1.242 & -2.836 \\ -3.216 & 0 & 3.216 \\ 0 & -2.432 & 2.432 \\ 1.594 & 0 & -1.594 \\ 0 & 1.242 & -1.242 \end{bmatrix}$$

$$G_{14} = 10^4 \times \begin{matrix} \theta_9 & \theta_{13} & \theta_{14} \end{matrix}$$
$$\begin{bmatrix} \mathbf{33.006} & \mathbf{9.799} & \mathbf{-42.805} \\ 9.799 & 19.107 & -28.906 \\ -42.805 & -28.906 & 71.711 \end{bmatrix}$$

Based on all the discussions we have above, the corresponding row in the system gain matrix can be derived for bus 9:

$$G(9) = 10^4 \times \begin{bmatrix} 10.536 \\ 10.686 \\ -60.566 - 86.965 + 53.623 \\ 44.674 \\ -427.18 + 10.740 - 378.03 \\ 65.458 \\ 1157.9 + 10.329 + 258.95 + 372.60 + 33.006 \\ -553.80 - 435.23 \\ 62.625 \\ 9.799 \\ -116.35 - 42.805 \end{bmatrix}^T \begin{matrix} \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \\ \theta_{11} \\ \theta_{13} \\ \theta_{14} \end{matrix} = 10^4 \times \begin{bmatrix} 10.536 \\ 10.686 \\ -93.908 \\ 44.674 \\ -794.47 \\ 65.458 \\ 1832.8 \\ -989.03 \\ 62.625 \\ 9.799 \\ -159.16 \end{bmatrix}^T \begin{matrix} \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_7 \\ \theta_8 \\ \theta_9 \\ \theta_{10} \\ \theta_{11} \\ \theta_{13} \\ \theta_{14} \end{matrix}$$

And for bus 9, node K matrix can be given:

$$K_9 = 10^3 \times \begin{bmatrix} 0.041 \\ 1.010 \\ 1.492 \\ -1.771 \\ -0.771 \end{bmatrix}^T \begin{matrix} \theta_4 \\ \theta_7 \\ \theta_9 \\ \theta_{10} \\ \theta_{14} \end{matrix}$$

Each node in the system will carry out the above graph-based algorithm in parallel until the system gain matrix and K matrix are formed. Then LU decomposition can be applied and obtain system state.

4.5 Case Study

To explore the feasibility of graph computing in power system state estimation, the proposed graph computing methodology is implemented in a Linux server. The testing environment is listed in Table 4.1.

Table 4.1. Test Environment

Hardware Environment	
CPU	2 CPUs × 6 Cores × 2 Threads @ 2.10 GHz
Memory	64 GB
Software Environment	
OS	CentOS 6.8
Graph Database	TigerGraph v0.8.1

The flat-start testing results, including number of iterations and estimation accuracy, are displayed in Table 4.2. IEEE standard systems, a real provincial system in China and a large system with 10790 buses, are tested. As shown in Table 4.2, the estimation results are very close to the system measurements with negligible mean squared errors. The convergence tolerances for voltage magnitude and phase angle are 1.0E-4.

Table 4.2. Number of Iterations and Computing Accuracy

Results / Systems	Number of Iterations	Mean Squared Error	
		Voltage Magnitude (p.u.)	Voltage Angle (degree)
IEEE 14-Bus	5	0	7.03E-8
IEEE 118-Bus	5	6.78E-14	3.75E-9
Sichuan-2433	7	1.42E-12	5.97E-10
MP-10790	8	1.55E-7	4.79E-7

The computation speed and parallelism of the graph-based state estimation is analyzed and presented as follows. In the PNNL technical report [23], the commercial EMS SE function was used as the benchmark, whose CPU frequency is 2.67 GHz. It costs ~5.70 seconds in average for the BPA system model, which has approximately 7500 buses, 9300 branches and 27000 measurements. The convergence tolerance for voltage magnitude and phase angle were 1.0E-2 and 2.0E-2, respectively. The SE function in Sichuan Grid EMS takes ~1.64 s at 2.30 GHz CPU frequency, which is around 1.30 s when normalized at 2.67 GHz. The Sichuan grid model is a 2433-bus system, having 2902 branches and 8786 measurements. In addition, its SE convergence thresholds are set at 1.0E-4 for both voltage magnitude and phase angle. Based on

the two commercial SE cases, it can be approximately estimated that, if using commercial EMS, the MP-10790 case SE function would consume 6~7 s at 2.67 GHz CPU frequency. With the proposed graph computing based parallel SE, the computation time of Sichuan-2433 and MP-10790 cases, if using 8 running threads, are respectively 93.96 ms and 401.30 ms at 2.10 GHz CPU frequency. Their normalized time are 73.90 ms and 315.63 ms at 2.67 GHz CPU frequency. The convergence tolerance is 1.0E-4 for both voltage magnitude and phase angle.

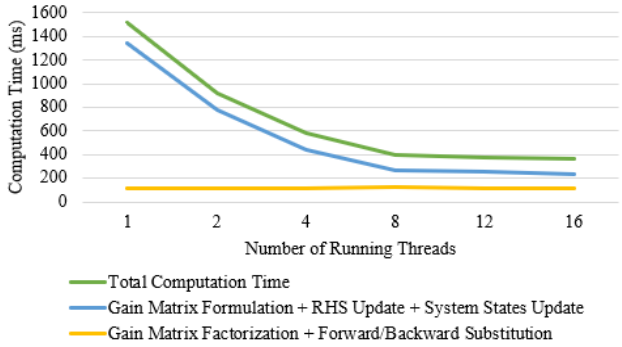

Figure 4.13: Computation time of MP-10790 cases with 1, 2, 4, 8, 12, 16 threads

The time consumption of SE equation solving part changes little, same as the results in [23], while the computation time spent on SE problem formulation and update almost decreases linearly as the number of running threads increases. Therefore, the total time cost changes in the same trend as the problem formulation and update part. Limited by the server's CPU configuration, the computation time changes little when the number of running threads exceeds 12.

Table 4.3 and Table 4.4 decompose the total computation time into detailed procedures for the two cases. These items include gain matrix formulation, gain matrix factorization, RHS vector update, forward/backward substitution (F/B substitution), and system states update. It can

be seen that the time costs of gain matrix formulation, RHS vector update, and system states update reduce with the increase of running threads, while the time consumption of gain matrix factorization and F/B substitution is stable with few variations. They are categorized into two components: 1) SE problem formulation and update component, including gain matrix formulation, RHS vector update, and system state update, 2) SE equation solving component, including gain matrix factorization and F/B substitution. Fig. 4.13 plots the trends of the computation time changes with the increase of running threads in MP-10790 case.

Table 4.3. MP-10790 Case Computation Time with 1, 4, 8, 12, 16 Threads

Threads	Gain Matrix Formulation	Gain Matrix Factorization	Per Iteration		
			RHS Vector Update	F/B Substitution	System State Update
1	349.88	101.95	113.06	1.83	11.37
2	211.24	100.99	63.72	1.83	6.89
4	112.72	103.27	37.03	1.81	4.39
8	72.06	105.77	22.44	1.77	2.51
12	70.84	102.68	19.87	1.73	2.58
16	71.50	101.59	18.42	1.75	2.54

Table 4.4. Sichuan-2433 Case Computation Time with 1, 4, 8, 12, 16 Threads

Threads	Gain Matrix Formulation	Gain Matrix Factorization	Per Iteration		
			RHS Vector Update	F/B Substitution	System State Update
1	70.89	13.57	23.48	0.34	2.77
2	39.94	13.21	14.12	0.33	1.87
4	28.13	13.12	8.92	0.31	1.36
8	18.83	13.38	6.10	0.31	0.94
12	15.88	13.64	6.10	0.34	0.96
16	15.66	13.36	6.05	0.31	1.03

4.6 Conclusion

In this work, a graph-computing based high-performance parallel state estimation algorithm is presented. With the graph database, system measurements are first partitioned and stored at corresponding nodes. Then node-based gain matrices, H matrices, and RHS vectors are derived and formulated using node-based parallel computing. To further save storage space and reduce the matrix sparsity, node-based dense matrices are presented. In addition, to improve the computation complexity of system-level gain matrix, system topology analysis is conducted to demonstrate that each row vector of system gain matrix can be obtained within 2-step graph traversal and stored as an attribute of the corresponding node. Then each row vector of system-level gain matrix is obtained locally and the whole system gain matrix is formulated in parallel. From the testing results, the computation time of gain matrix formulation, RHS vector update and system states update reduce in company with the increase of running threads. The proposed graph-based parallel computing algorithm can speed up the WLS SE to SCADA- rate and can help achieve real time monitoring of power systems.

REFERENCES

[1] F. C. Schweppe and J. Wildes, “Power system static-state estimation, Part I: exact model,” *IEEE Trans. on Power Apparatus and Systems, vol.* PAS-89, pp. 120–125, 1970.

[2] F. C. Schweppe and Rom D. B, “Power system static-state estimation, Part I: approximate model,” *IEEE Trans. on Power Apparatus and Systems, vol.* PAS-89, pp. 125–130, 1970.

[3] E. Handschin, F. C. Schweppe, J. Kohlas and A. Fiechter, “Bad data analysis for power system state estimation,” *IEEE Trans. on Power Apparatus and Systems,* vol. 94, no. 2, pp. 329-337, 1975.

[4] Y. Zhou, and L. Xie, “Detection of bad data in multi-area state estimation,” in *Power and Energy Conference (TPEC),* pp. 1-6, Feb, 2017.

[5] J. H. Eto and R.J. Thomas, “Computational needs for the next generation electric grid,” in *Proc. DOE workshop*, 2011.

[6] C. W Brice and R. K. Cavin, “Multiprocessor static state estimation,” *IEEE Trans. on Power Apparatus and Systems, vol.* PAS-101, pp. 302–308, 1982.

[7] A. Abur and P. Tapadiya, “Parallel state estimation using multiprocessors,” *Electrical Power Systems Research, vol.* 18, pp. 67–73, Jan. 1990.

[8] Th. Van Cutsem, J.L. Howard, M. Ribbens-Pavella and Y.M. EI-Fattah, “Hierarchical state estimation,” *International Journal of Electrical Power & Energy Systems,* vol. 2, no. 2, pp. 70-80, Apr 1980.

[9] Th. Van Cutsem, J. L. Horward, and M. Ribbens-Pavella, “A two-level static state estimator for electric power systems,” *IEEE Trans. on Power Apparatus and Systems, vol.* PAS-100, pp. 3722–3732, 1981.

[10] G. N. Korres, "A distributed multiarea state estimation," *IEEE Trans. on Power Systems*, vol. 26, no. 1, pp. 73-84, Feb. 2011.

[11] L. Xie, D.-H. Choi, S. Kar, and H. V. Poor, "Fully distributed state estimation for wide-area monitoring systems," *IEEE Trans. on Smart Grid*, vol. 3, no. 3, pp. 1154-1169, Sep. 2012.

[12] D. M. Falcao, F. F. Wu, and L. Murphy, "Parallel and distributed state estimation," *IEEE Trans. on Power Systems,* vol. 10, no. 2, pp. 724-730, May. 1995.

[13] J. B. Carvalho and F. M. Barbosa, "A parallel algorithm to power systems state estimation," in *Proc. Power System Technology, POWERCON*, 1998, pp. 1213–1217.

[14] V. Kekatos and G.B. Giannakis, "Distributed robust power system state estimation," *IEEE Trans. on Power Systems,* vol. 28, no. 2, pp. 1617-1626, May. 2013.

[15] H. Sasaki, K. Aoki, and R. Yokoyama, "A parallel computation algorithm for static state estimation by means of matrix inversion lemma," *IEEE Trans. on Power Systems*, vol. 2, no. 3, pp. 624-632, Aug. 1987.

[16] Y. Chen, S. Jin, M. Rice, and Z. Huang, "Parallel state estimation assessment with practical data," in *Proc. IEEE PES GM,* Vancouver, BC, Canada, 2013, pp. 1–5.

[17] Y. Chen, M. Rice, and Z. Huang, "SCADA-rate parallel state estimation assessed with utility data," in *Proc. IEEE PES GM,* National Harbor, MD, USA, 2014, pp. 1–5.

[18] A. Minot, Y. M. Lu, and N. Li, "A distributed Gauss-Newton method for power system state estimation," *IEEE Trans. on Power Systems*, vol. 31, no. 5, pp. 3804-3815, Sep. 2016.

[19] V. Seshadri Sravan Kumar, D. Thukaram, "State estimation in power systems using linear model infinity norm-based trust region approach," *IET Generation, Transmission & Distribution,* vol 7, no. 5, pp. 500-510, May. 2013.

[20] H. Zhu, and G. B. Giannakis, “Power system nonlinear state estimation using distributed semidefinite programming,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 6, pp. 3804-3815, Dec. 2014.

[21] H. Karimipour and V. Dinavahi, “Parallel relaxation-based joint dynamic state estimation of large-scale power systems,” *IET Generation, Transmission & Distribution*, vol 10, no. 2, pp. 452-459, Feb. 2016.

[22] F. Harary, Graph Theory, Addison-Wesley, 1969, p.199

[23] Y. Chen, M. Rice, K. Glaesemann, S. Wang, and Z. Huang, “Sub-second parallel state estimation,” *Pacific Northwest National Laboratory*, 2014. [Online]. Available: http://www.pnnl.gov/main/publications/external/technical_reports/PNNL-23830.pdf.

[24] A. Abur and A. G. Exposito, *Power System State Estimation: Theory and Implementation*. Marcel Dekker, 2004.

[25] Y. Liu, P. Ning, and M. K. Reiter, “False data injection attacks against state estimation in electric power grids,” in *Proc. ACM Conf. Computer and Communications Security*, pp. 21-32, 2009.

[26] M. A. Rahman, and H. Mohsenian-Rad, “False data injection attacks with incomplete information against smart power grids,” in *Global Communications Conference*, pp. 3153-3158, Dec. 2012.

[27] S. Wang, and W. Ren, “Stealthy false data injection attacks against state estimation in power systems: Switching network topologies,” in *American Control Conference*, pp. 1572-1577, Jun. 2014.

[28] J. Kim, and L. Tong, “On topology attack of a smart grid: Undetectable attacks and countermeasures,” *IEEE J. Sel. Areas Commun.*, vol. 31, no. 7, pp. 1294-1305, Jul. 2013.

[29] X. Liu, Z. Bao, D. Lu, and Z. Li, "Modeling of local false data injection attacks with reduced network information," *IEEE Trans. on Smart Grid*, vol. 6, no. 4, pp. 1686-1696, July. 2015.

[30] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *Proc. 2010 First IEEE International Conference on Smart Grid Communications*, pp. 226-231, Oct. 2010.

[31] L. Xie, Y. Mo, and B. Sinopoli, "Integrity data attacks in power market operations," *IEEE Trans. on Smart Grid*, vol. 2, no. 4, pp. 659-666, Dec. 2011.

[32] L. Jia, J. Kim, R. J. Thomas and L. Tong, "Impact of data quality on real-time locational marginal price," *IEEE Trans. on Power Systems* vol. 29, no. 2, pp. 627-636, Mar. 2014.

[33] J. Zhao, G. Zhang, M. L. Scala, Z. Y. Dong, C. Chen and J. Wang, "Short-term state forecasting-aided method for detection of smart grid general false data injection attacks," *IEEE Trans. on Smart Grid*, vol. 8, no. 4, pp. 1580-1590, July. 2017.

[34] T. T. Kim, and H. V. Poor, "Strategic protection against data injection attacks on power grids," *IEEE Trans. on Smart Grid*, vol. 2, no. 2, pp. 326-333, Jun. 2011.

[35] F. F. Wu, and W. E. Liu, "Detection of topology errors by state estimation," *IEEE Trans. on Power Systems*, vol. 4, no. 1, pp. 176-183, Feb. 1989.

[36] M. Prais, and A. Bose, "A topology processor that tracks network modifications," *IEEE Trans. on Power Systems*, vol. 3, no. 3, pp. 992-998, Aug. 1988.

[37] I. S. Costa, and J. A. Leao, "Identification of topology errors in power system state estimation," *IEEE Trans. on Power Systems*, vol. 8, no. 4, pp. 1531-1538, Nov. 1993.

[38] M. B. Do Coutto Filho, and J. C. S. de Souza, "Forecasting-aided state estimation-Part I: Panorama," *IEEE Trans. on Power Systems*, vol. 24, no. 4, pp. 1667-1677, Sep. 2009.

[39] M. Hassanzadeh, and C. Y. Evrenosoglu, “Power system state forecasting using regression analysis,” in *Proc. IEEE PES GM,* San Diego, CA, USA, 2012, pp. 1–6.

[40] Y. Gu, T. Liu, D. Wang, X. Guan, and Z. Xu, “Bad data detection method for smart grids based on distributed state estimation,” in *Proc. IEEE ICC,* June 2013.

[41] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed., Duxbury, 2002.

[42] F. F. Wu, K. Moslehi, and A. Bose, “Power system control centers: Past, present, and future,” *Proc. IEEE,* vol. 93, no. 11, pp. 1890-1908, Nov. 2005.

[43] A. P. S. Meliopoulos, B. Fardanesh, and S. Zelingher, “Power system state estimation: Modeling error effects and impact on system operation,” *Proc. Hawaii International Conference on System Sciences*, pp. 682-690, Jan. 2001.

[44] C. Yuan, Y. Zhou, G. Zhang, G. Liu, R. Dai, X. Chen, and Z. Wang, “Exploration of graph computing in power system state estimation,” *arXiv preprint arXiv: 1803.03300,* 2018