Stony Brook University

Real-Time Power Flow Analysis & Short-term Electricity Load Forecasting in Smart Grid

A Dissertation Presented

by

Muqi Li

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

May 2015

Stony Brook University

The Graduate School

Muqi Li

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Eugene Feinberg - Dissertation Advisor
Distinguished Professor, Department of Applied Mathematics and Statistics

Roman Samulyak - Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

Jiaqiao Hu - Member
Associate Professor, Department of Applied Mathematics and Statistics

Thomas Robertazzi - Outside Member
Professor, Department of Electrical and Computer Engineering

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Real-Time Power Flow Analysis & Short-term Electricity Load Forecasting in Smart Grid

by

Muqi Li

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2015

This dissertation studies two problems in smart grid, one of which is real-time power flow analysis. Power flow analysis is used to obtain the steady-state voltage phasors for the power system. The ability to perform power flow analysis quickly is essential for the successful implementation of advanced real-time control of transmission systems. We describe a sensor placement algorithm for conducting real-time parallel transmission network power flow computations. In particular, Phasor Measurement Units (PMUs) can be such

sensors. Graph partitioning is used to decompose the system into several subsystems and to locate sensors in an efficient way. Power flow calculations are then run in parallel for each area. Test results on the IEEE 118- and 300-bus systems show that the proposed algorithm is faster than the traditional (serial) Newton's method, and is suitable for real-time applications.

Electricity load forecastingis another problem investigated in this dissertation. Electric load forecasting techniques are used by most electric utility companies for operation and planning. Many operational and financial decisions are based on load forecasting, such as reliability analysis, voltage control, unit commitment,security assessment, and in purchasing electric power. We focus on short-term electric load forecasting. For this problem we present two models that predict future electricity demands based on historical hourly load and hourly weather information. A data cleaning scheme is applied to make the models robust. The estimation of the next day load is performed with an Artificial Neural Network (ANN) method and a Modified Statistical Learning method (MSL). We compare the results obtained by ANN and MSL method. Numerical testing shows that both methods provide accurate predictions.

Key Words: load forecasting, power flow calculation, smart grid, Neural Network, PMU placement

To my Family

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to my adviser, Professor Eugene Feinberg, for suggesting this important and exciting thesis topic and for his advice, support, and guidance toward my Ph.D. degree. He taught me not only the way to do scientific research, but also the way to become a professional scientist. He is my adviser and a good friend.

I would like to thank Professors Roman Samulyak, Jiaoqiao Hu and Haipeng Xing who provided encouragement and valuable technical knowledge. I would like to acknowledge the support of Department of Applied Mathematics and Statistics and its staff.

I would like to thank all my friends during my years of study as a graduate student at Stony Brook for their friendship and encouragement. I would like to mention Dr. Eting Yuan, Dr. Jun Fei, Dr. Guoli Sun, Dr. Bowen Song, Dr. Qiangqiang Shi, Jefferson Huang and Manasa Mandava.

Throughout my academic career, the constant support and unconditional love of my parents have always motivated me to strive forward. My dissertation is dedicated to them.

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000220.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, pro process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report was prepared as an account of work performed by The Research Foundation of SUNY as sub-recipient of any award made by an agency of the United States Government to the Long Island Power Authority. Neither the Long Island Power Authority nor any of its trustees, employees or subsidiaries, nor the State of New York, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring the Long Island Power Authority, its trustees or employees, or by the State of New York. The views and

opinions of authors expressed herein do not necessarily state or reflect those of the Long Island Power Authority, its trustees or employees, or of the State of New York.

Chapter 1

Real-Time Power Flow Calculation

1.1 Introduction

Fast load flow analysis is essential for the successful implementations of advanced real-time control of transmission systems. The nature tool for this is parallel computation. To this end, various parallel methods have been proposed. Many traditional approaches [42, 66, 71] use factorization and the forward-backward solution of linear equations to achieve parallelism. As many serial computations are needed for such approaches, however, their parallel efficiency is not high; their performance is also dependent on computer architecture.

Another approach to alleviating the computational burden is to decompose a large problem into a number of small problems and perform computations for each sub-problem in parallel. These smaller sub-problems are usually coordinated by a master process. Rafian et al. [53] presented a method for load-flow analysis based on tearing the network into 2-3 subsystems. In every iteration, the subsystems are solved in parallel and will communicate with a

coordinating program. After the communication, the coordinating program is then conducted to determine the global solution for the original system. Chen [9] et al. proposed a parallel solution based on piecewise method. The Jacobian matrix is converted into a bordered block diagonal form. The block diagonal form leads to subproblems that can be solved independently, after which a problem corresponding to the border is solved to coordinate the subproblems and obtain a solution to the original problem. Amano et al. [43] employed a block-parallel method for load-flow analysis. In particular, the Jacobian matrix is constructed by applying the epsilon decomposition algorithm, which eliminates weak coupling elements from the matrix. One drawback of this approach is that the transformation of the matrix into a balanced diagonal matrix takes time; further, the speed of convergence is affected by the choice of partition method.

Traditional power flow calculations use only power injection measurements as the input. However, at the present time it is also possible to have synchronized voltage measurements at buses. For example, it is possible to use Phasor Measurement Units (PMUs) for this. These devices provide accurate real-time measurements at multiple remote points on the grid. A number of significant improvements in control and analytical capabilities have been made possible by this technology. However, due to their advanced features and the need for communications infrastructure, PMUs are relatively costly to implement and maintain. Hence it is typically not economical to install a PMU at each bus [45]. This has motivated a growing literature on the optimal placement of PMUs, and various PMU placement algorithms have been devel-

oped for different situations. In [8], the PMU placement problem was modeled as an integer linear programming problem with the constraint that the entire power system should be observable. A mixed integer linear programming formulation was introduced in [15] for PMU placement that accounts for line and PMU contingencies. Gou [7] presented a model that allows for redundant PMU placement and incomplete observability, which can be solved by integer linear programming. Chakrabarti and Kyriakides [59] proposed an exhaustive search algorithm for PMU placement that takes so-called zero injection buses, i.e. buses with neither generation nor load, into account. In particular, systems with such buses may need fewer PMUs.

In this chapter, we investigate the possibility of using additional sensor measurements, e.g. from PMUs, to decompose the power system into several parts and conduct power flow calculation in parallel. The idea of using PMU is motivated by decomposition method, e.g. [53, 9], for solving power flow problem. Instead of using a master process to coordinate the subsystems in each iteration, we achieve the coordination directly by using PMU measurements, which results in a simple and efficient solution. The placement of PMUs is based on solving a graph partition problem. By placing a relatively small number of additional PMUs, a large power system can be divided into several non-overlapping smaller subsystems that can be solved in parallel. Numerical results on 118 and 300 bus power networks demonstrate the efficacy of the proposed approach in significantly reducing the computation time.

The rest of the chapter is organized as follows. Section 2.2 briefly describes the power flow problem. Section 2.3 introduces the traditional parallel

method for power flow analysis. Section 2.4 gives a overview of PMU and graph partitioning method. Section 2.5 presents the proposed method and the details of PMU placement methods. Section 2.6 illustrates the implementation of the algorithm via numerical examples on two power systems.

1.2 Power Flow Analysis

Power flow analysis, commonly known as load flow analysis, form an important part of power system analysis. The goal of a power flow analysis is to obtain complete voltage angle and magnitude information for each bus in a power system under balanced three-phase steady state conditions. Load flow analysis can provide a balanced steady operation state of the power system, without considering system transient processes. Hence, the mathematical model of load flow problem is a nonlinear algebraic equation system.

The solution to the power flow problem begins with identifying the known and unknown variables in the system. The known and unknown variables are dependent on the type of bus. There are three types of buses which can be classified as:

1. **Load Bus:** A bus without any generators connected to it. Here the real power P and reactive Q are known. It is also known as PQ buses.

2. **Generator Bus:** A bus with at least one generator connected to it. Here the real power P and the voltage magnitude $|V|$ are known. It is also known as PV buses.

3. **Slack Bus** A bus to balance the active and reactive power in the system. Here the voltage magnitude $|V|$ and phase angle δ are known.

For each Load bus, both the voltage magnitude and angle are unknown and must be solved for; for each generator bus, the voltage angle must be solved for; there are no variables that must be solved for the slack bus. Suppose that a power system with n bus and r generators, there are then $2(n-1)-(r-1)$ unknowns. $2(n-1)-(r-1)$ power balance equations are introduced to solve these unknowns. The power balance equations can be written as follows:

$$P_i = \sum_{j=1}^{n} |V_i||V_j|(G_{ij}cos\theta_{ij} + B_{ij}sin\theta_{ij}) \tag{1.1}$$

$$Q_i = \sum_{j=1}^{n} |V_i||V_j|G_{ij}sin\theta_{ij} - B_{ij}cos\theta_{ij}) \tag{1.2}$$

Here P_i and Q_i are the real and reactive power injections at bus i; V and θ are the real bus voltage magnitude and phase angle; G and B are the real and imaginary parts of the bus admittance matrix.

Mathematically, the load flow problem is a problem of solving a system of nonlinear algebraic equations. Its solution usually obtained by some iteration process. Thus reliable convergence becomes the prime criterion for a load flow calculation method. Newton-Raphson method is one of the most wildly used method for solving nonlinear equations in power system.

The Newton-Raphson method is an efficient algorithm to solve nonlinear equations. It transform the procedure of solving nonlinear equations into the

procedure of repeatedly solving linear equations. Expanding the Equations (1.1) and (1.2) using Taylor series and ignoring the higher order terms results in the following linear equations.

$$\begin{bmatrix} \Delta P_2 \\ \vdots \\ \Delta P_n \\ \Delta Q_2 \\ \vdots \\ \Delta Q_n \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_n} & \frac{\partial P_2}{\partial |V_2|} & \cdots & \frac{\partial P_2}{\partial |V_n|} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_n}{\partial \theta_2} & \cdots & \frac{\partial P_n}{\partial \theta_n} & \frac{\partial P_n}{\partial |V_2|} & \cdots & \frac{\partial P_n}{\partial |V_n|} \\ \frac{\partial Q_2}{\partial \theta_2} & \cdots & \frac{\partial Q_2}{\partial \theta n} & \frac{\partial Q_2}{\partial |V_2|} & \cdots & \frac{\partial Q_2}{\partial |V_n|} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_n}{\partial \theta_2} & \cdots & \frac{\partial Q_n}{\partial \theta_n} & \frac{\partial Q_n}{\partial |V_2|} & \cdots & \frac{\partial Q_n}{\partial |V_n|} \end{bmatrix} \begin{bmatrix} \Delta \theta_2 \\ \vdots \\ \Delta \theta_n \\ \Delta |V_2| \\ \vdots \\ \Delta |V_n| \end{bmatrix} \tag{1.3}$$

where Δ P and Δ Q are called the mismatch equations:

$$\Delta P_i = -P_i + \sum_{j=1}^{n} |V_i||V_j|(G_{ij}cos\theta_{ij} + B_{ij}sin\theta_{ij}) \tag{1.4}$$

$$\Delta Q_i = -Q_i + \sum_{j=1}^{n} |V_i||V_j|G_{ij}sin\theta_{ij} - B_{ij}cos\theta_{ij}) \tag{1.5}$$

Equation (1.3) can be written as Equation (1.6) below.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} J_1 & J_2 \\ J_3 & J_4 \end{bmatrix} \begin{bmatrix} \Delta \theta \\ \Delta |V| \end{bmatrix} \tag{1.6}$$

Here J is a matrix of partial derivatives known as the Jacobian. The linearized system of equations is solved to determine the next guess of voltage magnitude $|V|$ and angle θ based on:

$$\theta^{m+1} = \theta^m + \Delta\theta \tag{1.7}$$

$$V^{m+1} = |V|^m + \Delta|V| \tag{1.8}$$

The process continues until a stopping condition is met.The stopping criteria is usually set as the norm of $\Delta\theta$, $\Delta|V|$ is less than 10^{-5}

1.3 Parallel Methods for Power Flow Analysis

As we discussed in the introduction, network decomposition method is a promising approach to reduce the problem size of power flow calculation. The basic idea behind such approach is to divide an interconnected network into independent sub-networks and a collection of cutting nodes [65]. The nodes in the jacobian matrix are divided into a bordered block diagonal form. The nodes in sub-block only connect to the nodes within the same sub-block and nodes in the bordered block, while nodes in the bordered block connect to the nodes in different sub-blocks. The block diagonal form leads to subproblems that can be solved independently, after which a problem corresponding to the border is solved to coordinate the subproblems and obtain a solution to the original problem. The bordered block diagonal form(BBDF) matrix with four subnetworks is illustrated in Figure 1.1.

The general parallel solution scheme for the BBDF matrix is that each sub-block block with its associated bordered block will be allocated to a processor and solved in parallel. After coordinate the results between each processor

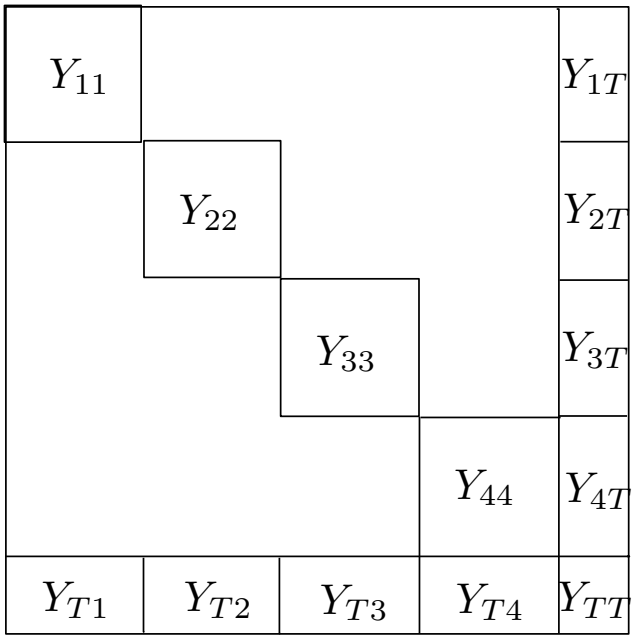

Figure 1.1: Example bordered block diagonal form partitioning when sub-block=4

and the bordered block, the global results can be determined. The details of the procedure is described as follows: Consider the Jacobian matrix Y has been decomposed into a BBDF form. Y_{ii}, Y_{iT} and Y_{Ti}, $i = 1, 2, \ldots, N$ are matrix blocks representing a sub-block. Y_{TT} represents cutting nodes and is referred as the last block. All the other blocks are all zeros. The linear equations sets in Newtons methods can be determined as follows:

$$\begin{bmatrix} Y_{11} & & & & Y_{1T} \\ & Y_{22} & & & Y_{2T} \\ & & \ddots & & \vdots \\ & & & Y_{NN} & Y_{NT} \\ Y_{T1} & Y_{T2} & \ldots & Y_{TN} & Y_{TT} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \\ X_T \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \\ b_T \end{bmatrix} \tag{1.9}$$

After the LU decomposition, the Jacobian matrix becomes:

$$\begin{bmatrix} Y_{11} & & & & Y_{1T} \\ & Y_{22} & & & Y_{2T} \\ & & \ddots & & \vdots \\ & & & Y_{NN} & Y_{NT} \\ Y_{T1} & Y_{T2} & \dots & Y_{TN} & Y_{TT} \end{bmatrix} = \begin{bmatrix} L_{11} & & & & \\ & L_{22} & & & \\ & & \ddots & & \\ & & & L_{NN} & \\ L_{T1} & L_{T2} & \dots & L_{TN} & L_{TT} \end{bmatrix} \begin{bmatrix} U_{11} & & & & U_{1T} \\ & U_{22} & & & U_{2T} \\ & & \ddots & & \vdots \\ & & & U_{NN} & U_{NT} \\ & & & & U_{TT} \end{bmatrix} \tag{1.10}$$

According to equation (1.10), the following can be derived:

$$Y_{ii} = L_{ii}U_{ii} \tag{1.11}$$

$$Y_{T_i} = L_{Ti}U_{ii} \tag{1.12}$$

$$Y_{iT} = L_{ii}U_{iT} \quad i = 1, 2, \dots, N \tag{1.13}$$

$$Y_{TT} = L_{TT}U_{TT} + \sum_{i=1}^{N} L_{Ti}U_{iT} \tag{1.14}$$

Equations(1.11)-(1.13) are solved in parallel first, then after all L_{iT} and L_{Ti} are calculated, equation (1.14) is solved to obtain the complete LU decomposition. the equation (1.9) now can be rewritten as :

$$\begin{bmatrix} L_{11} & & & & \\ & L_{22} & & & \\ & & \ddots & & \\ & & & L_{NN} & \\ L_{T1} & L_{T2} & \dots & L_{TN} & L_{TT} \end{bmatrix} \begin{bmatrix} U_{11} & & & & U_{1T} \\ & U_{22} & & & U_{2T} \\ & & \ddots & & \vdots \\ & & & U_{NN} & U_{NT} \\ & & & & U_{TT} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \\ X_T \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \\ b_T \end{bmatrix} \tag{1.15}$$

and the forward substitution becomes:

$$\begin{bmatrix} L_{11} & & & & \\ & L_{22} & & & \\ & & \ddots & & \\ & & & L_{NN} & \\ L_{T1} & L_{T2} & \dots & L_{TN} & L_{TT} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_N \\ Z_T \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \\ b_T \end{bmatrix} \tag{1.16}$$

In order to solve vector Z, following equations can be obtained.

$$L_{ii} Z_i = b_i \tag{1.17}$$

$$L_{Ti} Z_i = b_{Ti} \tag{1.18}$$

$$L_{TT} Z_T = b_T - \sum_{i=1}^{N} b_{Ti} \tag{1.19}$$

As there is no dependence existed in equation (1.17)(1.18), z_i and b_{Ti} can be solved in parallel for all i, After all result of Z_i and b_{Ti}, $i = 1, \dots, N$ are obtained. equation (1.19) can be solved to obtain Z_T

Similarly, the backward substitution can be formulated as follows:

$$\begin{bmatrix} U_{11} & & & & U_{1T} \\ & U_{22} & & & U_{2T} \\ & & \ddots & & \vdots \\ & & & U_{NN} & U_{NT} \\ & & & & U_{TT} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \\ X_T \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_N \\ Z_T \end{bmatrix} \tag{1.20}$$

It also can be solved in parallel.

This decomposition method gives us an idea to tear the network into pieces, and solve them in parallel. However, the communication time between subsystems can diminish performance gains. In our proposed method, instead of using a master process to coordinate the subsystems in each iteration, we achieve the coordination directly by using PMU measurements, which results in a simple and efficient solution. The detail of the PMU technique method and PMU placement methods will discuss in next two section 1.4 and 1.5.

1.4 Phasor Measurement Unit and Graph Partition Method

This section introduces the Phasor Measurement Units (PMUs) in subsection 1.4.1. The placement of PMU can be treated as a graph partitioning problem, the graph partitioning problem and several solution heuristics are subsequently described in subsection 1.4.2.

1.4.1 Phasor Measurement Unit

Synchronized phasor measurement techniques and hardware were first developed in the early 1980s. Phasor measurement units (PMUs) are contemporary metering devices installed on system buses to measure phasors of bus voltages and currents flowing across lines. After sampling windowing, phasor estimation, and time stamping, measurements are communicated to the intended application through phasor data connector. Merits of PMUs over conventional power meters include increased precision in measuring phasor angles due to network-wide synchronization, and higher sampling rates.It is considered one of the most important measuring devices in the future of power systems.

Using positive sequence voltage and current measurements, a linear estimator algorithm can be formulated. The more PMUs that are installed on the system, the more accurate the estimation will become. However, PMU penetration has so far been rather limited, mainly due to two factors:

1. For large power system, conducting power flow analysis based only on phasor measurements can prove a very expensive solution. The cost of PMUs limits the number that will be installed although an increased demand in the future is expected to bring down the cost. The placement sites are also limited by the available communication facilities, for large power system, the cost of communication facilities may be higher than that of the PMUs.

2. Conducting power flow analysis based only on PMUs would be very

vulnerable to measurement errors or telemetry failures. It is often desired that traditional SCADA (Supervisory control and data acquisition n) data and phasor data be combined to form a data set for a hybrid power flow calculation.

Methods for determining a minimum number of PMU installations have been actively investigated in recent years, these PMU placement algorithms are usually developed to achieve a full observability of an electric power system. Baldwin *et al.*[62] formulated the PMU placement problem as an integer linear programming problem with the constraint that the entire power system should be observable. If the number of available PMUs are not sufficient, the technique proposed in [49] based on incomplete observability can be used for optimal PMU placement. Other placement algorithms have been developed for specific applications such as islanding [48] , contingencies [45] and bad data processing [11].

We investigate the possibility of using additional PMUs to decompose the power system into several parts and conduct power flow calculation in parallel. PMUs will be installed at selected boundary buses to make sure the voltage phasors at all boundary buses are known. In other words, PMUs are placed to make all boundary buses observable. The cost associated with such a decomposition depends on the number of PMUs installed at boundary buses. Having roughly the same number of buses per subsystem is desirable from the standpoint of balancing computational load. Thus our objective is to minimize the number of PMUs installed at boundary buses subject to the condition that each subsystem has approximately the same number of buses. The procedure

of power system decomposition can be achieved by graph partition method.

1.4.2 Graph Partition Method

This subsection provides background on graph partitioning. We begin by defining the graph partitioning problem . Two partitioning heuristics are then described in the following section.

Problem Formulation

The graph partitioning problem is that given a graph $G = (V, E)$, where each vertex in V corresponds to a bus and each edge in E corresponds to transmission lines, we are considering divide the vertices of a graph into sets of smaller components with specific properties by graph separators. There are two kinds of graph separators. One is edge separator, another is vertex separator.

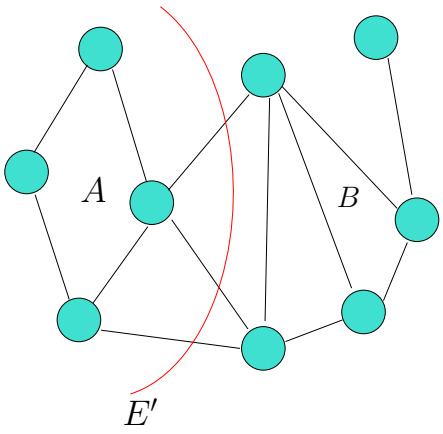

Figure 1.2: Example for Edge Separator

In edge separators. we partition the nodes into two sets, and we want to

minimize the number of edges $E' \subseteq E$, which is the number of edges between the two sets of nodes, here denoted by C. We also want to maintain some sort of balance between the sets, e.g. that the number of vertex in each set are as equal as possible. Figure 1.2 presents an example of edge separators which dived a graph into two sets.

In vertex separators, illustrated in Figure 1.3, we partition the nodes into three sets of nodes A, B and C, if the removal of C from the graph separates A and B into distinct connected components. The goal is to minimize the number of nodes in C, again maintaining an appropriate balance between A and B.

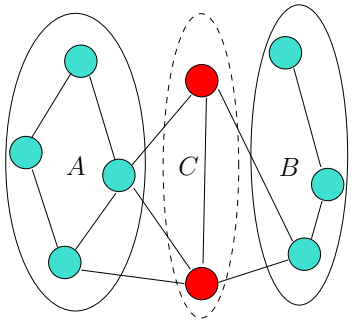

Figure 1.3: Example for Vertex Separator

Graph partitioning arises in many important scientific and engineering problem. Prominent examples include evenly distributing tasks among worker or computer processors, very-large-scale integrated (VLSI) circuit design, and public key cryptograph. Typically, graph partition problems fall under the category of $NP - hard$ problem. Solutions to these problems are generally derived using heuristics and approximation algorithm.

A numerous of heuristics have been developed to find approximate solu-

tions to the graph partitioning problem. The implementations of several of these heuristics such as spectral and multilevel k-way methods were used to decompose IEEE power system. These heuristics are described in following subsections.

Spectral Method

Hespanha [28] introduces a spectral partitioning method to solve k-way graph partitioning based on eigenvertor/eigenvalue decomposition. Given the undirected graph $G = (V, E)$ with vertex set V and edge set E. A k-way partition of V is a collection $\mathcal{P} = \{V_1, V_2, ..., V_k\}$ of k disjoint subsets of V, where $\cup_{i=1}^{k} V_i = V$. The cost associated with $\mathcal{P}$ is defined by

$$C(\mathcal{P}) := \sum_{i \neq j} \sum_{v \in V_i, \bar{v} \in V_j} c(v, \bar{v}) \tag{1.21}$$

where $c(v, \bar{v})$ is the cut cost, since the graph is undirected, $c(\bar{v}, v) = c(\bar{v}, v)$, $\forall v,\ \bar{v} \in V$.

There is a one-to-one correspondence between the set of k-partitions of $V := \{1, 2, ..., n\}$ and the set of k-partition matrices Π. The two correspondences can be defined by

1. Given a k-partition $\mathcal{P} = \{V_1, V_2, ..., V_k\}$ of V, a k-partition matrix $\Pi = [\pi_{vj}]$ can be defined by:

$$\pi_{vj} = \begin{cases} 1 & v \in V_j \\ 0 & v \in V_j \end{cases} \qquad \forall v \in V,\ j \in (1, 2, ..., k)$$

2. Given a k-partition matrix $\Pi = [\pi_{vj}]$, a k-partitions $\mathcal{P} = \{V_1, V_2, ..., V_k\}$ of V can be defined by

$$V_j \ := \{v \in V \ : \pi_{vj} = 1\}, \quad \forall j \in \{1, 2, ..., k\}$$

For these correspondences

$$\Pi'\Pi = diag[|V_1|, |V_2|, ..., |V_k|]$$

It turns out that it is easy to express the cost associated with a k-partition of V in terms of the corresponding k-partition matrix.

Lemma 1.4.1 *Let $\mathcal{P} = \{V_1, V_2, ..., V_k\}$ be a k-partition of $V \ := \{1, 2, ..., n\}$ and $\Pi = [\pi_{vj}]$ the corresponding k-partition matrix. Then*

$$C(\mathcal{P}) = 1_n^{'} A 1_n - trace(\Pi^{'} A \Pi),$$

where A is an $n \times n$ matrix whose $v\bar{v}$th entry is equal to $c(v, \bar{v})$ and $1_n$is a n-vector all entries equal to one.

The above lemma allows us to reformulate the cost function as follows:

$$\begin{aligned} maximize \quad & trace(\Pi^{'} A \Pi) \\ subject\ to \quad & \pi \in \{0, 1\}, \ \forall v, j \\ & \Pi \ orthogonal \\ & 1_n^{'} A 1_k = n \\ & 1_n \Pi e_i \leq l, \end{aligned} \tag{1.22}$$

where e_i denotes the ith vector in the canonical basis of R^k, l is the maximum vertices in subgraph.

In this method, the nonnegative symmetric matrix A need to be normalized to obtain the double stochastic matrix A', where $A' = \{a'_{ij}\}$satisfies $\sum_i a'_{ij} = \sum_j a'_{ij} = 1$ for each j and i.

For a doubly stochastic matrix A', all its eigenvalues are real and smaller than or equal to one, with one of them exactly equal to one. In this case the optimization is always smaller than or equal to n. This is because, for an arbitrary k-partition matrix Π that satisfies trace $\Pi'\Pi = n$, we must have

$$trace(\Pi^{'} A'\Pi) = \sum_{i=1}^{k} \pi_i^{'} A^{'} \pi_i \leq \sum_{i=1}^{k} \pi_i^{'} \pi = trace\Pi'\Pi = n \tag{1.23}$$

where π_i denotes $\Pi' s$ ith column. Here we used the fact that the largest eigenvalue of A' is no larger than one and therefor $\pi_i^{'} A'\pi_i \leq \pi_i^{'}\pi_i$.

Let $\lambda_1 = 1 \geq \lambda_2 \geq \cdots \geq \lambda_k$ be the larest eigenvalues of A' and $u_1, u_2, \ldots, u_k$ the corresponding orthonormal eigenvectors. It is convenient to define

$$U := [u_1\ u_2\ \ldots\ u_k],$$

$$D := diag[\lambda_1, \lambda_2, \ldots, \lambda_k],$$

for which $U'U = I_{k\times k}$ and $AU = DU$.

Suppose that we set $\Pi := UZ$, for some matrix $Z \in \mathbb{R}^{k\times k}$. In this case

$$\Pi'\Pi = Z^{'}U^{'}UZ = Z^{'}Z$$

and

$$trace(\Pi^{'}A^{'}\Pi) = traceZ^{'}U^{'}A'UZ = traceZ^{'}DZ$$

From this we conclude that the upper bound in (1.23) can be achieved exactly if the following conditions hold:

1. $D = I$

2. UZ is a k-partition matrix

3. $Z'Z \leq lI_{k\times k}$

According to [28], the algorithm approximately solves the graph partition when 1 holds approximately. It consists of setting

$$\Pi \ := argmin||\bar{\Pi} - UZ||_F$$

where the minimization is carried out over the k-partition matrices Pi and Z is a $k \times k$ matrix obtained by an heuristic algorithm aimed at making the conditions 2, 3 approximately hold.

Based on conclusion in [28], the computation of Z can be viewed as a clustering algorithm whose goal is to determine orthogonal vectors $z_1, z_2, \ldots, z_k$ around which the rows of U are clustered. This can be done by using clustering algorithm such as k-means or Expectation Maximization (EM) algorithm.

Multilevel partitioning scheme Method

Graph Partitioning problem is NP hard problem. The number of candidate partitioning grows exponentially as the number of vertices increase. Thus we do not expect the solve the graph partitioningproblem in polynomial time. Instead many approximate solutions with acceptable execution time are proposed.

Multilevel partitioning scheme is one of the most efficient algorithms for graph partitioning problem. Multilevel partitioning scheme reduces the original graph to a set of smaller graphs by collapsing vertices and edges. The smallest graphs is then partitioned using a heuristic method, and this partition is propagated back through the hierarchy of graphs with local refinement. The idea of multilevel partitioning is from the work of Barnard and Simon [4], in which they use a multilevel approach to calculate an eigenvector needed for a spectral partitioning algorithm. Hendrickson and Leland [27] generalized the concept of transferring eigenvectors between levels by instead transsferring partitions between levels. Karypis and Kumar [35] improved the efficiency of the algorithm, in which they compute a k-way partitioning of a graph $G = (V, E)$ in $O(|E|)$ time.

Here we introduce the multilevel method proposed by Karpis and Kumar [35]. As mentioned above, multilevel partitioning scheme has three stages, coarsening, inital partitioning and uncoarsen. First, a series of successively smaller graphs is derived from the input graph. Second, a partition of the coarsest graph is computed. and third, the graph is then de-coarsened in stages, and refined along the way.

Coarsening graph is important because that the number of possible partitions grows exponentially with the number of vertices in the graph, coarsening graph can significantly reduce the dimension of the problem. A good partition of coasrse graph is much easier to find than a good partition of the original graph. During the coarsening stage, a sequence of smaller graphs $G_i = (V_i, E_i)$, is constructed from the original graph $G = (V, E)$. Two vertices joined by an edge are merged to a single vertex, and repeating on the resulting graph until it is sufficiently small. Each vertex in a coarsened version of the original graph is assigned a weight equal to the sum of the weights of the original vertices it contains, and the adjacency structure of the original graph is perserved by making each coarse vertex adjacent to all the neighbors of its constituent original vertices. Edge weights are left unchanged unless both merged vertices are adjacent to the same neighbor, in this case the weight of new edge is equal to the sum of the weights of the two edges it replaces. This vertex and edge collapsing idea can be formally defined in terms of matchings. A matching in a graph $G = (V, E)$ is a subset of E whose endpoints are all distinct. Since the goal of collapsing vertices is to decrease the size of graph , the matching should be maximal. A matching is maximal if adding any edges would cause the set to no longer be a matching. Random matching, Heavy Edge Matching are two wildly used matching algorithm.

Once the original graph has been sufficiently coarsened, a multilevel k-way partitioning algorithm is used to compute a k-way partitioning of the coarse graph such that each partition contains roughly same vertex weight. Various algorithms can be used for this partition, however, these partitioning

algorithm must be able to handle edge and vertex weights, even if the original graph is unweighted. The implementation in METIS uses multilevel bisection algorithm [35]

In the uncoarsen stage, the coarse, partitioned graph is then projected back to the original graph. Each vertex in a coarse graph is simply the union of one or two vertices from a larger graph by maximum matching, we simply reverse this process to obtain the uncoarsened. Since uncoarsened graph has more degrees of freedom than the smaller coarse graph, it is possible to further improve the partitioning and thus decrease the edge cut. Hence, a local refinement algorithm is implemented to improve the partition quality.

A class of local refinement algorithms based on Kernighan-Lin algorithm can produce a good result. Kernighan-Lin algorithm first proposed in [38], is a $O(n^2 log(n))$ heuristic algorithm for solving the graph partitioning problem. Several improvements to the original KL algorithm have been developed over the years. Linear time implementation of KL algorithm is developed by Fiduccia and Mattheyses [20], Hendrickson and Leland [27] generalize the KL refinement algorithm for the case of k-way refinement. The use of KL algorithm is appropriate here since it is essentially a local, greedy optimization heuristic whose utility depends on the quality of the initial partition it is given. Given the initial partition, KL algorithm determines whether to move a vertex into a different set based on the concept of the "gain". The gain is simply the net reduction in the weight of cut edges that would result from switching one vertex to a different set. In particular, let w_{vw} be the weight of edge between vertices v and w, and c_{ab} is the symmetric inter-set cost metric for an edge

between sets a and b. Then the gain $g_b(v)$ associated with moving a vertex $v \in V$ from set a to set b can be defined as

$$g_b(v) = \sum_{(v,w)\in E} \begin{cases} w_{vw}c_{ab} & , if\ vertex\ w\ is\ in\ set\ b, \\ -w_{vw}c_{ab} & , if\ vertex\ w\ is\ in\ set\ a, \\ w_{vw}(c_{am} - c_{bm}) & , if\ vertex\ w\ is\ in\ set\ m,\ m\ is\ not\ neither\ set\ a\ or\ set\ b \end{cases}$$

If the gain associated with moving a vertex is positive, then making that move will reduce the total cost of the edges cut in partition. Details on how generalized KL algorithm selects moves can be found in [27], a simplified version of k-way refinement algorithms is presented in [35]. Test results in [35] shows that the multilevel k-way method is faster than the other method such as multilevel spectral bisection and multilevel recursive bisection. The quality of the partitioning produced by the multilevel k-way partitioning algorithm is comparable or better than other algorithms for a wide range of graphs.

1.5 Proposed Method

This section explains how the real-time power flow can be calculated by using the sensor measurement. The concept of the proposed the method is described in subsection 1.5.1. The details of the sensor placement methods are presented in subsection 1.5.2.

1.5.1 Concept of Proposed Method

We propose an approach for power flow calculation based on system decomposition and PMU placement. The proper placement of PMUs will decrease the dimensions of the sub-problems and improve the efficiency of computational procedures. Our method contains following steps:

A. Partition Power System and PMU Placement

A power system is decomposed into k non-overlapping subsystems of approximately the same size using a graph partition algorithm. PMUs will be installed at selected boundary buses to make sure the voltage phasors at all boundary buses are known. In other words, PMUs are placed to make all boundary buses observable [37]. Given a particular bus, installing a PMU obviously makes that bus observable. In addition, all adjacent buses to that bus become observable since their voltage phasors can be calculated from the branch current measured by the PMU and transmission line parameters. The procedure of power system decomposition and PMU placement will be described in subsection 1.5.2.

Let S denote the set of buses in a power system. Suppose this power system is decomposed into k subsystems. We denote by S_i the set of buses in subsystem $i = 1, \ldots, k$, where $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\cup_{i=1}^{k} S_i = S$. Thus, for subsystem i each bus belongs to one of the following types:

- Inner Bus: All its neighboring buses also belong to subsystem i.

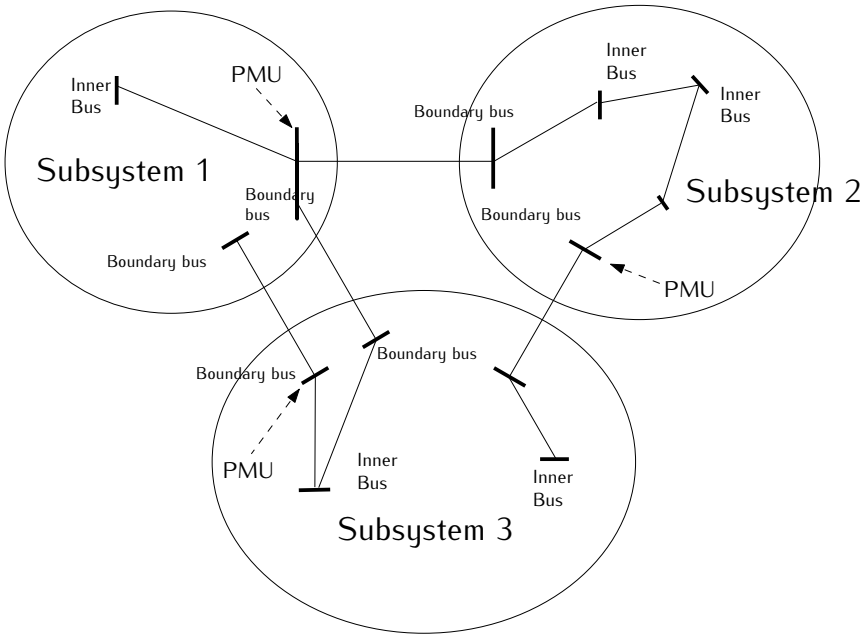

Figure 1.4: Power system decomposition and bus classification for k =3

- Boundary Bus: At least one of its neighboring buses belongs to a different subsystem.

This is illustrated in Figure 1.4.

B. Select Reference Buses

In order to perform power flow analysis for each subsystem, we need to choose a reference bus for each subsystem. Since the PMUs are placed so that all boundary buses are observable, any boundary bus can serve as a reference bus for the subsystem it belongs to.

C. Update Power at Boundary Buses

Before subsystem i can be solved independently of the others, the real and reactive power at each boundary bus needs to be updated to account for power flows from neighboring subsystems.

In particular, consider a boundary bus b that belongs to subsystem i. Let P_b and Q_b be the real and reactive power, respectively, at boundary bus b. For any bus c connected to b let $P_{b,c}$ and $Q_{b,c}$ be the real and reactive line power flow, respectively, on the line from bus b to bus c. Also, let $A(b)$ denote the set of boundary buses connected to bus b and not belonging to subsystem i. For any boundary bus b_1 connected to b the line power flows P_{b,b_1} and Q_{b,b_1} can be calculated from PMU measurements. The power flowing between bus b and its neighbors in subsystem i can be adjusted as follows:

$$\begin{aligned} P_b^{new} &= P_b - \sum_{c \in A(b)} P_{b,c}, \\ Q_b^{new} &= Q_b - \sum_{c \in A(b)} Q_{b,c}. \end{aligned}$$

D. Calculation for Subsystems and Aggregate the Results

Each subsystem run its own power flow calculation with respect to its reference bus, the voltage phasor of each reference bus are determined by PMU measurement. After each subsystem is solved in parallel, the solution to the whole system is then obtained by aggregating the solutions for each subsystem.

It is possible that some of the subsystems consist of multiple isolated connected components. In this case the number of connected component m is greater than k. In our computational experiments, this increase of components does not significantly increase the computational time.

1.5.2 PMU Placement Algorithm

Our real-time power flow method decomposes a large power system into several subsystems, which are then solved in parallel. The cost associated with such a decomposition depends on the number of PMUs installed at boundary buses. Having roughly the same number of buses per subsystem is desirable from the standpoint of balancing computational load. Thus our objective is to minimize the number of PMUs installed at boundary buses subject to the condition that each subsystem has approximately the same number of buses.

The basic structure of the PMU placement algorithm contains two steps. The first step is to divide the power system into k parts using a graph partitioning algorithm. This algorithm attempts to minimize the number of lines whose incident buses belong to different subsystems, while keeping the number of buses per subsystem approximately equal. The second step is to place PMUs based on power system decomposition. Given the partition k, the PMU locations are obtained by solving an integer linear programming problem.

Step 1: Power System Decomposition

Viewing the system as a graph $G = (V, E)$ where the vertex set V is the set of buses and the edge set E is the set of transmission lines, we have a k-way

partitioning problem. The k-way partitioning problem divides a graph into k sub-graphs with roughly the same number of vertices such that the edge cut, i.e. the number of edges connecting different sub-graphs, is minimized. This problem is NP-hard and several heuristics for its solution have been developed; see [21]. In this paper, we implement two heuristics for graph partitioning. One is a spectral partitioning algorithm, another one is a multilevel k-way partitioning algorithm.

Spectral partitioning: The spectral partitioning algorithm uses the eigenvectors of the adjacency matrix of a graph to find partitions. In this paper, we are using a spectral factorization based algorithm [28]. This spectral partitioning algorithm consists of four steps.

1. Form the Adjacency Matrix: For a power system with n buses, let $A = \{a_{i,j}\}$ be the corresponding $n \times n$ graph adjacency matrix, where the $a_{ij} = 1$, when bus i and bus j are connected by a transmission line, and equals zero otherwise.

2. Adjacency Matrix Normalization: Normalize the nonnegative symmetric matrix A to obtain a doubly stochastic matrix A', i.e. $A' = \{a'_{ij}\}$ satisfies $\sum_i a'_{ij} = \sum_j a'_{ij} = 1$ for each j and i.

3. Compute Eigenvectors: Compute the k largest eigenvectors u_i, $i = 1, 2, ..., k$, of matrix A'. It is convenient to define the $n \times k$ matrix $U := [u_1, u_2, ..., u_k]$.

4. Clustering: Obtain a partition of the network into k subsystems by clustering the n rows of U into k clusters using the k-means algorithm.

Multilevel k-way partitioning: We implemented the multilevel k-way partitioning algorithm from METIS [36]. The algorithm consists of three major steps:

1. Graph coarsening: A series of successively smaller graphs is derived from the input graph, where each successive graph is constructed from the previous graph by collapsing together a set of adjacent pairs of vertices.

2. Initial partitioning: A partition of the coarsest graph is computed using a relatively simple approach such as the multilevel bisection algorithm.

3. Uncoarsening and refinement: The partition of the smallest graph is projected to successively larger graphs by assigning vertices that were collapsed together to the same partition. After each projection step, the partition is refined using Kernighan-Lin method [38] that iteratively move vertices between sub-graphs as long as such moves improve the quality of the partition.

Step 2: PMU placement based on decomposition

Once a partition of the system is obtained, PMUs need to be installed to provide measurements for boundary buses. In most cases, the number of PMUs needed is not necessarily equal to the size of the edge cut. Based on the idea in [8], the optimal PMU placement problem can be formulated as an Integer Linear Programming(ILP) problem.

Given a partition of the system with n boundary buses, we can define an n by n constraint matrix M for the boundary buses. The entries of M are

defined as follows:

$$M_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 1, & \text{if boundary buses } i \text{ and } j \\ & \text{are connected} \\ 0, & \text{otherwise} \end{cases}$$

The PMU placement problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} c(i)x_i \\ \text{subject to} \quad & MX \geq \hat{1} \\ & x_i \in \{0,1\},\ i = 1, \ldots n, \end{aligned} \tag{1.24}$$

where n is the number of boundary buses, $c(i)$ is the cost of placing a PMU at boundary bus i, $\hat{1}$ is a vector of ones, and X is a binary decision vector whose entries are:

$$x_i = \begin{cases} 1, & \text{if a PMU is installed at boundary bus } i \\ 0, & \text{otherwise.} \end{cases}$$

For example, consider the power system shown in Figure 1.5 that has been decomposed into three subsystems. There are 4 transmission lines between 7 boundary buses. The optimal PMU placement problem can be solved as follows: First, initialize the constraint matrix M for the boundary buses. Building the M matrix for Figure 1.5 yields:

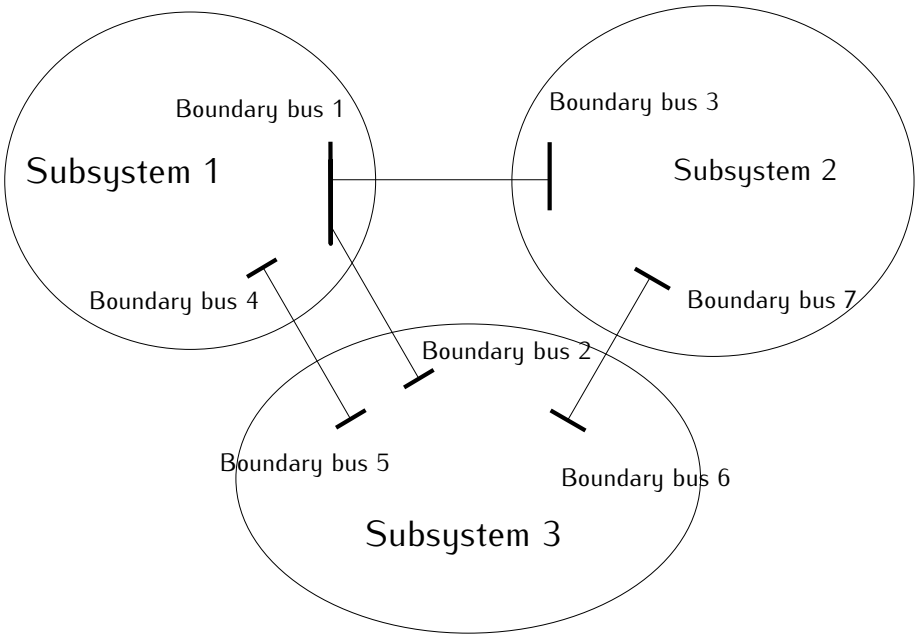

Figure 1.5: Example for optimal PMU placement

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Then the inequalities in (1.24) takes the following form:

$$\begin{cases} x_1 + x_2 + x_3 & \geq & 1 \\ x_1 + x_2 & \geq & 1 \\ x_1 + x_3 & \geq & 1 \\ x_4 + x_5 & \geq & 1 \\ x_4 + x_5 & \geq & 1 \\ x_6 + x_7 & \geq & 1 \\ x_6 + x_7 & \geq & 1 \end{cases}$$

The first constraint means a PMU must placed either on bus 1, bus 2 or bus 3 to make bus 1 observable. Similarly, the second constraint implies a PMU must installed at either bus 1 or 2 to make bus 2 observable. After solving the ILP problem for the boundary buses, the placement of PMUs for the entire system can be obtained.

1.6 Numerical Result

We use the IEEE 118 and 300 bus systems from [2] to illustrate the performance of our parallel algorithms. The 118 bus system was decomposed into 2, 4, and 8 subsystems, while the 300 bus system was decomposed into 2, 4, 8, and 16 subsystems. The power system was partitioned by both the spectral algorithm [28] and the multilevel k-way method [36] described subsection 1.5.2. The PMU measurements at the boundary buses were emulated using solutions obtained by traditional serial power flow methods. After decomposition, each of the resulting sub-networks was solved using Newton's method [58]. The convergence criterion was set to 10^{-5}, and the maximum iteration number for

Newton's method was set to 10. The system information is shown in Table 1.1. The partition results and computation times for different numbers of sub-networks are summarized in Tables 1.2 and 1.3.

Table 1.1: Network Information for two power system

System	118 bus	300 bus
number of nodes	118	300
number of branches	186	411
number of generator buses	53	68

Method	Partition number	Max subsystem size	Edge-Cut size [a]	PMU number	Calculation time (sec.)
Spectral Method	1	118	0	0	0.183
	2	79	5	3	0.090
	4	38	15	10	0.029
	8	22	26	13	0.016
Multilevel k-way	1	118	0	0	0.183
	2	78	5	3	0.089
	4	33	14	10	0.026
	8	19	29	14	0.014

[a] Edge-cut size: the number of branches connecting different subsystems

Table 1.2: Test Results on IEEE 118 bus system

The results obtained using our methods were compared with the corresponding results obtained using serial Newton's method. The maximum deviation of node voltage in our method compared to Newton's method was less than 10^{-4} p.u., which illustrates that our method is accurate and feasible.

Method	Partition number	Max subsystem size	Edge-Cut size [a]	PMU number	Calculation time (sec.)
Spectral Method	1	300	0	0	1.440
	2	184	6	5	0.570
	4	98	11	7	0.163
	8	56	19	13	0.075
	16	30	39	26	0.029
Multilevel k-way	1	300	0	0	1.440
	2	173	6	6	0.530
	4	97	11	10	0.162
	8	51	24	20	0.058
	16	25	49	32	0.025

[a] Edge-cut size: the number of branches connecting different subsubsystems

Table 1.3: Test Results on IEEE 300 bus system

1.6.1 Speedups compared to serial method

In Tables 1.2 and 1.3, the computation times T_s obtained by Newton's (serial) method are listed in the row where the partition number is 1. The rate of speedup can be obtained by the formula $S = T_s/T_p$, where T_p is the computation time of parallel method. Figures 1.6 and 1.7 show these speedups. Comparing the the details of these timing studies, we present the following conclusions:

1. High speedups and parallel efficiency are achieved in both the 118 and 300 bus systems by spectral and multilevel k-way method. Using spectral method as example, when the system is divided into two subsystems, a speedup by about a factor of 2 was obtained for both systems. Installing more PMUs usually leads to faster power flow calculations. In the 300-bus system, the traditional serial method takes 1.440 seconds, while with 5 PMUs the cal-

culation time is reduced to 0.570 seconds; here the speedup rate is 2.53. The speedup rate increases to 49.66 by using 26 PMUs. These speedups suggest that the parallel algorithm proposed in this paper is efficient.

2. The speedup associated with the parallel method increased as the size of the power system increased. For the case of splitting system into two subsystems , by using spectral method, the calculation time for the 118 bus system was sped up by a factor of 2.03, while for the 300 bus system it was sped up by a factor of 2.53. For the case of splitting system into four subsystems, the calculation time for the 118 bus system was 6.31 times faster, and for the 300 bus system it was 8.83 times faster. This suggests that the algorithms presented in this paper may perform well on larger power systems.

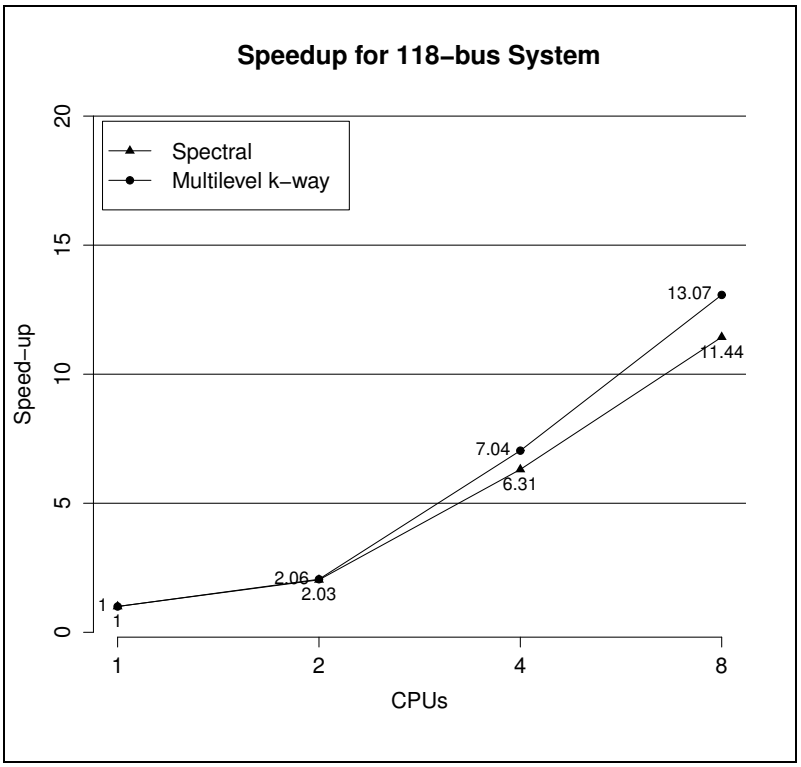

Figure 1.6: Speedup for 118 bus system

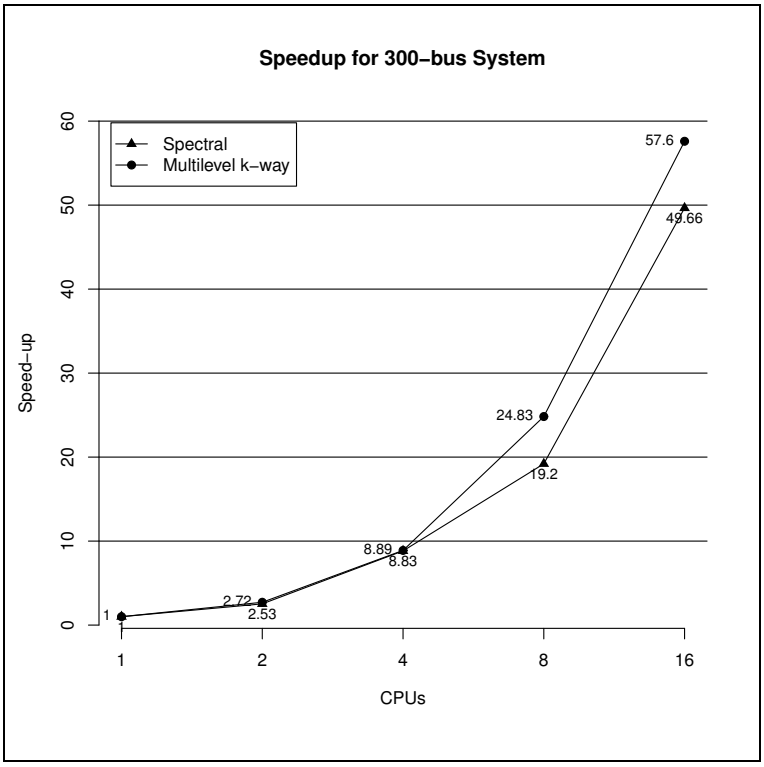

Figure 1.7: Speedup for 300 bus system

1.6.2 Performance of Spectral Partitioning vs. Multilevel k-way

Different partition methods will result in different PMU placements. In particular, the practicality of the partitioning algorithm becomes especially important when real systems are considered. We compared the spectral partitioning algorithm with the k-way partitioning method. As Figures 1.6 and 1.7 show, the multilevel k-way method outperforms the spectral partitioning algorithm in terms of calculation time. However, the multilevel k-way method usually needs more PMUs. Table 1.4 lists the partition results obtained via the

spectral method and the k-way partitioning method from the 300 bus system.

Table 1.4 shows that while the spectral partitioning method results in much fewer edge-cuts than the k-way partitioning algorithm, the allocation of buses to subsystems is more unbalanced. This imbalance may have contributed to the smaller speedups obtained when the spectral method was used instead of multilevel k-way; see Figures 1.6 and 1.7. On the other hand, less cut edges leads to fewer PMUs being placed, and the resulting cost reduction on PMU implementation and maintenance could be significant in the long run. This suggests that the spectral method may be more appropriate for our decomposition scheme.

Partition Number	Spectral algorithm			Multilevel k-way		
	Max Subsystem Size	Edge-cut Size	PMU Number	Max Subsystem Size	Edge-cut Size	PMU Number
2	184	6	5	173	6	6
4	98	11	7	97	11	10
8	56	19	13	51	24	20
16	30	39	26	25	49	32

Table 1.4: Comparison of the Spectral Method and Multilevel k-way in IEEE 300 bus system

1.7 Conclusion

In this chapter, we have provided a new approach to power flow analysis. Our problem formulation explicitly takes into account the placement of PMUs. A graph partition approach was proposed to partition the power system and determine the placement of PMUs, after which the power flow problem can be

solved in parallel. The effectiveness of the approach was illustrated on a 118 and 300 bus system using two different graph partitioning methods. In each of these cases, significant speedups compared to the serial Newton's method were obtained.

Chapter 2

Short-term Load Forecasting

2.1 Introduction

Load forecasting is an important tool for power system operation. Not only are accurate forecasts needed for the short-term operations and mid-term scheduling, but also utility operators need to have insight into the type of customers they have to supply as support for long-term planning. Many operational and financial decisions are based on load forecasting, such as reliability analysis, voltage control, unit commitment, security assessment and purchasing electric power. In addition, accurate estimated load is key input for electricity price forecasts. In the recent years, as the power system has become privatized and deregulated, the problem of accurate electric load forecasting has received more attention from utility companies.

There is no single forecast that can satisfy all of the needs of utilities. A common practice is to use different forecasts for different time horizon. Therefore, load forecasting can be broadly divided into four categories [23]: very short-term load forecasting (VSTLF), short-term load forecasting

(STLF), medium-term load forecasting (MTLF) and long-term load forecasting (LTLF). **VSTLF** predicts the future load up to few minutes ahead in a moving window manner. It is critical for automatic generation control (ACG) and resource dispatch, and it also ensures revenue adequacy for the independent system operator multi-settlement markets [24]. Due to the short time of the prediction horizon, Methods for VSTLF are usually based on the recently observed load pattern. **STLF** is usually from one hour to one week. Short-term load forecasting can improve purchasing decisions, generation decisions, and load switching. Since weather prediction is relatively accurate in the short time span (less or equal than one week), it plays a key role in STLF. **MTLF** are usually from a month up to three years. Temperature cannot be predicted accurately for the coming 3 years. Therefore, simulated scenarios of temperature based on the local temperature history can be used in the model. While the economics factor as well as population growth rate should be include in mid-term load forecasting. **LTLF** are over five years. The economic and infrastructure development factor is the major factor that drive the load. Therefore, economic and infrastructure development information is required in LTFL. On the other hand,weather information is hard to predict in the long run, simulated scenarios can be used.

The load forecasting for different time horizons are important for decisions such as trading, planning and operation. For example, energy purchasing is one of the most important decision for a utility. Utility needs to decide whether to purchases its own energy supplies from the market place or outsource this function to other parties. Load forecasting in different time horizons plays

an essential role in energy purchasing [30]. Accurate LTLF is the key to perform bi-lateral purchases and asset commitment in the long term, e.g. 10 years ahead. Meanwhile, MTLF and STLF are also required to adjust energy purchase in the Monthly or Daily day-ahead market. Transmission and distribution planning is another important decision for utility. The utilities need to properly maintain the upgrade the system to satisfy the growth of demand in the service territory and improve the reliability. Such planning involves determining the correct sizes, locations, interconnections, and timing of adding equipments or plants. An overestimate of demand can lead to an authorization of a plant or equipments that may not needed for several years. Furthermore an underestimate may be even worse since the utility needs 8-10 years to license and build a plant [22]. Accurate MTLF and LTFL can support the decisions in power system planning. In the daily operation of a utility, STLF obtains the load patterns for different regions , it is the critical information for system operators to make switching and loading decisions, and schedule maintenance outages.

With the deregulation of the energy industries, more and more parties have joined the energy markets, load forecasting is even more important.In the deregulated economy, electricity is traded as a commodity. Electric consumers will be able to choose their electric energy provider from a variety of power retailers [70], the consumers tend to shift electricity consumption from the expensive hours to other times when possible. Price information would affect the load profiles in such a price-sensitive environment [39]. With supply and demand fluctuating and energy prices increasing by a factor of ten or more

during peak situations [17], load forecasting is vitally important for electric utilities. Therefore, more accurate models should be developed to cope with all the uncertainties and changes in the deregulated electric power industry.

Over the past several decades, various methods have been proposed and applied for the load forecasting. In this dissertation, we are focusing on short-term load forecasting.The method for short-term load forecasting can be roughly categorized into two groups: one is statistical approaches, such as regression, time serious analysis and statistical learning. Another is artificial intelligence approaches. such as neural networks, fuzzy logic and support vector machine (SVM). Combination of these two approaches have also been studied and applied to STLF problems.

The organization of this chapter is as follows. Section 1.2 briefly describes the important factor for short-term load forecasting. Section 1.3 presents a literature review of STLF. Section 1.4 gives an overview of Artificial Neural Network (ANN) mehtod. Section 1.5 illustrates the implementation of the ANN method. Section 1.6 explains the Statistical Learning (SL) method and Modified Statistical Learning (MSL) method. The results of the proposed methods are given in Section 1.7.

2.2 Important Factor for STLF

There are many crucial factors that affect the energy consumption. They can be categorized as weather factors, calendar factors and economic factors.

Weather factors refer to the present condition of the meteorological elements such as temperature, humidity, wind, cloud cover, rainfall. etc. Other

variables such as Temperature Humidity Index (THI) , Wind Chill Indx(DWI) are also broadly used by researchers and utilities [50, 13]. THI is combination of temperature and humidity which measures the degree of discomfort experienced in warm weather , WCI is the measurement in winter for cold discomfort.

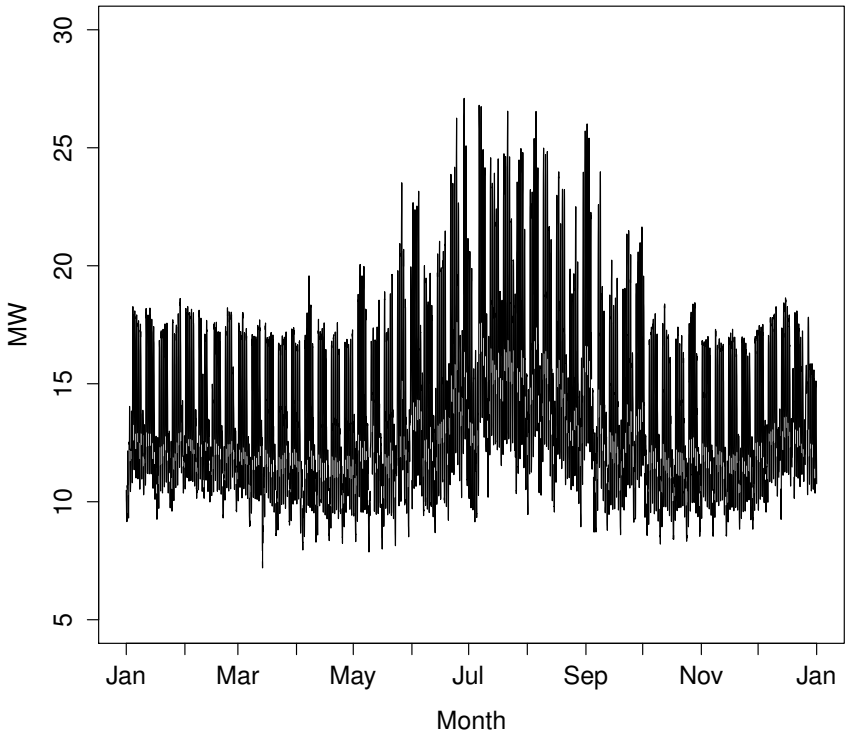

Figure 2.1: Example of substation hourly load from Jan 1st to Dec 31th in Long Island

Temperature is the most widely used factor for STLF. There are many type of temperature, such as dry bulb temperature and wet bulb temperature, can be used in load forecasting. Dry bulb temperature is the most widely used

one. It is well known that the load is strongly affected by the temperature, especially for the residential area. Load usually rises as temperature decrease in winter due to heating appliances; during the summer, load rises even more as temperature increase by the use of air conditioners. Figure 2.1 shows the hourly substation load from Jan 1st to Dec 31th of a substation in Long Island. From 2.1 one can see load during the summer is much higher than other seasons.

There are various ways to use weather information for modeling, we use temperature as example, we can select different set of temperature as input: previous hour temperature, current hour temperature, several consecutive hourly temperatures, maximum or minimum or average of the temperature of a day, etc. Also, the difference of temperature may have different impact for different season or territory. For example, in [60], it is reported that the change of temperature during spring , fall, and winter seasons is small, while the load changes dramatically in the summer due to the use of air conditioners.

Calendar factors are another important factors for STLF. Calendar information can be differentiated by time span. In the hourly resolution, energy consumption is different hour to hour within a day. Fig 2.2 shows the normalized load shape for each day of week. As we seen from Fig 2.2, daily load shape is similar to a sine wave. Energy consumption drops to lowest level at 5:00 am when most people are sleeping, it keep increase afterward when people wake up and go to work. Daily load usually reaches to the daily peak around 6:00 pm when people return back to their home.

The energy consumption in different days of week is also different. Fig

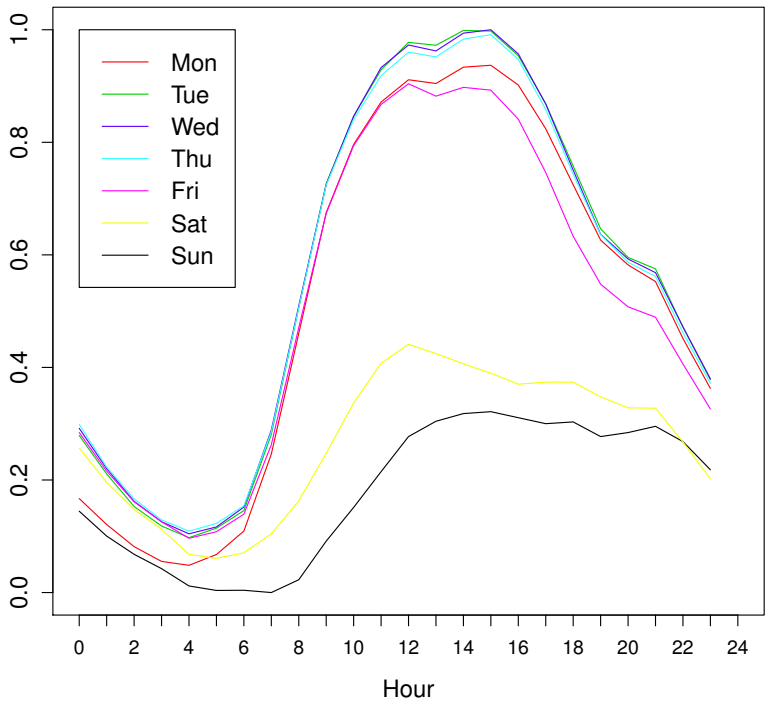

Figure 2.2: Normalized load shape for the year 2010

2.2 and 2.3 show the normalized weekly power consumption for each day of week. We can see a significant difference between weekday and weekend. This different may due to many different reasons, such as factory closed during the weekend, or people get up late during the weekend so that morning peak . Therefor, the day of week is usually classify into two groups: weekdays which are Monday to Friday and weekend which are Saturday and Sunday [13, 10].

Holidays and local festivities also affect the load demand. These events always lead to a lower demand for electric usage, because most of the offices and factories will closed that day. The influences of these events are highly depend on the customs of the area. On some major holidays such as Christmas

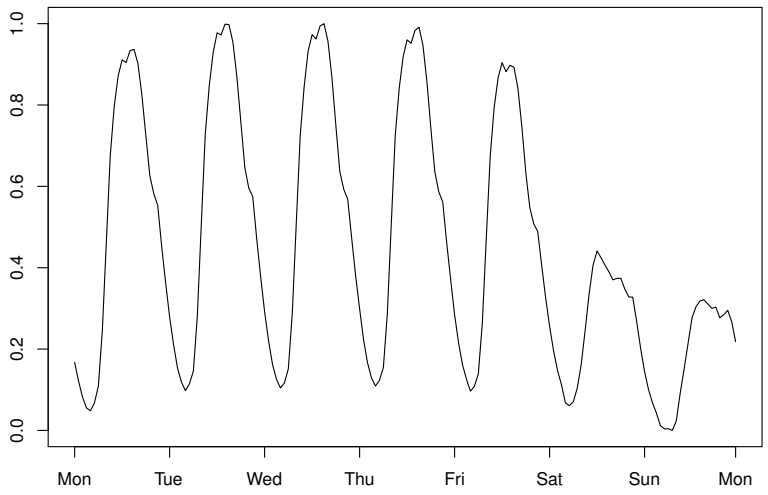

Figure 2.3: Normalized weekly load shape for the year 2010

or New Year, the load demand for electricity may be affected more compared with other holidays. These holidays are more difficult to forecast since holidays are much fewer than normal weekdays and weekend. A extra process usually be applied for accurate holiday load forecasting [69].

There are also important different in load consumption between different seasons. As show in Fig 2.1, The average load in summer is much higher than other seasons. Varies papers used four seasons (spring, summer, fall and winter) as grouping method.

Economic factors also play an important role to load forecasting. Based on the information from Department of Energy, during the year of 2009, which is the part of a recession in US, the energy consumption of US is much lower than the other year. This load decrease may mainly due to the bad econ-

omy, such that lots of business were closed and people were using power more conservatively.

For accurate STLF, we need to select useful factors into our model. Different information selection affects the model scheme and lead to different results. We will discuss the details of factor selection in section 1.5.2.

2.3 Literature Review

Thousands of methods and models have been developed for electric load forecasting in the past few decades. Most of the models for STLF require a mathematical model that represent load as function of different factors such as calendar, weather. Thus, the weather, calendar and load history are usually taken as the input. Before the modeling process, a data cleaning procedure is applied in order to clean the outliers and restore the missing value. After the relationship is found and represent as a mathematical model, with the weather forecasting and other information, load forecasting results can be obtained in an efficient way. The typical STLF process is shown in Fig. 2.4.

STLF methods can be roughly categorized into two approaches: one is statistical approaches, such as regression analysis and time series analysis. Another is artificial intelligence based approaches, such as Artificial Neural Networks, fuzzy logic and support vector machine. This section will give an overview of methodologies and models developed for load forecasting.

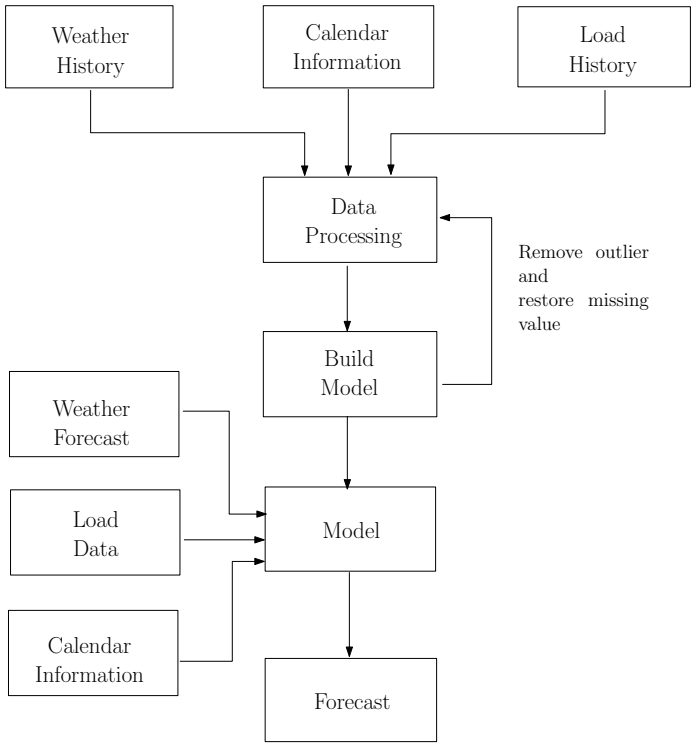

Figure 2.4: The typical process for load forecasting

2.3.1 Statistical Approaches

Regression: The general procedure for the regression approach is to assume basic function elements, and then using least square method to estimate the coefficients for the linear combination of the assumed basic function elements.

The regression models usually include two major areas: temperature modeling and holiday modeling. The model is based on a linear regression formulation. Holidays and other special effects are modeled using binary variables while normal days of week are handled by using appropriate independent variables for each day of the week and using weighted regression. Regression

model are more flexible comparing with other models, particularly ARIMA models. In regression model, missing data can be simply deleted or estimated by ad-hoc method [34]. The impact of outliers can be reduced by utilizing a weighted regression method. This methods make regression methods more practical in practice.

Papalexopoulos *et al.*[51] proposed a regression model for next day peak and hourly load forecast. Temperature variable is represented by heating and cooling degree function. It was an exciting idea, lots of later papers use similar piece-wise linear-quadratic functions of the temperature as input variables [40, 50]. Holiday effect is modeled by using binary variables(i.e., variables which take on values of either 0 or 1). This paper clearly introduced the regression approach for STLF, some later papers focus on different aspects of linear regression.

Haida and Muto [25] presented a regression based daily peak load forecasting method with a transformation technique. Seasonal and annual effect was considered in their model. A transformation function used to convert previous year's load into a set of new data set, the results will fit the shape of temperature-load relationship in this year. Ruzic *et al.*[57] proposed a regression based adaptive weather sensitive STLF algorithm which contains two steps. Total daily energy consumption was forecast in first step while hourly loads are predicted in the second step. Holidays were corrected by multiply by a correction coefficient.

One of the disadvantage of regression method is that it assumes the relationship between load and weather variables is linear which in fact may be

nonlinear. Conventional regression approach does no have the versatility to fit the nonlinear relationship, it will produce an average result instead. Therefore, an adaptable technique is required for performing accurate STLF.

Time Series Method : The idea of the time series approach is based on the understanding that a load pattern is composed of a time series signal and some known multiple seasonal effects. These periodicity give a rough prediction of the load at the given season, day of the week, and the time of the day [52]. The difference between the prediction and the actual load can be considered as a stochastic process. Time series method assumes the data has internal structure, which can be described by autocorrelation,trend or seasonal variation. By the analysis of such structure, we may get more accurate prediction.

To implement of time series model, the operation of difference is employed first such that the stationary time series can be formed. Then by use of the autocorrelation function (ACF) and partial autocorrelation function (PACF), the preliminary order identification of a model is confirmed. This can be followed by the application of maximum likelihood or weighted least-squares methods for parameter estimation [33]. This time series model is also called as Auto Regressive Moving Average (ARMA) model.

Double seasonal ARMA models are often as benchmarks in load forecasting studies. Taylor *et al.*[61] presented a model which can be expressed as $ARMA(p,q) \times (P_1,Q_1)_{s_1} \times (P_2,Q_2)_{s_2}$. The model was computed by maximum likelihood function based on the standard Gaussian assumption. The lag polynomials in their study was considered up to order three.

Periodic ARMA models usually be used on one week ahead electricity demand studies. That is because daily load shape is usually reasonably similar for the weekdays, but quite different for the weekends. This implies that the autocorrelation at a lag of one day is time-varying across the day of the week. ARMA model can not describe this attribute while AR model can capture this periodic features. Maybee and Uri [44] used periodic ARIMA models to forecast load one week ahead and the load duration curve one year ahead. Seasonal parameters are changed during seasons.

Autoregressive moving average with exogenous variables (ARMAX) is another widely used time series model in STLF. Since power load demand is sensitive to the weather variables, such as temperature, humidity, using history load only may not sufficient to forecast the load. Yang *et al.*[72] describe an implementation of ARMAX model to represent the relationship of load with temperature. An evolutionary programming approach was used to identify the ARMAX model parameters for one week ahead hourly load demand forecast. Huang *et al.*[32] proposed particle swarm optimization (PSO) approach to identify ARMAX model for one day to one week ahead hourly load forecasting.

The disadvantage of this time series approach include the inaccuracy of prediction and numerical instability. One of the reasons this method often gives inaccurate results is that it does not utilize weather information. There is a strong correlation between the behavior of power consumption and weather variables such as temperature, humidity, wind speed. This is especially true in residential areas. The time series approach mostly utilizes computationally cumbersome matrix-oriented adaptive algorithms which may be unstable.

2.3.2 Machine Learning Approach

Artificial Neural Network : Artificial neural network (ANN) traces previous load patterns and predicts a load pattern using recent load data. It has been widely studied since 1990(e.g. [52, 50]). The biggest advantage of ANN is that it able to perform non-linear modeling and adaptation while does not require assumption of any functional relationship between load and weather variables in advance.

An ANN can be defined as a highly connected array of elementary processors called neurons. A widely used model called the multi-layer perceptron (MLP) ANN.The MLP type ANN consists of one input layer, one or more hidden layers and one output layer. Each layer has several neurons and each neuron in a layer is connected to the neurons in the adjacent layer with different weights. With the exception of the input layer, each neuron receives signals from the neurons of the previous layer with different weight. The neuron then produces its output signal by passing the summed signal through a sigmoid function. By simply learning historical samples, a mapping between the inputs and the load is reconstructed and then can be adopted for the prediction theoretically. Hippert *et al.*[29] offered a high-level methodology to develop ANN models for STLF in their review paper

Park *et al.*[52] proposed a method to train multi-layer perceptron ANN by delta rule [55]. The weights parameters in ANN model is updated by backward propagation error. Once the neural network is trained, it produces very fast output for a given input data, only requires a few multiplications, additions, and calculations of sigmoid function.

The ANN model consistently outperforms the regression model in terms of both average errors and number of "large" errors. Chen [13] presents a similar day-based neural networks to forecast tomorrow's load. Similar day load was selected as the input load based on correlation analysis. Reis and Alves da Silva [54] describe an hybrid ANN method. A wavelet transform method performed for the load before applying ANN mehthod. The load was decomposed into high frequency signals and low frequency signals. Then ANN method could capture more information with these signals compared to orginal load.

Neural networks have been shown to have the ability not only to learn the time-series load curves but also to model an unspecified nonlinear relationship between load and weather variables.the ANNs are based on a soft computing technique and do not require to explicitly model the underlying physical system. Compared to the other methods such as time series and regression, the ANN allows more flexible relationships between temperature and load pattern. The disadvantage of ANN algorithms is that, since the neural network simply interpolates among the training data, it will give high error with the test data that is not close enough to any one of the training data.

Support Vector Machine: Support vector machine (SVM) is a supervised learning method that analyzed data and recognize patterns, used for data classification and regression. Support Vector Machine was first introduced by V.Vapnik [63] in 1995. SVMs were originally developed to solve pattern recognition problems, implementing the idea of structural Risk Minimization. The basic concept of SVM for classification problem is to generate input-output

classification functions from a set of labeled training data. A nonlinear kernel functions are often used to transform input data to a high-dimensional feature space in which the input data become more separable compare to the original input space [64].

Drucker *et al.*[14] proposed a version of SVM for regression which was also known as support vector regression (SVR). This generalization of the SVM algorithm to regression estimation is based on $\epsilon - insensitive$ loss function. In order to train a support vector machine one needs to solve a constrained quadratic optimization problem, so that the solution of SVM is always unique and globally optimal. Once the SVM algorithm was generalized to regression problems, the researchers started applying it to various problems of estimating real valued functions. SVMs have demonstrated highly competitive performance in numerous real-world applications, such as bioinformatics, text mining, face recognition, and image processing, which has established SVMs as one of the state-of- the-art tools for machine learning and data mining, along with other methods.

Chen et al.[5]are the pioneers for proposing a SVM method, their program was the winning entry of the competition organizad by thee EUNITE network. They briefly introduce support vector regression method for midterm load forecasting. Temperature information was not used in their work due to the lack of information. Instead, the previous load time series data was embedded into the model to predict future load. They conduct cross-validation to choose suitable parameters.The results indicate that SVMs compare favorably against the time series method.

Hong [68] presented a SVR model with immune algorithm (IA) to forecast the Taiwan regional electric loads, IA is applied to determine the parameter in SVR model. The test results indicate that their model results better performance than other methods such as regression and ANN model. make

There are some drawbacks for SVM: Solving a SVM implies solving a quadratic programming problem, SVMs scale suffer from a widely known scalability problem in terms of computational time. Furthermore, the correct choice of kernel parameters is crucial for obtaining good resultsand this often complicates the task [46].

2.4 Artificial Neural Network: Mathematical Formulation

In this section we will present a recently developed methodology for short-term load pocket forecasting, called Artificial Neural Network. ANN has been applied to STLF for two decades with varying degree of success.

2.4.1 Framework of Artificial Neural Network

An artificial network consists of a pool of simple processing units (Neurons) which communicate by sending signals to each other over a large number of weighted connections. The artificial neural network contains following :

Processing units (Neurons): Processing units in an ANN are also known as neurons. These neurons are interconnected by means of information channels called interconnections. Each neuron can have multiple inputs, while

there can be only one output. Each unit performs a relatively simple job: receive input from neighbors or external sources and use this to compute an output signal which is propagated to other units. Figure 2.5 shows this process. Inputs to a neuron could be from external stimuli or could be from output of the other neurons. The single output that comes from a neuron could be input to many other neurons in the network. It is also possible that one of the copies of the neuron's output could be input to itself as a feedback.

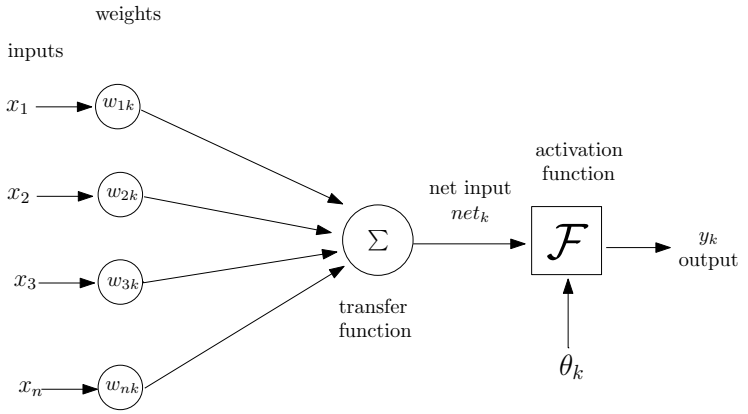

Figure 2.5: Diagram of an artificial neuron

Input weight: There are weights between neurons and its inputs. In most cases we assume that there is additive relationship between neurons and their input. Suppose that neural k has several input x_j, $j = 1, ..., n$. Let w_{jk} denotes the weight between x_j and neural k. θ_k is the bias in neural k. The total input to neuron k can be represented as:

$$s_k = \sum_{j=1}^{n} w_{jk} x_j + \theta_k \tag{2.1}$$

We call equation (2.1) as propagation rule, the units with a propagation

rule is called as sigma units.

Activation function: the activation function $\mathcal{F}$ is a nondecreasing function of the total input of the unit, the output of nerual k can be represented as:

$$y_k = \mathcal{F}_k(s_k) = \mathcal{F}_k(\sum_j w_{jk}x_j + \theta_k)$$

There are a number of common activation functions in use with neural networks. We list three of them:

1. Step Function: A step function is a function like that used by the original Perceptron. The output is a certain value, A_1, if the input sum is above a certain threshold and A_0 if the input sum is below a certain threshold. The values used by the Perceptron were $A_1 = 1$ and $A_0 = 0$.

2. Linear activation function: A linear combination is where the weighted sum input of the neuron plus a linearly dependent bias. Specifically:

$$y_k = \mathcal{F}(s_k) = a * s_k + b$$

Here a is a weight, b is the bias.

3. Sigmoid Function: A sigmoid function, also known as a logistic function, is given by the relationship

$$y_k = \mathcal{F}(s_k) = \frac{1}{1 + e^{-\beta s_k}}$$

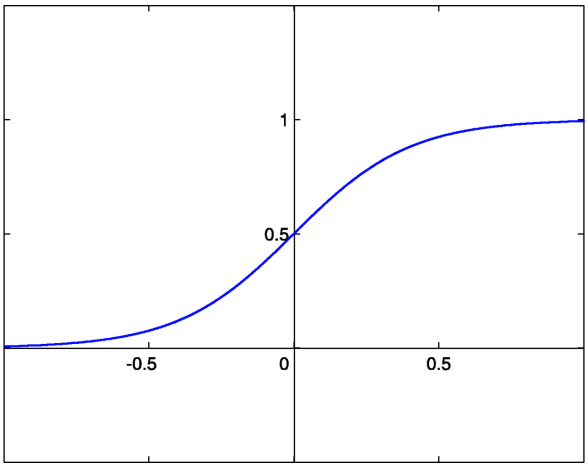

Figure 2.6: sigmoid activation function

Where β is a slop parameter. The sigmoid has the property of being similar to the step function, but with the addition of a region of uncertainty. Sigmoid functions in this respect are very similar to the input-output relationships of biological neurons, although not exactly the same.

Artificial neural networks topology: There are many types of ANN topology. Feedforward neural network is the most wildly used networks. A feedforward neural network is an artificial neural network where connections between the units do not form a directed cycle. It was the first and most simple type of artificial neural network [1]. In this network the information moves in only one direction: from the input nodes, through the hidden nodes and to the output nodes. There are no cycles or loops in the network .

2.4.2 Single layer perceptron network

Single-layer perceptron network is the simplest feedforward neural network, which consists of a single layer of output nodes; the inputs connect

directly to the outputs via a series of weights. In this way it can be considered the simplest kind of feed-forward network. Fig 2.7 shows the simplest case of this network which has only two inputs and a single output. The input of the neuron is the weighted sum of the inputs plus the bias term. Suppose that neural k has several input x_j, $j = 1, ..., n$. Let w_{jk} denotes the weight between x_j and neural k. θ_k is the bias in neural k. The output of the network is formed by the activation of the output neuron, which is a function of the input:

$$y = \mathcal{F}(\sum_{i=1}^{2} w_i x_i + \theta)$$

The sum of the products of the weights and the inputs is calculated in each node. The output of the network is formed by the activation $\mathcal{F}(s)$ of the output neuron.

$$\mathcal{F}(s) = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{otherwise} \end{cases}$$

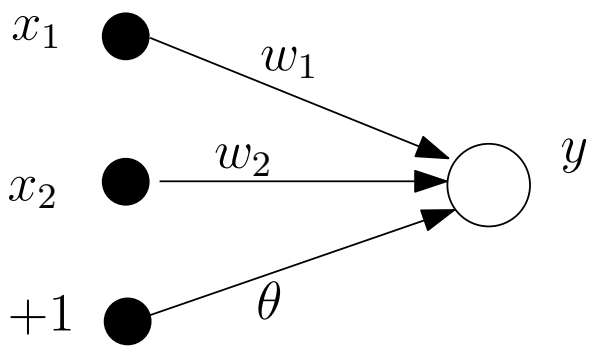

Figure 2.7: Single layer perceptron network with one output and two input.

The output of the network is either $+1$ or -1, depending on the input. This network can be used for the classification task. If the sum of input is positive, the sample will be assigned to class 1, it will belong to class 2 otherwise. The separation between the two classes in this case is a straight line, given by the equation:

$$w_1 x_1 + w_2 x_2 + \theta = 0$$

A geometrical representation of the linear threshold neural network is given in figure 2.8. We can see that the weights determine the slope of the line and the bias determines the 'off set'.

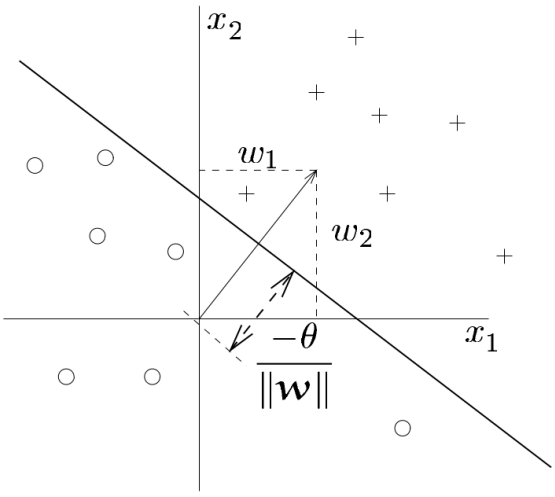

Figure 2.8: Single layer perceptron network with one output and two input

2.4.3 Multi Layer Perceptron Network

A single-layer perceptron network has severe restrictions: the class of tasks that can be accomplished is very limited, only linear functions can

be represented. Minsky and Papert [47] showed in 1969 that a multilayer feed-forward network can overcome many restrictions. Multilayer perceptron (MLP) model maps sets of input data onto a set of appropriate outputs. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. However, they did not present a solution to the problem of how to adjust the weights from input to hidden units.

An answer to this question was presented by Rumelhart *et al.*[56] in 1986, and similar solutions appeared to have been published earlier (Werbos [67] (1974), LeCun [41] (1985)). MLP utilizes a supervised learning technique called backpropagation for training the network. Back-propagation can also be considered as a generalization of the delta rule for non-linear activation functions and multilayer networks. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable. It has been shown [31, 26] that only one layer of hidden units suffices to approximate any function with finitely many discontinuities to arbitrary precision.

In a multilayer network (Fig. 2.9) containing hidden units, that is, units that are neither input nor output units. The error signal can be formed as before, but many neurals can give rise to the error, not just the ones at the output units. Since we usually do not know what the target outputs of the hidden units are, we cannot directly compute the error signal for hidden units.

The "generalized delta rule" is suggested by Rumelhart *et al.*[55] and gives a recipe for adjusting the weights on internal units based on the error at

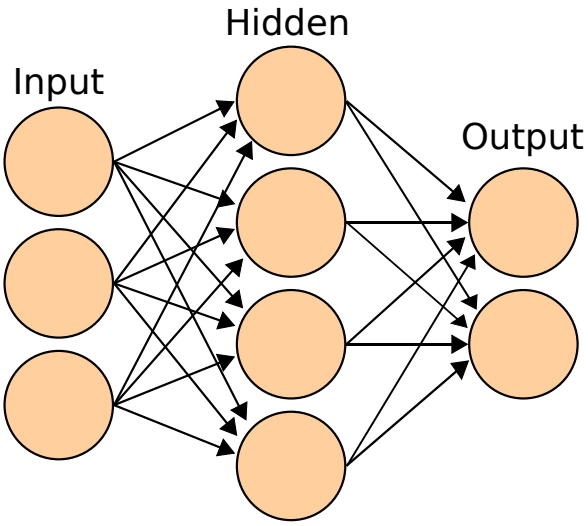

Figure 2.9: Diagram of a two-layer perceptron network

the output. To be more specific, let

$$E = \frac{1}{2}\sum_{k}(t_k - y_k)^2 \tag{2.2}$$

be the measure of the error with k outputs units, where t_k is the target output for kth component of the output and y_k is the kth component of output produced by the network. The network is specified as

$$y_k = \mathcal{F}(s_k) \tag{2.3}$$

in which

$$s_k = \sum_{j} w_{jk}x_j + \theta_k \tag{2.4}$$

Here $\mathcal{F}$ is a differentiable and nondecreasing activation function, and w_{jk} is a weight between input x_j and neuron k which need to be adjusted. The

function $\mathcal{F}$ is normally a sigmoid type function as shown in Fig .

To obtain a rule for adjusting weights Δw_{jk}, the gradient of E with respect to w_{jk} is used and it is represented as follows:

$$\frac{\partial E}{\partial w_{jk}} = -\frac{\partial E}{\partial s_k}\frac{\partial s_k}{\partial w_{jk}} \tag{2.5}$$

By equation (2.4) we can get the derivative of second factor in (2.5) as:

$$\frac{\partial s_k}{\partial w_{jk}} = x_j \tag{2.6}$$

Thus quation (2.5) can be represented as follows:

$$\frac{\partial E}{\partial w_{jk}} = \delta_k x_j \tag{2.7}$$

Here δ_k defines as

$$\delta_k = -\frac{\partial E}{\partial s_k} \tag{2.8}$$

If a neuron is an output neuron, δ_k is given by

$$\delta_k = (t_k - y_k)\mathcal{F}_k^{'}(s_k) \tag{2.9}$$

and for a neuron in an arbitrary hidden layer

$$\delta_k = \mathcal{F}^{'}(s_k)\sum_o \delta_o w_{ok} \tag{2.10}$$

Here o represents all the neurons in next layer.

The gradient descent algorithm adapts the weights according to the gradient error, the rule of adjusting weights can be drived using equation (2.7) and given as:

$$\Delta w_{jk}(n+1) = \gamma \delta_k x_j + \alpha \Delta w_{jk}(n)$$

where γ is the learning rate parameter and α is the momentum constant to determine the effect of past weight changes, and n is the iteration number.

Once the neural network is trained, it produces very fast output for a given input data. It only requires a few multiplications, additions and calculations of sigmoid function.

2.5 Application of ANN to One Day Ahead Electric Load Forecasting

In section 2.4, we explained that multilayer perceptron neural network to solves the optimization problem: Using the software Matlab [6], we applied ANN to short-term electrical load forecasting. The input data set include historical hourly load data and hourly weather data. Historical weather data were provided by NCDC (National Climatic Data Center).

2.5.1 Data cleaning

Before data is ready to be used as input to neural network, we first need to clean the data, by removing outliers, missing values or any irregularities. Neural network is sensitive to the irregular load data, during the training, bad

data would result in large forecast error. In our method, we use four rules to clean our data.

1. **Three sigma rule:** Most widely used method is called "three-sigma" rejection rule. The "three-sigma" rule consists in rejecting observations that are outlying beyond mean estimation plus three times of the estimated standard deviation. From the observation, we find the bad data usually last more than one day. we calculate the daily MAPE, if the MAPE of that day is greater than average daily MAPE plus three times of standard deviation, all the loads of that day will be treated as outlier.

2. **Remove duplicate data and restore missing value:** Electric load is different by hour. From observation, when the meter is failure or any abnormal things happens, the the meter reading would be the exactly same for a long time. Thus, if there are consecutive same loads for more than six hours, the load for that day will be considered as outlier. All the missing data will be replaced by the imputed load.

3. **Remove the holidays:** Holiday electricity load are different from normal day, and different region by region. Since we only have two years data, accurate holiday forecasting is difficult to achieve. Six holidays were removed from our data and replaced by imputed data.

2.5.2 Input Selection

After data cleaning, we need to decide what data information should incorporate into the ANN models. The key issue for the success implementation

of any load forecasting method is finding a suitable selection of input variables. For short-term load forecasting factors such as history load, calendar and weather factors should be included into the model.

One of the most important factors is calendar factor. Calendar factor include time of the year, the day of the week and the hour of the day. The weekday load shapes are typically different from the weekend profiles. Also the load on different weekdays can behave differently. To differential the weekday impact, weekday index [34] which using dummy variables to represent day of week is used in our work . Various weekday index has been tested. Based on the test result we choose a 3-bit binary number as our weekday index, i.e., 100 for Monday through Friday, 010 for Saturday and 001 for Sunday.

Weather factors is another important factor. The main variable to be included is the dry bulb temperature, since it has been well known that the demand rises on cold days and hot days. The next commonly used variable is dew point, it will affect human satisfaction when temperature is high. The other variables such as wind velocity, cloudiness etc can also have effects on load demand. We experimented model with temperature and humidity and models with other weather factors. The differences between the models are not significant. Thus, our model we only include temperature and humidity.

Beside calendar and weather factors we also include historical load information. To forecast the load of tomorrow(Load T), the common practice is to use the most recently available load, i.e., the load 24 hours ago(T-24), and the load of one week ago(T-168)[12, 3, 50]. We experimented with several models before we decide what load information to use. The best results were obtained

when using one hour ago (T-1), one day ago(T-24) and one week ago(T-168) data in training. The next day load forecasting is then conducted hour by hour. The next hour forecasting is based on the prediction of previous one. The model that did not have any lagged load variables gave the worst results.

Thus, our training data set includes calendar, weather and past load information. The last step in our data preparation is to linearly scale our data sets as proposed in [54]. Scaling the data before applying ANN is very important. The main advantage is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. The input weather, load and calendar information are normalized to value between 0 to 1.

2.5.3 Model Selection

Beside deciding what variables to include in the ANN model, there are some parameters need to be determined before training the ANN. These parameters include number of output neurons, hidden neurons and activation function.

The number of output neurons affects the structure of the ANN. Typically, the number of output neurons is set to either 24 or 1 for short-term load forecasting. For the case of 24 output neurons, the single-model multivariate forecasting method is used [29]. This method uses a multivariate algorithm to forecast all the loads at once, so that each profile is represented by a 24-dimensional vector. One drawback is that training the ANN with 24 outputs takes time, and also treating each day as a vector will results in very few number of input data. e.g. two month of data will yield only 60 data points,

which is too few for the large MLPs . In our method, we forecast next load by iterative forecasting method [54] which forecast one hourly load at a time and then aggregating the load consecutively, so that the forecasts for the later hours will be based the forecasts for the earlier ones.

Determining the number of neurons in the hidden layer may be more difficult than determining the size of the input or output layers. There is again little theoretical basis for the decision. If the neuron is too few, the model will not be flexible enough to model the data well; if they are too many, the model will over fit the data. Here we experimented on several models, the optimal number of neurons we selected is 8.

Even though there are only few activation functions that are commonly used, we need to decide which one we will use. We tried two activation functions,the hyperbolic tangent sigmoid function and log-sigmoid. The results from the latter were more adequate and we end up using the sigmoid function.

To avoid the over-fitting problem, a cross-validation technique [6] has been incorporated in our solution strategy in which the progress of the training phase is controlled by the validation error instead of training error. Validation set is a part of training samples which is removed from the training set and so becomes unseen for the neural network. Minimum of the validation set error is selected as the optimal point of the training phase, where it is expected that the generalization capability of the neural network be maximized. Since the validation error often does not follow a neat curve and there may be several local minimal, its behavior over a sufficiently large number of training iterations is evaluated.

After deciding what variables to include into our model, determining the number of output neurons, the number of hidden hidden neurons and activation function, we are able to train our ANN model. The results of our model will be presented in Section 2.7.

2.6 Statistical Learning Method for One Day Ahead Forecasting

In this section, we first introduce the Statistical Learning (SL) method for short-term and long-term load forecasting. A Modified Statistical Learning (MSL) Method then proposed which is more suitable for the short-term load forecasting of the system level data.

2.6.1 Statistical Learning Method

Feinberg and Genethliou [18] built a statistical learning model for load forecasting, this model can be used for both middle-term load forecasting and short term load forecasting. Statistical Learning model builds a multiplicative statistical model that takes into account time factors such as the day of the week and the hour of the day, as well as weather factors such as temperature and humidity. The developed multiplicative model has the following form:

$$y_t = C(d_t, h_t) \cdot f(w_t) + e(t) \tag{2.11}$$

where y_t is the original load at time t, d_t is the day of the week, h_t is the hour of the day, w_t is the weather data such as temperature and humidity, $C(d,h)$ is a function of calendar factor, $f(d,h)$ is a function of weather factor, and $e(t)$ is the random error. In particular, $w(t)$ is a vector that consists of the current and lagged weather variables. This reflects the fact that electric load depends not only on the current weather conditions but also on the weather during the previous hours and days.

To optimally estimate the parameters, The least square method is used to minimize the total squared residues, i.e.,

$$min \sum_t (y_t - C(d_t, h_t) \cdot f(w_t))^2 \tag{2.12}$$

Due to the excessive number of parameters and the mixture of discrete and continuous parameters in the model. Conventional method such as newton's method and gradient decent method is not efficient. Instead of using traditional method, a recursive iteration method has been applied [16, 19].

To estimate the weather factor $f(w)$, the regression model has been used

$$f(w) = \beta_0 + \sum \beta_j X_j$$

where X_j are explanatory variable which are nonlinear functions of current and past weather parameters and β_0, β_j are the regression coefficients. The parameters of the model can be calculated iteratively. $C(d,h)$ was set as one initially. Then we use the above regression model to estimate $f(w)$. The

$C(d, h)$ was calculated and so on. This process continues until equation (2.12) meet the stop criteria.

This statistical learning model was compared with other method such as time series and converntional regression analysis and gave more adequate results. It is relatively simple, robust and reliable. It converges quickly, mostly in less than 10 steps, and can be easily used for different levels of forecasting [16].

2.6.2 Modified Statistical Learning Method

Statistical Learning method developed a multiplicative statistical model that takes into account two factors. The first one is calender factor such as day of the week, hour of the day, the second one is weather factors such as temperature and humidity. Besides the time and weather factors, historical load is also a very important factor for short-term forecasting. It is a common sense that today's load will similar to tomorrow's load. From our experiment, the correlation between the load (Load T) and 24 hours ago load (Load T-24) is usually more than 0.7. Since the load series is strongly autocorrelated, previous load information can be a good indicator for the one day ahead load forecasting.

In this subsection, we present a modified statistical learning algorithm for one day ahead load forecasting by using historical load. This idea of using historical load is motivated by the forecasting methods in papers [12, 3, 50, 34] where historical load is used as one of the inputs. We modified the SL model structure by adding the historical load parameter in the function $f(w)$. In

mathematical terms the model is presented as :

$$y_t = C(d_t, h_t) \cdot f(w_t, L_{t'}) + e(t) \tag{2.13}$$

Here $L_{t'}$ is the lagged load.

There are many different ways to choose lagged load. Some researchers treat the load as 24 hour vectors, and use the one past day load or two past days as input. The other researchers use the Box and Jenkins methodology for fitting ARIMA models, and selected the lags by the analysis of the autocorrelation functions and partial autocorrelation functions. For the regression model, choose too many lagged load variables can cause the problem of multi-collinearity. We experimented different models by using different lag load variables. The best results were obtained by selecting the one day ago load(T-24) as the input.

After include the lagged load variables, the objective function for parameter estimation now becomes

$$min \sum_t (y_t - C(d_t, h_t) \cdot f(w_t, L_{t'}))^2 \tag{2.14}$$

We train the MSL model using the same iterative method as described in section 2.6.1, the converge rate of MSL method is approximately same as SL method.

2.7 Numerical Results

We evaluate the performance of the ANN and MSL models by calculating the Mean Absolute Percentage Error (MAPE) and its standard deviation. The formula for the MAPE is given by

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\frac{|L_{ACT(i)} - L_{FOR(i)}|}{L_{ACT(i)}}$$

where $L_{ACT(i)}$ and $L_{FOR(i)}$ are actual and forecast load of hour i, respectively; N is the forecast horizon. The ANN method is tested using a substation data in long island, the MSL method is tested on a system level data used in [54].

2.7.1 Test Results of ANN method for the Substation Data

We use the substation data in long island to test our ANN algorithm. For this substation we have less than two year data from Jan 2010 to Nov 2011. We try to predict the data from Feb to Nov in 2011. The training data contains two parts. The first part is a four-week windows before forecasted day. The second part is the equivalent periods of one year before (e.g. we try to predict load demand on Nov. 1st 2011, then the training data is the data in Oct of Year 2011 and Oct of Year 2010). The training data set has calendar information, weather information and historical load.

The ANN model was trained by applying a back propagation algorithm with cross validation. After training, one day ahead prediction are computed.

The load forecasters are retrained at the end of each day to incorporate the most recent load information. The concatenation of four week training windows, for a particular day, is shifted one day ahead, and the forecasts for the next 24 hour are evaluated. This test procedure is repeated from February to November in 2011.

Effect of Data Cleaning Procedure

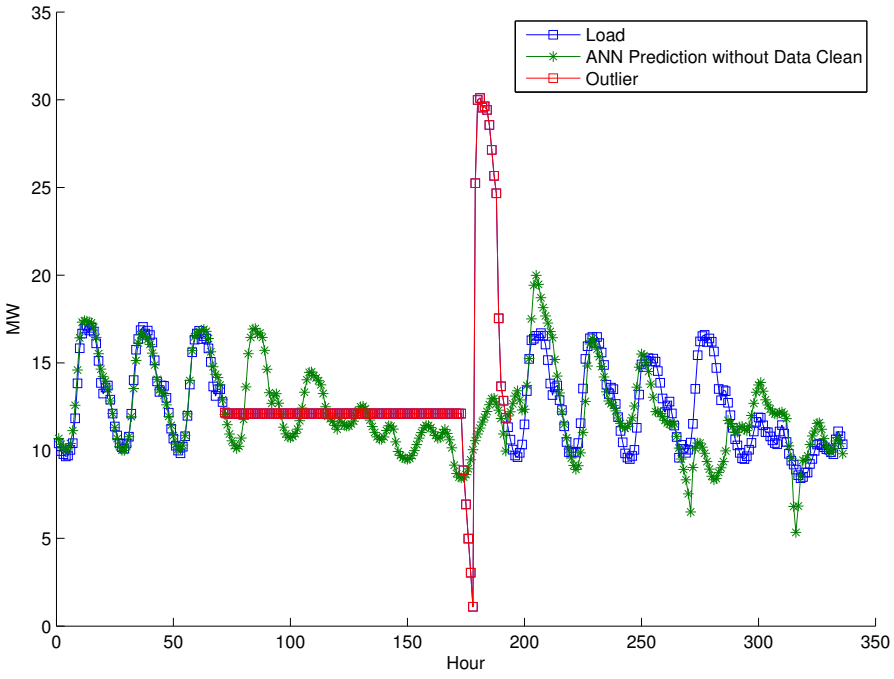

Figure 2.10: ANN forecasting without data cleaning

The data cleaning procedure described in subsection 2.5.1 was applied for the model training. Fig 2.10 and 2.11 presented two week ANN forecasting results with data cleaning procedure and without data clean procedure.

From the Figures 2.5.1 and we can see the outlier has a long-term effec-

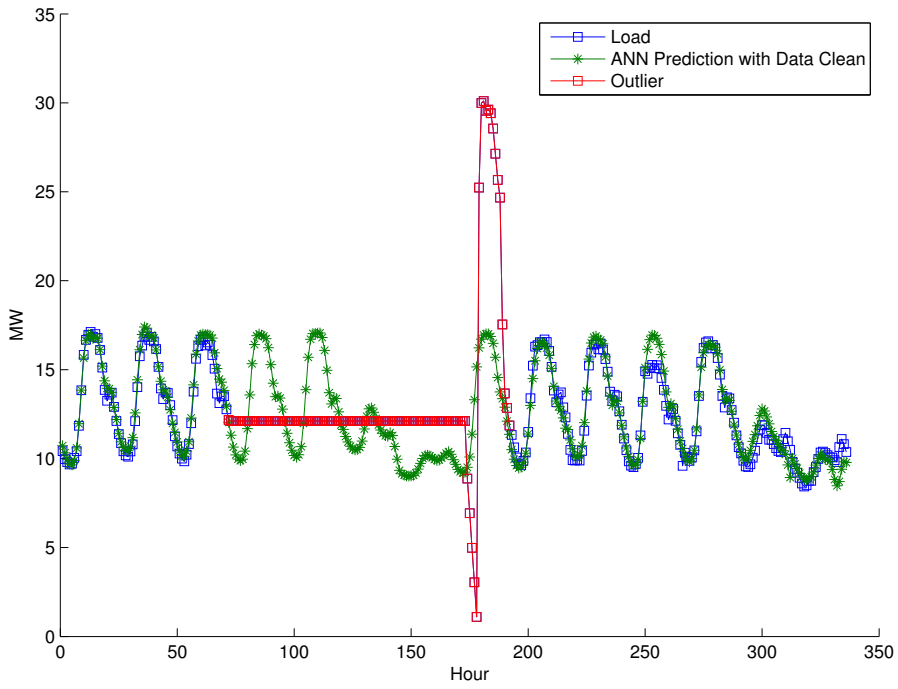

Figure 2.11: ANN forecasting with data cleaning

tiveness to load forecasting. Without data cleaning procedure, the outlier data will be reused in training session for the prediction model. Since prediction model is highly dependent on the quality of training data, the prediction model would become unstable without cleaning such outlier. Fig 2.11 shows that after applying data cleaning procedure, the forecasting errors have a significant decrease.

The performance of ANN method

The scatter plot of the actual load and forecasted load by ANN is shown in Fig 2.12. The almost linear relationship between actual load and forecasted load indicate that ANN method provides accurate predictions.

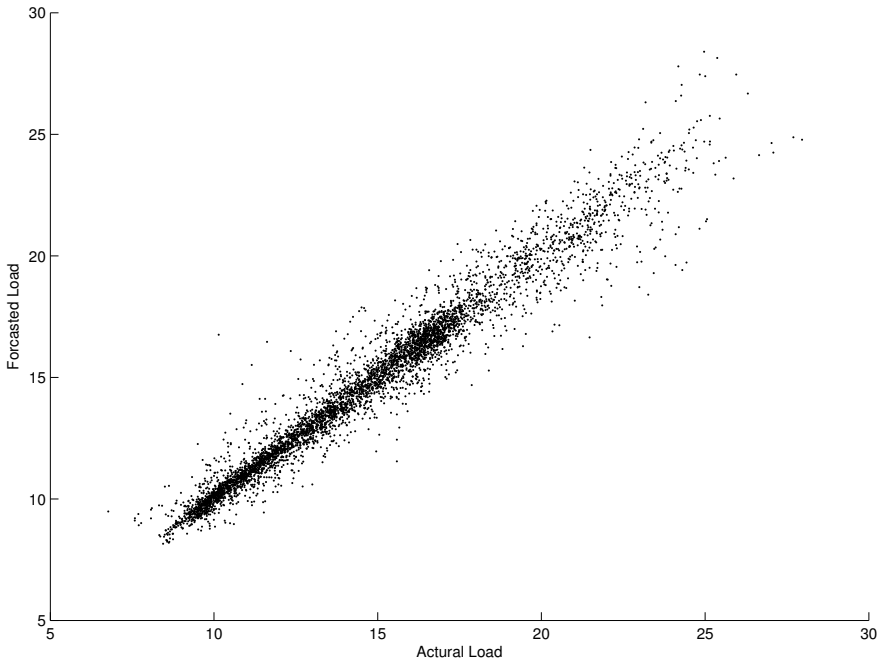

Figure 2.12: Scatter plot of actual load versus the forecasted load by ANN for a substation

The results of ANN method is shown in Table 2.1. The first column in Table 2.1 indicates the month we used to calculate the MAPE, "All" stands for average hourly MAPE from February to November. Second column shows the average hourly MAPE with weekend. The overall MAPE for ANN is 3.19. Since electricity usage in weekend is usually low, utility companies pay more attentions for load forecasting in workdays. The load forecasting without weekend is presented in third column of Table 2.1. As seen, by taking only workdays into account, ANN method has some improvement in over all MAPE. This may due to the more observations of workdays than weekend.

To give a graphical view about the STLF accuracy of the proposed ANN

	with weekend	without weekend
Feb	2.15	2.10
Mar	3.31	3.23
Apr	3.16	2.86
May	2.49	1.95
Jun	4.95	5.17
Jul	3.76	3.45
Aug	2.93	2.52
Sep	3.24	3.10
Oct	3.23	2.83
Nov	2.52	2.33
ALL	3.19	2.96

Table 2.1: Test results of ANN method for the substation data

methods, one week load forecasting by ANN method for one week in July and February are shown in Figure 2.14 and 2.13. Actual load and forecasted load by ANN are represented by blue and green line respectively. As seen, ANN model is more accurate in workdays compare to weekends.

2.7.2 Test Results of MSL method for the System Data

For the MSL algorithm, our test case include hourly load and temperature data from a North-American electric utility, the available data from January 1st 1991 to September 30th 1992. This data set has also been considered in [54]. The MSL algorithm is used to train our models and predict next day load, we evaluate the performance by forecasting the load demand from February to September for 1992. Test results are shown in Table 4.2. We compare our results with the SL method which described in the previous section.

For this test case, MSL method has better performance than SL method. The overall MAPE for MSL is 2.56 while SL is 3.85. For the workday predic-

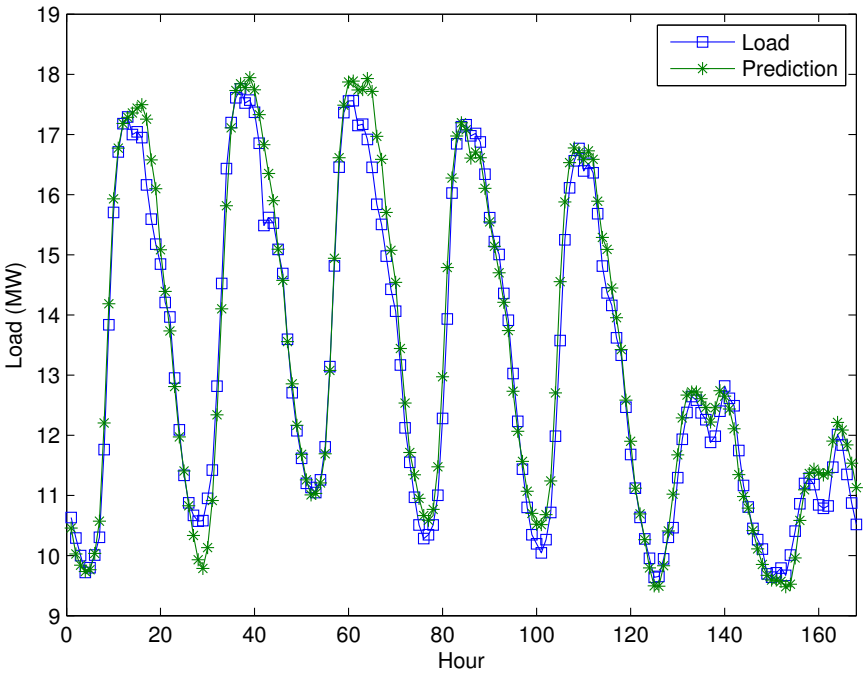

Figure 2.13: Forecasting by ANN method for one week in February using substation data

tion, MSL also outperform the SL method. This indicates that the previous load is a good predictor for the one day ahead load forecasting. Figure 2.15 and Figure 2.16 show one week load forecasting by MSL method for one week in February and July respectively.

2.8 Conclusions

Load forecasting is essential for the successful implementation of advanced control for a utility, this study deals with one day ahead load forecasting problem. ANN and MSL methods have been presented. These two methods require only historical load and weather data to forecast one day

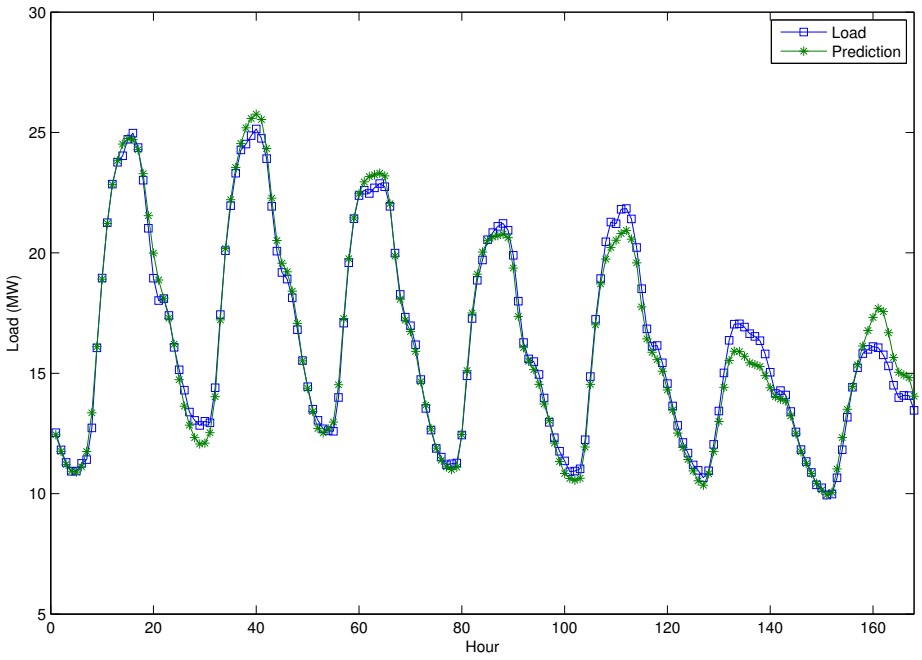

Figure 2.14: Forecasting by ANN method for one week in July using substation data

ahead load.

The concepts of the ANN algorithm has been introduced in section 2.4. We investigate the applicability of ANN method for one day ahead load forecasting. The ANN algorithm has been applied for a substation data. The MSL method is based on a multiplicative model. The historical load is used as a dependent variable, which improve the model significantly. An iterative algorithm to estimate the model parameters is developed. The algorithm performs iteratively linear regressions of the load as functions of weather parameters. MSL method has been tested using a system level data. Computation results illustrate that both methods can produce accurate predictions.

	with weekend		without weekend	
	SL	MSL	SL	MSL
ALL	3.85	2.56	3.82	2.51
Feb	3.35	2.97	3.17	2.75
Mar	2.96	2.43	2.74	2.31
Apr	5.08	3.12	4.87	3.16
May	5.61	3.72	5.92	3.75
Jun	4.16	2.75	4.36	2.71
Jul	2.86	1.81	2.87	1.89
Aug	2.32	1.38	2.28	1.30
Sep	4.51	2.52	4.25	2.31

Table 2.2: Comparison of MSL and SL method for a system level data

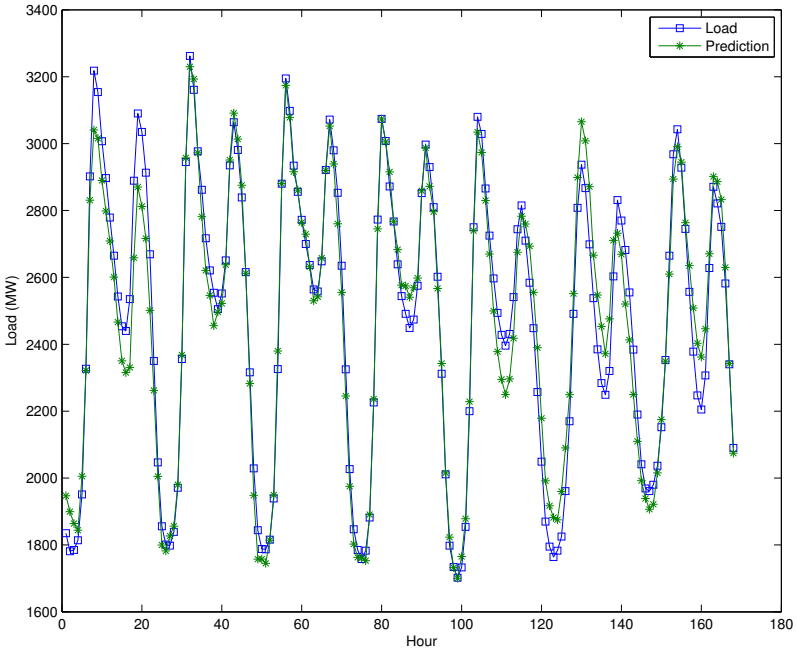

Figure 2.15: Forecasting by MSL method for one week in February using system level data

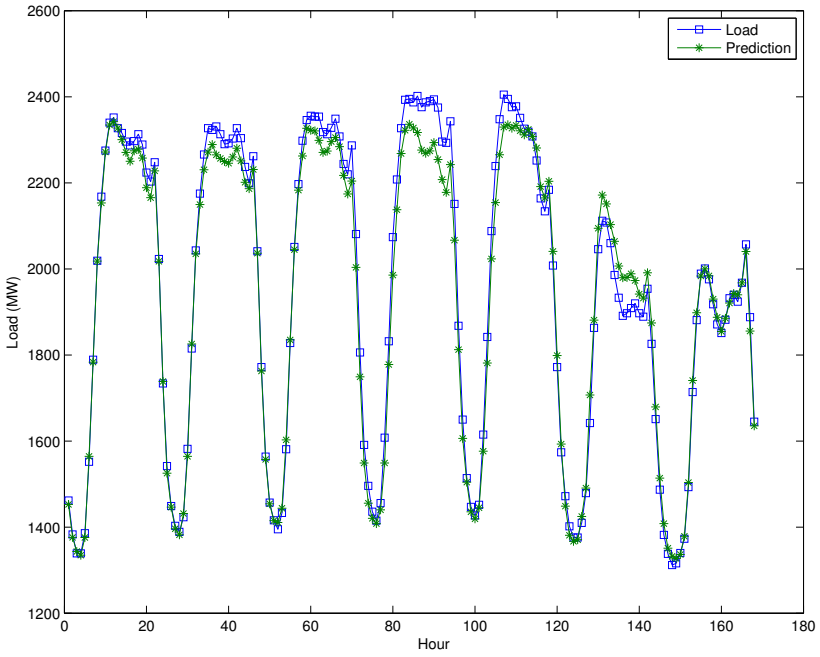

Figure 2.16: Forecasting by MSL method for one week in July using system level data

Bibliography

[1] Feeder forward neural network. "http://en.wikipedia.org/wiki/feedforward neural network", "Accessed: 2015-01-01".

[2] Power systems test case archive. "https://www.ee.washington.edu/research/pstca", "Accessed: 2015-01-01".

[3] A.S. Alfuhaid, M.A. El-Sayed, and M.S. Mahmoud. Cascaded artificial neural networks for short-term load forecasting . *IEEE Transactions on Power Systems*, 1997.

[4] Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: practice and experience*, 1994.

[5] B.Chen, M.Chang, and C.Lin. Load forecasting Using Support Vector Machines. *IEEE Transactions on Power Systems*, 19(4):1812–1830, 2004.

[6] Mark Hudson Beale, Martin T. Hagan, and Howard B. Demuth. *Neural Network Toolbox*. The MathWorks Inc., Natick, MA, 2013.

[7] B.Gou. Generalized Integer Linear Programming Formulation for Optimal PMU Placement. *IEEE Transactions on Power Systems*, 23(03), 2008.

[8] B.Xu and A.Abur. Observability Analysis and Measurement Placement for Systems with PMUs. In *Power Systems Conference and Exposition, 2004. IEEE PES*, volume 2, pages 943–946, 2004.

[9] K.W. Chan, R.C. Dai, and C.H. Cheung. A Coarse Grain Parallel Solution Method for Solving Large Set of Power Systems Network Equations . *IEEE Transactions on Power Systems*, 26, 2002.

[10] Bo-Juen Chen, Ming-Wei Chang, and Chih-Jen Lin. Load forecasting using support vector Machines: a study on EUNITE competition 2001. *IEEE Transactions on Power Systems*, 2001.

[11] Jian Chen and Ali Abur. Improved bad data processing via strategic placement of PMUs. *Power engineering society general meeting*, 1:509–513, 2005.

[12] S.-T. Chen, D.C. Yu, and A.R. Moghaddamjo. Weather sensitive short-term load forecasting using nonfully connected artificial neural network. *IEEE Transactions on Power Systems*, 1992.

[13] Ying Chen, P.B. Luh, Che Guan, and Yige Zhao. Short-Term Load Forecasting: Similar Day-Based Wavelet Neural Networks . *IEEE Transactions on Power Systems*, 2009.

[14] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support Vector Regression Machines. *IEEE Transactions on Power Systems*, 1996.

[15] F.Aminifar, A.Khodaei, M.Fotuhi-Firuzabad, and M.Shahidehpour. Contingency-constrained PMU placement in power networks. *IEEE Transactions on Power Systems*, 25(1):516–523, 2010.

[16] Eugene Feinberg, Jun Fei, Janos Hajagos, and R.J. Rossin. Smart Grid Software Applications for Distribution Network Forecasting. *Proceedings of ENERGY 2011: The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, 2011.

[17] Eugene Feinberg and D. Genethliou. *Applied Mathematics for Restructured Electric Power Systems: Optimization, Control, and Computational Intelligen*. M.I.T. Press, 2005.

[18] Eugene Feinberg, D. Genethliou, J.T. Hajagos, and B.G. Irrgang. Load pocket forecasting software . *IEEE PES Power Systems Conference and Exposition*, 2004.

[19] Eugene Feinberg, Dora Genethliou, and Janos Hajagos. Load Pocket Forecasting. *Power and Energy Systems, Proceedings of the Second IASTED International Conference*, 2002.

[20] R.M. Fiduccia, C.M.and Mattheyses. A Linear-Time Heuristic for Improving Network Partitions. *19th Design Automation Conference*, 1982.

[21] Per-Olof Fjällström. Algorithms for Graph Partitioning: A Survey. *Linköping Electronic Articles in Computer and Information Science*, 1998.

[22] C.W. Gellings. Demand Forecasting for Electric Utilities. *The Fairmont Press, Lilburn, GA*, 1996.

[23] Dora Genethliou. Statistical Approaches to Electric Load Forecasting. *Stony Brook University, Dept. of AMS*, 2005.

[24] Che Guan, P.B. Luh, L.D. Michel, Yuting Wang, and P.B. Friedland. Very Short-Term Load Forecasting: Wavelet Neural Networks With Data Pre-Filtering . *IEEE Transactions on Power Systems*, 28, 2013.

[25] T. Haida and Shoichi Muto. Regression based peak load forecasting using a transformation technique . *IEEE Transactions on Power Systems*, 1994.

[26] Eric Hartman, James D. Keeler, and Jacek M. Kowalski. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 1990.

[27] Bruce Hendrickson and Robert Leland. A multilevel algorithm for partitioning graphs. *Proceeding Supercomputing '95 Proceedings of the 1995 ACM/IEEE conference on Supercomputing*, 1995.

[28] J.P. Hespanha. An efficient MATLAB Algorithm for graph partitioning. *Technical Report, University of California, Santa Barbara, CA*, 2004.

[29] H.S. Hippert, C.E. Pedreira, and R.C. Souza. Neural Networks for Short-Term Load Forecasting: A Review and Evaluation. *IEEE Transactions on Power Systems*, 2001.

[30] Tao Hong. Short Term Electric Load Forecasting . *North Carolina State University , Raleigh, North Carolina*, 2010.

[31] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Proceedings of Cognitiva*, 1985.

[32] Chao-Ming Huang, Chi-Jen Huang, and Ming-Li Wang. A particle swarm optimization to identifying the ARMAX model for short-term load forecasting . *IEEE Transactions on Power Systems*, 2005.

[33] Shyh-Jier Huang and Kuang-Rong Shih. Short-Term Load Forecasting Via ARMA Model Identification Including Non-Gaussian Process Considerations. *IEEE Transactions on Power Systems*, 2003.

[34] J.Chen, W.Li, A.Lau, J.Cao, and K.Wang. Automated Load Curve Data Cleansing in Power Systems. *IEEE Transactions on Smart Grid*, 2010.

[35] George Karypis and Vipin Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs . *JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING*, 1998.

[36] George Karypis and Vipin Kumar. A fast and highly quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[37] Rajesh Kavasseri and Sudarshan K. Srinivasan. Joint Placement of Phasor and Power Flow Measurements for Observability of Power Systems. *IEEE Transactions on Power Systems*, 26, 2011.

[38] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell System Technical Journal*, 1970.

[39] A. Khotanzad, Z. Enwang, and H. Elragal. A neuro-fuzzy approach to short-term load forecasting in a price-sensitive environment. *IEEE Transactions on Power Systems*, 2002.

[40] S. J. Kiartzis, C. E. Zoumas, J. B. Theocharis, A. G. Bakirtzis, and V. Petridis. Short-term load forecasting in an autonomous power system using artificial neural networks. *IEEE Transactions on Power Systems*, 1997.

[41] Yann LeCun. A learning scheme for asymmetric threshold networks. *Proceedings of Cognitiva*, 1985.

[42] S. Lin and J.E. Van Ness. Parallel solution of sparse algerbraic equations. *IEEE Transactions on Power Systems*, 9(2):743–749, 1994.

[43] M.Amano, A.I. Zecevic, and D.D. Siljak. An improved block-parallel newton method via epsilon decompositions for load-flow calculations. *IEEE Transactions on Power Systems*, 11(03):15191525, 1996.

[44] S. Maybee and N.D Uri. Time series forecasting of utility load duration curves . *Electrical Engineering Journal*, 1979.

[45] M.Esmaili, K.Gharani, and H.A. Shayanfar. Redundant observability PMU placement in the presence of flow measurements considering contingencies. *IEEE Transactions on Power Systems*, 28(4), 2013.

[46] David Meyer. Support Vector Machines. *Technicishe University, Wien, Austria,*, 2010.

[47] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. M.I.T. Press, 1969.

[48] S. A. Nezam-Sarmadi, S. Nouri-Zadeh, A. M. Ranjbar, and M. R. Pishvaie. An islanding algorithm to restore a PMU installed power system. *Power and Energy Engineering Conference (APPEEC), 2010 Asia-Pacific*, 2010.

[49] R.F. Nuqui and A.G. Phadke. Phasor measurement unit placement techniques for complete and incomplete observability. *IEEE Transactions on Power Delivery*, 20(4):2381–2388.

[50] A. D. Papalexopoulos, Shangyou Hao, and T.-M. Peng. An implementation of a neural network based load forecasting model for the EMS. *IEEE Transactions on Power Systems*, 1994.

[51] Alex D. Papalexopoulos and Timothy C. Hesterberg. A regression-based approach to short-term system load forecasting. *Transactions on Power Systems*, 5(4), 1990.

[52] D.C. Park, M.A. El-sharkawi, R.J. Marks, L.E. Atlas, and M.J. Damborg. Electric load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 6(2), 1991.

[53] M. Rafian, M.J.H. Sterling, and M.R. Irving. Decomposed load-flow algorithm suitable for parallel processor implementation. *Generation, Transmission and Distribution, IEE Proceedings, 132,Pt. C, 6*, 1985.

[54] A.J.R. Reis and A.P. Alves da Silva. Feature extraction via multiresolution analysis for short-term load forecasting. *IEEE Transactions on Power Systems*, 2005.

[55] D. Rumelhart and J. McClelland. *Learning Internal Representations by Error Propagation*. MIT Press, 1987.

[56] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by backpropagating errors. *Nature*, 1986.

[57] S. Ruzic, A. Vuckovic, and N. Nikolic. Weather sensitive method for short term load forecasting in Electric Power Utility of Serbia. *IEEE Transactions on Power Systems*, 2003.

[58] H. Saadat. *Power System Analysis.* New York: McGraw-Hill, 1999.

[59] S.Chakrabarti and E.Kyriakides. Optimal placement of phasor measurement units for power system observability . *IEEE Transactions on Power Systems*, 23(3), 2008.

[60] Kyung-Bin Song, Seong-Kwan Ha, Jung-Wook Park, and Dong-Jin Kweon. Hybrid load forecasting method with analysis of temperature sensitivities. *IEEE Transactions on Power Systems*, 2006.

[61] J. Taylor and P.McSharry. Short-term load forecasting methods: an evaluation based on european data. *IEEE Transactions on Power Systems*, 22(04):2213–2219, 2007.

[62] T.L.Baldwin, L.Mili, M.B.Boisen, and R.Adapa. Power system observability with minimal phasor measurement placement. *IEEE Transactions on Power Systems*, 8(2), 1993.

[63] Vladimir N. Vapnik. *the nature of statistical learning theory.* Springer-Verlag, New York, 1987.

[64] Lipo Wang. *Support Vector Machines: Theory and Applications.* Springer-Verlag, New York, 2005.

[65] Xiaofang Wang, Sotirios G. Ziavras, Chika Nwankpa, Jeremy Johnson, and Prawat Nagvajara. Parallel solution of Newton's power flow equations on configurable chips. *International Journal of Electrical Power and Energy System*, 2006.

[66] Y.Q. Wang and H.B. Gooi. New ordering methods for sparse matrix inversion via diagonalization. *IEEE Transactions on Power Systems*, 12(3):1298–1305, 1997.

[67] P. Werbos. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. *PhD thesis, Harvard University*, 1974.

[68] W.Hong. Electric load forecasting by support vector modelling. *Applied Mathematical Modelling*, 33(5):2444–2454, 2009.

[69] Young-Min Wi, Sung-Kwan Joo, and Kyung-Bin Song. Holiday Load Forecasting Using Fuzzy Polynomial Regression With Weather Feature Selection and Adjustment. *IEEE Transactions on Power Systems*, 2011.

[70] H.L. Willis. *Spatial Electric Load Forecasting*. Marcel Dekker,New York, 1996.

[71] J.Q. Wu and A. Bose. Parallel solution of large sparse matrix equations and parallel power flow. *IEEE Transactions on Power Systems*, 10, 1995.

[72] Hong-Tzer Yang, Chao-Ming Huang, and Ching-Lien Huang. Identification of ARMAX model for short term load forecasting: an evolutionary programming approach . *IEEE Transactions on Power Systems*, 1995.