

Big Data Framework for Analytics in Smart Grids and Applications on Electric
Vehicle Loads

By
Amr Munshi

A thesis submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in
Computer Engineering

Department of Electrical and Computer Engineering
University of Alberta

©Amr Munshi, 2019

Abstract

The traditional electric grid based on centralized generation plants and unidirectional transmission and distribution systems is transitioning to a smart grid that is decentralized and multidirectional with high integration of information and communication technologies. With the rapid development of smart grids, large amounts of smart meters and sensors are deployed with huge coverage. As a result, large amounts of multi-sourced heterogeneous smart grid data are being produced. This massive amount of data needs to be sufficiently managed to increase the efficiency, reliability, and sustainability of the smart grid. Interestingly, the nature of smart grids can be considered as a big data challenge that requires advanced informatics techniques and cyber-infrastructure to deal with huge amounts of data and their analytics to take the smart grid a step forward in the big data era.

In this thesis, a big data framework that potentially promotes innovative smart grid data analytics is presented. Further, the framework is developed to comply with the Lambda architecture that is capable of performing parallel batch and real-time operations on distributed data. Implementations of the frameworks on cloud-computing based platforms are presented, and various applications are applied on top of the framework, including visualization, load monitoring, and data mining. The framework is able to acquire, store, process and query massive amounts of smart grid data in near real-time, which is milliseconds in this study. This suggests that the framework is feasible in performing further smart grid data analytics.

The second part of the thesis presents various smart grid applications that are applied on top of the smart grid big data framework. First, an unsupervised algorithm to extract electric vehicle charging loads (EVCLs) non-intrusively from the smart meter data is proposed. The proposed

algorithm can run on low-frequency smart meter sampling data and requires only the real power smart meter measurement, which is the type of data recorded and communicated by most smart meters. Validation results on real aggregated residential household loads have shown that the proposed approach is efficient in extracting EVCLs and effective in mitigating the interference of other appliances, such as cloth dryers and air condition systems, that have similar load behaviors as electric vehicles (EVs). Secondly, a method to define flexibility for the collective EV charging demand is presented. This is achieved by analyzing the time-variable patterns of the aggregated EV charging behaviors. Furthermore, a case study on real residential data to analyze EV charging trends and quantify the flexibility achievable from the aggregated EV load in different time periods is presented. To verify the effectiveness of the approach, the EVCL extraction algorithm was applied on real residential datasets. The results of extracting the EVCLs from residential households were satisfactory. The extracted EVCLs were segmented into weekdays and weekends, and the flexibility achievable from the collective EV charging behavior was analyzed. Further, statistical indicators that represent time periods where trends in EV charging may occur are discussed. Finally, a method to group EV charging customers into clusters to reshape the aggregated EVCL is presented. This part of the thesis promotes the reliability and economical operation of smart grids. The utilized indicators based on statistical analysis can potentially assist operators and researchers in understanding time periods where trends in EV charging behaviors may arise and act accordingly.

Preface

This thesis is an original work by Amr Munshi. As detailed in the following, some chapters of this thesis have been published or accepted for publication as scholarly articles in which Prof. Yasser A.-R. I. Mohamed was the supervisory author and has contributed to concepts formations and the manuscript composition.

Chapter 3 of this thesis has been published as **A. A. Munshi** and Y. A.-R. I. Mohamed, “Big data framework for analytics in smart grids,” *Electric Power Systems Research*, vol. 151, pp. 369-380, Oct. 2017.

Chapter 4 of this thesis has been published as **A. A. Munshi** and Y. A.-R. I. Mohamed, “Data lake lambda architecture for smart grids big data analytics,” *IEEE Access*. vol. 6, pp. 40463-40471, Aug. 2018.

Chapter 5 of this thesis has been accepted for publication as **A. A. Munshi** and Y. A.-R. I. Mohamed, “Unsupervised non-Intrusive extraction of electrical vehicle charging load patterns,” *IEEE Transactions on Industrial Informatics*. DOI: 10.1109/TII.2018.2806936

Chapter 6 of this thesis has been published as **A. A. Munshi** and Y. A.-R. Mohamed, “Extracting and defining flexibility of residential electrical vehicle charging loads,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 448-461, Feb. 2018.

Acknowledgment

I would like to express my gratitude to my supervisor, Prof. Yasser A.-R. I. Mohamed, for his continuous guidance and assistance throughout my Ph.D studies. His valuable discussions, insightful suggestions, and constant feedback were always helpful and inspiring. Also, his continuous support and encouragement were my greatest motive to aim for the best.

I would like to express my thanks to the examiners committee for their support, valued time and interests in my thesis.

I would like to show my deepest gratitude and respect to my family, especially my parents, to whom I owe all the success in my life. No words can express my gratitude, but I pray that God will bless them and reward them.

Many thanks to my wife, without whom I could have never been able to achieve this work. Her patience and encouragement were always a source of strength for me.

Many thanks to my daughter and two sons, who accepted trading our playing time together with research time. I will make it up to them.

Table of Contents

Abstract	ii
Preface	iv
Acknowledgment	v
Table of Contents	vi
List of Tables.....	xii
List of Figures	xiii
List of Acronyms.....	xvi
Chapter 1	1
Introduction	1
1.1 Research Motivations	1
1.2 Thesis Objectives.....	2
1.3 Thesis Contributions.....	3
1.4 Thesis Outline.....	4
Chapter 2	5
Literature Survey.....	5
2.1 Smart Grid Big Data.....	5
2.2 Hadoop Platform.....	7
2.3 Lambda Architecture	8
2.4 Electric Vehicle Load Monitoring.....	8
2.5 Defining Flexibility of Residential Electric Vehicle Loads	10
Chapter 3	12
Big Data Framework for Analytics in Smart Grids	12

3.1	Introduction	12
3.2	Big Data Core Components for Smart Grids.....	12
3.2.1	Data Acquisition Component.....	13
3.2.2	Distributed Data Storing and Processing Components	14
3.2.3	Data Querying Component.....	15
3.2.4	Data Analytics Components.....	15
3.3	Features of Hadoop’s Platform for Smart Grids.....	15
3.3.1	Scalability.....	15
3.3.2	Flexibility	16
3.3.3	Fault Tolerance.....	16
3.4	Proposed Big Data Framework for Smart Grids	16
3.4.1	Data Generation.....	17
3.4.2	Data Acquisition.....	18
3.4.3	Data Storing and Processing	18
3.4.4	Data Querying	19
3.4.5	Data Analytics.....	19
3.5	Implementation on a Cloud Computing Platform	19
3.5.1	Cloud Platform	20
3.5.2	Flume.....	22
3.5.3	Hadoop Platform	24
3.5.4	Hive.....	25
3.5.5	Impala.....	26
3.5.6	Visual Analytics	27
3.6	Practical Applications of the Framework.....	27

3.6.1	Single-house Application	28
3.6.2	Smart Grid Application	31
3.7	Conclusion	33
Chapter 4	35
Lambda Architecture for Smart Grids Big Data Analytics	35
4.1	Introduction	35
4.2	Features of the Lambda Architecture for Smart Grids	35
4.2.1	Robustness and Fault Tolerance.....	35
4.2.2	Low Latency.....	36
4.2.3	Scalability.....	36
4.2.4	Generalization and Flexibility.....	36
4.3	Smart Grid Big Data Lambda Architecture Eco-system	36
4.3.1	Smart Grid Data	37
4.3.2	Data Collecting.....	38
4.3.3	Lambda Architecture (Data Storing and Processing).....	38
4.3.4	Data Querying	38
4.3.5	Analytics.....	39
4.4	Implementation on a Cloud Computing Platform	39
4.5	Practical Applications of the Smart Grid Big Data Lambda Architecture	42
4.5.1	Storing and Organizing the Data.....	42
4.5.2	Visualization of Smart Grid Loads.....	43
4.5.3	Clustering Residential Customer Daily Loads	44
4.6	Conclusions	47
Chapter 5	48

Unsupervised Non-intrusive Extraction of Electrical Vehicle Charging Load Patterns (EEVCLP).....	48
5.1 Introduction	48
5.2 Theoretical Background of ICA for Extracting EVCLs.....	48
5.3 Independent Component Analysis for Extracting Electric Vehicle Loads.....	49
5.4 Proposed EEVCLP Algorithm.....	51
5.4.1 General Aspects.....	51
5.4.2 Iterative Process	54
5.4.3 Extraction of Gradual Increase in the EVCL (Stage1).....	60
5.4.4 Extraction of Gradual Decrease in the EVCL (Stage3) and Correction Phase ...	61
5.5 Verifications and Discussions	63
5.5.1 Verification and Comparison on Dataset#1 on Extracting Stage2 Patterns.....	64
5.5.2 Verification on Dataset#1 and #2 on Extracting All Stages.....	69
5.5.3 Verification on Extracting Hourly EVCLs.....	71
5.5.4 Lower Sampling Rates	73
5.5.5 Extracting EVCLs of Different Categories	74
5.6 Conclusions	75
Chapter 6	77
Defining Flexibility of Residential Electrical Vehicle Charging Loads	77
6.1 Introduction	77
6.2 Modeling Demand Variations Using Bayesian Maximum Likelihood	77
6.2.1 Binomial Representation of Variations.....	78
6.2.2 Bayesian Maximum Likelihood Estimation.....	78
6.3 Flexibility Definitions for EVCLs.....	81

6.3.1	Flexibility Index of EV Aggregated Demand (FIEVAD).....	81
6.3.2	Flexibility Percentage Level (FPL).....	82
6.4	Applications and Discussions.....	84
6.4.1	Extracting and Comparing the Aggregated EVCLs.....	84
6.4.2	Case Study on Defining Flexibility.....	87
6.5	Conclusions	94
Chapter 7		95
Clustering and Targeting EV Charging Customers for Load Shaping		95
7.1	Introduction.....	95
7.2	Clustering Customers into Groups.....	96
7.3	Methodology to Target Customers and Reshape the EVCL.....	96
7.3.1	Retrieving and Clustering the EV Charging Customers	96
7.3.2	Using Flexibility Indices to Reshape EV Load.....	97
7.4	Case Study on Choosing Customers to Reshape EVCL	98
7.5	Conclusion.....	104
Chapter 8		105
Summary and Future Work.....		105
8.1	Summary.....	105
8.2	Future Work.....	107
Bibliography.....		108
Appendix A		116
Flume Agent Configuration		116
Appendix B		118
Commands to Configure Spark for Matlab and Read/Store into HDFS		118

Appendix C	119
Preprocessing of ICA	119
Appendix D	121
Modified F-Score	121
Appendix E.....	122
K-means Clustering Algorithm	122

List of Tables

Table 4.1 Robustness Test Execution Times for The Smart Grid Big Data Eco-System on Clustering Residential Customer Daily Loads	47
Table 5.1 Stage2 Charging Amplitudes of EVs in Global Markets.....	50
Table 5.2 The Parameters and Descriptions for the EEVCLP Algorithm.....	53
Table 5.3 Performance Comparison of the Proposed Algorithm with Estimation <i>Methods 1- 4</i> , $ps = 5$ and $fs = 15$ to Extract <i>stage2</i>	66
Table 5.4 Performance Comparison of the Proposed Algorithm with Estimation <i>Methods 1-4</i> , $ps = 10$ and $fs = 20$ to Extract <i>stage2</i>	66
Table 5.5 Performance of the Algorithm in [39] on Dataset#1	68
Table 5.6 Performance of the Algorithm on Dataset#1 and Dataset#2 for All Stages.....	70
Table 5.7 Performance of the Proposed Algorithm with Lower Sampling Rates.....	73

List of Figures

Figure 2.1: The three stages of the EV charging loads pattern.....	10
Figure 3.1: A hierarchical architecture of the core components for smart grid big data, including the components of data acquisition, data storing, data processing, data querying and data analytics.....	13
Figure 3.2: A basic Flume topology to ingest data into HDFS.....	14
Figure 3.3: The framework to deal with smart grid big data for visual analytics. The framework covers the lifecycle of smart grid data from data generation to data analytics and forms a learn and response loop.....	17
Figure 3.4: The IP address and host name of the machines identified at each cluster node in the <i>/etc/hosts</i> file.....	20
Figure 3.5: Commands to setup an SSH connection.....	21
Figure 3.6: Sample of the csv file that includes attributes of timestamp, smart meter's ID, generated power and zip code.....	22
Figure 3.7: Flume configuration file <i>flume.conf</i> that defines how the source, sink and channel are wired together to form the data flows.....	23
Figure 3.8: HDFS distributes file blocks among cluster nodes.....	25
Figure 3.9: CDH distributes the work out to the nodes.....	25
Figure 3.10. The Hive SQL-like query to build a table that includes the time stamp, ID and consumption of the smart meters.....	25
Figure 3.11: The Impala SQL-like query to build a table that includes the timestamp, consumption, pvpower, windpower and zip code.....	26
Figure 3.12: Dashboards for power status. (a) cumulative consumption and generation with one-minute time resolution. (b) power status of the house. (c) Map with pie-chart for consumption (red) and generation for the house.....	30
Figure 3.13: Dashboards for power status in the smart grid. (a) Power consumption of 6,436 Irish home and businesses updated every 30-minutes. (b) Power consumption of 11 selected smart meters.....	32

Figure 4.1: The smart grid big data eco-system to deal with the smart grid big data from data collecting to data analytics, with visualization and feedback loop capabilities.....	37
Figure 4.2: Hierarchical view of the utilized components to implement the smart grid big data eco-system.....	40
Figure 4.3: Energy consumption observation for the residential customers for Jan. 8, 2017.....	43
Figure 4.4: Radoop RapidMiner nest process to preprocess the data, apply the data mining clustering K-means algorithm and store the results into the HDFS repository.....	45
Figure 4.5: Visualization of the five cluster representatives of the K-means clustering algorithm.....	46
Figure 4.6: Zoomed view into cluster# 4 as it represents abnormal or interesting load consumption behavior.....	46
Figure 5.1: The extraction of EVCL from the aggregated load problem. Source1 is the aggregated load pattern without the EVCL. Source2 is the EVCL. Mixture is the aggregated load pattern.....	50
Figure 5.2: Flow-chart of the EEVCLP.....	53
Figure 5.3: Zoomed view for EVs from different categories (a) <i>stage1</i> durations. (b) <i>stage3</i> durations.....	61
Figure 5.4: The result of the proposed algorithm with estimation <i>Method 3</i> . (a) the EVCL after applying ICA. (b) EVCL after removing the FPs. (c) final EVCL (transparent yellow) vs. the actual EVCL (red).....	65
Figure 5.5: The actual EVCL vs. the results from [39] and the proposed algorithm on extracting <i>stage2</i>	68
Figure 5.6: The actual EVCL vs. the extracted EVCL from the proposed algorithm with estimation <i>Method 3</i> for house#19 after undergoing correction phase.....	71
Figure 5.7: The actual EVCL vs. the extracted EVCL from the proposed algorithm with all stages.....	71
Figure 5.8. (a) The extracted EVCLs from Dataset#1 and Dataset#2. (b) The cumulative EVCLs for the neighborhood.....	72
Figure 5.9: Extracting EVCLs of different categories. (a) The actual EVCL of house#1 added to an actual ALP. (b) The resulted EVCL of the first run of the algorithm. (c) The resulted EVCL of	

the second run of the algorithm.....	74-75
Figure 6.1. The daily residential loads for July 2016. (a) weekdays. (b) weekends.....	85
Figure 6.2. The daily EVCLs for July 2016. (a) weekdays. (b) weekends.....	86
Figure 6.3. Comparison between the aggregated actual and extracted EVCLs. (a) weekdays. (b) weekends.....	87
Figure 6.4: General layout of the procedure.....	88
Figure 6.5: The probability of increase in EVCL and its upper and lower limits. (a) weekdays. (b) weekends.....	89
Figure 6.6: The FIEVAD values. (a) weekdays. (b) weekends.....	90
Figure 6.7. The FPL percentage. (a) weekdays. (b) weekends.....	92
Figure 6.8. The 15-min aggregated EV load (blue), $+VPL$ in kW (red) and $-VPL$ in kW (yellow). (a) weekdays. (b) weekends.....	93
Figure 7.1. Relationship between chapters 5, 6 and 7.....	95
Figure 7.2. The procedure followed to change the EV charging behaviours to reshape the aggregated EVCL (AggEVL) using the average daily load (ADL), ρ the upper FIEVAD and β lower FIEVAD.....	98
Figure 7.3: The FIEVAD values.....	99
Figure 7.4: The groupings EVCLs. (a) Cluster#1. (b) Cluster#2.....	101
Figure 7.5: The resulted centroids.....	101
Figure 7.6: The aggregated EVCL before (red) and after customer accepting changes (blue)..	102
Figure 7.7: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 50% at each time period.....	102
Figure 7.8: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 70% at each time period.....	103
Figure 7.9: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 70% at each time period.....	103

List of Acronyms

ADL: Average Daily Load

ALM: Appliance Load Monitoring

ALP: Aggregated Load Pattern

ANN: Artificial Neural Networks

CDH: Cloudera Distribution of Hadoop

CSA: Cloud Security Alliance

DBMS: Data Base Management System

DR: Demand Response

DDR: Dynamic Demand Response

ECS: Extracted Charging Session

EEVCLP: Extraction of Electric Vehicle Charging Load Pattern

EPRI: Electric Power Research Institution

EV: Electric Vehicle

EVCL: Electric Vehicle Charging Loads

G2V: Grid-to-vehicle

HDFS: Hadoop Distributed File System

HMM: Hidden Markov Model

ILM: Intrusive Load Monitoring

K-NN: K-Nearest Neighbour

NILM: Non-Intrusive Load Monitoring

ODBC: Open Database Connection

PV: Photovoltaic

SSH: Secure Shell

SVM: Support Vector Machines

TCP: Transmission Control Protocol

TOU: Time-of-use

V2G: Vehicle-to-grid

YARN: Yet Another Resource Negotiator

Chapter 1

Introduction

1.1 Research Motivations

Ever growing volume of data production is the reality we are living in. The recent technological advancements have led to a deluge of data from various domains, such as social networks, scientific sensors, smart cities, and the Internet. The global data volume from 2005 to 2020 is predicted to grow by a factor of 300, from 130 exabytes to 40,000 exabytes, representing a double growth every two years [1]. To cope with the volume, velocity and variety of data produced, the term “big data” was brought up to capture the meaning of this evolving trend of data.

Big data are becoming a new technology focus in science and engineering domains. Big data includes a set of tools and mechanisms to acquire, store, and process disparate data while leveraging the massively parallel processing power to perform complex transformations and analysis. However, designing and deploying a big data framework system for a specific application is not a straightforward task [2], [3]. This is due to the fact that data comes from multiple, heterogeneous and autonomous sources with complex and evolving relationships, and keeps on growing. Moreover, the rise of big data applications where data collection has grown tremendously is beyond the capability of current commonly used hardware and software platforms to manage, store and process within a tolerable amount of time [2].

Many utilities are transferring to smart meters and smart grids as part of long-range planning to improve the reliability of power supply, incorporate distributed generation resources, develop storage solutions, use the power plants efficiently, and enable customers to participate in controlling their energy use. The IEEE 2030 standard [4] states that the smart grid system is based on an interconnection of three systems: 1) the electric power system which emphasise the power generation, delivery and consumption. 2) the communication system which emphasis the communication connectivity among systems, devices, and applications. 3) the information

technology system which includes technologies that store, process and manage data information for decision making on the power system operation. The latter leads to incorporate other challenges, for example, going from a system that reads the meter once a month to a smart meter that can provide meter readings every few minutes leads to millions of reads per hour. The result is a massive increase in data that is overwhelming, if not managed properly. This generated data, if managed efficiently, can provide a better understanding of customer behavior and assist in defining electric tariffs. For example, time-of-use (TOU) pricing encourages customers to operate certain higher voltage appliances at off-peak periods. Consequently, customers save money and less power is generated. For this, developing a framework that is able to handle the smart grid big data is considered in this thesis.

The growth of electric vehicles (EVs) poses a challenge to electricity systems, but also a promising opportunity to reduce petroleum use. Accordingly, many economic and environmental issues can be overcome. For this purpose, EVs have received increasing attention recently. The U.S. electric power research institution (EPRI) projects that 62% of the entire U.S. vehicle fleet will consist of EVs by 2050 [5]. Those EV loads bring large impacts on smart grids, including an increase in system peak demand and voltage unbalances, especially at the electric power distribution level [6]. The gravity of this impact depends on the EV charging behaviors. As the rapid popularization of EVs introduces many new load peaks to the electrical grid, extracting and aggregating those EV charging loads is essential to allow smart grid operators to make intelligent and informed decisions about conserving energy and promoting the reliability of the electrical grid.

1.2 Thesis Objectives

This thesis aims at proposing a big data framework that is able to handle the smart grid big data. Also, providing a cost-effective development environment for a non-data scientist to perform smart grid research. Once the smart grid big data framework is developed, smart grid-related applications are applied on top of the framework to promote the reliability of the grid and assist the smart grid operators in planning and making informed decisions.

First, the thesis objectives for the framework development are summarized as follows.

- Proposing a comprehensive big data framework for smart grids, which covers handling the smart grid data from generation to analytics.
- Including a feedback loop in the framework to monitor the effects of made decisions.
- Utilizing open source state-of-the-art prevalent Hadoop platform to address smart grids big data challenges.
- Adopting open source tools to provide an easy and cost-effective development environment for practicing engineers to develop similar tools for their demanding smart grid applications.
- Perform various smart grid data analytical applications on top of the framework.

Secondly, smart grid data analytic applications are applied on top of the framework aiming at reducing the effects of EV charging on the electric grid. The objectives of this part are summarized as follows.

- Proposing a new unsupervised approach for solving the EV disaggregation problem.
- Extracting the EV charging loads non-intrusively from the smart meter real power data.
- Mitigating the interference of other household appliances that have similar load patterns as EVs.
- Define the EV charging demand by analyzing the time-variable patterns of the aggregated EV charging behaviors.
- Quantifying the flexibility amount achievable from the aggregate EV load in different time periods.
- Choosing groups of customers that can be targeted to reshape the collective EV load.

1.3 Thesis Contributions

The key contributions of this thesis are as follows:

- Developing an eco-system for smart grid big data applications to improve decision making and acquire further advantages for operating the smart grid.
- Providing a big data eco-system that can be built and utilized by non-data scientists for their smart grid big data applications.
- Enabling data mining and knowledge discovery from massive distributed datasets of smart grid big data.

- Proposing an unsupervised non-intrusive algorithm to extract EV charging loads from the aggregated residential loads.
- Quantifying the amount of the flexibility that can be achieved from the aggregated EV charging load demand at different time periods.
- Presenting a method to choose groups of customers to target for EV load reshaping.

1.4 Thesis Outline

The remainder of the thesis is organized as follows. A literature survey is presented in Chapter 2. Chapter 3 introduces the smart grid big data framework that covers the lifecycle of smart grid data from data generation to data analytics. The core components of the framework are discussed in details. Further, an implementation of the framework on a secured cloud computing platform and the application on two real scenarios are presented. In 2015, a big data architecture, namely the Lambda architecture, that allows batch and real-time computing was introduced. For this, in Chapter 4, the framework of Chapter 3 is extended to comply with the Lambda architecture allowing to perform parallel batch and real-time operations on distributed smart grid data. An implementation of the Lambda architecture for smart grid big data analytics on a cloud computing platform and a data mining application are presented in the same chapter. Chapter 5 presents an unsupervised algorithm to extract EV charging load patterns non-intrusively from the aggregated residential meter readings. A method to define flexibility indices for the collective EV changing demand by analyzing the time-variable patterns of the aggregated EV charging behaviors is presented in details in Chapter 6. Also, details on how the smart grid operators could utilize the flexibility indices are shown. Chapter 7 uses the methods in the previous chapter to present a method to group customers into distinct clusters based on their EV charging behaviors. Furthermore, a case study is carried out to test the feasibility of this methodology in reshaping the aggregated EV charging loads. Finally, the summary of the thesis and future work are presented in Chapter 8.

Chapter 2

Literature Survey

2.1 Smart Grid Big Data

There has been much discussion about what big data actually means [2], [7], [8]. However, the most common definition in literature is the “Vs” definition [2], [9]-[14] which includes several characteristics of big data beginning with the letter “V”. The “3Vs” definition includes [14]: volume, variety and velocity.

- **Volume:** big data implies enormous volumes of data. This data is generated by machines, networks, social media, etc. Thus, the volume of data to be analyzed is massive.
- **Variety:** refers to the various sources and formats of data (structured, semi-structured and unstructured). As data comes in the form of photos, videos, logs, sensor devices, etc., this variety of data formats creates challenges for storage, mining and analyzing data.
- **Velocity:** the velocity of data is the rate at which data arrives. This also includes the time that it takes to process and understand the acquired data to assist in decision making.

In [2], [15]-[19] various challenges and issues in adopting big data technology were discussed. From their research, it was concluded that refining a unified framework suitable for every module is not straight-forward due to the diversity of applications. The actual challenge of big data is not in collecting it, but in managing it as well as making sense of it. In general, the challenges in designing a big data system can be summarized as [2]:

- First, due to the variety of disparate data sources and sheer volume, it is challenging to collect and integrate data with scalability from distributed sources.
- Second, big data systems need to store and manage the massive heterogeneous gathered data, while providing function and performance guarantee, in terms of fast retrieval, scalability, and privacy protection.
- Third, big data analytics must effectively mine massive datasets at different levels in real-time or near real-time, including modeling, visualization, prediction, and optimization,

such that inherent promises can be revealed to improve decision making and acquire further advantages.

Many utilities are transferring to smart meters and smart grids as part of long-range planning to improve the reliability of power supply, incorporate distributed generation resources, develop storage solutions, use the power plants efficiently, and enable customers to participate in controlling their energy use. To accomplish this, utilities are deploying smart meter systems as a first step. This leads to incorporate other challenges. For example, going from a system that reads the meter once a month to a smart meter that can provide meter readings every few minutes leads to millions of readings per hour. The result is a massive increase in data that is overwhelming, if not managed properly. This generated data, if managed efficiently, can provide a better understanding of customer behavior and assist in defining electric tariffs. For example, time-of-use (TOU) pricing encourages customers to operate certain higher voltage appliances at off-peak periods. Consequently, customers save money and less power is generated.

Developing frameworks that address the challenges of smart grid big data are of research interest. In [17], [18] several main elements of big data and database technologies that are beneficial within the utility eco-system are scratched on, but a comprehensive idea on how big data elements can construct a framework to deal with smart grid data has not been presented. The work in [20] analyzed several challenges of big data and suggested that high-performance computing platforms are required to unleash the power of big data. In [21], a mathematical model for healthcare big data analytics was presented. A big data value chain in [2] was presented; it decomposed big data into four sequential modules, namely data generation, data acquisition, data storage, and data analytics. More in detail, numerous approaches for each module were highlighted, and a prevalent framework for addressing big data challenges was suggested. The work presented in [7] demonstrates a cloud-based dynamic demand response (DDR) platform project that is being deployed in the University of Southern California campus as a testbed for transforming Los Angeles utility into a smart grid in the future. In [22] a secure cloud computing based framework for big data information management in smart grids was proposed. However, a platform to apply big data analytics on the distributed cloud structure has not been included. Works related to smart grid big data mainly describe possible theoretical

frameworks and challenges, and lack practical implementation. As handling smart grid big data is becoming an area of research interest, developing an effective and comprehensive big data framework for analytics in smart grids is of interest to take the smart grids a step forward in the big data era.

2.2 Hadoop Platform

Hadoop [23] is an open-source platform that supports storing and processing large amounts of data. It relies on distributed hardware to store and process data, which enables processing large amounts of data on distributed clusters of commodity servers. The core of Hadoop consists of two components: a storage component which is Hadoop's Distributed File System (HDFS) [24] and a processing component called MapReduce [25].

HDFS [24], [25] is a distributed storage file system that is developed to run on commodity hardware. An HDFS cluster consists of NameNode(s) that manage the file system metadata, and numerous DataNodes that store data. A file is split into blocks, and these blocks are stored in a set of DataNodes. Each block has several replications distributed in different DataNodes for reliability purposes.

MapReduce [26] is the processing component of Hadoop. It mainly consists of master node(s) (JobTracker) and slave nodes (TaskTracker) per cluster. The master is responsible for scheduling jobs for the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. The MapReduce and HDFS run on the same set of nodes, which allows tasks to be scheduled on the nodes in which data are already available. In 2013, Hadoop was released with Yarn [27] (Yet Another Resource Negotiator). The fundamental idea of Yarn is to split the two major responsibilities of the JobTracker/TaskTracker of the MapReduce into separate entities. Yarn basically consists of a global ResourceManager and per-node slave NodeManager for managing applications in a distributed manner. An ApplicationMaster in Yarn negotiates resources from the ResourceManager and works with the NodeManager(s) to execute and monitor the component tasks. Each ApplicationMaster has responsibility for negotiating appropriate resource containers from the scheduler, tracking their status, and monitoring their progress.

2.3 Lambda Architecture

Lambda architecture is a data-processing architecture designed to handle massive amounts of data by taking advantage of batch and real-time methods. The basis of the Lambda architecture is to compute arbitrary functions on distributed datasets in real-time; also, to combine batch and real-time processing capabilities to balance data latency throughput and fault tolerance. However, there is no single tool that can accomplish this task. Instead, a variety of tools and techniques are used to build a complete big data system. The Lambda architecture addresses the problem of computing arbitrary functions parallel on distributed data in real-time by presenting a three-layered architecture that consists of a *batch layer*, a *speed layer*, and a *serving layer* [28].

The *batch layer* is mainly responsible for two tasks. The first is to store the constantly growing master data in a distributed file system manner, which is, in this case, a Hadoop distributed file system (HDFS) [23]. The second task is to precompute batch views for this distributed data by using the MapReduce [25] processing paradigm. Those batch views can be used to answer incoming queries with low read latency. It should be noted that the Hadoop platform, which contains the HDFS and MapReduce components, can fulfill the functionality of the batch layer.

Differently from the *batch layer*, the *speed layer* does not precompute the views for the entire data. Instead, it uses an incremental approach which stores and updates the real-time views of the data. Thus, it supplements the gap that is left by the *batch layer*. The *speed layer* only computes views for recent data, due to the fact that older data is absorbed into the *batch layer*.

The *serving layer* is a specialized distributed database that indexes the batch views so that they can be queried in a low-latency and ad-hoc manner. It is responsible for merging the results of *batch* and *speed* layer computations on the data. This way the *serving layer* can provide the real-time computation results over all the data.

2.4 Electric Vehicle Load Monitoring

Appliance load monitoring (ALM) is an essential solution for energy management that allows obtaining appliance-specific energy consumption statistics that can further be used to devise load scheduling strategies for optimal energy utilization. In literature, there are two major approaches

to ALM, namely intrusive load monitoring (ILM) and non-intrusive load monitoring (NILM). Different approaches have been used for load disaggregation. Artificial neural networks (ANN) [29], [30], support vector machines (SVM) [31], [32], and nearest neighbor (k-NN) [29], [33], are among the algorithms that have been popular for appliance disaggregation. However, the construction and training of ANNs [34] are arbitrary, and tuning can result in local maxima and overfitting. SVM [34] use optimal line separation between classifications and also requires training. The k-NN classifies unlabeled data that is nearest to each other based on a distance function. However, this method is impractical due to large storage requirements and can be susceptible to the curse of dimensionality. Recently, methods that use a hidden Markov model (HMM) [34] have become of research interest [35]-[38]. However, these methods require extensive training and computation [39]. There has been a devoted effort in extracting appliance signatures from aggregated data. Most methods require smart meter data with high sampling rates higher than 1-Hz. Also, different combinations of electrical measurements are used to disaggregate appliances, such as the active power (P), reactive power (Q) and current (I) [40]-[43]. Recent research [33], [35], [44] propose methods that utilize the real power measurements. The main reason for using only the real power measurement is because it is the type of data recorded and communicated by most smart meters.

EVs are a topic undergoing intense research, and methods to extract EV charging loads (EVCL) are recently emerging. The EVCL pattern can be decomposed into three stages: 1) gradual increase in charging load (*stage1*), 2) steady charging load (*stage2*), and 3) gradual decrease in charging load (*stage3*). Fig. 2.1 illustrates the typical stages of the EVCL pattern. In [45], a method for extracting *stage2* EVCLs is presented. However, it cannot extract the EVCLs if other appliances were operating at the time the EV was plugged-in. Also, it requires a sampling rate of 1-Hz (1-second). Moreover, it requires the active and reactive power measurements, which is not the reality of most deployed smart meters. In [39], an algorithm to extract *stage2* EVCLs with amplitudes from 3 kW to 4 kW, from the aggregated signal is presented. The algorithm did not require training and was able to outperform the HMM algorithms on extracting EVCLs. Previous works of extracting EVCLs don't include *stage1* and *stage3* patterns. Non-intrusive methods to extract various amplitudes of EVCLs including all

three stages of charging are of interest. The extraction and aggregation of such load behaviors can be aggregated and open further smart grid analyses and studies that could promote the reliability of the smart grid in spite of the rapid growth of EVs.

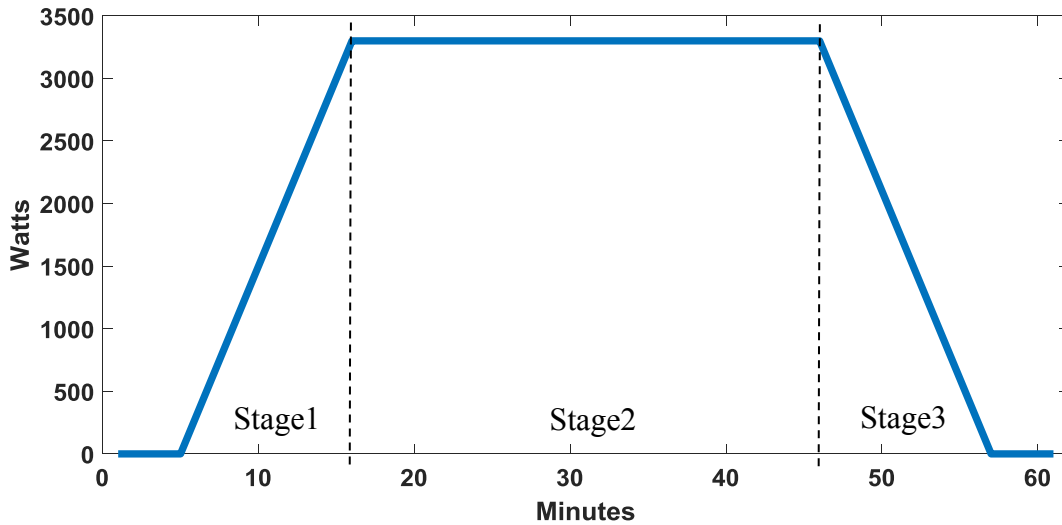


Figure 2.1: The three stages of the EV charging loads pattern.

2.5 Defining Flexibility of Residential Electric Vehicle Loads

The rapid popularization of EVs introduces many new load peaks to the electrical grid. Extracting and aggregating those EVCLs is essential to allow smart grid operators to make intelligent and informed decisions about conserving energy and promoting the reliability of the electrical grid. Hence, a huge amount of research was conducted on EVs and their upcoming challenges to the grid in recent years. Due to the instability that EVs introduce to the grid, research studies focus mainly on smart charging of EVs [46]-[49]. Also, investigating the effects of EVs on distribution networks [50] and allocating of optimal capacity and location of commercial and residential EV charging stations [51]-[55]. To promote the reliability of the grid and minimize infrastructural changes, many utilities use time-of-use (TOU) based electricity demand response (DR) programs, which can potentially promote the reliability of the grid by

shifting EV load demand from peak load periods to off-peak load periods. In a TOU system, the price of electricity differs during peak and off-peak hours. This encourages customers to charge EVs during off-peak hours [56], [57]. However, currently only 1% of residential customers in North America are billed with TOU rates, and 5% of utilities provide TOU pricing to residential customers [58]. Although the number of utilities adopting TOU-DR programs for residential customers is increasing [56], the coordination of DR resources is a challenging task due to the lack of communication with each load. The current terminology has adopted the term flexibility to indicate the capacity to adapt across temporal, circumstances, intention and area of application [59]. For the applications to the electrical grid, flexibility refers to the possibility of deploying the resources to respond reliably to the load and generation variations at acceptable costs. One of the current challenges is to address the quantification of the flexibility amount. Recent works focus on obtaining flexibility from the generation and demand side [60]-[67]; however, no works address the extraction and quantification of the flexible amount of the aggregated demand of EVs. For this purpose, quantifying the flexibility amount of the aggregated EV charging demand is a key factor in promoting the reliability of the grid in spite of the rapid growth of EVs.

Chapter 3

Big Data Framework for Analytics in Smart Grids

3.1 Introduction

This chapter highlights the big data core components used in the framework for smart grids. The features of using the Hadoop platform in the smart grid environment are highlighted. The framework's stages including, data acquisition, data storing and processing, data querying, and data analytics components are discussed in details. Also, the components that can be beneficial for smart grid big data from the smart grid applications point of view are discussed. Then the framework that covers the lifecycle of smart grid data from data generation to data analytics is presented. Also, this chapter includes an implementation of the framework on a secured cloud-computing platform. To verify the effectiveness of the framework, this chapter presents the application of the framework on two scenarios: the first scenario is a single-house that includes micro-generators (i.e., wind turbine, photovoltaic (PV) roof panels and EV), the second scenario includes a real smart metering electricity behavior dataset from the Irish Social Science Data Archive for 6436 participating Irish homes and businesses. The framework is capable of handling smart grid data from various resources including, transmission, distribution and consumption data. Finally, the conclusions and benefits of utilizing the framework for smart grids are summarized.

3.2 Big Data Core Components for Smart Grids

Figure 3.1 illustrates the hierarchical architecture of the core components of the framework for smart grid big data. In the following subsections, the components of the data acquisition, data storing and processing, data querying and, data analytics components are introduced. The data storing and processing components are included in the same subsection as they are both under the same platform (Hadoop).

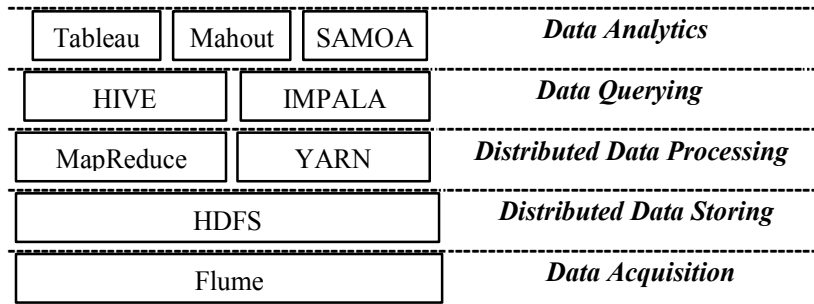


Figure 3.1: A hierarchical architecture of the core components for smart grid big data, including the components of data acquisition, data storing, data processing, data querying and data analytics.

3.2.1 Data Acquisition Component

Flume [68] is a distributed system developed by Apache that efficiently collects, aggregates, and transfers large amounts of log data from disparate sources to a centralized storage. However, flume can be used to ingest large amounts of streaming data such as social media and sensor data into the Hadoop Distributed File System (HDFS) which will be introduced in the following subsection. Figure 3.2 depicts a representative Flume topology [68] and the following components make-up the Flume tool:

- **Event:** a stream of data that is transported by Flume.
- **Source:** the entity through which data enters into Flume. A source can actively poll for data or wait for data to be delivered to it.
- **Sink:** the entity that delivers the data to the destination. A variety of sinks allows data to be streamed to multiple destinations. Here the HDFS sink that writes events to HDFS storages is used.
- **Channel:** the conduit between the source and the sink. Sources ingest events into the channel, and the sinks drain the channel.

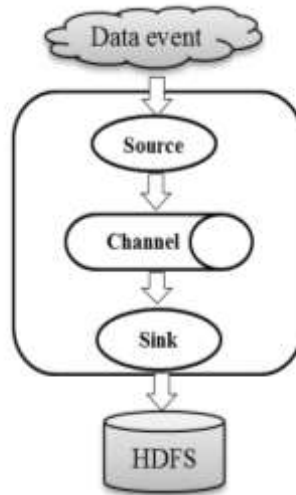


Figure 3.2: A basic Flume topology to ingest data into HDFS.

3.2.2 Distributed Data Storing and Processing Components

Hadoop [23] the open-source software platform that supports massive data storage and processing enables distributed processing of large amounts of data on clusters of commodity servers. The core of Hadoop consists of two main components: a storage component which is the Hadoop Distributed File System (HDFS) [24] and a processing component called MapReduce [26]. Hadoop is considered to be a major part of any architecture in big data.

Hadoop was inspired by Google’s work on its distributed file system, Google’s File System (GFS) [25] and the MapReduce programming model. In 2006, these software components became an Apache project, called Hadoop. MapReduce was sufficient at a few specific kinds of batch processing, which was an obstacle in taking advantage of Hadoop’s flexibility with data and storage. In 2013, Hadoop was released with Yarn [27] (Yet Another Resource Negotiator) or MapReduce2. Yarn the general-purpose resource manager for Hadoop enabled applications from other processing frameworks to run on a Hadoop cluster in a distributed manner.

3.2.3 Data Querying Component

Hive [69] and Impala [70] are two SQL-like high-level declarative languages that express big data analysis tasks. They facilitate querying and managing big data residing in distributed storage. Hive express big data analysis tasks in MapReduce operations. Whereas, Impala is an interactive SQL query tool on big data [80]. Impala doesn't have to translate an SQL query into another processing framework, such as MapReduce operations. The execution of an Impala query is executed in parallel in each node's memory in the cluster. The intermediate results of the nodes are transmitted and aggregated then returned.

3.2.4 Data Analytics Components

Mahout [71] is a data mining library implemented on top of Hadoop and provides batch machine learning processing. It contains a few core algorithms for scalable performant machine learning applications.

SAMOA [72] (Scalable Advanced Massive Online Analysis) is a distributed streaming machine learning framework that contains a programming abstraction for distributed streaming algorithms for some common data mining and machine learning tasks.

Tableau [73] is an interactive data visualization tool that enables users to analyze, visualize and share information and dashboards.

3.3 Features of Hadoop's Platform for Smart Grids

Hadoop has attracted substantial attention from both industry and scholars. In fact, Hadoop has long been the mainstay of the big data movement. Hadoop has many advantages, and the following features make it suitable for smart grid big data management and analysis:

3.3.1 Scalability

Hadoop allows hardware infrastructure to be scaled up and down without affecting the existing data [2]. The system can automatically re-distribute data and computation tasks to accommodate hardware changes. For example, if new neighborhoods or generation utilities are added to the

grid, additional nodes and storage devices can be added to the existing cluster without affecting the functionality of the existing nodes.

3.3.2 Flexibility

Hadoop is free of schema and able to absorb various types of data from numerous sources. Moreover, different types of data from numerous sources can be aggregated for further analysis. Hence, many challenges of the various types of smart grid data can be addressed.

3.3.3 Fault Tolerance

Missing data and computation failures are common in smart grid data. Hadoop can recover the data and computation failures caused by node breakdown or network congestion by storing the data at many nodes and distributing the computation work to other healthy nodes in the cluster.

3.4 Proposed Big Data Framework for Smart Grids

The framework to deal with smart grid big data is presented in Figure 3.3. The framework covers the lifecycle of smart grid data from data generation to data analytics. The following sub-sections discuss the framework stages in details.

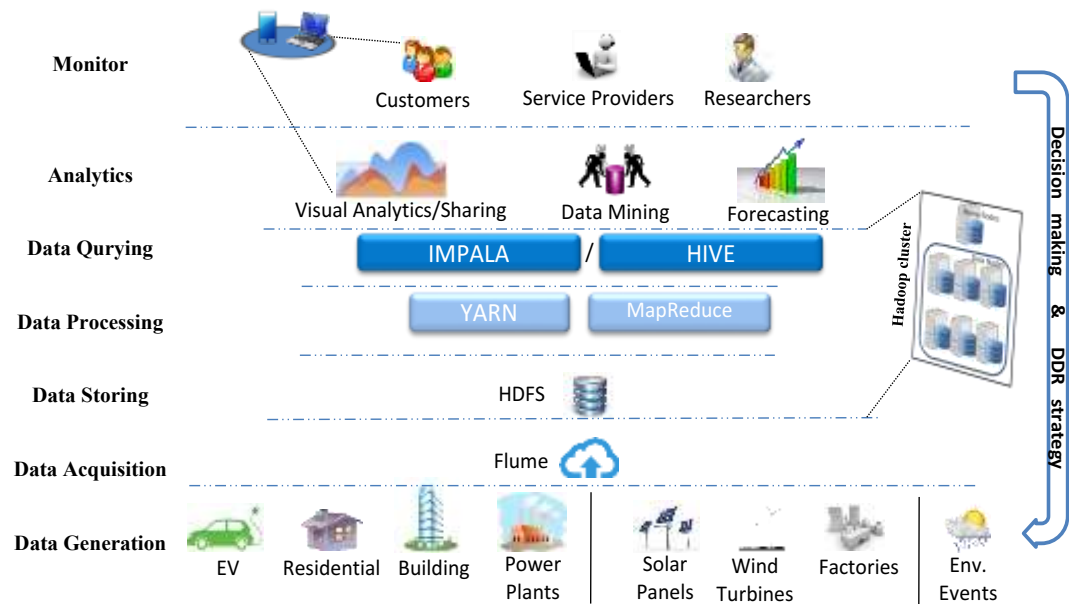


Figure 3.3: The framework to deal with smart grid big data for visual analytics. The framework covers the lifecycle of smart grid data from data generation to data analytics and forms a learn and response loop.

3.4.1 Data Generation

Streaming data is generated from thousands of smart meters in the smart grid. The generated data may belong to a supplier site or a demand site. In addition, environmental events such as weather conditions from weather stations can be beneficial. For example, to predict the amount of power that can be generated from a certain power resource such as, wind farms. In this framework, data from various sources are considered these include EVs, residential homes, commercial buildings, industrial factories, solar panels, wind turbines and various power plants. Considering data from such sources could increase the grid's reliability as technology changes are starting to permeate through the entire smart grid, from generation to transmission to distribution. For example, renewable power resources are being included in the generation mix, not just by the power generation utilities, but also by consumers through rooftop solar panels, residential wind turbines, EVs and other micro-generators that can act as positive supply to the grid.

3.4.2 Data Acquisition

The data acquisition for smart grids' data can be decomposed into three sub-tasks, namely, data collection, data transmission, and data pre-processing. The data generated from the previous stage are collected proactively by centralized/distributed agents. The collected data is then transmitted to a master node in the Hadoop cluster. Once the raw data are gathered, it is transferred to a data storage infrastructure for subsequent processing. Due to the diverse source of data, the collected data may have different formats and information. Accordingly, data pre-processing is required. Data integration techniques aim to combine data from different sources and provide a unified view of the data [2]. In this framework, the data is transferred to comma-separated value (csv) files. The attributes of the data contain information such as, the timestamp, smart meter ID, generated/consumed power and location. Also, in the pre-processing of data, inaccurate and incomplete data can be amended or removed to improve the quality of data; also, this can be done at a further step in the process.

Flume can fulfill the function of the data acquisition. It can collect, aggregate, and transfer the large amounts of generated data from various sources to a Hadoop master node. When a Flume source receives data, it stores it into one or more channels. The channel is a passive store that keeps the event until it is consumed by a flume sink. The flume sink removes the event from the channel and puts it into an external repository. In this framework, the data files are ingested into an external HDFS repository.

3.4.3 Data Storing and Processing

After the smart grid data has been acquired, in this stage, the HDFS handles storing the data for further processing. An HDFS cluster consists of a single NameNode that manages the file system metadata, and collections of DataNodes that store the actual data. The received smart grid data is split into one or more blocks, and these blocks are stored in a set of DataNodes. Hadoop Yarn is the computation core for big data analysis. The HDFS and Yarn run on the same set of nodes, which allows tasks to be processed on the nodes in which smart grid data are already present.

3.4.4 Data Querying

Hive and Impala are used in this framework to read the smart grid data from an HDFS repository and select, analyze or generate data of interest. For example, the consumption of electricity for a certain region or the aggregated power produced from wind farms can be obtained. The data querying stage runs on top of a Hadoop cluster which allows obtaining prompt results.

3.4.5 Data Analytics

The smart grid data acquired must be shared to improve the efficiency of the smart grid. For example, this data can be utilized by analytics for proposing curtailment, researchers for data mining and correlations, and consumers visualizing and gaining knowledge of their power profiles.

The data analytics stage has two main objectives, to learn and to respond. Sharing the grid's status between utilities and consumers promotes the reliability of the smart grid. Also, consumers act as an active part in the reliability of the grid. This can be achieved through visualization dashboard portals, which provide a visualization of the smart grid's status that can be accessed via the Internet or mobile apps. Consequently, a DR strategy by analytics can be suggested to determine customers and buildings to target during a peak load period. Moreover, dynamic power pricing and incentives for reducing load during peak periods can be advertised for.

3.5 Implementation on a Cloud Computing Platform

This section demonstrates the implementation of the framework on a cloud computing platform. Then a method to establish a secure connection between the cloud cluster nodes is briefly presented. Further, the settings of the components that are used at each stage of the framework are discussed. For simplicity, in the data storing and data processing stages a Hadoop distribution namely, Cloudera distribution Hadoop (CDH) that contains MapReduce/Yarn and HDFS is used instead of setting up each component separately.

3.5.1 Cloud Platform

Cloud computing can be deployed as the infrastructure layer for big data systems to meet certain infrastructure requirements, such as cost-effectiveness, improved accessibility, and scalability. Based on the requirements of the proposed framework, Infrastructure as a Service (IaaS) clouds [74] is appropriate to use to implement the smart grid big data framework. Cloud service providers such as, Amazon AWS and Google can be utilized to build a cluster that will host the framework. In this implementation, a Google cloud platform cluster with six machines is used. Five machines running CentOS Linux operating system will be deployed for the Hadoop platform. The remaining machine will be running Windows operating system to perform the visual analytical tasks. In the Hadoop cluster there will be one master node and four slave nodes. The IP address and hostname of the nodes are identified at each node in the `/etc/hosts` file (Figure 3.4):

```
10.240.0.2  master.sgf  #IP and node name for master
10.240.0.3  slave1.sgf  #IP and node name for slave1
10.240.0.4  slave2.sgf  #IP and node name for slave2
10.240.0.5  slave3.sgf  #IP and node name for slave3
10.240.0.6  slave4.sgf  #IP and node name for slave4
```

Figure 3.4: The IP address and hostname of the machines identified at each cluster node in the `/etc/hosts` file.

It should be noted that at any time a new node is added to the cluster, it must be defined to all other cluster nodes in the `/etc/hosts` file.

Often smart grid big data analysis is conducted with a vast array of data sources that come from many sources. For that, it is required to be aware of the security and governance policies that apply to various smart grid data sources. The data that remains will need to be secured and governed. Therefore, a well-defined security strategy is required. It should be notated that security is something that requires frequent update strategies because the state-of-the-art is constantly evolving. In order to establish the smart grid framework in a secure cluster environment, Secure Shell version 2 (SSH) protocol [75] is adopted. SSH is a cryptographic network protocol that allows network systems to operate securely over an

unsecured network. The SSH provides a secure encrypted link in a client-server architecture, which connects a client with a server. In addition, SSH provides authentication, encryption and data integrity to secure network communications. The setup of SSH enables different operations on the cluster, such as starting, stopping, and distributing operations to nodes. In the context of our cluster, it provides secured connection between the slaves and master(s) nodes. Also, specifies how a node can connect securely to another node, and then use the resulting secure connection to access the other nodes resources. An advantage of implementing the framework using known cloud service providers is that their cloud service complies with the Cloud Security Alliance (CSA) which promote the use of best practices for providing and ensuring security within cloud computing. To implement the SSH in the cloud cluster, the public key authentication method is used. Public key authentication in SSH is considered to be a popular strong authentication method. To accomplish this, a manually generating public and private key are generated on a node. The public key will be given to all cluster nodes that require authentication. Any data encrypted with that public key, will be decrypted with the corresponding private key. Thus, every cluster node has a file which contains the complete list of the other node keys. While authentication is based on the private key, the key itself is not transferred through the network during authentication. The SSH only verifies if the same node offering the public key also owns the matching private key. This will prevent a node that doesn't belong to the cluster to connect as an authenticated node (eavesdropping). The commands to setup an SSH connections are presented in Figure 3.5.

```
~]$ ssh-keygen
~]$ cd ~/.ssh
~]$ cp id_rsa.pub authorized_keys #copying node key to list of authorized keys
```

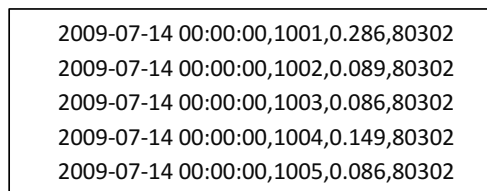
Figure 3.5: Commands to setup an SSH connection.

Then the public key (*id_rsa.pub*) is copied to each node. Hence, every node in the cluster has the complete list of the other node keys.

The components that are needed to implement the framework (i.e., Flume, Hadoop, Hive, Impala) are setup on the cluster. Fortunately, those components can be found in open source distributions. For simplicity, in this framework CDH is used. During the setup of CDH, the master and slave nodes are identified by their hostname or IP address. The master node will run the master “daemons” and it knows where the slaves are located and how many resources they have. A node identified as master runs several services; the most important is the ResourceManager which decides how to assign the resources. Nodes identified as slaves announce themselves to the ResourceManager. Periodically, they send heartbeats to the ResourceManager. Each slave node offers resources to the cluster. The resource capacity is the amount of memory and the number of cores. At run-time, the ResourceManager will decide how to use this capacity.

3.5.2 Flume

Once the data are available from smart meters, it is sent to a local node. An advantage of using the cloud service is that the data can be sent from any location as long as there is an Internet connection and the security protocol of the cloud allows it. For example, a neighborhood’s data can be sent to a node using one Internet connection. It should be noted that the network communication and aggregation of smart meter data is beyond the scope of this thesis. Nodes that act as flume agents can be master or slave nodes. In this implementation, a flume agent receives data in a csv file format. The attributes of the file can include data such as, the timestamp (Datetimes), smart meter’s ID (ID), generated (Gen)/consumed (Cons) power and zip code (Zip) (Figure 3.6).



```
2009-07-14 00:00:00,1001,0.286,80302
2009-07-14 00:00:00,1002,0.089,80302
2009-07-14 00:00:00,1003,0.086,80302
2009-07-14 00:00:00,1004,0.149,80302
2009-07-14 00:00:00,1005,0.086,80302
```

Figure 3.6: Sample of the csv file that includes attributes of timestamp, smart meter’s ID, generated power and zip code.

When a flume source receives an event it stores it into a channel that keeps the file event until it's consumed by a flume sink (Figure 3.2). The sink removes the file event from the channel and puts it into the HDFS sink. The file events are removed from the channel only after they are stored in the HDFS repository for reliability. The flume agent configuration is stored in a local configuration file (*/etc/flume-ng/conf/flume.conf*). This is a text file that follows the Java properties file format. Configurations for one or more agents can be specified in the same configuration file. The configuration file includes properties of each source, sink and channel in an agent and how they are wired together to form data flows. Figure 3.7 is the configuration for the Hadoop cluster, where the flume agent is the master node.

```
# Initialize agent's source, channel and sink
agent.sources = SGFExampleDir
agent.channels = memoryChannel
agent.sinks = flumeHDFS
# Setting the source to spool directory where the file exists
agent.sources.SGFExampleDir.type = spooldir
agent.sources.SGFExampleDir.spoolDir =
/usr/local/flumeSGF
# Setting the sink to HDFS repository
agent.sinks.flumeHDFS.type = hdfs
agent.sinks.flumeHDFS.hdfs.path =
hdfs://master/user/flume/sgf
agent.sinks.flumeHDFS.hdfs.fileType = DataStream
# Write format can be text or writable
agent.sinks.flumeHDFS.hdfs.writeFormat = Text
# Connect source and sink with channel
agent.sources.SGFExampleDir.channels = memoryChannel
agent.sinks.flumeHDFS.channel = memoryChannel
```

Figure 3.7: Flume configuration file *flume.conf* that defines how the source, sink and channel are wired together to form the data flows.

3.5.3 Hadoop Platform

In this illustration, the Cloudera Manager Hadoop distribution is used for simplicity. The reason for using such distribution is to provide an easy development environment for practicing users that are not familiar with CentOS operating system to develop similar tools for their demanding smart grid applications. Cloudera Manager automates the installation of the essential configuration, debugging, SQL database, CDH agents, and other components.

During the installation, the cluster nodes are specified using the IP address and hostname (Figure 3.4). It should be noted that an existing link between the cluster nodes should exist. This has been completed by establishing the SSH secure encrypted link. Also, the cluster nodes are assigned master and slave roles, and there can be more than one master node in the cluster. The nodes are assigned roles that run on them. For example, nodes that perform MapReduce tasks are specified, and nodes that perform a Hive task are also specified. A cluster node may be specified to perform more than one task. The status and usage of nodes can be monitored through Cloudera Manager. This can assist decision makers in adding/reducing the number of nodes in the cluster and monitoring the reliability of nodes.

Identifying the master and slave nodes was completed during the setting-up of CDH for the cluster. In the previous step, Flume ingested the data into the HDFS repository. HDFS manages storage on the cluster by breaking the incoming files into blocks, and storing each of the blocks redundantly across the slaves. In the common case, HDFS stores three complete copies of each file by copying each block to three different nodes for reliability (Figure 3.8). Each block size is 128 MB by default, and can be changed to meet the application demand. However, decreasing the block size could lead to a huge number of blocks throughout the cluster, which causes the master node to manage an enormous amount of metadata.

The CDH processing component, YARN, takes advantage of this data distribution by distributing the work involved in a task to many different nodes in the cluster. Each of the nodes runs the task on its own block of the file (Figure 3.9). The results are collated and digested into a single result after each involved block has been processed. The CDH monitors jobs during execution, and will restart work lost due to node failure if necessary.

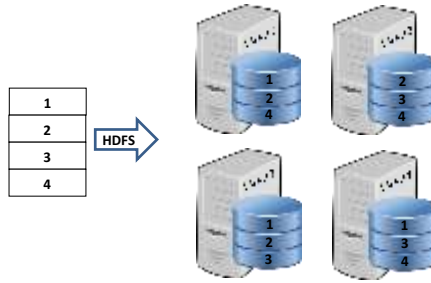


Figure 3.8: HDFS distributes file blocks among cluster nodes.



Figure 3.9: CDH distributes the work out to the nodes.

3.5.4 Hive

Hive facilitates reading, writing, and managing the data stored in the HDFS repository using an SQL-like interface. In this framework, Hive reads the smart grid data file from the HDFS repository and generates a data table of interest. In this implementation, it is desired to build a table that includes only the time stamp, smart meter ID, and consumption of the smart meter. The SQL-like query can achieve this (Figure 3.10):

```
CREATE EXTERNAL TABLE ConsumptionsTable (
  Datetimes TIMESTAMP,
  ID BIGINT,
  Cons FLOAT )
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LOCATION 'user/flume/sgf';
```

Figure 3.10. The Hive SQL-like query to build a table that includes the time stamp, ID and consumption of the smart meters.

This query will read data from 'user/flume/sgf' (the path where Flume sinks the data in Figure 3.7), and anytime new data is ingested into this directory, the “ConsumptionsTable” table will be updated automatically.

3.5.5 Impala

Similar to Hive, Impala is able to read, write, and manage the data stored in the HDFS repository using SQL-like queries. However, Hive and Impala differ in the way they function and how SQL-like statements are written. The following query is an example that deals with a single house smart meter data, including power generated from solar panel (pvpower) and a residential wind turbine (windpower). The query creates a table that includes the timestamp, consumption, pvpower, windpower and zip code (Figure 3.11). The “Invalidate metadata” statement is required after a table is created, to update the metadata. To respond to queries, Impala must have current metadata about the data and tables that clients query directly. Therefore, if the data table used by Impala is modified, the information cached by Impala must be updated. In cases where a delay of data occurs due to power outages, Flume will ensure the data is stored into the HDFS storage, and Impala/Hive will update the tables when the data becomes available in “user/flume/sgf”. Adding the “ORDER BY” statement will reorder the data based on the timestamp in this case.

```
CREATE EXTERNAL TABLE SingleHouse (  
  datetimes TIMESTAMP , Cons float, pvpower float,  
  windpower float, Zip STRING )  
ROW FORMAT DELIMITED  
FIELDS TERMINATED BY ','  
LOCATION 'user/flume/sgf';  
Invalidate metadata;  
ORDER BY (datetimes) DESC;
```

Figure 3.11: The Impala SQL-like query to build a table that includes the timestamp, consumption, pvpower, windpower and zip code.

3.5.6 Visual Analytics

In this implementation, the Tableau software [73] is used for the smart grid big data visual analytics. The Tableau presents interactive data visualizations by means of SQL queries. Here it is desired to send SQL queries to Hive/Impala to build a visualization of data of interest. In order to accomplish this, a connection between Tableau and Hive or Impala has to be established. Open database connectivity (ODBC) interface allows applications to access data in database management systems (DBMS) using SQL as a standard for accessing the data. The remaining machine running Windows operating system is used to setup Tableau and install Hive/Impala ODBC drivers [76]. In the Tableau software, a Hadoop server is chosen to connect to, and the machine and port are determined. The IP address or hostname of one of the machines that run Hive/Impala is entered. For example, to connect to the master node, the IP address should be 10.240.0.2 (from Figure 3.4). In the port field, the port is the number of the transmission control protocol (TCP) port that the Hive/Impala server uses to listen for client connections. In CDH, the Hive TCP port is 10000, and the Impala TCP port is 21050. Once a connection is established, the desired HDFS data can be reached. The visualizations in Tableau are built by sending SQL-like Hive/Impala queries generated by Tableau. The Hive/Impala perform on top of the Hadoop platform which allows vastly improved speed on big datasets for prompt visual analytics.

3.6 Practical Applications of the Framework

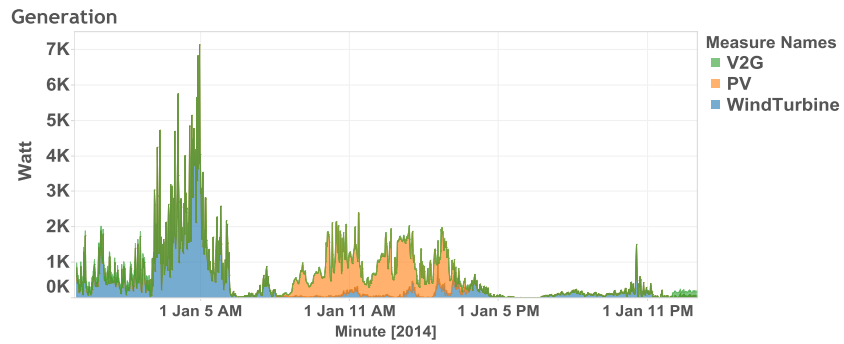
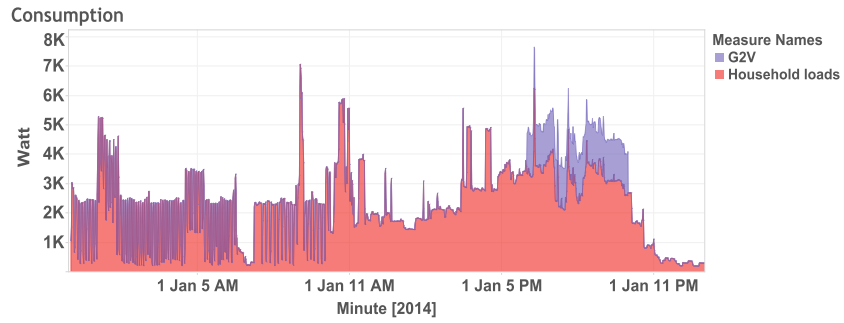
The framework described in Section 3.4 can be applied to manage energy for a single-house, neighborhood or the entire grid. In this section, the cloud platform and Hadoop cluster to implement the proposed framework are presented. Furthermore, the application of the framework is applied to two scenarios. In the first scenario, the framework is applied on a single-house to manage its power usage, to save power and contribute to a smooth and efficient functioning of the smart grid. In the second scenario, the framework is applied on a recently available smart metering dataset that consists of 6,436 homes and businesses. Based on the requirements of the framework, such as cost-effectiveness, improved accessibility, and scalability, Infrastructure as a Service (IaaS) clouds [77] are appropriate to use in implementing the framework. An advantage of using such service is that the data can be sent from any location as long as there is an Internet

connection and the security protocol [78] of the cloud allows it. The framework was hosted on a Google cloud platform that consists of six machines. A Hadoop cluster is setup on five machines with one master node and four slave nodes. The components that are needed to implement the framework (i.e., Flume, Hadoop, Hive, Impala) are setup on the cluster. The master node is a 2.6 GHz, 7.5 GB RAM running 64-bit Linux operating system. All slave nodes were 2.6 GHz, 3.75 GB RAM running 64-bit Linux operating system. The remaining machine was a 2.6 GHz, 3.75 GB RAM running 64-bit Windows operating system to run Tableau to perform the visualization tasks. A secure encrypted link between the cluster nodes was setup using the SSH protocol [78].

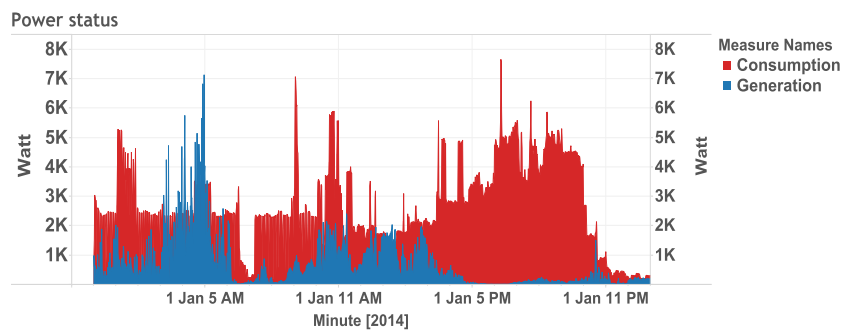
3.6.1 Single-house Application

In the first scenario, beside the typical household appliances that consume power, this house includes micro-power generators namely, a residential wind turbine and rooftop photovoltaic (PV) solar panels. Furthermore, an EV is included in the scenario, as EVs are a subject undergoing intense study in smart grids. The household electric power consumption data was obtained from the UCI data repository [79]. It includes the global active power and three sub-meterings. The first sub-metering corresponds to the kitchen, containing mainly a dishwasher, an oven, and a microwave. The second sub-metering corresponds to the laundry room, containing a washing machine, a dryer, a refrigerator and a light. The third sub-metering corresponds to a water heater and an air-conditioner. To calculate the power output of the wind turbine and PV solar panels, the wind speed, temperature and irradiation data were obtained from [80] with the latitude of 39.74°N and longitude of 105.18°W. Due to the lack of datasets that include the power consumption and micro-power generators' data, it is assumed that the house exists in a location where the data of micro-power generators is available. For that, it is assumed that the house is located with the latitude of 39.74°N and longitude of 105.18°W, which is the same coordinates of the data of the wind speed, temperature, and irradiation. The wind turbine considered was a 3 kW residential turbine. The wind turbine's power output was calculated using the weather data (i.e., the wind speed and air density) of the specified location and the wind turbine's data sheet [81] put in the formulas of [82]. The number of rooftop PV solar panels [83] was ten, and the method to calculate the power output with respect to the weather data (i.e., ambient temperature

and solar irradiation) for the location is described in details in [84]. The EV charging (G2V) profile was obtained from a study of EV driver recharging habits in the north east of England [85], and the EV discharging (V2G) habit was obtained from [86] as it suggested that 10% of the EV's energy can be discharged into the network. Figure 3.12(a) illustrates the consumption and generation of the house with a one-minute time resolution using the Tableau visualization tool. While Figure 3.12(b) illustrates a dashboard for the power status of the aforementioned house updated every one minute. The consumption of power (household loads and G2V) is in red color whereas the aggregated generation from the PV panels, wind turbine and V2G is in blue (Figure 3.12(b)). A pie-chart can be added on the location of the houses to observe the average consumption/generation power (Figure 3.12(c)). This can be applied to other locations on the map if data were available.



(a)



(b)



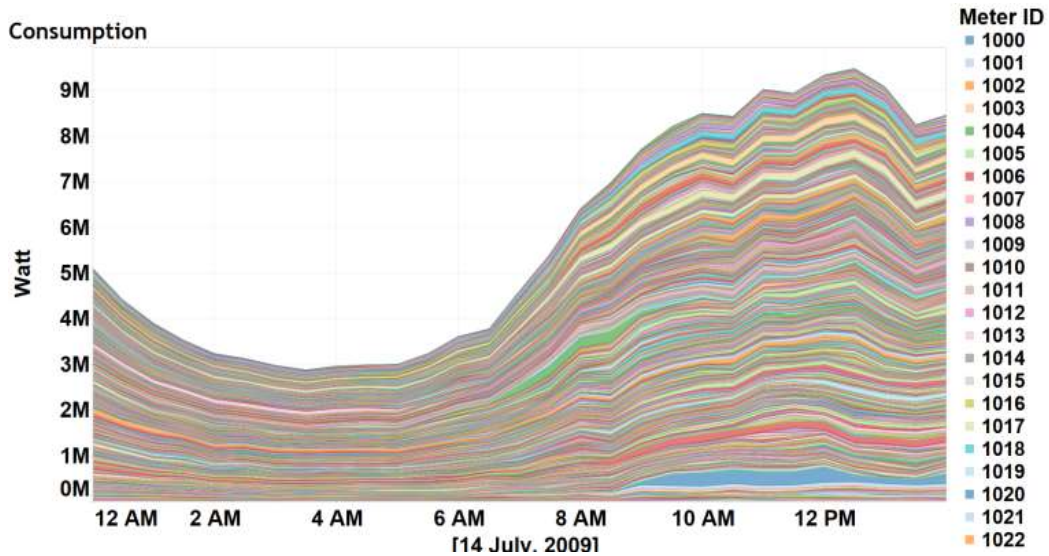
(c)

Figure 3.12: Dashboards for power status. (a) cumulative consumption and generation with one-minute time resolution. (b) power status of the house. (c) Map with pie-chart for consumption (red) and generation for the house.

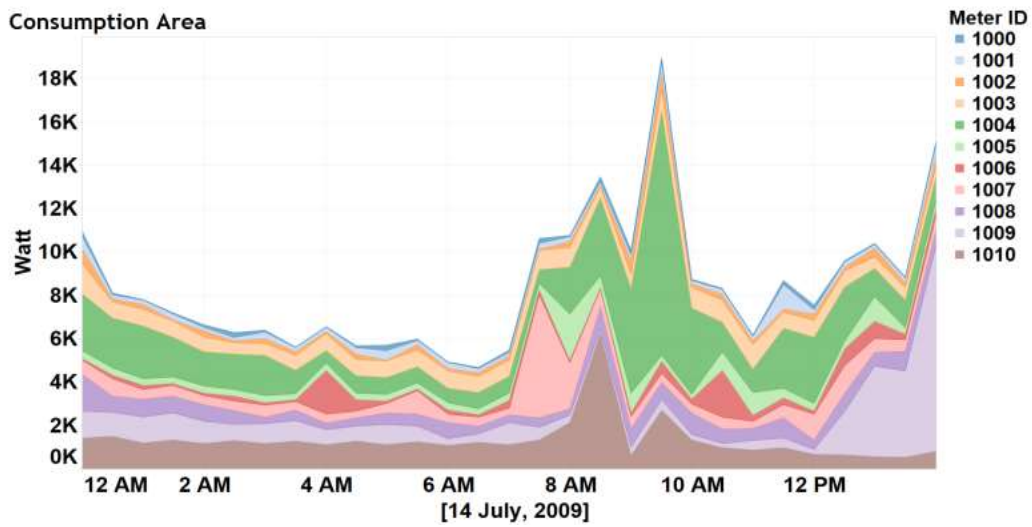
3.6.2 Smart Grid Application

Due to the lack of lengthy datasets similar to the data used in the first scenario that consider micro-generators, in the second scenario a large smart grid data set is considered. The reason for applying the first scenario was to show that micro-generators and EV systems data can be included in the framework. In the second scenario, it is also desired to test the efficiency of the framework in handling lengthy smart grid data.

In the second scenario, a smart metering electricity behavior dataset from the Irish Social Science Data Archive [87] that took place from July 2009 to December 2010 for 6,436 Irish homes and businesses with a 30-minute time resolution is used to test the feasibility of the framework. Each smart meter produced around 25,730 electricity consumption time-series readings during the mentioned period. This corresponds to over 165 million electricity consumption readings to be ingested. The data generation rate of each smart meter was 30-minutes. Thus, 6,436 smart meter data observations were ingested every 30-minutes. Each observation contains the timestamp, smart meter ID, and electricity consumption. Utility companies may have access to additional data about their customers, e.g., location and square footage of the home. However, this information is usually not available to third-party applications. The data were individually ingested to the master node in the Hadoop cluster using Flume. Once the data are collected, it is stored into the HDFS infrastructure. The SQL queries to read and arrange tables from the electricity consumption of the 6,436 smart meter readings are run in Hive and Impala. As Hive and Impala differ in the way they function, it was desirable to observe which data querying component is able to perform faster. In this Hadoop cluster, both Hive and Impala were able to produce/update the table with the new 6,436 smart meter data readings (i.e., the timestamp, smart meter ID and electricity consumption from each smart meter every 30-minutes) in a comparable amount of time (less than one second).



(a)



(b)

Figure 3.13: Dashboards for power status in the smart grid. (a) Power consumption of 6,436 Irish home and businesses updated every 30-minutes. (b) Power consumption of 11 selected smart meters.

In this application, the focus is to develop a dashboard to visualize the status of the smart grid for DDR purposes. To achieve this, Tableau visualization software was used. The Tableau software connects with the Hadoop cluster through Hive or Impala queries to achieve near real-time visualization. Here Impala was able to outperform Hive in updating the visualization of the smart grids status. This suggests that Hive can be suitable for big data batch processing, whereas, Impala can satisfy the requirement of near real-time big data processing. Figure 3.13(a) presents a dashboard for the status of the on-hand smart grid with the aggregated consumption of 6,436 Irish homes and businesses. Consequently, a DR strategy by analytics can be suggested during peak periods. Moreover, dynamic power pricing and incentives for reducing loads during peak periods can be advertised for. The locations of the smart meters' can be visualized on a map similar to Figure 3.12(c). Also, a view of the power consumption for a certain region or specific smart meters can be obtained (Figure 3.13(b)) for further analysis.

3.7 Conclusion

This chapter presented a big data framework for smart grids. The concept of big data and the core components of the framework were highlighted. The framework's stages including, data acquisition, data storing and processing, data querying, and data analytics components were discussed in details. Furthermore, the functionality of the Hadoop platform and the features that make it suitable for the smart grid big data management and analysis were highlighted.

To verify the effectiveness of the framework, the framework was implemented on a cloud-based platform. Furthermore, the application of the framework was applied to two scenarios. The first scenario was a single-house that included micro-generators (i.e., wind turbine, PV roof panels and EV). The second scenario included a real smart metering electricity behavior dataset from the Irish Social Science Data Archive for 6,436 participating Irish homes and businesses. The framework was able to acquire, store, process and query the massive amount of data in near real-time. Also, this chapter covered a DDR task, by enabling the smart grid users to share and visualize its information. This can present the following benefits:

- The integration of renewables by using demand-side management to address supply fluctuations is promoted.

- The grid's reliability can be increased by using the customers as a virtual power resource during peak periods (negative demand is equivalent to a positive supply).
- This can elude the need to build new power plants for standby generation, by lowering the peak periods and advertising for incentives for reducing loads.
- The environmental impacts can be limited as monitoring micro-generators are included, especially vehicle-to-grid (V2G) power.

Finally, the impact of the presented framework goes beyond visual analytics, but in this work, the main objective was to introduce a framework that can sufficiently handle the massive smart grid data. The application of the two scenarios and the visualization of the grid's status, suggests that this framework is feasible in performing further smart grid data analytics.

Chapter 4

Lambda Architecture for Smart Grids Big Data Analytics

4.1 Introduction

The previous chapter presented a framework that covers the life-cycle of smart grid big data from generation to analytics. That framework utilized state-of-the-art big data components to address the smart grid big data challenges. However, the previous presented framework can be developed to scale with big data applications that require real-time updates. For that, in this chapter the framework is developed to comply with Lambda architecture to handle massive quantities of data by taking advantage of both batch and real-time processing methods.

In this chapter the features of utilizing the Lambda architecture for smart grid big data are highlighted. Then the smart grid big data Lambda architecture eco-system is presented. Further, the implementation of the eco-system on a cloud computing platform is presented. In addition, visualization and data mining applications are presented. Finally, the conclusions of the chapter are drawn.

4.2 Features of the Lambda Architecture for Smart Grids

The following features make the Lambda architecture suitable for smart grid big data management and analytics.

4.2.1 Robustness and Fault Tolerance

Computation failures and node breakdown are common in smart grid systems. The Lambda architecture is tolerant to machine failure and data corruption. The batch and real-time views can always be recomputed from the master data. Also, replicates of the data are stored in many nodes, and the computation tasks can be distributed to other healthy nodes in the cluster in case of machine failures or breakdowns.

4.2.2 Low Latency

The Lambda architecture brings parallel computation, and allows achieving real-time read and updating without compromising robustness. This allows smart grid operators to monitor the status of the grid in real-time, in addition, to propose demand-side-management decisions to produce desired changes in the aggregated load shape during rapid imbalances in the smart grid.

4.2.3 Scalability

Layers of the Lambda architecture can be scaled independently. This enables the system to automatically redistribute data and computation tasks to accommodate hardware changes without affecting the functionality of the cluster nodes. Also, this supports installing new smart meters to the grid by adding computation nodes and storage devices to the existing Lambda architecture cluster without affecting the existing infrastructure.

4.2.4 Generalization and Flexibility

The Lambda architecture is able to store and compute views for various types of data from numerous sources, which makes it feasible to be used across a large number of different smart grid big data applications.

4.3 Smart Grid Big Data Lambda Architecture Eco-system

The architecture for smart grid data can be decomposed into five subsequent stages:

- 1- Smart grid data generation.
- 2- Smart grid data collecting.
- 3- Data storing and processing
- 4- Data querying.
- 5- Data analytics.

This smart grid big data eco-system includes a feedback loop that could assist the smart grid operators in observing the results of their decisions on the reliability of the grid. Figure 4.1

presents the smart grid big data cycle based on the Lambda architecture, and the following subsections discuss the stages of this architecture.

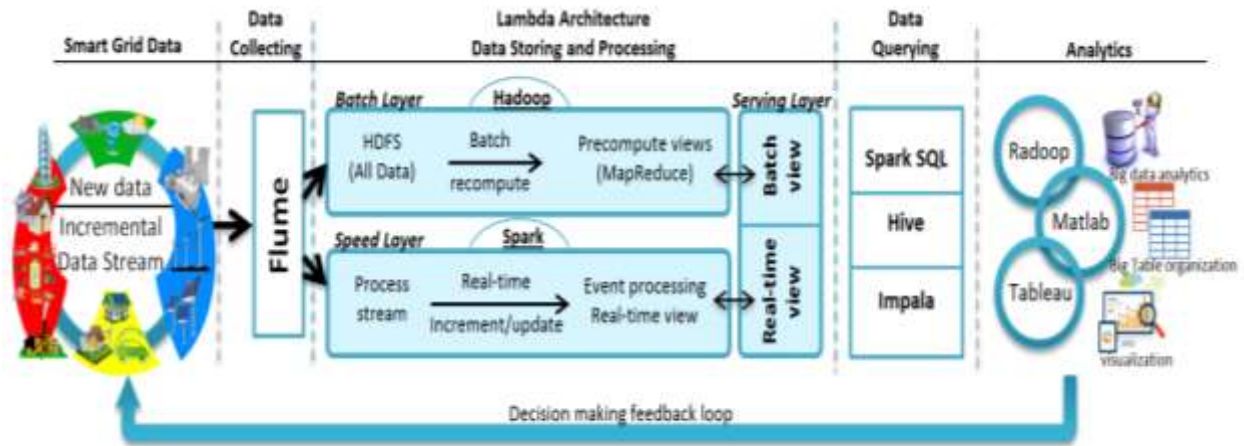


Figure 4.1: The smart grid big data eco-system to deal with the smart grid big data from data collecting to data analytics, with visualization and feedback loop capabilities.

4.3.1 Smart Grid Data

The deployment of smart meters and sensors throughout the grid results in massive amounts of data. This includes generation side data (wind farms and PV plants), consumption side data (residential homes, factories, and EV charging stations), prosumers data (residential PV panels and V2G) and, weather and natural disasters data can be included in the smart grid system. Also, images and video footage could be included to detect physical attacks (California transmission substation sniper attack [88]) or investigate power outages. The smart grid data is considered to be large in volume, high in velocity and wide in variety. The value of this smart grid big data becomes useful when integrated with multi-sourced existing smart grid data in an analytics environment, and can potentially enhance the functionality of the smart grid.

4.3.2 Data Collecting

The multi-source smart grid data generated from the previous layer are to be sent pro-actively to the utility center. To accomplish this, the smart meter data is transmitted to a cloud storage platform through an Internet connection. A reliable tool that is capable of collecting the smart grid data from a single/multisource is Flume [68]. In this smart grid big data eco-system, Flume pulls the data and transfers it to a specific master node in the Hadoop cluster. An advantage of using Flume for data aggregation and transmission is that it ensures the data is stored in the desired final destination. This ensures that the data will be delivered even in cases where disconnections and outages occur. Flume accomplishes this by keeping the actual data in a virtual memory channel until it is completely ingested into the data repository.

4.3.3 Lambda Architecture (Data Storing and Processing)

From the previous layer, the data is sent to a Hadoop master node. The HDFS component in the batch layer of the Lambda architecture eco-system manages storing the data across multiple nodes in the cluster. Also, the batch layer performs its second task, to precompute batch views for this distributed data by using the MapReduce processing component. Meanwhile, the speed layer in the Lambda architecture stores, updates and computes the real-time views of the data collected from Flume. As mentioned in Section 2.3, the Hadoop platform can accomplish the operation of the batch layer of the Lambda architecture. In order to accomplish the functionality of the speed layer of the Lambda architecture in this smart grid big data eco-system, Apache Spark [89], [90] is used. The main feature of Spark is its in-memory cluster computation capability that increases the processing speed of an application. The serving layer of the Lambda architecture merges the results of batch and speed layer computations to provide real-time computation results for the subsequent data querying layer.

4.3.4 Data Querying

The data querying layer includes tools that enable to extract, load and aggregate data stored in HDFS form. In this smart grid big data eco-system, three querying tools are used. Each querying

tool differs in the way it functions and executes various parallel operations on a cluster. Hive [69] uses MapReduce operations to retrieve data. Impala [70] uses the nodes memory to execute the queries. While, Spark SQL [91] is a component on top of Spark in the speed layer of the Lambda architecture that uses resilient distributed dataset, which is a collection of objects partitioned across the nodes of the cluster that can be operated-on in parallel. The reason for including those querying tools is to enable using the suitable querying tool for a specific type of application. For example, for real-time visualization of the grid load, Impala and Spark SQL maybe options to use in this application. However, to compute the amount of power consumed by a specific region during the past year, Hive would be a suitable option to use. Also, including those three data querying components make the smart grid big data eco-system a more compact system.

4.3.5 Analytics

The data analytics is the most important stage in this developed smart grid big data eco-system. The main objectives of this stage are to extract useful information and insights, and assist the smart grid operators in making informed decisions that could essentially promote the reliability and operation of the smart grid. Also, to observe the effects of the decisions made on the smart grid by using the feedback loop. In this eco-system, three analytical tools are presented. These tools can cover data mining and knowledge discovery, statistical and table manipulation, and visual analytics applications. It should be noted that other analytical tools maybe used on top of this eco-system, however, the tools illustrated in this section are able to perform an entire smart grid big data analytical application including, observing the results of decision making using the feedback loop.

4.4 Implementation on a Cloud Computing Platform

In this section the implementation of the smart grid big data eco-system on a cloud computing platform cluster and the method to establish secure connections between the cluster nodes are presented. A hierarchical view of the utilized tools to implement the smart grid big data Lambda

architecture eco-system is shown in Figure 4.2. Further, the settings to connect the analytical tools to the Lambda architecture are highlighted.

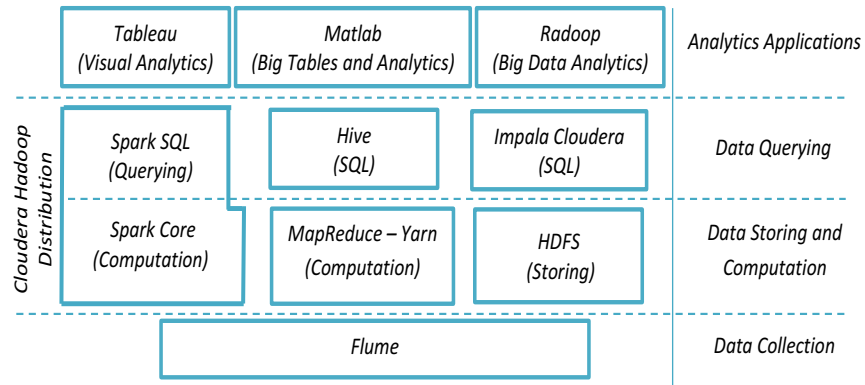


Figure 4.2: Hierarchical view of the utilized components to implement the smart grid big data eco-system.

The IaaS cloud computing platform can meet the smart grid big data requirements, which provide reliability, scalability, cost-effectiveness, and service provisions for hardware services such as, virtual machines and storage. For this implementation, the Google cloud computing is used to host the eco-system. Six machines were used to build-up the eco-system cluster. This six node cluster consists of one master node and five worker nodes. The master node is 8 vCPUs 30 GB RAM machine and the worker nodes are 4 vCPUs 15 GB RAM machines, all running 64-bit Linux operating system. To establish a secure encrypted connection between the cluster nodes, the secure-shell (SSH) [75] connection is used.

Once a secured connection is established, the various types of smart grid data are pro-actively sent to the IP address of the master node. The advantage of using a cloud computing platform is that the data can be sent from any location through an Internet connection. The data collection component, Flume, is setup on the master node to actively poll for data, and is responsible to sink the data in the HDFS repository. To organize the data present in the repository, multiple Flume agents are setup. For example, residential smart meter and power plant generation data are

stored in separate directories. This prevents “Data Swamp”, which is a phenomenon in data lakes that occurs due to large volume of unorganized ingestion of data, from happening.

To implement the Lambda architecture the following components were setup on the cluster nodes:

- Hadoop platform: includes the storing component HDFS and the processing components MapReduce and Yarn.
- Spark Core [89]: contains the basic functionality of Spark, including components for scheduling, memory management, fault recovery, interacting with the storage system.
- Spark SQL to query the data [91].
- Hive: facilitates reading, writing, and managing the data stored in the repository using queries.
- Impala Cloudera: to read, write, and manage the data stored in parallel on the node’s memory.

In order to install and configure the above components, Linux and programming background experience is required. Alternatively, Hadoop open-source distributions can be used, such as, Cloudera distribution of Hadoop (CDH) and Hortonworks Hadoop distribution. Those distributions include most of the smart grid big data components that are used in this eco-system, and the average user is able to configure the settings. It should be noted that using the CDH distribution is more convenient; also, it includes the Impala querying tool.

To perform data analytic applications on top of the smart grid Lambda architecture eco-system, a machine that runs Windows operation system is used. The RapidMiner Radoop [92], Matlab [93] and Tableau [73] tools are installed on this machine. Then a remote connection between those analytic tools and the cluster through the master node is established. This connection allows access to the data and run queries using Spark SQL, Hive and Impala. Those connections are established by using Open Database Connectivity (ODBC) drivers [76] and configuring the TCP port. For example, the TCP ports in the CDH distribution for Hive and Impala are 10000 and 21050, respectively. In this smart grid big data eco-system, Radoop is used for data mining, Matlab tall arrays for table organization, and Tableau for visual analytics. Matlab tall arrays provide working with data backed by a distributed data store. It should be

noted that Radoop and Matlab can be used for many other applications, such as data mining and statistical applications.

Once the eco-system has been built, and a remote connection to the cluster exists, analytical applications can be performed on top of the eco-system.

4.5 Practical Applications of the Smart Grid Big Data Lambda Architecture

In this application of the smart grid big data eco-system it is desired to perform an unsupervised data mining application and visualization of smart grid data. The data mining application is clustering the residential customer daily loads, and visualizing the residential load consumption of the smart grid. The utilized smart grid data is from the Pecan Street Dataport [94]. This data includes the smart meter data for 359 real households in Texas, Colorado, and California recorded every ten minutes. The following subsections present the steps to accomplish an analytics task on top of the smart grid big data Lambda architecture eco-system.

4.5.1 Storing and Organizing the Data

Every ten-minute, Flume ingests the smart meter data into the batch layer and speed layer. This data includes the timestamp, ID, and load of the smart meter. For the specific application of clustering the daily residential loads, Matlab accesses this data through Spark SQL and organize it into daily loads of tall arrays. This corresponds to 144 ten-minute load observations per day from each smart meter. The Flume file and commands to configure Spark for Matlab, and read/store into HDFS are presented in Appendix A and Appendix B, respectively.

In order to extract, process and store the data, the Extract, Process, Store (EPS) process [95] is used. This EPS process uses the smart grid data stored by extracting appropriate information, structuralizing it, processing and querying, and storing the data in the desired form. The daily residential loads are stored into the HDFS repository using Hive queries. The results of this step are:

- 1- Real-time ten-minute smart meter readings of the smart grid for visualizing the residential load consumption.
- 2- The daily residential loads for clustering.

4.5.2 Visualization of Smart Grid Loads

The objective of this step is to visualize the residential loads as they are recorded every ten minutes from the smart meters in real-time. Differently from the visualization task in the previous chapter where Tableau was connected to the distributed smart grid data through an Impala connection, here the connection and visualization are run through Spark. The disadvantage of using Impala is that the recent smart meter data is first stored in the HDFS repository in the batch layer then can be accessed. This causes delays in the visualization especially when the number of smart meters within the smart grid is high, accordingly real-time visualizations of the smart grid loads cannot be achieved. However, in the Lambda architecture, this delay is overcome by introducing the speed layer for real-time access to the recent data. This Spark connection to Tableau allows the smart grid operators to monitor the loads in real-time. The result of this step is a real-time visualization of the residential loads (Figure 4.3). It is worth noting that the visualizations may be shared with customers through online dashboards to monitor the status of the smart grid, which makes them an active component in the reliability and operation of the grid.

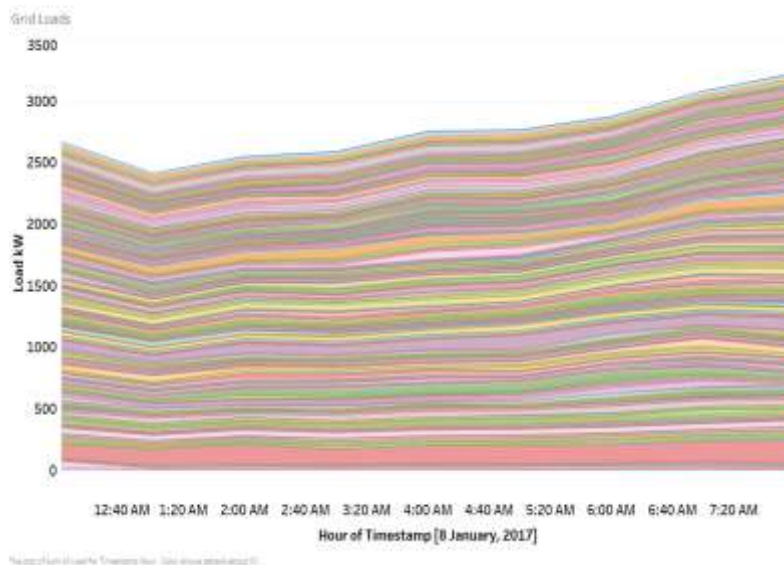


Figure 4.3: Energy consumption observation for the residential customers for Jan. 8, 2017.

4.5.3 Clustering Residential Customer Daily Loads

In this step of the application, it is desired to partition the residential customers into groups based on the shape of the load consumption on top of the smart grid big data eco-system. To accomplish this big data mining task, the Radoop tool is configured to connect to the cluster nodes through Spark. The Radoop process to preprocess the data and group the residential customers is presented in Figure 4.4. The first step in this process is to retrieve the daily residential customer loads from the distributed HDFS repository. This is achieved by the serving layer which merges the results of the batch and speed layers to compute views of the data. Once views of the desired data are available, it is necessary to preprocess the retrieved data, such as dropping daily loads with missing values and selecting the load attributes and customers to cluster. After the data is preprocessed, a suitable clustering algorithm can be applied to the data. It should be noted that the chosen clustering algorithm should be designed to be applied in parallel on distributed data. Data mining algorithms that run parallel on distributed data are still narrow and are a topic that is under research. In this application, the well-known K-means clustering algorithm is used to partition the residential customers into groups based on their load profiles. The results of applying the K-means algorithm are stored back into the HDFS repository for further analysis. This formation of customer groups based on the load consumptions can assist the smart grid operators in the tariff formation process. Also, those clusters can present detailed knowledge of the consumption nature to promote demand response programs. In this clustering application, the customers are grouped into five clusters. It should be noted that choosing the number of clusters is beyond the scope of this application. In this smart grid big data eco-system, Tableau, and Matlab can be used for collecting and visualizing the results of the K-means algorithm. Figure 4.5 shows the clusters' representatives for January 2017. It can be observed that customers in cluster# 4 have abnormal load consumptions when comparing it with the other clusters. This cluster has higher load consumption profiles, and may be of interest to apply demand response strategies. The smart grid operators may decide to target those customers to promote the reliability of the smart grid or reassign tariff formation. Figure 4.6 shows a zoomed view of cluster# 4. It should be noted that this data mining application can be applied to a larger number of customers, and as the scale of data is larger, more accurate knowledge and

insights maybe achieved that could essentially assist the smart grid operators in decision making and promoting the reliability of the smart grid.

To test the robustness of the presented eco-system in situations where cluster nodes breakdown or are disconnected, in the first case two nodes are randomly disconnected from the cluster, then the clustering of the daily residential loads application is run on those four remaining nodes. The remaining four nodes were able to retrieve the data and present the same results. However, the execution time of the Radoop process increased. This was accomplished by the storing of smart grid data in multiple nodes and distributing the computation work to other healthy nodes in the cluster. Further, the same application was run on three nodes; this includes a master node and two worker nodes. The same Radoop clustering results were successfully achieved. This suggests that the presented smart grid big data Lambda architecture eco-system is robust to network outages and node failures. The execution times for the robustness tests for six, four and three nodes are presented in Table 4.1. The success of a test in Table 4.1 indicates that the required data to perform the clustering of residential daily loads were retrieved from the healthy nodes in the cluster.

The main objective of this data mining application was to test the feasibility of the smart grid Lambda architecture eco-system in performing data analytical applications on large-scale distributed smart grid big data. The results obtained from this application suggest that various data analytical applications can be applied on top of the presented eco-system.

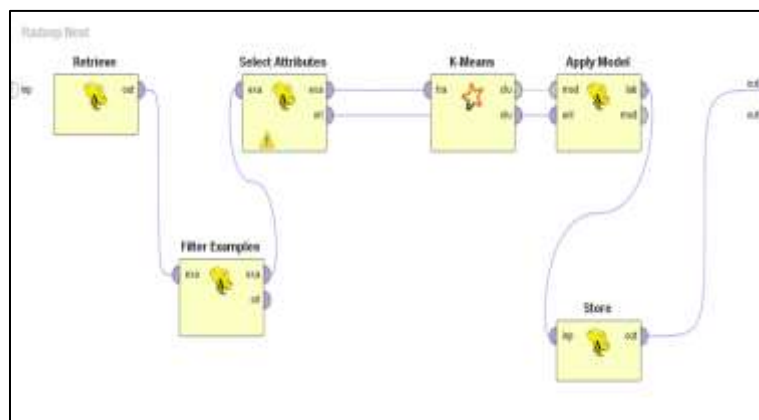


Figure 4.4: Radoop RapidMiner nest process to preprocess the data, apply the data mining clustering K-means algorithm and store the results into the HDFS repository.

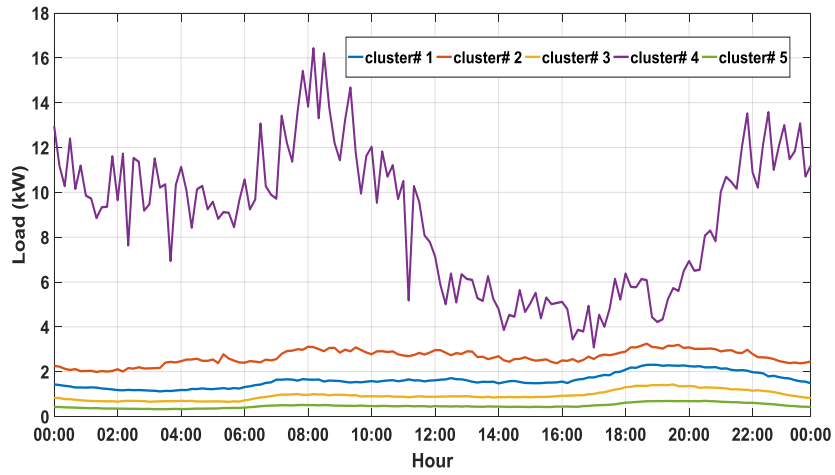


Figure 4.5: Visualization of the five cluster representatives of the K-means clustering algorithm.

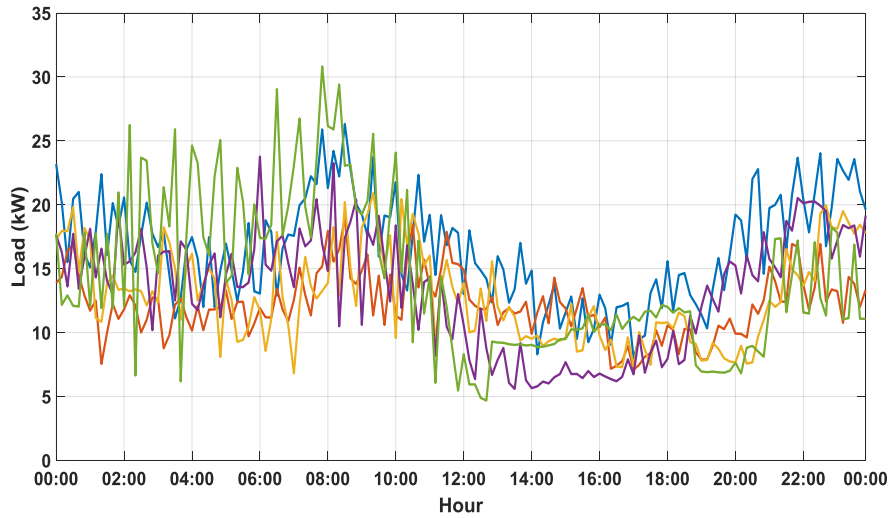


Figure 4.6: Zoomed view into cluster# 4 as it represents abnormal or interesting load consumption behavior.

Table 4.1 Robustness Test Execution Times for the Smart Grid Big Data Eco-System on Clustering Residential Customer Daily Loads

Number of cluster nodes (including one master node)	6 nodes	4 nodes	3 nodes
Success	✓	✓	✓
Radoop process execution time	0.27 sec	0.59 sec	1.29 sec

4.6 Conclusions

This chapter presented a smart grid big data eco-system based on the Lambda architecture. The Lambda architecture design and principals for building batch and real-time processing systems were discussed. This eco-system is able to handle massive quantities of smart grid data by taking advantage of batch and real-time processing methods. Furthermore, this eco-system collects then stores the smart grid big data into a cloud. This allows collecting various types of smart grid data including smart meter data, and image and video data to enable data mining in digital image and video processing applications.

The presented eco-system was implemented and setup on a cloud computing platform. Furthermore, data mining and visualization applications on real smart grid data were performed. The data mining application was to partition the daily smart meter readings into groups based on the load consumption. In the visualization application, the presented eco-system was able to overcome the delay in real-time visualization of the previous smart grid big data frameworks by utilizing the Lambda architecture. In addition, the robustness tests carried out proved the robustness of the eco-system in cases of network outages or node failures.

Due to the lack of a mature tool that can perform data analytic applications, this eco-system utilized a combination of analytic tools that were sufficient to perform the desired applications of visualization and data analytics. Also, it is worth noting that other analytic tools can be used on top of the eco-system, and as data mining algorithms develop to work parallel on distributed data, they can be applied on top of the presented eco-system.

Chapter 5

Unsupervised Non-intrusive Extraction of Electrical Vehicle Charging Load Patterns (EEVCLP)

5.1 Introduction

The previous chapter presented an eco-system that is capable of collecting and storing the smart grid data, including the smart meter data, for further analysis. With the rapid popularization of EVs and the issues it introduces to the electrical grid, in this chapter it is desired to extract the EV charging loads (EVCLs) from the aggregated smart meter daily loads of residential households. Extracting and aggregating those EVCLs is essential to allow smart grid operators to make intelligent and informed decisions about conserving energy and promoting the reliability of the grid.

In this chapter, an unsupervised algorithm to extract the EVCLs non-intrusively from the smart meter data is proposed. The proposed algorithm can run on low-frequency smart meter sampling data and only requires the real power measurement, which is the type of data communicated and recorded by most smart meters. Furthermore, validation results and case studies are shown.

5.2 Theoretical Background of ICA for Extracting EVCLs

ICA [96], [97] is a signal processing technique whose goal is to express a set of random variables as linear combinations of statistically independent component variables. One of the main applications of ICA is in blind source separation [97]. In the basic form of ICA [96], $\mathbf{x} = [x_1, \dots, x_m]$ is a random time-varying observed signal vector, and likewise $\mathbf{s} = [s_1, \dots, s_m]$ is a random vector with the original signal elements. Then the linear relationship is given by:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \quad (5.1)$$

where \mathbf{A} is an unknown $m \times n$ mixing matrix to be estimated.

The statistical model in (5.1) is called the ICA model. The ICA model is a generative model, which means that it describes how the observed data are generated by a process of mixing the components \mathbf{s} . The independent components are latent variables that cannot be directly observed. Also, the mixing matrix \mathbf{A} is assumed to be unknown. The only observation is the random vector \mathbf{x} , and it is desired to estimate \mathbf{A} and \mathbf{s} using it. To obtain this, it is assumed that the components s_i are statistically independent. Also, the independent component has non-gaussian distributions. However, in the basic model, these distributions are not known (if they are known, the problem is considerably simplified). For simplicity, it is assumed that the unknown mixing matrix \mathbf{A} is square. Then, after estimating the matrix \mathbf{A} the inverse can be computed (\mathbf{W}), and the independent components are obtained by:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \quad (5.2)$$

Before applying the ICA algorithm on data, it is useful to do some preprocessing, to make the problem of ICA estimation simpler and better conditioned. The used preprocessing techniques including centering and whitening are presented in Appendix C.

Although ICA separates the signals, the exact amplitude and sign of the independent components cannot be determined. This is considered to be a drawback in applications where the amplitudes of the signals are desired.

5.3 Independent Component Analysis for Extracting Electric Vehicle Loads

In the context of applying the independent component analysis (ICA) for extracting the steady charging load (*stage2*) of EVs, \mathbf{x} is the observed aggregated signal composed of two mixed signals. The first is the aggregated signal without the EVCL signal, and the other is the EVCL signal. Figure 5.1 illustrates the problem of the ICA in the context of extracting EVCLs from a mixture of aggregated appliance signals. Initially, it is assumed that the amplitude of the EV signal is known. This means that one independent distribution is known and, hence, this simplifies the problem. Accordingly, the ICA model can act as extracting the distribution pattern of the EV load from the observed aggregated signal. Although the EVCL can be extracted from the aggregated signal using the ICA method, the amplitude and sign of the extracted EVCL

cannot be determined. This is due to the unavoidable ambiguities of ICA. For that, the grouping of EVs in categories will assist in estimating the amplitude of the extracted EVCL signal in Section 5.3. For that, various EV charging amplitudes have been collected. Table 5.1 presents some of the collected charging amplitudes at *stage2* for existing EVs in global markets [98], including 100% EVs and plug-in hybrid vehicles. The EV signal that the ICA needs to extract is called a template. A template is a row vector that contains N replicates of the EV charging amplitude. The EVs can be grouped into categories based on their charging amplitudes. For example, EVs that have charging amplitudes of 3xxxW are grouped in the same category and EVs that have charging amplitudes of 6xxxW are in the same category, and so on.

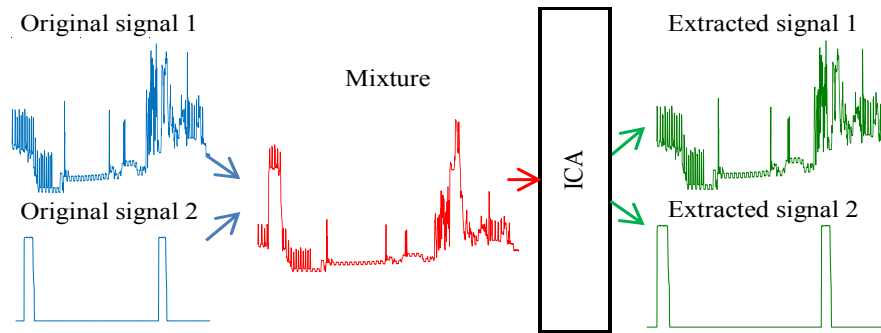


Figure 5.1: The extraction of EVCL from the aggregated load problem. Source1 is the aggregated load pattern without the EVCL. Source2 is the EVCL. Mixture is the aggregated load pattern.

Table 5.1 Stage2 Charging Amplitudes of EVs in Global Markets

Model	Max Charge	Category#
Porsche Panamera S	3 kW	1
Audi A3 e-tron, Cadillac ELR, Chevy Spark, Chevy Volt, Ford Fusion, Hyundai Sonata, Mercedes S550, Mitsubishi i-MiEV, Nissan LEAF	3.3 kW	
Porsche Cayenne S, Volkswagen e-Golf	3.6 kW	
Tesla	6 kW	2
Fiat 500e, Ford Focus, Honda Accord, Kia Soul, Nissan LEAF	6.6 kW	
BMW i3, Volkswagen e-Golf	7.2 kW	3
Tesla Model S, Tesla Model X	10 kW	4

5.4 Proposed EEVCLP Algorithm

5.4.1 General Aspects

This section illustrates the general proposed extraction of EV charging load pattern (EEVCLP) algorithm. The initial data is the aggregated load pattern (ALP) vector $\mathbf{x} = [x_t, \dots, x_T]$, for $t = 1, \dots, T$. The ALP contains T observations with a time-series resolution of one minute ($\tau=1$). The mathematical formulation presented here is described by considering all vectors as row vectors. The flowchart of the EEVCLP algorithm is presented in Figure 5.2 and the parameters are presented in Table 5.2.

The vector $\mathbf{s} = [s_{nm}, \dots, s_{Nm}]$ represents the m -th template to be extracted from the ALP \mathbf{x} . The whole set S of templates is represented by the matrix $\mathbf{S} = [s_m, \dots, s_M]$ for $m = 1, \dots, M$. From Table 5.1, the number of templates $M=7$. Each template s_m contains N points of *stage2* charging power amplitudes to be extracted. In this illustration, the ICA is applied with a time frame window size of $N=10$. The window is then shifted progressively to the next frame until the entire ALP (\mathbf{x}) is covered. For example, if \mathbf{x} contains the daily ALP with one-minute time-steps ($T=1440$), accordingly, the ICA will be applied $T/N = 144$ times to cover the whole sequence of \mathbf{x} . The templates are grouped into categories based on their charging amplitude. For example, EVs that have charging amplitudes of 3xxxW are in the same category and EVs that have charging amplitudes of 6xxxW are in the same category, and so on. This assumption will assist in the estimation of the amplitude of the extracted EVCL in the iterative process (Section 5.4.2).

The extracted EVCL can include more than one extracted charging session (ECS). An ECS consists of the index time t when the EV starts charging ($start_e$) and the index time t when the EV ends its charging session (end_e). The ECSs are registered as couples in a vector $D_e = [start_e, end_e]$ and the entire couples of ECSs for an extracted EVCL are represented in the matrix $\mathbf{D} = [D_e, \dots, D_E]$.

As mentioned previously, in the context of applying the ICA for extracting the EVCL, the observed signal vector is \mathbf{x} and one of the sources \mathbf{s} is assumed to be known. This means that one independent distribution is known and, hence, this simplifies the problem. Accordingly, the ICA

model can act as extracting the distribution pattern of the template s_m from the aggregated power x .

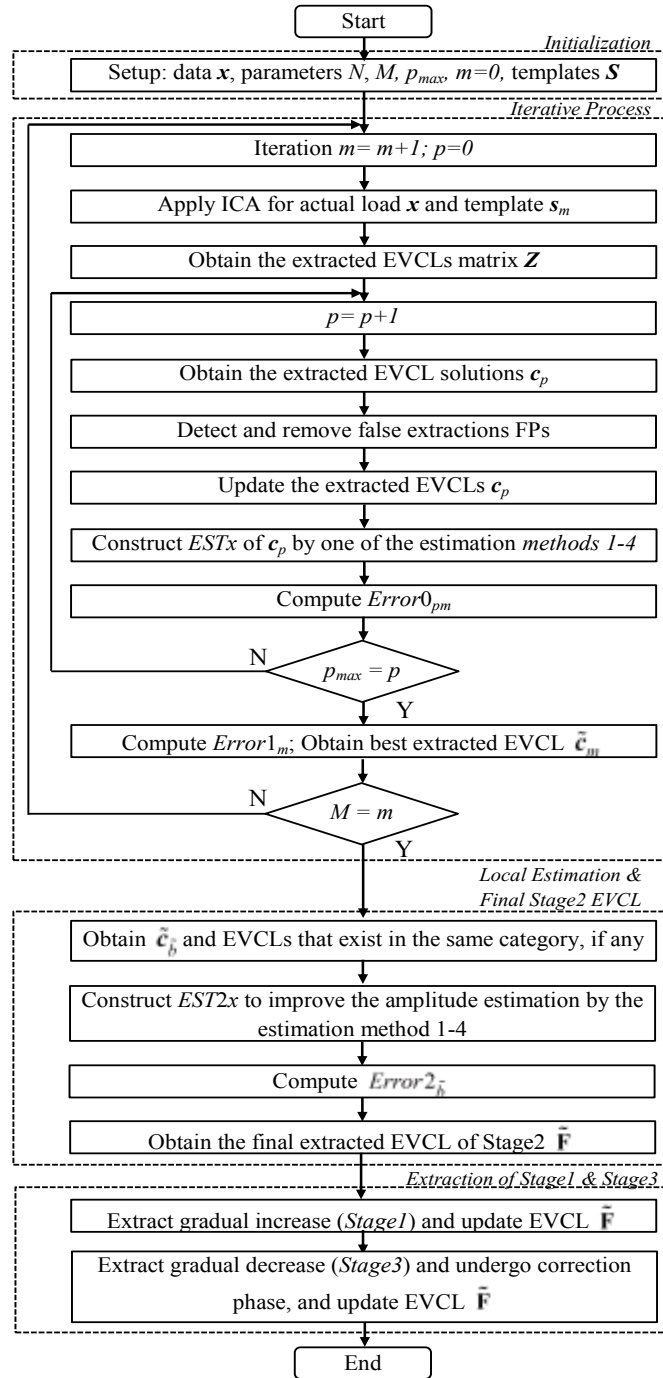


Figure 5.2: Flow-chart of the EEVCLP.

Table 5.2 The Parameters and Descriptions for the EEVCLP Algorithm

\tilde{a}	Index of error1
\tilde{b}	Index of error2
\mathbf{C}	Solution vector
\tilde{g}	Index of overall minimum error
\mathbf{S}	Vector template of maximum charging amplitude
τ	Time resolution
\mathbf{X}	Original aggregated load pattern
ν	Component of <i>stage1/stage3</i>
ν	Pattern window of <i>stage3</i>
\mathbf{Z}	Extracted signal by independent component analysis
B_e	Base load
\mathbf{C}	Matrix of solution vectors
\mathbf{D}	Vector of start and end couple
\mathbf{D}	Matrix for start and end couples
<i>Error#</i>	Error between the reconstructed and original signal
<i>ESTx</i>	Reconstructed load pattern
<i>ESTx2</i>	Reconstructed load pattern to improve the estimated amplitude
$\tilde{\mathbf{F}}$	Final extracted <i>stage2</i> signal
\mathbf{P}	The preceding components
\mathbf{F}	The following components
$start_e$	Start time index of charging session
end_e	End time index of charging session
M	Number of templates
N	Window size of sequence to be extracted
P	Counts of non-zero extracted signals
\mathbf{S}	Matrix of templates s
V	Duration of <i>stage1/ stage3</i>
\mathbf{Z}	Extracted signals matrix of independent component analysis

5.4.2 Iterative Process

Initialization

The initialization includes the setting-up of the number of minimum desired sequence of EVCL to be extracted at each window (N), and the amplitudes and number of templates (M).

Successive Iteration

Application of ICA (Step1): The ICA is applied with a window size of $N=10$ on the entire sequence of \mathbf{x} to extract the m -th EVCL (\mathbf{s}_m). The result is a vector $\mathbf{z}_n = [z_{nt}, \dots, z_{nT}]$, for $t = 1, \dots, T$ that represents the extracted EVCL from \mathbf{x} . The vector \mathbf{z}_n will contain detections of EVCLs if the whole \mathbf{s}_m template window matches. This will lead to missing EVCLs that have not started at the beginning of the window. For this purpose, the ICA is repeatedly applied N times. At the beginning of every repetition n , the window is shifted one position to the right. Thus, ten extracted EVCLs are obtained and stored in an $N \times T$ matrix $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$, for $n = 1, \dots, N$.

Extracting the EVCL vector (Step2): Here a variable p for $p = 1, \dots, N/2$ is introduced. This variable represents the number of non-zero extracted EVCL occurrences in the t -th column of matrix \mathbf{Z} . From each column t of matrix \mathbf{Z} , a solution component c_t , for $t = 1, \dots, T$ is generated. The component c_t is zero, unless there are p non-zero occurrences then c_t becomes:

$$c_t = \begin{cases} \left(\sum_{n=1}^N Z_{nt} \right) / p, & \text{number of occurrences} = p \\ 0, & \text{otherwise} \end{cases} \quad (5.3)$$

The result is an extracted EVCL solution vector $\mathbf{c}_p = [c_{p1}, \dots, c_{pT}]$ and the set of solutions \mathbf{C} is represented by the matrix $\mathbf{C} = [\mathbf{c}_1, \dots, \mathbf{c}_p]$, for $p = 1, \dots, 5$.

Removing false positive extractions of EVCLs (Step3): Due to appliances that may have similar operation cycles and amplitudes as EVs (e.g., A/C units and dryers) the generated solution \mathbf{c}_p can contain false positive (FP) detections of EVCLs. It is desired to shave or at least mitigate these FP extractions. To achieve this, the signatures of household appliances that may cause FP

detections are studied to take advantage of their behavior in reducing the FP detections rate. It was observed that most of the FPs occur from dryers, especially, when s is from Category#1.

The following steps were generated after studying the appliances that interfere, and have similar amplitudes and patterns to EVs.

1- If the length between two consecutive couples of ECSs is:

$$abs(start_e - end_{e-1}) \leq 18 \text{ AND } (abs(start_e - end_e) < 120 \text{ OR } abs(start_{e+1} - end_{e+1}) < 120) \quad (5.4)$$

then remove both couples of ECSs from c_p and update the ECS matrix D .

2- If the length of a couple ECS is $abs(start_e - end_e) < 20$, remove the ECS couple from c_p and update the ECS matrix D .

The latter will lead the proposed algorithm not to extract EVCLs that have charging sessions < 20 -minutes. Such short charging sessions can be neglected when there is a trade-off between accuracy and detecting shorter charging sessions. The state-of-the-art algorithms [39] can only extract EVCLs that have charging sessions > 30 -minutes.

The result from this step is an updated matrix C that has zero/reduced FP detections of EVCLs.

Estimating the amplitude (Step 4): As mentioned in Section 5.2, the ICA algorithm is unable to detect the sign and estimate the amplitude of the separated signal. In the context of EV charging, the sign of the extracted EVCL must always be positive as it consumes power. Accordingly, this drawback of ICA can be overcome. However, the remaining issue of estimating the amplitude of the extracted EVCL needs to be addressed. For this purpose, four estimation methods will be introduced to estimate the amplitude of the extracted EVCLs.

It should be noted that, for EV templates (s) that have greater amplitude than the ALP (x), there will be no ECS detected. For example, if s_7 (from Category#4) is selected as a template, where the amplitude is 10 kW, and the entire ALP (x) has amplitude less than 10 kW, there will be no extracted EVCL (c_p). Accordingly, there is no reason to estimate the amplitude and the algorithm will return zero EVCLs.

In this algorithm, it is assumed that the amplitude can be estimated from the components of x that precede (P) and/or follow (F) a D_e charging couple:

$$P = x(D_e(start_e) - ps, \dots, D_e(start_e) - fs) \quad (5.5)$$

$$F = x(D_e(end_e) + ps, \dots, D_e(end_e) + fs) \quad (5.6)$$

where a duration of 10-minutes that precede and/or follow are considered, $ps = 5$ and $fs = 15$. In a second formation of the algorithm, $ps = 10$ and $fs = 20$. This means that 10-minutes that precede and/or follow the ECS are used to estimate the amplitude of the ECS of the extracted EVCL. The reason behind skipping the components that directly precede (*stage1*) and follow (*stage3*) the ECS is due to the gradient nature of EVs charging pattern behavior during the plugging and unplugging, which could penalize the results of the amplitude estimation methods.

In the following methods the steady amplitude of *stage2* is estimated by computing the load before and/or after an ECS occurs (baseload). The baseload is then added to the extracted EVCL to reconstruct the original ALP (\mathbf{x}). Then the amplitude of the ECS can be the one that has a reconstructed load pattern ($ESTx$) closest to the original \mathbf{x} . This estimation of the amplitude using the baseload ensures that the best chosen EV template s_m does not overestimate the actual EV amplitude. For example, if the ALP exceeds 10 kW and the actual EV charging amplitude is 7 kW (in Category#3), the amplitude estimation using the baseload will ensure that the extracted EVCL will not be overestimated and assigned to a template in Category#4 with an amplitude of 10 kW. Following are four estimation methods; it should be noted that *Methods 1-3* are the same when computing $ESTx$, but differ when computing $ESTx2$ in the following subsection.

Method 1, 2, 3:

For each ECS couple, the baseload (B_e) is computed:

$$B_e = \begin{cases} \max\{\max(P), \max(F)\}, & \begin{array}{l} D_e(start_e) \geq fs \\ \text{AND } T - D_e(end_e) \leq fs \end{array} \\ \max(F), & D_e(start_e) \leq fs \\ \max(P), & T - D_e(end_e) \leq fs \end{cases} \quad (5.7)$$

Then the baseload (B_e) is used to reconstruct an estimation of the original ALP (\mathbf{x}).

$$\begin{aligned}
ESTx &= x \\
ESTx_t &= \left(\begin{array}{l} ((x_t - s_{m1}) / B_e) \times \\ \max(x(start_e, \dots, end_e)) \end{array} \right) + C_{pt} \quad , \quad \text{for } start_e \leq t \leq end_e
\end{aligned} \tag{5.8}$$

where $ESTx$ is a reconstruction of x .

Method 4:

The two-bin histogram ($hb1$, $hb2$) is computed [99] for P and F , and the mean of the components of each bin is computed, then:

$$B_e = \begin{cases} \max\{\text{mean}(hb2(P)), \text{mean}(hb2(F))\} , & D_e(start_e) \geq fs \\ & \text{AND } T - D_e(end_e) \leq fs \\ \text{mean}(hb2(F)) , & D_e(start_e) \leq fs \\ \text{mean}(hb2(P)) , & T - D_e(end_e) \leq fs \end{cases} \tag{5.9}$$

The baseload (B_e) is used to reconstruct the original ALP (\mathbf{x}):

$$\begin{aligned}
ESTx &= x \\
ESTx_t &= ((x_t - S_{m1}) / B_e) \times C_{pt} \quad , \quad \text{for } start_e \leq t \leq end_e
\end{aligned} \tag{5.10}$$

where $ESTx$ is a reconstruction of x .

Now the error between the original x and the reconstructed ALP ($ESTx$) is computed:

$$Error0_{pm} = \sum_{t=1}^T (x_t - ESTx_t) \tag{5.11}$$

where p represents the extracted EVCL solution \mathbf{c}_p and m represents the EV template \mathbf{s}_m .

For the remaining solution vectors $\mathbf{C} = [\mathbf{c}_2, \dots, \mathbf{c}_p]$, the same operations of **Step3** and **Step4** are performed with the shaving of FP extractions, the reconstruction of $ESTx$, and the computation of $Error0_{pm}$. After the $Error0$ has been computed for each p , the minimum error ($Error1_m$) of the extracted EVCL solution vectors \mathbf{C} for iteration m is computed:

$$Error1_m = \min(abs(Error0_m)) \tag{5.12}$$

$$\tilde{a}_m = \arg \min(\text{abs}(\text{Error}0_m)) \quad (5.13)$$

and the related extracted EVCL solution vector c_p and index are stored in matrix \tilde{c} and vector \tilde{a} , respectively. The corresponding c_p is interpolated as the best extracted EVCL \tilde{c} for iteration m , $\tilde{c}_m = [c_{\tilde{a}_m 1}, \dots, c_{\tilde{a}_m T}]$.

For the successive iterations $m=1, \dots, M$, the group of operations are the same as the ones described in **Step1** to **Step4**, this includes the application of ICA, the generation of extracted EVCL solution vectors \mathbf{C} , the shaving of FP extractions, the reconstruction of $ESTx$, the computation of $\text{Error}1_m$, and storing the best estimations of the extracted EVCLs in \tilde{c} .

Before the final best-extracted EVCL and amplitude template are obtained the next stage is an attempt to improve the accuracy of the extracted EVCL amplitudes.

Local Estimation Improvement

In this stage, an attempt to improve the accuracy of the amplitude of the extracted EVCL is presented. From the vector $\text{Error}1$, the minimum index is located:

$$\tilde{b} = \arg \min(\text{abs}(\text{Error}1)) \quad (5.14)$$

then the corresponding best EVCL $\tilde{c}_{\tilde{b}}$ and the EVCLs that exist in the same category, if any, are of interest for further processing. For example, if $\tilde{b}=2$ then the EVCLs \tilde{c}_1 and \tilde{c}_2 are involved as they fall in the same category (Category#1). For each one of these $\tilde{c}_{\tilde{b}}$, the ECSs matrix \mathbf{D} is computed. Then similar to the reconstruction of the original ALP (\mathbf{x}) part in **Step4**, for each couple D_e , the reconstructed load pattern $ESTx2$ is computed by following the same method (*Methods 1-4*) chosen from **Step4**, including the previously reconstructed load pattern ($ESTx$):

$$ESTx2 = x \quad (5.15)$$

Method 1:

$$B2_t = (x_t - ESTx_t) \quad , \quad \text{for } start_e \leq t \leq end_e \quad (5.16)$$

$$ESTx2_t = ESTx_t + B2_t \quad , \quad \text{for } start_e \leq t \leq end_e \quad (5.17)$$

Method 2:

B_e is computed using (5.7)

$$B2_e = \begin{cases} B_e - \left((10^{\text{length}(\text{Error1}_{\tilde{b}})} \right) / (end_e - start_e + 1) \quad , & \sum_{t=start_e}^{end_e} (x_t - ESTx_t) < 0 \\ B_e + \left((10^{\text{length}(\text{Error1}_{\tilde{b}})} \right) / (end_e - start_e + 1) \quad , & \text{otherwise} \end{cases} \quad (5.18)$$

$$EST2x_t = \left(((x_t - \tilde{c}_{\tilde{b}}) / B_e) \times B2_e \right) + \tilde{c}_{\tilde{b}} \quad , \quad \text{for } start_e \leq t \leq end_e \quad (5.19)$$

Method 3, 4:

$$B2_e = \left(\sum_{t=start_e}^{end_e} (x_t - ESTx_t) \right) / (end_e - start_e + 1) \quad (5.20)$$

$$ESTx2_t = ESTx_t + B2_e \quad , \quad \text{for } start_e \leq t \leq end_e \quad (5.21)$$

Now the error between the original x and the reconstructed ALP ($EST2x$) is computed:

$$\text{Error2}_{\tilde{b}} = \sum_{t=1}^T (x_t - EST2x_t) \quad (5.22)$$

The final extracted EVCL (\tilde{F}) with the best-estimated amplitude can be obtained by:

$$\tilde{g} = \arg \min \{ \min \text{abs}(\text{Error1}), \min \text{abs}(\text{Error2}) \} \quad (5.23)$$

$$\tilde{\mathbf{F}} = \tilde{\mathbf{c}}_{\tilde{\mathbf{g}}} \quad (5.24)$$

The final extracted *stage2* EVCL ($\tilde{\mathbf{F}}$) is the $\tilde{\mathbf{c}}_{\tilde{\mathbf{g}}}$ vector that has the minimum error between the reconstructed load pattern and the original ALP (\mathbf{x}). As mentioned previously, this step was to improve the selection of the best estimate from the templates (\mathbf{S}).

5.4.3 Extraction of Gradual Increase in the EVCL (Stage1)

To extract the gradual increase in EVCLs, sufficient *stage1* samples must be studied. Thus the data for 100 EVs from different categories are obtained. Figure 5.3(a) shows a zoomed view of *stage1*. It can be observed that it takes no more than two-minutes ($V = 2$) to reach the maximum charging amplitude for all EVs from different categories. This corresponds to three time observations. The first observation in *stage1* is when the EV is plugged-in, $\tilde{\mathbf{F}}_{start_e-2} = 0$, and the third observation is the maximum charging amplitude, $\tilde{\mathbf{F}}_{start_e}$. In order to compute the second observation, the curve fitting process [100] is used to express the behavior of *stage1*. The general function that computes the second observation, $\tilde{\mathbf{F}}_{start_e-1}$, of *stage1* for all EVs categories can be presented as:

$$\begin{aligned} H(\nu) &= 2970.5 \times \log(\nu) - 69.82, \quad \text{for } \nu = 2 \\ \hat{H} &= (H(\nu) + 69.82) / 3263.4 \\ \tilde{\mathbf{F}}_{start_e-1} &= \hat{H} \times \tilde{\mathbf{F}}_{start_e} \end{aligned} \quad (5.25)$$

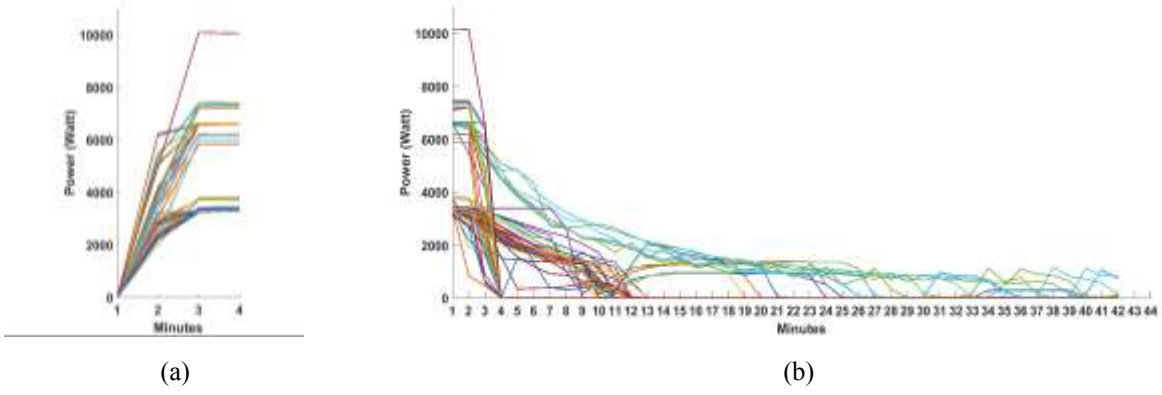


Figure 5.3: Zoomed view for EVs from different categories (a) *stage1* durations. (b) *stage3* durations.

5.4.4 Extraction of Gradual Decrease in the EVCL (Stage3) and Correction Phase

In order to study the behavior of EVCLs at *stage3*, various EVCLs from different categories are obtained. Figure 5.3(b) shows a zoomed view of *stage3* for the gathered data. Now, differently from *stage1*, it can be observed that EVCLs from different categories have different behaviors at *stage3*. Moreover, EVs that belong to the same category have different patterns when ending their charging sessions. EVs from Category#1 and #2 have two *stage3* patterns, while EVs from Category#3 and #4 have one *stage3* pattern.

EVs from Category#1 have two patterns. In the first pattern, it takes up to ten-minutes ($V=10$) to reach zero watts. In the second pattern, it takes two minutes to reach zero watts. The curve fitting process is used to express those two behaviors of Category#1 EVs. For the ten-minute pattern, the first observation is the maximum charging amplitude, $\tilde{\mathbf{F}}_{end_e}$, and the last observation is $\tilde{\mathbf{F}}_{end_e+10}=0$. Each remaining observation, ν , in-between is computed by:

$$\begin{aligned}
 H(\nu) &= -1773.1 \times \log(\nu) - 5164.4, \quad \text{for } 1 \leq \nu \leq 9 \\
 \hat{H}(\nu) &= (H(\nu) + 1081.7) / 4082.7 \\
 \tilde{\mathbf{F}}_{end_e+\nu} &= \hat{H}(\nu) \times \tilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.26}$$

Similarly, the two-minute ($V = 2$) *stage3* pattern of Category#1 can be computed by:

$$\begin{aligned}
 H(\nu) &= -2804.6 \times \log(2) - 3608.4, \quad \text{for } \nu = 2 \\
 \widehat{H} &= (H(\nu) + 6689.6) / 9395.9 \\
 \widetilde{\mathbf{F}}_{end_e+1} &= \widehat{H} \times \widetilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.27}$$

The algorithm begins extracting *stage3* with assuming $V = 10$. During the extraction of *stage3*, the algorithm performs a correction phase. At this phase the resulted *stage3* pattern window $\nu = (\nu_1, \dots, \nu_V)$ is placed at the end of the ECS, which corresponds to $x_{end_e}, \dots, x_{end_e+V}$. If the pattern ν matches the ALP ($\partial = 1$):

$$\partial = \begin{cases} 1, & x_{end_e} - x_{end_e+V} = \widetilde{\mathbf{F}}_{end_e} \\ 0, & otherwise \end{cases} \tag{5.28}$$

Then the end of the charging session was determined adequately, but if the pattern ν doesn't exist ($\partial = 0$), this means that the duration of the ECS was overestimated due to overlapping with other appliances, and the correction phase is entered. At this phase the window, ν , is shifted one position backwards until the condition $\partial = 1$ is satisfied. If the window ν reaches the start of the ECS, i.e., $\nu_1 = x_{start_e}$, and $\partial = 0$. This means that the EVCL doesn't match the 10-minute pattern and is likely to match the 2-minute pattern of Category#1.

Now similarly, the algorithm begins extracting *stage3* with assuming $V = 2$. During the extraction the correction phase is undergone. The resulted pattern window ν is placed at the end of the ECS, i.e., $\nu_1 = x_{end_e}$, and the window is shifted one position backwards until $\partial = 1$. It should be noted that, the correction phase could correct the duration of an ECS to under 20-minutes. However, removing the FP detection rule of eliminating ECS < 20-minutes (Section 5.4.2) will cause detecting FPs and more importantly disable the signal amplitude estimation.

For EVCLs from Category#2, the *stage3* pattern has two patterns. The first could take up to 40-minutes. However, after 20-minutes the charging amplitude is much lower and sometimes

reaches zero and bounces up, therefore it can be neglected to capture higher amplitudes. For this, the first pattern is limited to 20-minutes ($V = 20$):

$$\begin{aligned}
 H(\nu) &= -2652 \times \log(\nu) + 8268, \quad \text{for } 1 \leq \nu \leq 19 \\
 \hat{H} &= (H(\nu) + 324.1180) / 7944.7 \\
 \tilde{\mathbf{F}}_{end_e+1} &= \hat{H} \times \tilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.29}$$

The two-minute ($V = 2$) *stage3* pattern of Category#2 can be computed by:

$$\begin{aligned}
 H(\nu) &= -9066.9 \times \log(2) + 1225.4, \quad \text{for } \nu = 2 \\
 \hat{H} &= (H(\nu) + 8735.6) / 9961 \\
 \tilde{\mathbf{F}}_{end_e+1} &= \hat{H} \times \tilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.30}$$

The group of operations are the same as the ones described for Category#1, this includes the extraction of *stage3* and undergoing the correction phase.

For EVCLs from Category#3 and #4, *stage3* has a two-minute duration pattern that can be computed respectively by:

$$\begin{aligned}
 H(\nu) &= -1021.9 \times \log(2) + 1465.8, \quad \text{for } \nu = 2 \\
 \hat{H} &= (H(\nu) - 343.1281) / 1122.7 \\
 \tilde{\mathbf{F}}_{end_e+1} &= \hat{H} \times \tilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.31}$$

and,

$$\begin{aligned}
 H(\nu) &= -1423.3 \times \log(2) + 2064.2, \quad \text{for } \nu = 2 \\
 \hat{H} &= (H(\nu) - 500.5451) / 1563.7 \\
 \tilde{\mathbf{F}}_{end_e+1} &= \hat{H} \times \tilde{\mathbf{F}}_{end_e}
 \end{aligned} \tag{5.32}$$

5.5 Verifications and Discussions

To verify the proposed approach, the EEVCLP algorithm is first applied on the same dataset used by [39]. It should be noted that the algorithm in [39] can only extract EVCLs of *stage2*. To perform a fair comparison, in this first verification study the proposed algorithm is limited to only extract *stage2* patterns. In the latter verification studies, the results are presented by

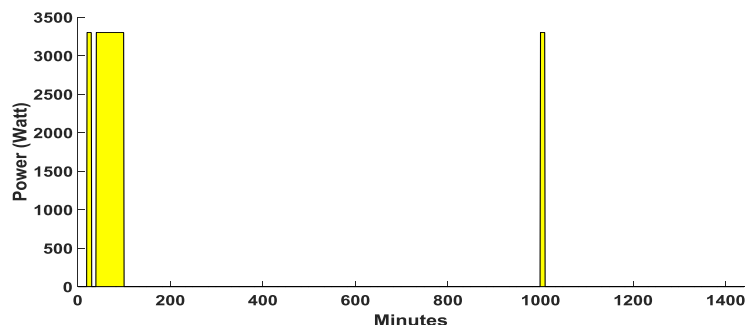
applying the proposed algorithm including the extraction of *stage1*, *stage3* and entering the correction phase.

5.5.1 Verification and Comparison on Dataset#1 on Extracting Stage2 Patterns

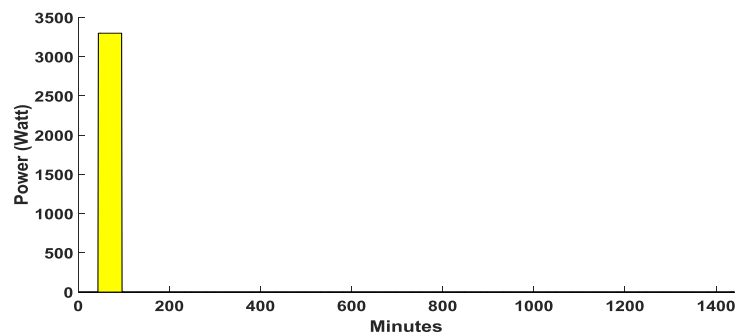
The EEVCLP algorithm was tested by using data from the Pecan Street Inc. [94], which contains up to 1-minute time-steps of circuit-level and house-level electricity data from 1391 households in Texas, Colorado, and California. It should be noted that not all households have EVs. To verify the proposed approach, the available dataset used by [39] was used as a benchmark. This dataset (Dataset#1) includes the daily aggregated power signals for 23 households, and the ground truth of the EVCLs from [94]. The EVs considered in this dataset have amplitudes that fall in Category#1. It should be noted that houses#4, #5, #13 and #23, do not include any EVCLs, in order to test the extraction of FPs from the EVCLs. The proposed algorithm was applied each time with one of the four estimation methods (*Method 1* to *Method 4*) twice (two-runs). In the first run, the duration of 10-minutes that precede and/or follow an extracted EVCL to estimate its amplitude was, $ps = 5$ and $fs = 15$. In the second run, $ps = 10$ and $fs = 20$. Thus, the proposed algorithm was run eight times in total with different combinations of estimation methods, and ps and fs . Figure 5.4, illustrates how the extracted *stage2* EVCL was obtained for house#2 using the proposed algorithm with estimation *Method 3*, and $ps = 10$ and $fs = 20$. Differently from [39] where only accuracy was used as an evaluation metric, here an evaluation metric from the information retrieval domain, modified F-score [101] is used. The reason for involving evaluation metrics other than the accuracy is due to the fact that with power disaggregation, the accuracy can be very skewed if the appliance is rarely used. For example, if the EV was plugged-in for only 20-minutes in a 24-hour period, and the algorithm was unable to extract any charging session, it will still achieve high accuracy. For that, the results of the proposed algorithm are evaluated using a modified F-score, and the accuracy and error are measured only when EVCLs are present either in the EVCL ground truth or the extracted EVCL. This not only measures the accuracy of classification of the state of the appliance, it also

measures the accuracy of the estimated amplitude. The utilized evaluation measures are presented in [101] and Appendix D, and the modified F-score was computed by:

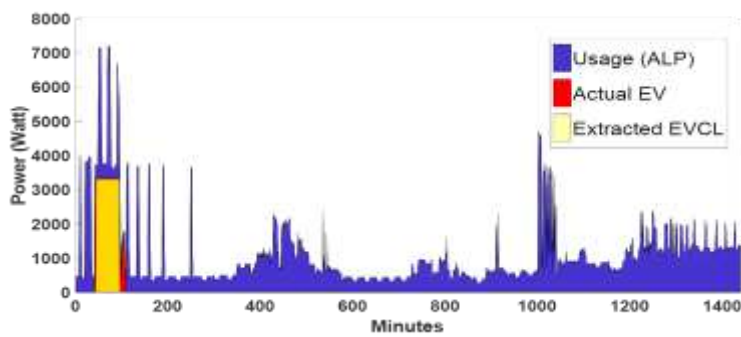
$$F_{score} = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (5.33)$$



(a)



(b)



(c)

Figure 5.4: The result of the proposed algorithm with estimation *Method 3*. (a) the EVCL after applying ICA. (b) EVCL after removing the FPs. (c) final EVCL (transparent yellow) vs. the actual EVCL (red).

Table 5.3 Performance Comparison of the Proposed Algorithm with Estimation Methods 1- 4, $ps = 5$ and $fs = 15$ to Extract stage2

EEVCLP with $ps = 5$ and $fs = 15$																				
Car	Estimation Method 1					Estimation Method 2					Estimation Method 3					Estimation Method 4				
	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score
1	1	0.99	89.29	100	94.34	1	11.09	89.29	100	94.34	1	0.99	89.29	100	94.34	1	0.99	89.29	100	94.34
2	1	-100	0.00	0.00	0.00	1	5.20	90.38	95.92	93.07	1	5.20	90.38	95.92	93.07	1	5.20	90.38	95.92	93.07
3	1	3.37	83.58	95.73	89.24	1	17.57	92.13	100	95.90	1	3.37	83.58	95.73	89.24	1	7.23	84.17	100	91.41
4	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
5	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
6	1	-9.99	98.21	98.21	98.21	1	2.23	100	94.64	97.25	1	-0.99	98.21	98.21	98.21	1	2.23	100	94.64	97.25
7	1	-9.78	99.00	99.00	99.00	1	-9.78	99.00	99.00	99.00	1	-1.75	100	99.00	99.50	1	6.09	100	98.00	98.99
8	1	-5.39	95.45	100	97.67	1	-5.39	95.45	100	97.67	1	10.95	97.67	100	98.82	1	51.38	64.06	97.62	77.36
9	1	-3.62	99.05	96.30	97.65	1	4.14	100	96.30	98.11	1	-9.88	99.07	99.07	99.07	1	-9.88	99.07	99.07	99.07
10	1	-10.7	99.54	98.64	99.09	1	6.09	100	98.18	99.08	1	-10.7	99.54	98.64	99.09	1	6.09	100	98.18	99.08
11	1	-9.25	98.85	99.61	99.23	1	6.39	100	98.45	99.22	1	6.39	100	98.45	99.22	1	6.39	100	98.45	99.22
12	1	-9.58	99.54	100	99.77	1	8.01	100	100	100	1	-9.58	99.54	100	99.77	1	-0.53	99.54	100	99.77
13	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
14	1	0.25	90.00	100	94.74	1	0.25	90.00	100	94.74	1	0.25	90.00	100	94.74	1	0.25	90.00	100	94.74
15	1	-7.87	97.92	100	98.95	1	-0.77	100	100	100	1	-0.77	100	100	100.00	1	-0.77	100	100	100
16	1	-7.06	97.10	100	98.53	1	-7.06	97.10	100	98.53	1	3.45	100	95.52	97.71	1	-7.06	97.10	100	98.53
17	1	-4.46	94.29	100	97.06	1	1.55	96.77	90.91	93.75	1	1.55	96.77	90.91	93.75	1	4.82	96.88	93.94	95.38
18	1	-6.65	96.67	100	98.31	1	-6.65	96.67	100	98.31	1	-6.65	96.67	100	98.31	1	8.28	98.28	98.28	98.28
19	1	-100	0.00	0.00	0.00	2	-100	0.00	0.00	0.00	2	-100	0.00	0.00	0.00	2	-100	0.00	0.00	0.00
20	1	-11.5	99.10	97.36	98.22	1	-11.5	99.10	97.36	98.22	1	5.24	100	97.36	98.66	1	-3.53	100	97.36	98.66
21	1	-9.94	98.91	98.91	98.91	1	4.55	100	96.74	98.34	1	4.55	100	96.74	98.34	1	-2.01	100	98.91	99.45
22	1	-8.46	98.49	100	99.24	1	5.99	100	97.96	98.97	1	-8.46	98.49	100	99.24	1	-8.46	98.49	100	99.24
23	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
Overall		16.78	86.05	88.61	87.27		11.27	91.88	92.91	92.34		10.03	91.53	92.92	92.16		12.16	89.85	93.17	91.25

Table 5.4 Performance Comparison of the Proposed Algorithm with Estimation Methods 1-4, $ps = 10$ and $fs = 20$ to Extract stage2

EEVCLP with $ps = 10$ and $fs = 20$																				
Car	Estimation Method 1					Estimation Method 2					Estimation Method 3					Estimation Method 4				
	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score	Category	Error%	Precision	Recall	F-score
1	1	0.99	89.29	100	94.34	1	11.09	89.29	100	94.34	1	19.02	90.91	100	95.24	1	11.09	89.29	100	94.34
2	1	-100	0.00	0.00	0.00	1	5.20	90.38	95.92	93.07	1	5.20	90.38	95.92	93.07	1	5.20	90.38	95.92	93.07
3	1	7.23	84.17	100	91.41	1	7.23	84.17	100	91.41	1	7.23	84.17	100	91.41	1	7.77	92.13	100	95.90
4	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
5	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
6	1	-9.99	98.21	98.21	98.21	1	-9.99	98.21	98.21	98.21	1	-9.99	98.21	98.21	98.21	1	-9.99	98.21	98.21	98.21
7	1	-9.78	99.00	99.00	99.00	1	-9.78	99.00	99.00	99.00	1	-2.75	100	98.00	98.99	1	6.09	100	98.00	98.99
8	1	-5.39	95.45	100	97.67	1	-5.39	95.45	100	97.67	1	1.71	97.67	100	98.82	1	5.79	97.56	95.24	96.39
9	1	-9.88	99.07	99.07	99.07	1	-4.87	100	87.96	93.60	1	-4.87	100	87.96	93.60	1	-9.88	99.07	99.07	99.07
10	1	-10.36	99.54	99.09	99.32	1	5.11	100	97.27	98.62	1	-1.40	99.54	99.09	99.32	1	-10.36	99.54	99.09	99.32
11	1	-9.25	98.85	99.61	99.23	1	6.39	100	98.45	99.22	1	-0.94	99.61	99.61	99.61	1	6.39	100	98.45	99.22
12	1	-9.58	99.54	100	99.77	1	8.01	100	100	100	1	-9.58	99.54	100	99.77	1	8.01	100	100	100
13	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
14	1	0.25	90.00	100	94.74	1	0.25	90.00	100	94.74	1	10.28	90.00	100	94.74	1	12.48	88.24	100.00	93.75
15	1	-11.71	97.83	95.74	96.77	1	-9.22	100	91.49	95.56	1	1.34	100	93.62	96.70	1	-7.87	100	85.11	91.95
16	1	-9.75	98.51	98.51	98.51	1	-9.75	98.51	98.51	98.51	1	-9.75	98.51	98.51	98.51	1	-9.75	98.51	98.51	98.51
17	1	-4.46	94.29	100	97.06	1	1.55	96.77	90.91	93.75	1	1.55	96.77	90.91	93.75	1	1.55	96.77	90.91	93.75
18	1	-6.65	96.67	100	98.31	1	8.28	98.28	98.28	98.28	1	0.97	98.31	100	99.15	1	8.28	98.28	98.28	98.28
19	1	39.38	64.62	100	78.50	1	63.65	64.47	97.62	77.65	1	53.32	64.62	100	78.50	1	-100	0.00	0.00	0.00
20	1	-12.30	99.55	96.92	98.21	1	-3.97	100	96.92	98.43	1	4.29	100	96.48	98.21	1	-3.97	100	96.92	98.43
21	1	-9.94	98.91	98.91	98.91	1	-0.93	98.91	98.91	98.91	1	-9.94	98.91	98.91	98.91	1	-0.93	98.91	98.91	98.91
22	1	-8.46	98.49	100	99.24	1	7.09	100	98.98	99.49	1	7.09	100	98.98	99.49	1	-8.46	98.49	100.00	99.24
23	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-	0	0.00	-	-	-
Overall		14.49	89.57	93.95	91.48		9.35	94.91	97.28	95.81		8.48	95.11	97.69	96.10		11.83	91.86	92.24	91.96

A detailed comparison of the results from each run of the proposed method on extracting *stage2* is presented in Table 5.3 and Table 5.4. It was observed that the proposed algorithms achieve relatively high F-scores, except for house#19. The resulted EVCL when $ps = 5$ and $fs = 15$, was unable to detect the actual EVCLs. This suggests that considering observations of ALP that are closer to an ECS to estimate the amplitude may extract FPs or not extract any EVCLs. It should be noted that extracting the EVCL for house#19 (Figure 5.5) is a challenging task due to the high degree of overlapping of other appliances that have similar EV load patterns. The best extracted *stage2* EVCL for house#19 was extracted using the proposed algorithm with *Method 3*. The overall results of the proposed algorithm, regardless of the used estimation method, outperformed [39] on all evaluation measures. The overall results of the proposed algorithm on extracting *stage2* were satisfactory, and the best approach was using *Method 3* with $ps = 10$ and $fs = 20$, as it had the best overall performance and presented the best result on house#19. The results of the approaches discussed were compared with the state-of-the-art [39] (Table 5.5). Their algorithm extracted the second ECS of house#19 as two separate ECSs. Although their algorithm performed better on house#19 (Figure 5.5), it has failed to extract the EVCLs for houses#2 and #17, and overestimated house#3. A drawback of [39] is introducing definitions: *effective width* (the width of an EVCL segment at the bottom) and *effective height* (the height at which the EVCL segment's width is 80% of the bottom width). Those definitions restrict the detection of EVCLs to certain charging patterns. In house#17, their algorithm failed because it can only extract EVCLs that have durations longer than 30-minutes and shorter than 200-minutes. In order to extract EVCLs that have less than a period of 30-minutes, FPs extractions may appear. On the other hand, the algorithm proposed in this work can extract EVCLs that have durations as low as 20-minutes. From this, there is a trade-off between detecting shorter EVCLs and extracting FPs that occur from other appliances. As noted previously in this verification study, the intention was to only extract the *stage2* EVCLs for fair comparison with the state-of-the-art method. For this purpose, the correction phase, and extracting *stage1* and *stage3*, has not been undertaken. In the following verification study the overestimation of house#19 will be addressed.

Table 5.5 Performance of the Algorithm in [39] on Dataset#1

Car	Category	Error %	Precision	Recall	F-score
1	1	25.71	89.29	100	94.34
2	0	-100	0.00	0.00	0.00
3	1+	49.84	0.00	0.00	0.00
4	0	0.00	-	-	-
5	0	0.00	-	-	-
6	1	0.46	98.25	100	99.12
7	1	1.58	99.01	100	99.50
8	1	13.41	95.45	100	97.67
9	1	4.64	99.08	100	99.54
10	1	-0.55	98.65	100	99.32
11	1	-0.55	98.85	100	99.42
12	1	3.90	99.08	100	99.54
13	0	0.00	-	-	-
14	1	14.58	86.54	100	92.78
15	1	0.80	97.92	100	98.95
16	1	1.72	97.10	100	98.53
17	0	-100	0.00	0.00	0.00
18	1	2.14	96.67	100	98.31
19	1	-8.97	99.47	89.05	93.97
20	1	6.46	98.66	97.36	98.00
21	1	-0.32	98.92	100	99.46
22	1	18.36	97.00	98.98	97.98
23	0	0.00	-	-	-
Overall		18.63	81.57	83.44	82.44

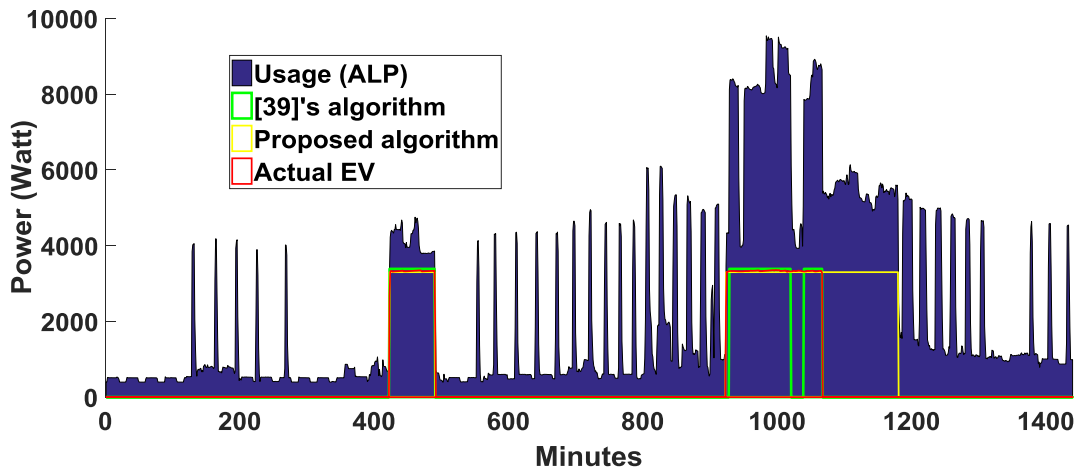


Figure 5.5: The actual EVCL vs. the results from [39] and the proposed algorithm on extracting *stage2*.

5.5.2 Verification on Dataset#1 and #2 on Extracting All Stages

In the previous verification, the proposed algorithm was applied on Dataset#1 that contains EVs from Category#1 to extract *stage2* EVCL patterns. To further verify the proposed algorithm, a dataset from the Pecan Street Inc. [94] that contains eleven daily aggregated power signals and the ground truth of EVs from Category#2, #3 and #4 is considered (Dataset#2). In this verification study, the proposed algorithm is applied to extract the EVCLs including *stage1* and *stage3*, and enter the correction phase. The proposed algorithm using *Method 3* with $ps = 10$ and $fs = 20$, which had the best overall performance in extracting *stage2* is used. The evaluation measures are presented in Table 5.5, and the EVCL for house#19 is presented in Figure 5.6. It can be observed that during the extraction of *stage3* the correction phase was able to correct the overestimated duration of the second ECS (Figure 5.6). This suggests that the correction phase can capture the duration of the EVCL when overlapping with appliances that have similar EV load patterns. The EVCL for house#30 is presented in Figure 5.7. It can be observed that the ALP exceeds 10 kW at some periods, which may cause to estimate the amplitude of the EV to a template in Category#4 (10 kW), but this is unlikely to happen. This is due to that the estimation method involves the baseload in estimating the amplitude of the EVCL. Also, it can be observed from Figure 5.7 that the second charging session was not extracted. This is because the proposed algorithm neglects charging periods less than 20-minutes to reduce the FPs that may occur. The results of applying the algorithm presented in [39] on Dataset#2 are also presented in Table 5.6. Differently from the previous application, the evaluation measures are computed with including all three stages (*stage1*, *stage2* and *stage3*). It can be observed that their method presented less accurate results on EVCLs with higher amplitudes. This is due to the fact that, as previously mentioned, it restricts the detection of EVCLs to certain charging patterns. Also, it is observed that the F-score decreased, this is because the algorithm in [39] doesn't involve the extraction of *stage1* and *stage3* of the EVCLs. The results of applying the proposed algorithm to extract EVCLs that have higher charging amplitudes (e.g., Tesla EVs) suggest that the algorithm is capable of extracting various types of EVCL amplitudes.

Table 5.6 Performance of the Algorithm on Dataset#1 and Dataset#2 for All Stages

Car	EEVCLP			[39]'s algorithm		
	Category	Error %	F-score	Category	Error %	F-score
Dataset#1						
1	1	12.34	90.09	1	12.35	87.52
2	0	0.17	90.74	0	-100	0.00
3	1	3.07	89.14	1+	37.05	0.00
4	0	0.00	-	0	0.00	-
5	0	0.00	-	0	0.00	-
6	1	-10.33	98.25	1	-2.45	98.52
7	1	-5.25	97.03	1	-3.35	97.59
8	1	-2.96	95.56	1	0.00	91.41
9	1	-6.03	92.82	1	-0.68	96.03
10	1	-3.45	98.21	1	-5.01	97.28
11	1	-2.40	98.29	1	-5.83	96.83
12	1	-10.01	99.54	1	2.61	98.22
13	0	0.00	-	0	0.00	-
14	1	4.39	92.16	1	0.93	90.04
15	1	-2.47	93.75	1	-13.44	92.86
16	1	-12.64	97.84	1	-10.37	94.29
17	0	-3.49	88.57	0	-100	0.00
18	1	-2.23	95.08	1	-12.66	92.62
19	1	-0.03	98.37	1	-10.19	93.34
20	1	3.70	96.92	1	1.35	92.95
21	1	-19.79	93.92	1	-9.95	94.89
22	1	5.28	97.24	1	7.47	88.37
23	0	0.00	-	0	0.00	-
Dataset#2						
24	2	-11.97	96.92	2	-14.78	86.53
25	2	-8.21	93.08	3	-36.18	75.18
26	3	-10.54	99.31	0	-100	0.00
27	3	1.07	91.94	3	-4.10	98
28	3	16.40	79.92	0	-100	0.00
29	2	10.87	74.24	3	-52.40	49.62
30	2	-20.67	89.91	2	-47.35	63.02
31	2	-6.95	97.37	2	-45.38	67.79
32	2	-3.39	99.58	2	-1.89	99.05
33	2	0.08	96.87	2	9.03	92.15
34	4	-6.37	97.10	3	-19.59	80.17
Overall		6.88	93.99		23.48	77.34

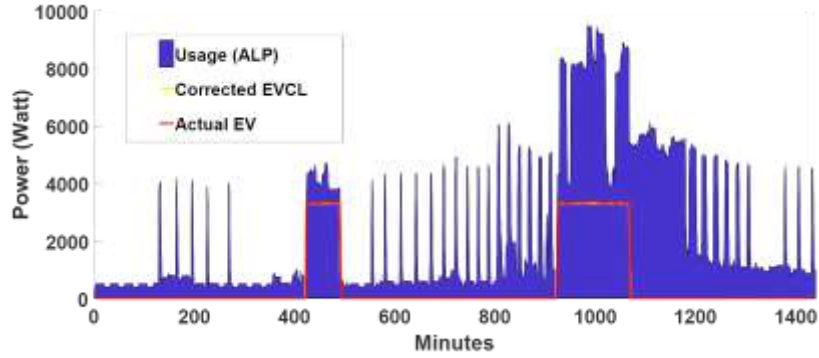


Figure 5.6: The actual EVCL vs. the extracted EVCL from the proposed algorithm with estimation *Method 3* for house#19 after undergoing correction phase.

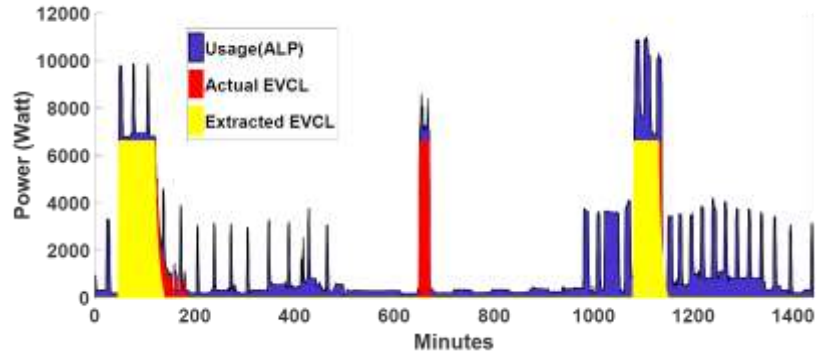
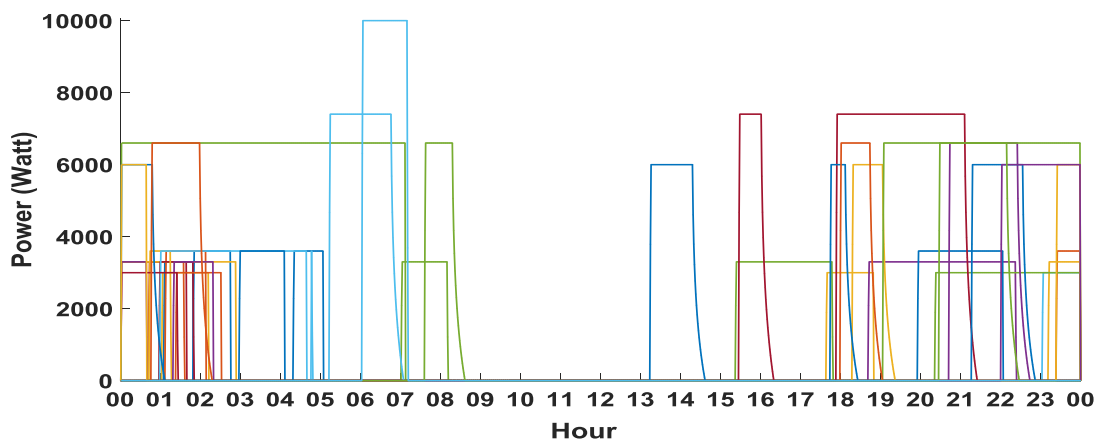


Figure 5.7: The actual EVCL vs. the extracted EVCL from the proposed algorithm with all stages.

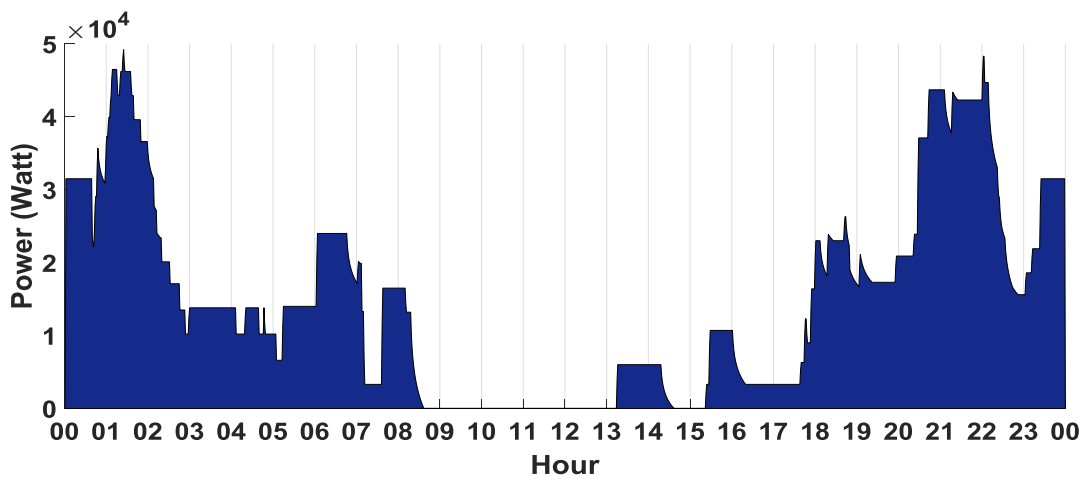
5.5.3 Verification on Extracting Hourly EVCLs

To further test the effectiveness of the proposed algorithm, the household data of Dataset#1 and Dataset#2 are considered. It is assumed that those 34-houses are located in the same neighborhood. Then differently from the previous applications, the data is ingested in segments of 1-hour, in order to monitor the charging behavior of a certain neighborhood during the past hour. The results of this application are shown in Figure 5.8. From the cumulative plot (Figure 5.8(b)) it can be observed that this neighborhood plug-in their EVs from around 18:00 to 06:00. An accurate estimation of the aggregated charging demand is critical for utilities to evaluate the power delivery. Also, analyzing and studying such behaviors can assist the smart grid operators

in planning and dynamic demand response strategies. In this application (Figure 5.8(a)) some EVCLs may have been missed due to the rule of neglecting charging sessions less than 20-minutes, For example, if the on-hand hour is 18:00 to 18:59, and a 30-minute charging session started at 18:45 (relies between two following hours) then it will be neglected as it was extracted as a less than 20-minute charging session. It should be noted that the state-of-the-art algorithms neglect charging sessions <30-minutes; accordingly, those algorithms tend to miss many EVCLs, and are infeasible in capturing the charging behaviors of the EVs during the past hour or shorter time periods.



(a)



(b)

Figure 5.8. (a) The extracted EVCLs from Dataset#1 and Dataset#2. (b) The cumulative EVCLs for the neighborhood.

5.5.4 Lower Sampling Rates

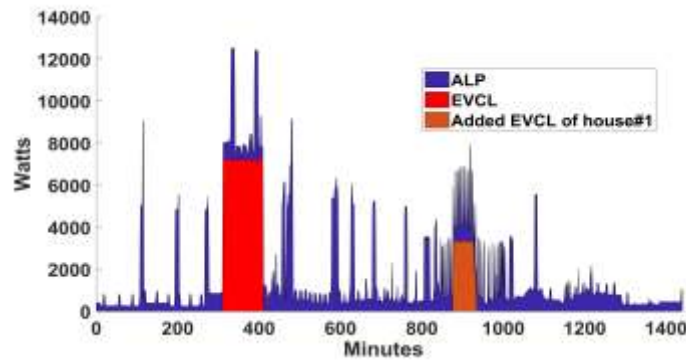
In the previous applications, the proposed algorithm was tested on datasets with 1-minute sampling rates ($\tau=1$). In this application, the effect of sampling the data at lower rates is tested. For this purpose, Dataset#1 and Dataset#2 are sampled at rates from $\tau=2$ to $\tau=5$. Then the EEVCLP algorithm is applied. The results are presented in Table 5.7. From the results, it was observed that at lower sampling rates, the accuracy measures of the extracted EVCLs decrease.

Table 5.7 Performance of the Proposed Algorithm with Lower Sampling Rates

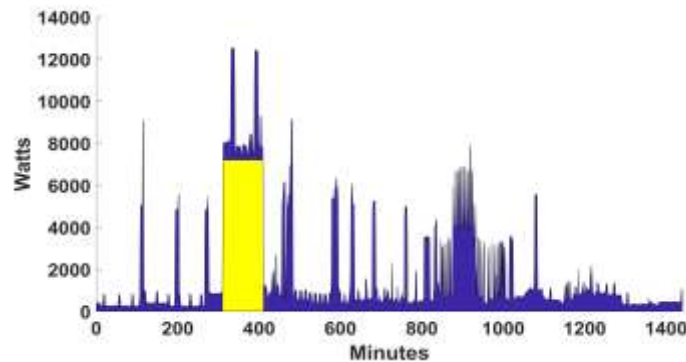
Car	N = 10, $\tau = 2$		N = 10, $\tau = 3$		N = 10, $\tau = 4$		N = 10, $\tau = 5$	
	Error %	F-score	Error %	F-score	Error %	F-score	Error %	F-score
1	51.42	74.63	1.81	96.97	-0.87	95.65	18.92	90.91
2	8.11	95.83	14.86	96.97	-36.97	73.68	-100	0.00
3	7.79	87.60	-9.59	82.50	-16.11	75.00	5.45	92.31
4	0.00	-	0.00	-	0.00	-	0.00	-
5	0.00	-	0.00	-	0.00	-	0.00	-
6	-4.52	98.18	-6.51	97.14	-48.56	72.73	-100	0.00
7	-11.65	98.99	-3.57	98.51	-9.86	96.00	-9.87	95.00
8	3.16	92.68	6.40	96.55	-0.73	95.24	-100	0.00
9	-16.53	96.15	-17.92	87.50	4.26	98.11	-44.23	76.47
10	-5.46	96.74	-14.31	92.75	-14.86	95.33	-23.45	87.18
11	-2.43	99.22	-5.56	97.62	-13.48	88.89	-23.76	91.67
12	7.01	99.53	-9.24	95.65	-2.83	99.07	-26.74	89.74
13	0.00	-	0.00	-	0.00	-	0.00	-
14	3.33	97.67	1.03	96.55	6.63	91.67	0.21	94.74
15	-14.21	88.37	-18.89	85.71	-9.55	100.00	0.07	94.74
16	-0.85	100.00	62.23	75.86	-47.50	69.23	-2.95	96.30
17	277.64	41.56	211.37	44.90	-100	0.00	-100	0.00
18	37.05	79.45	-13.20	88.89	-3.44	96.55	8.11	95.65
19	34.67	76.34	30.63	74.71	-100	0.00	20.08	69.39
20	-5.25	97.76	-3.58	98.65	4.20	98.18	5.66	98.88
21	39.00	76.92	40.60	80.00	-0.89	95.65	-14.68	97.30
22	-65.77	48.06	-66.46	50.57	-66.91	46.88	-63.96	50.00
23	0.00	-	0.00	-	0.00	-	0.00	-
24	0.77	100	-8.39	75.00	-100	0.00	-44.26	94.74
25	-0.12	100	-42.72	93.15	-47.61	90.20	-52.24	97.67
26	-10.45	99.44	-0.43	99.58	-46.97	73.68	-58.45	92.83
27	-7.76	94.62	2.72	100	2.72	100	-100	0.00
28	18.34	91.51	18.48	91.55	-0.19	100	-0.17	100
29	8.38	89.95	11.10	88.19	12.23	47.14	3.77	42.59
30	-23.79	91.30	-59.43	83.15	-30.73	86.57	-70.95	65.22
31	-15.69	92.96	-32.80	81.82	-100	0.00	-100	0.00
32	-0.05	96.49	-1.03	96.00	-3.87	94.55	-62.70	76.92
33	-8.09	97.87	-2.49	98.90	-6.79	98.59	-68.94	63.64
34	-24.48	98.59	-23.38	97.87	-22.25	97.14	-26.59	100
Overall	23.79	89.94	24.69	88.10	28.70	75.85	41.87	68.46

5.5.5 Extracting EVCLs of Different Categories

In this application, it is desired to extract EVCLs for households that have more than one EV. In cases where the EVs have the same charging amplitudes, the proposed algorithm can extract the EVCL at one run. However, in cases where the EVs have different charging amplitudes, the algorithm is run twice. In the first run, the algorithm will extract the EV that has higher charging amplitude. The extracted EVCL is subtracted from the ALP. Then, in the second run, the algorithm extracts the EV having the lower amplitude. In order to verify this, the actual EVCL of house#1 (has an EV of Category#1) is added to the ALP of a house that has an EV of Category#3. The actual EVCL of house#1 was added during the operation of other appliances (Fig 5.9 (a)) to further test the extraction during overlapping with other appliances. The EVCL from the first run and second run are shown in Figure 5.9 (b) and (c), respectively.



(a)



(b)

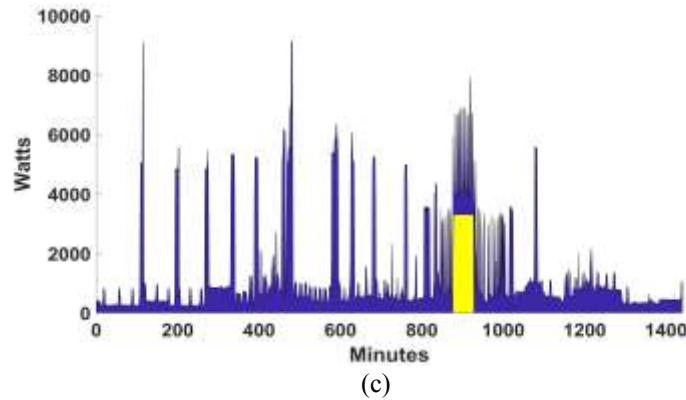


Figure 5.9: Extracting EVCLs of different categories. (a) The actual EVCL of house#1 added to an actual ALP. (b) The resulted EVCL of the first run of the algorithm. (c) The resulted EVCL of the second run of the algorithm.

5.6 Conclusions

This chapter introduced an algorithm related to extracting EV charging loads (EVCL). The proposed algorithm (EEVCLP) is unsupervised and is able to run on low-frequency smart meter data. In addition, the proposed algorithm only requires the real power smart meter measurement, which is the type of data recorded and communicated by most smart meters. The proposed algorithm utilizes a signal processing method, ICA, to extract the EVCLs from the aggregated load pattern of households. The amplitude and sign of the extracted EVCLs obtained by applying the ICA method have been addressed by introducing amplitude estimation methods. In addition, the proposed algorithm can effectively mitigate the interference of other appliances that have similar load behaviors as EVs.

In order to verify the effectiveness of the proposed approach, the EEVCLP algorithm was applied on real household datasets and compared with the state-of-the-art algorithms. Further, the proposed algorithm includes extracting the gradual increase, steady and gradual decrease in the EV charging patterns. The results of extracting the EVCLs from the daily ALP of households were satisfactory. Furthermore, the proposed algorithm was tested by ingesting 1-hour segments to monitor the EV charging behaviors of a neighborhood during the past hour. The results were satisfactory and suggested that the proposed algorithm is able to run on shorter time frames.

Aggregating the EVCLs could provide an accurate estimation of the aggregated charging demand that is critical for utilities to evaluate the power delivery, and can assist smart grid operators in planning and dynamic demand response strategies. Also, further analyses and studies can be carried out from aggregating such charging behaviors of EVs. It should be noted that the EEVCLP was applied on top of the smart grid big data eco-system.

Chapter 6

Defining Flexibility of Residential Electrical Vehicle Charging Loads

6.1 Introduction

In the previous chapter, the EEVCLP algorithm extracted the EV charging loads (EVCLs). Those extracted EVCLs can be aggregated for further analysis. In this chapter, a method to define flexibility for the collective EV charging demand by analyzing the time-variable patterns of the aggregated EV charging behaviors is presented. Those flexibility indices reflect the collective trend in EV charging in certain time periods. This information is useful for system operators to identify how flexible the aggregated EVCL is at different time periods and plan for demand response programs accordingly.

6.2 Modeling Demand Variations Using Bayesian Maximum Likelihood

From the previous chapter, each extracted EVCL pattern is represented by $f = [f_1, \dots, f_T]$, for $t = 1, \dots, T$. The matrix of all extracted EVCLs is represented by the matrix F_k , for $k = 1, \dots, K$. The EV load variations referring to an increase or decrease from one-time observation to the following one is computed for all extracted EVCLs f :

$$\Delta f_{kt} = f_{kt} - f_{k(t-1)}, \quad \text{for } t = 2, \dots, T \quad (6.1)$$

The extracted EVCL and its variation can be represented by the following vectors:

$$f_k = [f_1, \dots, f_T] \quad (6.2)$$

$$\Delta f_k = [\Delta f_1, \dots, \Delta f_T] \quad (6.3)$$

6.2.1 Binomial Representation of Variations

For the particular problem in this chapter, the variations in EVCL may be positive or negative. Accordingly, the demand variations can be modeled with the binomial distribution with two response variables: 1) increasing in EVCL demand, 2) decreasing in EVCL demand. Each observation can be considered as a Bernoulli trial with two outcomes. The outcome of each trial at a particular time observation t is:

$$\mu_{kt} = \begin{cases} UPos_{kt} = 1, & \Delta f_{kt} > 0 \\ UNeg_{kt} = 1, & \Delta f_{kt} < 0 \end{cases} \quad (6.4)$$

6.2.2 Bayesian Maximum Likelihood Estimation

From the binomial model presented in the previous subsection, the probability of an increase (ω_i) is unknown, and a suitable estimation method is needed. The Bayesian maximum likelihood estimation method [102] is used. The reason for using such method is due to it providing a natural and principled way of combining prior information about EV charging behaviors with data, within a solid decision theoretical framework. In addition, it incorporates past information about the increase in EVCLs and forms a prior distribution for future increase in EVCLs analysis. All inferences logically follow from Bayes' theorem. Bayes formula is presented by [102]:

$$p(\omega | f^{data}) = \frac{p(f^{data} | \omega) p(\omega)}{p(f^{data})} \quad (6.5)$$

This formula can be expressed informally in English by:

$$posterior = \frac{likelihood \times prior}{evidence} \quad (6.6)$$

The Bayes formula converts the prior information about the increase in EVCLs into a posterior probability $p(\omega | f^{data})$ by using the likelihood function $p(f^{data} | \omega)$.

To compute the probability of an increase (ω_t) using the Bayesian maximum likelihood estimation method, first, the prior probability is computed. To compute the prior probability of EVCL increase, EVCL pattern data is needed. This data can be extracted by the EEVCLP algorithm (Chapter 5) for past years for the same location where the posterior probability is desired. If prior historical data is unavailable for that location, data for another location can be used instead to compute the prior. For example, if historical EVCLs data for Texas is available, the prior probability of an increase in EVCLs can be computed, and recent data from Texas can be used to compute the likelihood. However, if it is desired to compute the likelihood of EVCL increase for California where historical data (prior) is unavailable, the computed Texas prior probability can be used. Following are the steps to compute the Bayesian likelihood estimation for the increase in EVCL.

Step1 (Prior): Previous aggregated residential load patterns that include EVCLs are obtained. The EEVCLP algorithm is applied (Chapter 5) then the variations are modeled with the binomial distribution (Section 6.2.1). The number of trials (increase/decrease) is computed:

$$n = \sum_{t=1}^T UPos_t + UNeg_t \quad (6.7)$$

The probability of success (increase in EVCL) is computed:

$$m = \sum_{t=1}^T UPos_t / n \quad (6.8)$$

Step2 (Likelihood): Recent residential load patterns that include EVCLs are obtained. Then similar to Step1, the EEVCLP algorithm is applied, and the variations are modeled with the binomial distribution. The number of trials from this data is computed:

$$\aleph = \sum_{t=1}^T UPos_t + UNeg_t \quad (6.9)$$

The number of successes (increase in EVCL) is computed:

$$Y = \sum_{t=1}^T UPos_t \quad (6.10)$$

Step3 (Posterior): In this step (6.5) converts our prior belief about the increase in EVCL (before seeing data) into a posterior probability by using the likelihood function (Step2). The posterior probability of increase for this binomial process can be computed by:

$$a = Y + (n \times m) - 1 \quad (6.11)$$

$$b = \aleph - Y + (n \times (1 - m)) - 1 \quad (6.12)$$

$$\omega_t = \text{mean} = \frac{a}{a + b} \quad (6.13)$$

and, the confidence intervals are computed by:

$$\text{var} = \sqrt{(a + b) / (a + b)^2 (a + b + 1)} \quad (6.14)$$

$$\overline{\omega}_t = \text{mean} + \text{var} \quad (6.15)$$

$$\underline{\omega}_t = \text{mean} - \text{var} \quad (6.16)$$

The probability of an increase in EVCL and its upper and lower limits can be represented by the following vectors:

$$\omega_t = [w_2, w_3, \dots, w_T] \quad (6.17)$$

$$\overline{\omega}_t = [\bar{w}_2, \bar{w}_3, \dots, \bar{w}_T] \quad (6.18)$$

$$\underline{\omega}_t = [\underline{w}_2, \underline{w}_3, \dots, \underline{w}_T] \quad (6.19)$$

6.3 Flexibility Definitions for EVCLs

In the previous section, the probability of demand increase in EVCL at each time observation was modeled using the Bayesian maximum likelihood method. In this section, the probability of demand increase in EVCL (ω_t) is used to define a flexibility index of EV aggregated demand (FIEVAD). Further, the flexibility percentage level (FPL), that expresses the percentage of flexible demand associated with the flexibility index (FIEVAD), is defined.

6.3.1 Flexibility Index of EV Aggregated Demand (FIEVAD)

Let ${}^+\bar{f}$ and ${}^-\bar{f}$ be mean values of the load variations for increase and decrease in demand, respectively, and can be computed by:

$${}^+\bar{f}_t = \sum_{t=1}^T \Delta f_t / Y, \quad \text{for } \Delta f_t > 0 \quad (6.20)$$

$${}^-\bar{f}_t = \sum_{t=1}^T \Delta f_t / (S - Y), \quad \text{for } \Delta f_t < 0 \quad (6.21)$$

Now, the minimum between the probability of an increase in demand and the complementary probability is computed:

$$\pi_t = \min_{\forall \omega_t}(\omega_t, 1 - \omega_t) \quad (6.22)$$

By definition, the value of each entry π_t is in the range of [0, 0.5]. In fact the minimum complementary value of entries ω_t and $(1 - \omega_t)$ is equal to 0.5 when $\omega_t = 0.5$, and is equal to zero when $\omega_t = 1$. In order to obtain the formulation of the FIEVAD in a range of [0, 1] in line with the probabilistic limits, the probability values of π_t are multiplied by two, which are the number of variations (increase and decrease in demand) of the binomial probability distribution:

$$FIEVAD_t = \pi_t \times 2, \quad \text{with } FIEVAD \in [0, 1] \quad (6.23)$$

It can be noted from the formation of the FIEVAD value that it gives information about the possible probabilistic change to the nearest optimum, 0 or 1, for each binomial category (increase or decrease in demand). Also, by definition, the FIEVAD values are symmetric with respect to 0.5. This corresponds to that any change in ω_t determines opposite changes in ${}^+\overline{f}$ and ${}^-\overline{f}$.

The flexibility index (FIEVAD) is not a quantitative margin expressed in watts, but reflects EV charging behavioral interpretation in terms of collective trend. Also, the FIEVAD incorporates both the possibility of increasing/decreasing in the aggregated EVCL and accepting variations. If the FIEVAD value is close to 1, this indicates that the EV charging behavior is very random and there is no collective trend in the corresponding observation time. Accordingly, the flexibility to change the behavior is high, and there is a high chance to obtain DR benefits. This information can assist operators and researchers to improve the reliability and economical operation of the grid by managing supply using flexibility indices. FIEVAD values close to zero, indicate that there is a collective behavior (trend) in EV charging. Accordingly, the aggregated demand is rigid to accept DR changes.

6.3.2 Flexibility Percentage Level (FPL)

The flexibility percentage level (FPL) expresses the percentage of flexible demand associated with the flexibility index (FIEVAD). This indicator represents what percentage of aggregated demand can be reduced or increased without affecting the average change in the collective demand. The FPL can be computed by:

$$FPL_t = \frac{{}^+\overline{f}_t - {}^-\overline{f}_t}{\overline{f}_t} \left(\frac{FIEVAD_t}{2} \right) \times 100 \quad (6.24)$$

where \overline{f}_t is the mean value of the extracted EVCLs f at time observation t . The FPL increases as customers change their EV charging behavior from an increase in EVCL demand towards a decrease in EVCL demand and vice versa. However, increasing the FPL is challenging when

customers follow a trend behavior (i.e., ω_t is close to 0 or 1). For example, if $\omega_t=0.9$, which means there is a trend in EV charging behavior, it is hard to reverse this trend by reducing ω_t to zero. Also, when $\omega_t=0.1$, which means there is no trend in EV charging, it is more reasonable to reduce ω_t to zero than increasing it to 1. This is one of the reasons to define the FIEVAD as the minimum of the two binomial probabilities of increase and decrease in EV charging demand in (6.23).

The FPL takes into account the increase and decrease in collective demand together. Separate information for the maximum increase ($+VPL$) and decrease ($-VPL$) in aggregate demand can be obtained by:

$$+VPL_t = \frac{+\bar{f}_t - -\bar{f}_t}{f_t} (1 - w_t) \times 100 \quad (6.25)$$

$$-VPL_t = \frac{+\bar{f}_t - -\bar{f}_t}{f_t} (w_t) \times 100 \quad (6.26)$$

These indicators represent the maximum demand variation that may be obtained in the case were all the increasing EV charging demand changes to decreasing in EV charging demand, and vice versa. It should be noted that ($+VPL$) and ($-VPL$) refer to load variability and not to the flexibility in collective EV load demand. The FPL, $+VPL$ and $-VPL$ can be expressed in watts by:

$$FPLwatt_t = f_t \times (FPL_t / 100) \quad (6.27)$$

$$+VPLwatt_t = f_t \times (+VPL_t / 100) \quad (6.29)$$

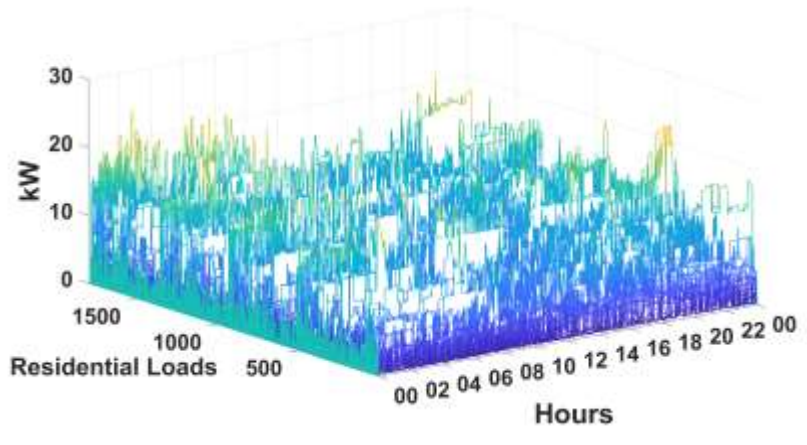
$$-VPLwatt_t = f_t \times (-VPL_t / 100) \quad (6.30)$$

6.4 Applications and Discussions

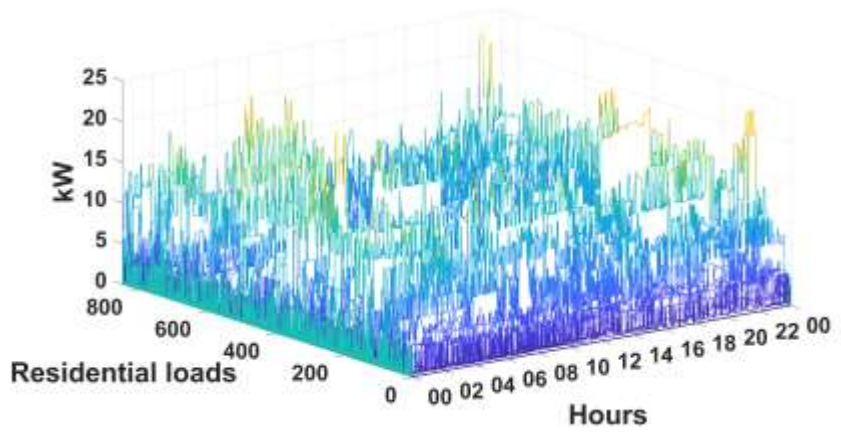
In this section, the application of the previously mentioned methods to extract EVCLs and define flexibility for the extracted aggregated EVCLs is presented. In the following subsection, a verification of the EEVCLP algorithm on extracting EVCLs and comparing the aggregated EVCLs to the aggregated ground-truth is presented. The resulted data is used in a case study in order to quantify the flexibility achievable from the aggregate EV load in different time periods. It should be noted that the proposed approach is applied off-line in the following verifications.

6.4.1 Extracting and Comparing the Aggregated EVCLs

In this verification of the EEVCLP algorithm, data from the Pecan Street Inc. [39] for each month of 2015 and 2016 are considered. Each month is divided into weekdays and weekends. The reason for this separation is because EV charging behaviors vary between weekdays and weekends. For each month, the EEVCLP algorithm is applied to extract the EVCLs (F matrix). Figure 6.1 shows the daily residential loads for weekdays and weekends of July 2016. The extracted EVCLs from the daily residential loads of Figure 6.1, using the EEVCLP algorithm, are presented in Figure 6.2. Those extracted EVCLs are aggregated and compared with the actual aggregated EVCL demand pattern in Figure 6.3. The error between the actual and extracted aggregated EVCL demand for weekdays and weekends for July 2016 was -3.12% and -4.75%, respectively.

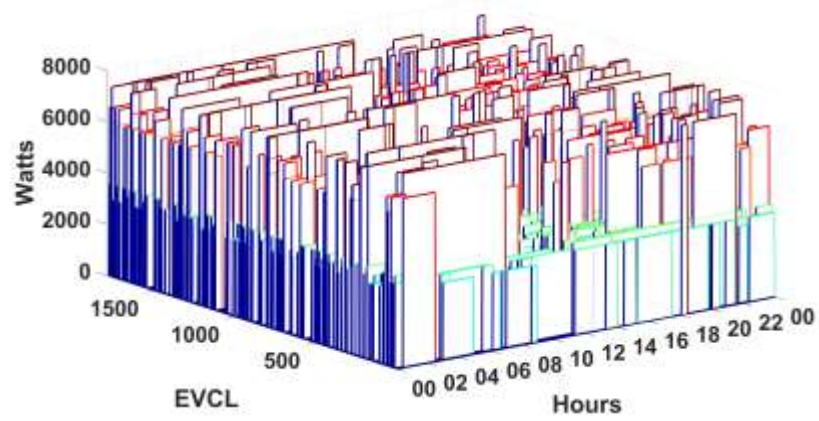


(a)

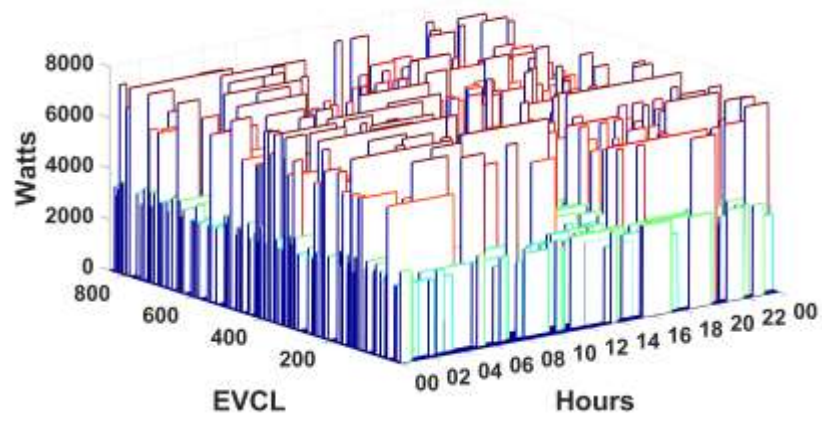


(b)

Figure 6.1. The daily residential loads for July 2016. (a) weekdays. (b) weekends.

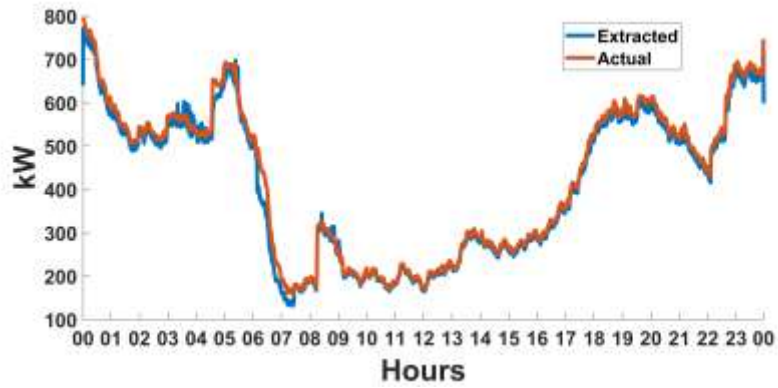


(a)

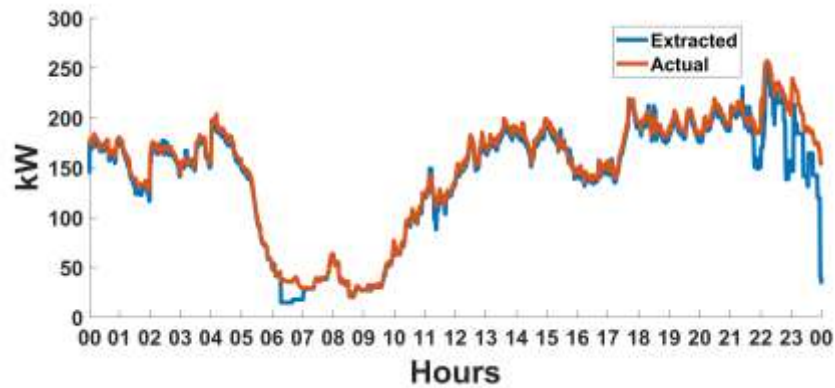


(b)

Figure 6.2. The daily EVCLs for July 2016. (a) weekdays. (b) weekends.



(a)



(b)

Figure 6.3. Comparison between the aggregated actual and extracted EVCLs. (a) weekdays. (b) weekends.

6.4.2 Case Study on Defining Flexibility

In this case study, it is intended to define the flexibility for a certain time frame. For this purpose, the time observation duration is 15-minutes. Each time observation is the sum of EVCLs in that 15-minute period. The general layout of the procedure of this case study is presented in Figure 6.4.

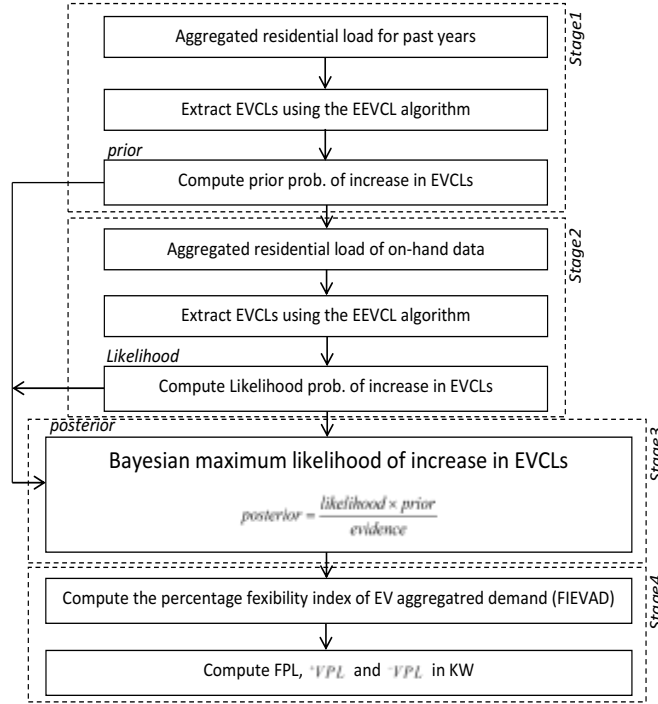


Figure 6.4: General layout of the procedure.

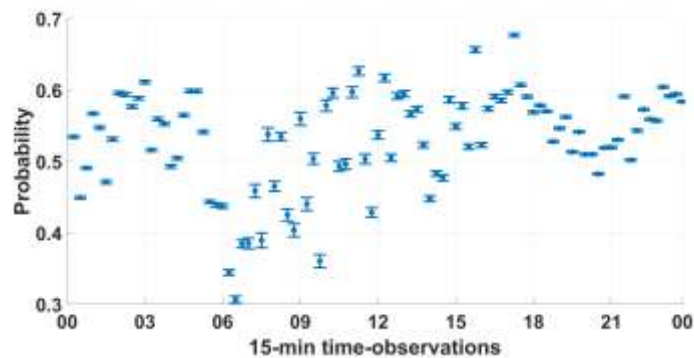
The demand variations for the extracted EVCLs (F) for each month are first modeled using the Bayesian maximum likelihood method (Section 6.2.2). In this illustration, we continue with presenting the results of July. To compute the probability of an increase in EVCL (ω_i):

- 1- First, the prior probability of an increase in EVCLs is computed using (6.7) and (6.8). This corresponds to data of July 2015. However, as mentioned in Section 6.2.2 if prior historical data is unavailable for a certain location, data for another location can be used instead to compute the prior.
- 2- The likelihood of the recent data is computed using (6.9) and (6.10). This corresponds to the extracted EVCLs for July 2016 (Figure 6.3).
- 3- The posterior probability, which corresponds to ω_i can now be achieved by (6.11)-(6.13).

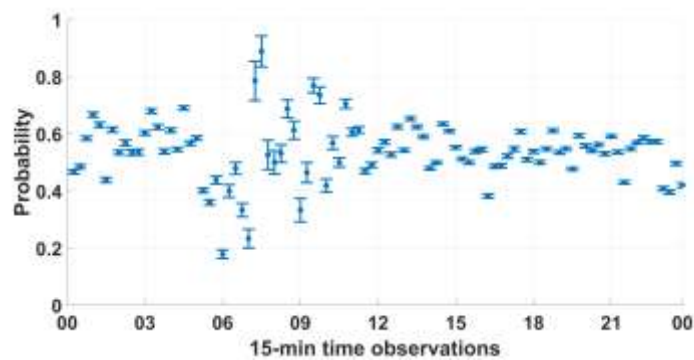
Figure 6.5 presents the probability of an increase in EVCL and its upper and lower limits for July. It can be observed that there is a high probability of an increase in EV charging around

hour 11:00 and hour 17:00 in weekdays. The same effect can be observed between hour 7:00 and 8:00 in weekends. The EV charging behavior between hour 5:00 and hour 7:00 is similar for weekdays and weekends.

The flexibility index (FIEVAD) is computed using (6.20)-(6.23). Figure 6.6 presents the FIEVAD values. It can be noted that lower FIEVAD values appear during the morning and evening periods in weekdays, where there is a collective trend, and the aggregated demand becomes more unlikely to induce DR changes. The same effect can be observed between hour 7:00 and 8:00 in weekends. As mentioned previously, FIEVAD closer to one indicates that the EV charging behavior is random and there is no collective trend, accordingly, there is more chance for inducing DR changes. This effect can be observed more in weekends.

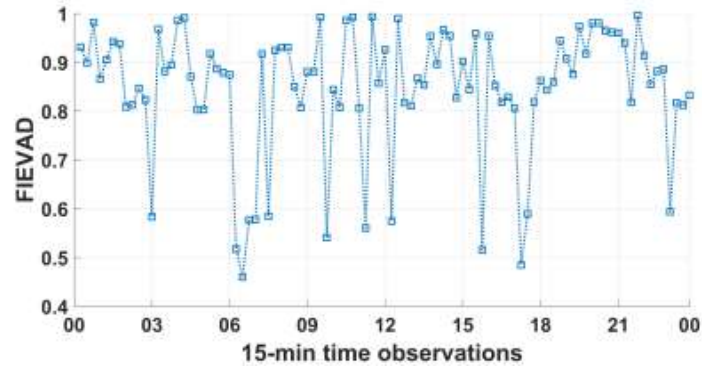


(a)

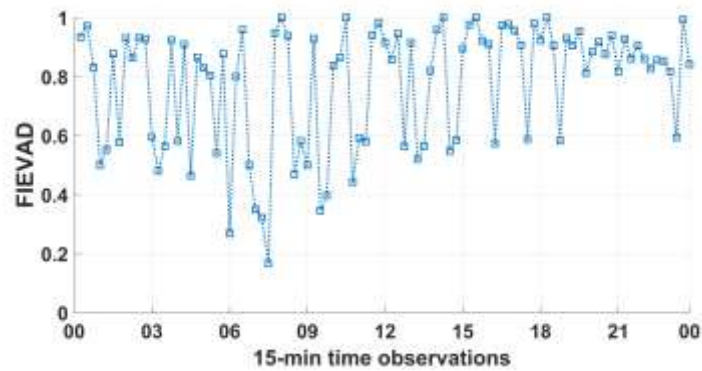


(b)

Figure 6.5: The probability of increase in EVCL and its upper and lower limits. (a) weekdays. (b) weekends.



(a)



(b)

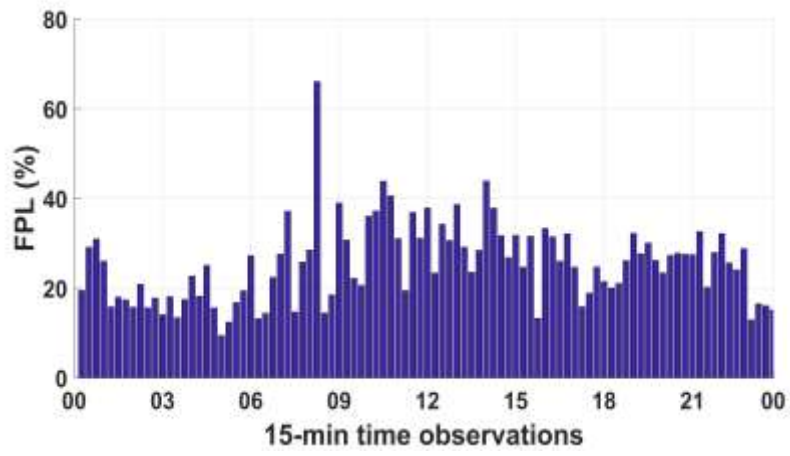
Figure 6.6: The FIEVAD values. (a) weekdays. (b) weekends.

The flexibility percentage (FPL) is presented in Figure 6.7. This represents the percentage of the aggregated EV charging demand that can be reduced or increased without affecting the average change in the collective EV charging. It can be observed that when trend periods arise, the probability to achieve higher FPL values is low.

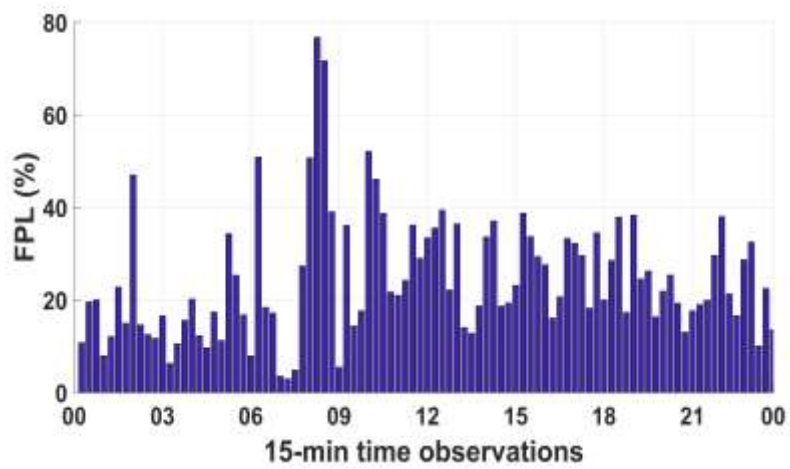
The variability $+VPL$ and $-VPL$ are presented in Figure 6.8. As mentioned previously, these indicators represent the maximum demand variation that could be obtained when all the increase in demand changes to decrease in demand, and vice versa. The yellow bars represent the maximum decrease in collective demand if all the increase in demand changes to decrease in the collective EV charging demand at that time period. Whereas, the red bars represent the maximum increase in demand for a time period if all the decrease in demand changes to increase

in the collective EV charging demand. This can be useful for grid operators to make decisions on which time periods can be selected to establish EV DR programs. The grid operators may involve other inputs for effective decision-making such as, peak loads and electricity price.

The procedure followed in this application is useful for system operators to identify how flexible the aggregated EVCL in different periods of time. This can assist the operators in deciding whether or not it can be viable to incentivize EV charging customers to change their charging demand patterns, taking into account the flexibility information identified to represent the collective behavior of the aggregated EV charging. On the basis of the flexibility indices, the expected response to DR programs can be higher in some time periods and lower in other time periods; accordingly, it may be useless for the operators to propose incentives to customers in time periods where the EV aggregate demand has low flexibility. In particular, in the time periods where the values of the FIEVAD and FPL indicators are low, the decisions aiming to reshape the aggregate EV charging demand is unlikely to be effective. This is because the collective EV charging behavior is following a trend in these time periods, which limits the overall demand flexibility. In other time periods where the FIEVAD and FPL values are higher, the collective behavior of the consumers is random, and there is no clear trend in the changing demand. This suggests that consumers are more available to accept changes, leading to a better ability to reshape the aggregated EV charging demand. The operators could use the FPL as an indicator to represent the percentage of the aggregated EV charging demand that can be reduced or increased without affecting the average change in the collective EV charging. Then, the variability $+VPL$ and $-VPL$ can be used to represent the maximum variation limits when all the increase in demand changes to decrease in demand, and vice versa. This can assist operators in the reshaping of the EV charging demand at the selected time periods, as decreasing the aggregated EV charging demand at one time period may imply an increase in another.

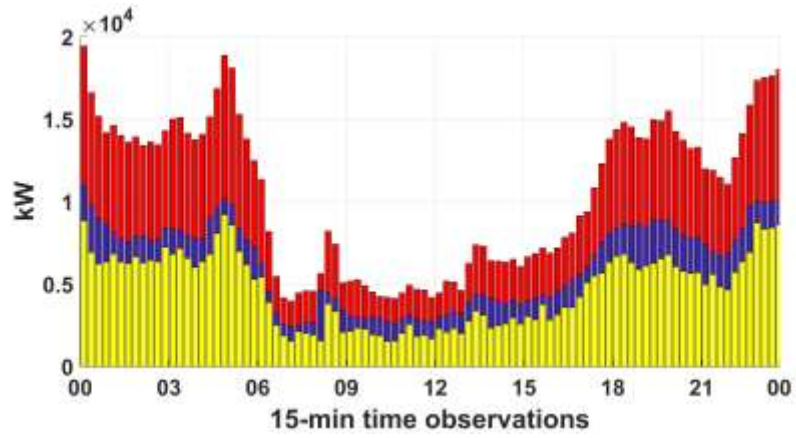


(a)

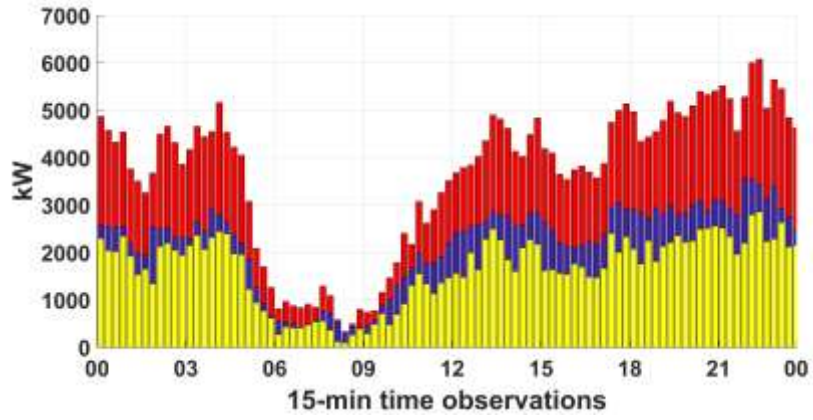


(b)

Figure 6.7. The FPL percentage. (a) weekdays. (b) weekends.



(a)



(b)

Figure 6.8. The 15-min aggregated EV load (blue), +VPL in kW (red) and -VPL in kW (yellow). (a) weekdays. (b) weekends.

6.5 Conclusions

In this chapter, a method to model the probability of an increase in EVCLs, using the Bayesian maximum likelihood, and quantify the flexibility of the aggregated EVCL demand is presented. The flexibility index of EV aggregated demand (FIEVAD) expresses the EV charging behavior in terms of collective trend. A FIEVAD close to one indicates that the EV charging behavior is very random and there is no collective trend. Accordingly, the probability of customers to change their charging behavior is high at those time periods. In contrast, FIEVAD values close to zero, indicates that there is a collective trend in EV charging and the customers are less likely to accept demand response changes at those time periods.

The work presented in this chapter promotes the reliability and economical operation of smart grids and can be useful for grid operators to plan for smart charging decisions on which time periods can be selected to establish DR programs. Also, the utilized indicators based on statistical analysis can assist operators and researchers in understanding time periods where trends in EV charging behaviors may arise and act accordingly.

Chapter 7

Clustering and Targeting EV Charging Customers for Load Shaping

7.1 Introduction

In the previous chapter, a method to quantify the amount of flexibility achievable from the collective EV charging load was presented. In this chapter, it is desired to choose which customers to target for reshaping the EV charging load in a way that promotes the reliability of the smart grid based on those flexibility indices. Figure 7.1 describes the relationship between chapters 5, 6 and this Chapter 7. A case study on the same data used in Chapters 5 and 6 is presented in this chapter.

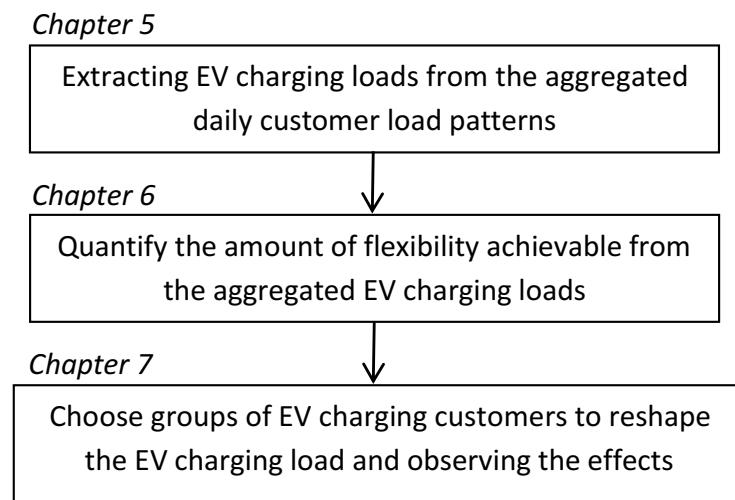


Figure 7.1. Relationship between chapters 5, 6 and 7.

7.2 Clustering Customers into Groups

At each time period, the customers can be grouped into two distinct clusters. The first cluster includes customers that have a low collective or no charging profile during that time period. The second cluster includes customers that are charging their EVs and have a high collective charging profile during the same time period. For example, in hours between 6:00 am and 9:00 am the customers can be grouped into two clusters based on their EV charging behavior. The first group consists of customers who are not charging their EVs, and the other group consists of customers who are charging their EVs at this time period. To achieve this grouping of customers, data mining methods are needed. In this approach, the K-means clustering algorithm is used to group customers into two distinct clusters. Appendix E describes the K-means clustering algorithm in details. The cluster representatives of each cluster could show which group of customers are charging during that time period. RapidMiner Radoop [92] was used to apply the K-means clustering algorithm on the distributed data on top of the eco-system of Chapter 4.

7.3 Methodology to Target Customers and Reshape the EVCL

In this section, customers that are not charging or have low EV charging profiles will be referred to as Cluster#1. Customers in this cluster are used in cases where the EV charging demand is lower than the average daily load or what the operators desire at that time period. The average daily load (ADL) can be computed from historical data, such as the same day of the passed week or year. This group of customers are targeted to assist in increasing the loads. On the other hand, customers that are charging their EVs or have high charging profiles will be referred to as Cluster#2. This group of customers are targeted to assist in reducing the load during peak or high load periods. As reducing the EVCLs at some time periods will increase the charging load at other time periods, customers of Cluster#1 and Cluster#2 are targeted depending on the flexibility indices presented in Chapter 6.

7.3.1 Retrieving and Clustering the EV Charging Customers

To achieve the Cluster#1 and Cluster#2 groupings, the K-means clustering algorithm is run on top of the smart grid big data Lambda architecture eco-system of Chapter 4. The eco-system

retrieves the customers EVCLs from the distributed HDFS repositories. The data is then preprocessed; this includes selecting the time period to be clustered and removing customer profiles with missing data. Once the desired data is preprocessed, the K-means clustering algorithm is applied, and the grouping of EV charging customers is achieved.

7.3.2 Using Flexibility Indices to Reshape EV Load

Once the groupings of customers are obtained, at each time period the flexibility index of EV aggregated demand (FIEVAD) and flexibility percentage level (FPL) are computed as shown in Chapter 6. The FIEVAD expresses the amount of flexibility at a time period. FIEVAD values closer to one indicate that the collective behavior in EV charging is random. At those time periods it is more likely that customers will accept changes. On the other hand, FIEVAD values closer to zero indicate there is a collective trend in EV charging or a collective trend in no charging of EVs. The aggregated EVCL and probability of increase are used to distinguish between a trend in charging and no charging of EVs cases. At time periods when it is desired to reduce the aggregated EVCLs, the FIEVAD value is an indicator to the possibility of achieving that. The FPL express the percentage of EVCL that could be increased or decreased without affecting the average collective EV charging demand. The aggregated EVCL and probability of increase indicate to whether the aggregated load should be increased or decreased.

In cases were the aggregated EVCL is above the average daily load or above what the operators desire, the aggregated EVCL should be reduced by the FPL percentage at that time period and vice-versa. However, the suggested FPL may only be achieved if the customers of the targeted cluster accept to change their behavior. In this case study, it is assumed that each customer has a 50% probability of accepting changes in their EV charging behavior at a specific time period. For that, the suggested FPL at a certain time period may not be reached. The procedure followed to change the EV charging behaviors to reshape the aggregated EVCL (AggEVL) is shown in Figure 7.2. In this chapter the time period to perform clustering is 3-hours.

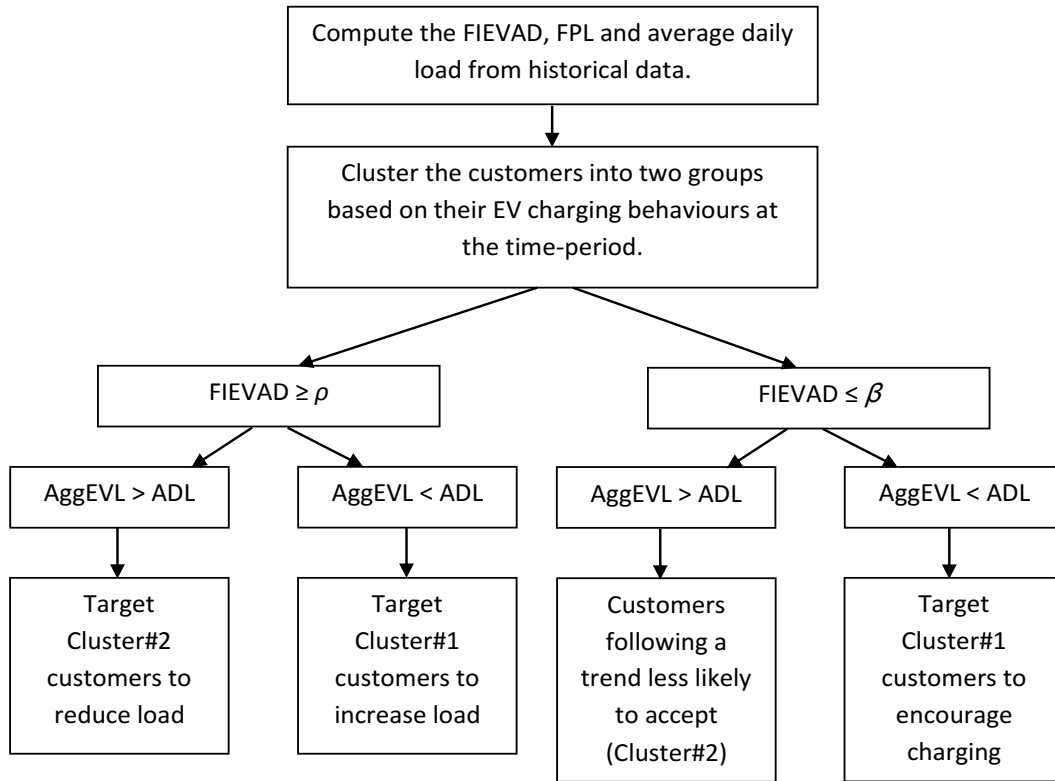


Figure 7.2. The procedure followed to change the EV charging behaviors to reshape the aggregated EVCL (AggEVL) using the average daily load (ADL), ρ the upper FIEVAD and β lower FIEVAD.

7.4 Case Study on Choosing Customers to Reshape EVCL

In this case, it is desired to follow the procedure of Figure 7.2 to choose customers that can potentially smooth the aggregated EVCL closer to the daily average pattern or how operators desire. Then the effects of the changes of customers' behaviors are observed to test the feasibility of the approach in reshaping the EV load.

The first step is to compute the FIEVAD and FPL. From the previous chapter (Chapter 6) the FIEVAD and FPL are used in this case study. Figure 7.3 presents the FIEVAD. The flexibility indices were computed for time periods of 15-minutes.

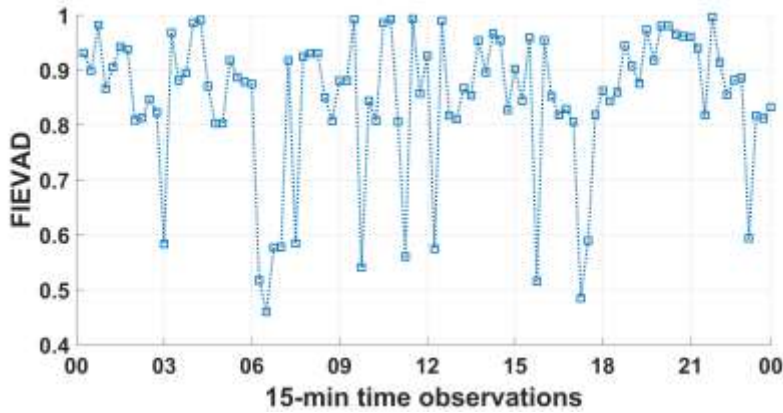


Figure 7.3: The FIEVAD values.

In the next step, the customers are grouped into two clusters every 3-hours. This is because customers often change their EV charging behaviors. For example, in the time period of 6:00 am to 9:00 am, a customer may be classified in Cluster#1 and then in the following time period (9:00 am to 12:00 pm) the customer may be classified as in Cluster#2. Figure 7.4 and Figure 7.5 show the groupings of customers and centroids of the time period from 21:00 to 00:00, respectively. It can be observed from Figure 7.4 (a) that this group are not charging their EVs or have relatively low charging profiles during this 3-hour time period. By contrast, it can be observed in Figure 7.4 (b) that this group have high charging profiles at the same 3-hour period. Figure 7.5 presents the centroids of each cluster. It can be clearly observed that Cluster#1 have a low profile pattern and Cluster#2 have a high profile pattern at this 3-hour time period. In this case study, if the FIEVAD is $\rho \geq 0.80$, this is considered a high value of flexibility, and at this time period the EV charging behavior is random and the customers are more likely to accept change, whereas, if the FIEVAD is $\beta \leq 0.30$, this indicated there is a strong trend in the EV charging behavior and

customers are rigid to accept changes at this time period. The results of targeting customers in Cluster#2 is shown in Figure 7.6. The blue bars of Figure 7.6 express how the aggregated EVCL is after some customers accepting to change their behaviour and not to charge their EVs during this time period. The probability of a customer accepting change was set to 50%. The results of the application through the entire day are shown in Figure 7.7. It can be observed that between the hours of 19:00 and 05:00 the aggregated EVCL was high (red) and by grouping and targeting customers that may have potential in changing their charging behavior the aggregated EVCL was relatively reduced during some time periods. Also, it can be seen around the hours of 09:00, 11:00 and 16:00 there was an increase in the aggregated EVCL. This is because reducing the aggregated EVCL at some periods will cause an increase in other time periods. By following the methodology in this case study, the customers during the hours of 09:00, 11:00 and 16:00 were encouraged to charge their EVs and have accepted to change their charging behaviors. The effect of increasing the probability of customers accepting to change their EV charging behaviours at 70% and 100%, at each time period independently, are presented in Figure 7.8 and 7.9, respectively. The high FIEVAD was set to $\rho \geq 0.80$ and the low FIEVAD was set to $\beta \leq 0.30$. It should be noted that, the probability of customers accepting to change their charging behaviours can be increased by demand-side-management strategies such as, TOU and DR programs.

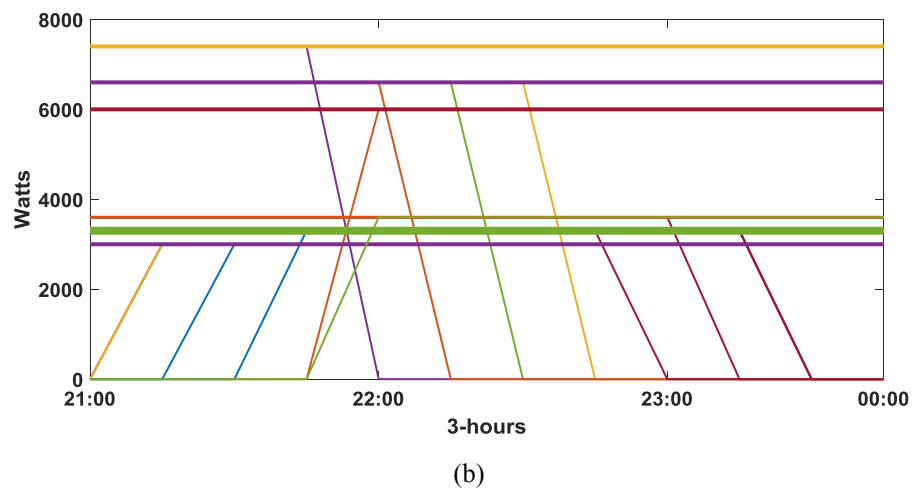
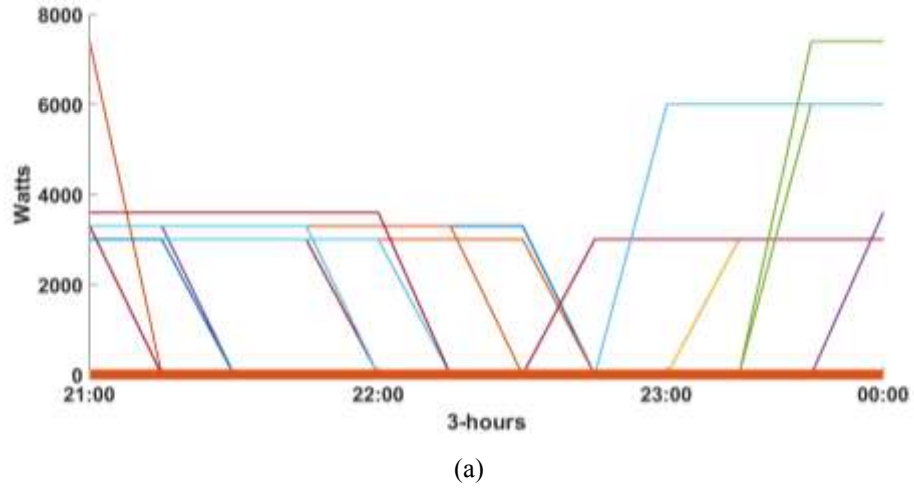


Figure 7.4: The groupings EVCLs. (a) Cluster#1. (b) Cluster#2. The thickness of the lines indicates to multiple EVs.

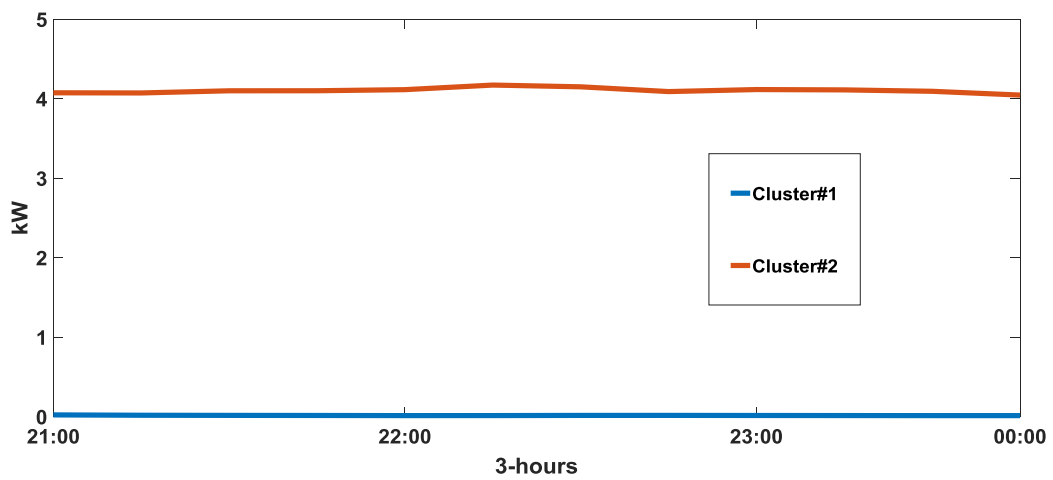


Figure 7.5: The resulted centroids.

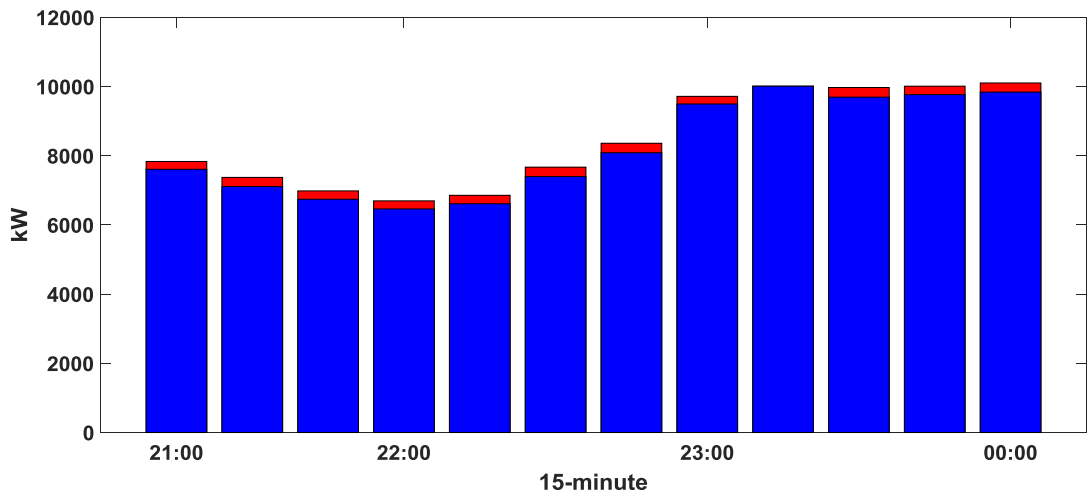


Figure 7.6: The aggregated EVCL before (red) and after customer accepting changes (blue).

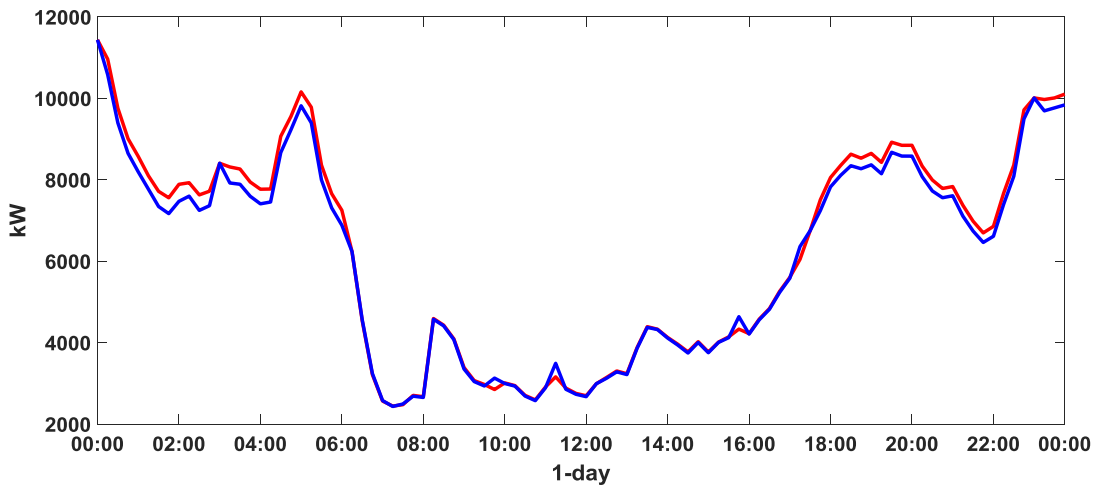


Figure 7.7: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 50% at each time period.

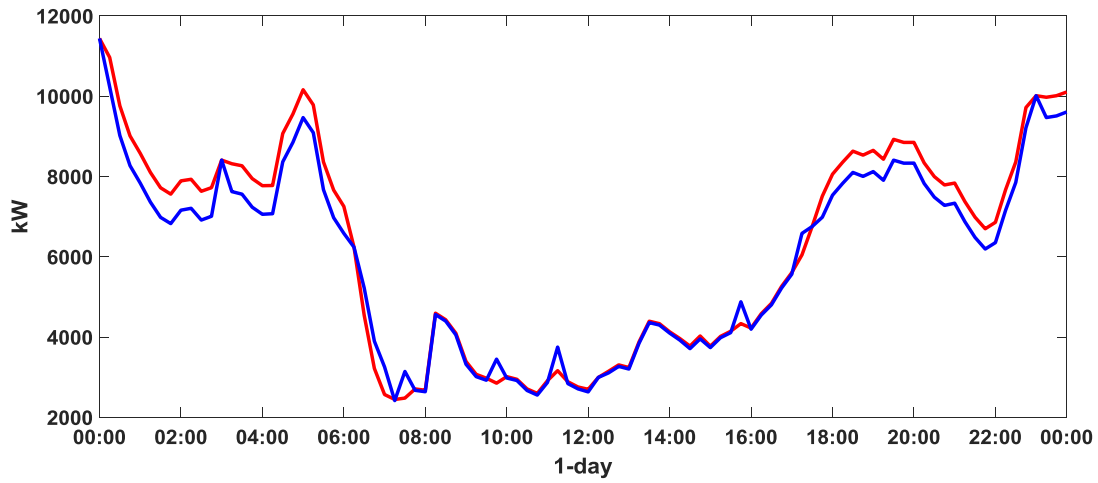


Figure 7.8: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 70% at each time period.

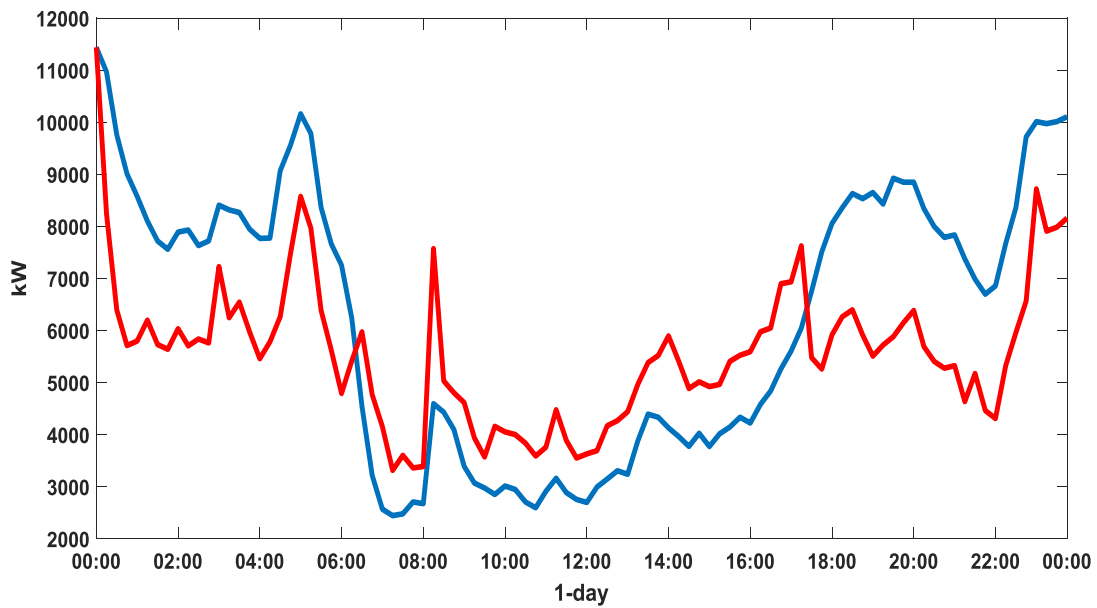


Figure 7.9: The aggregated EVCL before (red) and after customer accepting changes (blue) for an entire day with the accepting probability of 100% at each time period.

7.5 Conclusion

In this chapter, a method to group customers into distinct groupings based on their EV charging behaviors is presented. The customers in those groups are then targeted to assist in reshaping the aggregated EVCL in a way that promotes the reliability of the smart grid. Furthermore, a case study was carried out to test the feasibility of this methodology in reshaping the aggregated EVCL. The FIEVAD and FPL were utilized to express the possibility of customers to accept changes in the EV charging behavior at different time periods. The results of the case study suggest that the methodology could potentially encourage customers to change their EV charging behavior and obtain desired changes to the collective aggregated EVCL. This essentially promotes the reliability and operation of the smart grid. To further encourage the customers to accept changes, incentivizing studies can be performed. Also, studies to reshape the aggregated load in ways that consumes renewable power can be included to reshape the EV load.

Chapter 8

Summary and Future Work

8.1 Summary

This thesis presented a framework to deal with the smart grid big data covering the lifecycle of smart grid from data generation to data analytics. The framework also includes a learn-and-response loop that enables to monitor the effects of the decisions made on the grid. The primary objective was to build a comprehensive framework capable of handling smart grid big data and applying innovative studies related to EVs. To achieve these objectives, state-of-the-art big data components were utilized; also, various analytical methods were proposed to extract EV charging loads, quantify the flexibility of the aggregated EV charging load demand and, choose EV charging customers to target in reshaping the EV collective load. The proposed methods and studies were run on top of the smart grid big data Lambda architecture eco-system.

Chapter 3 presented a big data framework for smart grids. The framework's stages including, data acquisition, data storing and processing, data querying, and data analytics components were discussed in details. Furthermore, the functionality of the Apache Hadoop platform and the features that make it suitable for the smart grid big data management and analysis were discussed. The framework was implemented on a cloud-based platform. Furthermore, visual analytical applications on real data were performed.

Chapter 4 extends the framework of Chapter 3 by presenting a smart grid big data eco-system based on the Lambda architecture. This eco-system is able to handle massive quantities of smart grid data by taking advantage of batch and real-time processing methods. The Lambda architecture design and principals for building batch and real-time processing systems were discussed. The eco-system collects then stores the smart grid big data into a cloud. This allows collecting various types of smart grid data including smart meter data, and image and video data to enable data mining in digital image and video processing applications.

Data mining and visualization applications on real smart grid data were performed. The data mining application was to partition the daily smart meter readings into groups based on the load

consumption. In the visualization application, the presented eco-system was able to overcome the delay in real-time visualization of the previous smart grid big data frameworks by utilizing the Lambda architecture.

In Chapter 5, an algorithm to extract EV charging loads (EVCL) non-intrusively was proposed. The proposed algorithm (EEVCLP) is unsupervised and is able to run on low-frequency real power data. The proposed algorithm utilizes a signal processing method, ICA, to extract the EVCLs from the aggregated load pattern of households. In addition, the proposed algorithm can effectively mitigate the interference of other appliances that have similar load behaviors as EVs.

To verify the effectiveness of the EEVCLP algorithm, the algorithm was applied on real household datasets on top of the smart grid big data Lambda architecture eco-system. The results of extracting the EVCLs from the daily ALP of households were satisfactory. Furthermore, the proposed algorithm was tested by ingesting 1-hour segments to monitor the EV charging behaviors of a neighborhood during the past hour. The results were satisfactory and suggested that the proposed algorithm is able to run on shorter time frames. Aggregating the extracted EVCLs can potentially provide an accurate estimation of the charging demand that is critical for utilities to evaluate the power delivery, and can assist smart grid operators in planning and demand response strategies. Also, further analyses and studies can be carried out from aggregating such charging behaviors of EVs.

Chapter 6 presents a method to model the probability of an increase in EVCLs and quantify the flexibility of the aggregated EVCL demand. The proposed flexibility index of EV aggregated demand (FIEVAD) expresses the EV charging behavior in terms of collective trend. FIEVAD values close to one indicate there is no collective trend in EV charging. In contrast, FIEVAD values close to zero, indicate that there is a collective trend in EV charging and the customers are less likely to accept changes at those time periods. This can be useful for grid operators to plan for informed charging decisions on which time periods can be targeted to establish demand response programs.

In Chapter 7, a methodology to group customers into distinct clusters based on their EV charging behaviors is presented. The customers in those groups are then targeted to assist in

reshaping the aggregated EVCL in a way that promotes the reliability of the smart grid. Furthermore, a case study was performed to test the feasibility of this methodology in reshaping the aggregated EVCL. The results of the case study suggest that the methodology could potentially encourage customers to change their EV charging behavior and obtain desired changes to the collective aggregated EVCL. This essentially promotes the reliability and operation of the smart grid. The probability of customers accepting to change their behaviour could be increased by demand-side-management strategies including, TOU and incentivizing programs.

8.2 Future Work

The following research directions can be followed as an extension out of this thesis:

- Extending the smart grid big data Lambda architecture with components that enable real-time image and video processing applications.
- Proposing data mining algorithms that can be applied on distributed data.
- Proposing supervised algorithms to predict outages.
- The trend of further developing the smart grid big data Lambda architecture eco-system for non-data scientists.
- Enhancing the method of choosing EV charging customers by increasing the probability of accepting changes.
- Consumer-oriented applications to provide feedback to end-users on reducing electricity consumption and saving money through smartphone push notifications.
- Modeling and extracting other major appliances from the aggregated load.

Bibliography

- [1] J. Gantz, and D. Reinsel, “The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east,” *IDC Anal. Future*, 2012.
- [2] H. Hu, Y. Wen, T-S. Chua, and X. Li, “Toward scalable systems for big data analytics: A technology tutorial,” *IEEE Access*, vol. 2, pp. 652-687, May 2014.
- [3] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, “Data mining with big data,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97-107, Jan. 2014.
- [4] IEEE Std 2030-2011, “Guide for smart grid interoperability of energy technology and information technology operation with the electric power system (EPS), and end-use applications and loads,” 2011.
- [5] M. Duvall and E. Knipping, “Environmental assessment of plug-in hybrid electric vehicles,” EPRI Jul. 2007.
- [6] A. Dubey, and S. Santoso, “Electric vehicle charging on residential distribution systems: impacts and mitigations,” *IEEE Access*, vol. 3, pp. 1871-1893, 2015.
- [7] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou, and V. Prasanna, “Cloud-based software platform for data-driven smart grid management,” *IEEE/AIP Comput. Sci. Eng.*, vol. 15, no. 4, pp. 38–47, 2013.
- [8] O. R. Team, *Big Data Now: Current Perspectives from O'Reilly Radar*, Sebastopol, CA, USA: O'Reilly Media, 2011.
- [9] M. Grobelnik. (2012, Jul.). *Big Data Tutorial*. Available [Online]: http://videlectures.net/eswc2012_grobelnik_big_data/
- [10] J. Manyika *et al.*, *Big data: The Next Frontier for Innovation, Competition, and Productivity*. San Francisco, CA, USA: McKinsey Global Institute, 2011, pp. 1-137.
- [11] P. Zikopoulos, and C. Eaton, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, New York, NY, USA: McGraw-Hill, 2011.
- [12] E. Meijer, “The world according to LINQ,” *Commun. ACM*, vol. 54, no. 10, pp. 45-51, Aug. 2011.
- [13] D. Laney, “3d data management: Controlling data volume, velocity and variety,” Gartner, Stamford, CT, USA, White Paper, 2001.
- [14] P. Zikopoulos, D. deRoos, C. Bienko, R. Buglio, and M. Andrews, *Big Data: Beyond the Hype*, New York, NY, USA: McGraw-Hill, 2015.

- [15] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and good practices," *Proc. Contemporary Computing*, pp. 404-409, 2013.
- [16] Y. Demchenko, C. deLaat, and P. Membrey, "Defining architecture components of the big data ecosystem," *Proc. Collaboration Technologies and Systems*, pp. 104-112, May 2014.
- [17] C. L. Stimmel, *Big Data Analytics Strategies for the Smart Grid*, CRC Press, pp. 155-169, 2015.
- [18] X. He, Q. Ai, C. Qiu, W. Huang, L. Piao, and H. Liu, "A big data architecture design for smart grids based on random matrix theory," *IEEE Trans. on Smart Grid*, vol. 8, no. 2, pp. 674-686, 2017.
- [19] A. A. Munshi, and Y. A. I. Mohamed, "Cloud-based visual analytics for smart grids big data," *Proc. IEEE Innovative Smart Grid Technologies*, 2016.
- [20] X. Wu, X. Zhu, G. Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97-107, Jan. 2014.
- [21] H. Khazaei, C. McGregor, M. Eklund, K. El-Khatib, and A Thommandram, "Toward a big data healthcare analytics system: a mathematical modeling perspective," *IEEE World Congress on Services*, 2014.
- [22] J. Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang, "A secure cloud computing based framework for big data information management of smart grid," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 233-244, 2015.
- [23] T. White, *Hadoop: The Definitive Guide*, Sebastopol, CA, USA: O'Reilly Media, Yahoo! Press, 2012.
- [24] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," *Proc. IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1-10, May 2010.
- [25] S. Ghemawat, H. Gobioff, and S-T. Leung, "The google file system," *Proc. 19th ACM Symposium on Operating System Design and Implementation*, Oct. 2003.
- [26] J. Dean, and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Proc. 6th Symposium on Operating System Design and Implementation (OSDI'04)*, Dec. 2004.
- [27] V. K. Vavilapalli *et al.*, "Apache hadoop YARN: Yet another resource negotiator," *Proc. 4th Symposium on Cloud Computing*, 2013.

- [28] N. Marz, and J. Warren, *Big Data: Principles and Best Practices of Scale realtime data systems*, Manning, Apr. 2015.
- [29] M. –S. Tsai, and Y. –H. Lin, “Modern development of an adaptive non-intrusive appliance load monitoring system in electricity energy conservation,” *Applied Energy*, vol. 96, pp. 55-73, 2012.
- [30] H. –H. Chang, C. –L. Lin, and J. –K. Lee, “Load identification in nonintrusive load monitoring using steady-state and turn-on transient energy algorithms,” in *Proc. CSCWD*, Shanghai, China. 2010, pp. 27-32.
- [31] M. Figueiredo, A. de Almeida, and B. Ribeiro, “Home electrical signal disaggregation for non-intrusive load monitoring (nilm) systems,” *Neurocomputing*, vol. 96, pp. 66-73, 2012.
- [32] J. Kolter, S. Batra, and A. Ng, “Energy disaggregation via discriminative sparse coding,” in *Proc. Neural Information Processing Systems*, 2010.
- [33] M. E. Berges, E. Goldman, H. S. Matthews, and L. Soibelman, “Enhancing electricity audits in residential buildings with nonintrusive load monitoring,” *Journal of Industrial Ecology*, vol. 14, no. 5, pp. 844-858, 2010.
- [34] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 2nd ed., Chapman & Hall/CRC, Boca Raton: USA, 2015.
- [35] M. Zeifman, “Disaggregation of home energy display data using probabilistic approach,” *IEEE Trans. Consumer Electronics*, vol. 58, no. 1, pp. 23-31, 2012.
- [36] T. Zia, D. Bruckner, and A. Zaidi, “A hidden markov model based procedure for identifying household electric loads,” in *Proc. IEEE IECON*, Melbourne, Nov. 2011. pp. 3218-3223.
- [37] J. Kolter, and T. Jaakkola, “Approximate inference in additive factorial hmms with application to energy disaggregation,” *Journal of Machine Learning Research*, vol. 22, pp. 1472-1482, 2012.
- [38] M. J. Johnson, and A. S. Willsky, “Bayesian nonparametric hidden semi-markov models,” *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 673-701, 2013.
- [39] Z. Zhang, J. H. Son, Y. Li, M. Trayer, Z. Pi, D. Y. Hwang, and J. K. Monn, “Training-free non-intrusive load monitoring of electric vehicle charging with low sampling rate,” in *Proc. IEEE IECON*, Dallas, 2014. pp. 5418-5425.
- [40] G. Hart, “Nonintrusive appliance load monitoring,” in *Proc. IEEE*, Dec. 1992. pp. 1870-1891.

- [41] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *IEEE Power and Energy Magazine*, vol.1, no. 2, pp. 56-63, 2003.
- [42] R. Fisera, and K. Macek, "Virtual sub-metering via combined classifiers," in *Proc. IDAACS*, vol. 1, pp. 126-131, 2011.
- [43] H. -H. Chang, C. -L. Lin, and J. -K. Lee, "Load identification in nonintrusive load monitoring using steady-state and turn-on transient energy algorithms," in *Proc. CSCWD*, pp. 27-32, 2010.
- [44] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Proc. AAAI*, pp. 356-362, 2012.
- [45] P. Zhang, C. Zhou, B. G. Stewart, D. M. Hepburn, W. Zhou, and J. Yu, "An improved non-intrusive load monitoring method for recognition of electric vehicle battery charging load," *Energy Procedia*, vol. 12, pp. 104-112, 2011.
- [46] M. A. Ortega-Vazquez, "Optimal scheduling of electric vehicle charging and vehicle-to-grid services at household level including battery degradation and price uncertainty," *IET Gener. Transm. Distrib.*, vol. 8, no. 6, pp. 1007–1016, Jun. 2014.
- [47] T. Zhang, W. Chen, Z. Han, and Z. Cao, "Charging scheduling of electric vehicles with local renewable energy under uncertain electric vehicle arrival and grid power price," *IEEE Veh. Technol.*, vol. 63, no. 6, pp. 2600–2612, Jan. 2013.
- [48] Y. Liu, R. Deng, and H. Liang, "Game-theoretic control of PHEV charging with power flow analysis," *AIMS Energy*, vol. 4, no. 2, pp. 379-396, Mar. 2016.
- [49] M. Wang, H. Liang, R. Zhang, R. Deng, and X. Shen, "Mobility-aware coordinated charging for electric vehicles in VANET-enhanced smart grid," *IEEE Journal on Selected Areas in Communications - Smart Grid Communications Series*, vol. 32, no. 7, pp. 1344-1360, July 2014
- [50] V. Aravinthan and W. Jewell, "Controlled electric vehicle charging for mitigating impacts on distribution assets," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 999–1009, Jan. 2015.
- [51] M. Moradijoz, M. Parsa Moghaddam, M. R. Haghifam, and E. Alishahi, "A multi-objective optimization problem for allocating parking lots in a distribution network," *Int. J. Elect. Power Energy Syst.*, vol. 46, pp. 115– 122, Mar. 2013.
- [52] M. R. Aghaebrahimi, M. M. Ghasemipour, and A. Sedghi, "Probabilistic optimal placement of EV parking considering different operation strategies," in *Proc. 17th IEEE Mediterr. Electrotech. Conf. (MELECON)*, Apr. 2014, pp. 108–114.

- [53] Z. Hu, Y. Song, and Z. Luo, "Optimal siting and sizing of electric vehicle charging stations," in *Proc. IEEE Int. Elect. Veh. Conf.*, 2012, pp. 1–6.
- [54] A. El-Zonkoly, H. Ashour, and A. Ahmed, "Optimal allocation, sizing and energy management of PHEV parking lots in distribution system," in *Proc. 5th Int. Renew. Energy Congr. (IREC)*, 2014, pp. 1–5.
- [55] M. Moradijoz, A. Ghazanfarimeymand, M. P. Moghaddam, and M. R. Haghifam, "Optimum placement of distributed generation and parking lots for loss reduction in distribution networks," in *Proc. 17th Conf. Elect. Power Distrib. Netw. (EPDC)*, 2012, pp. 1–5.
- [56] "2012 assessment of demand response and advanced metering," U.S. Federal Energy Regulatory Commission, Washington, DC, Dec. 2012.
- [57] I. Sulyma, K. Tiedemann, M. Pedersen, M. Rebman, and M. Yu, "Experimental evidence: A residential time of use pilot," in *Proc. ACEEE Summer Study Energy Efficiency Buildings*, 2008, pp. 292–304.
- [58] P. L. Joskow and C. D. Wolfram, "Dynamic pricing of electricity," *Amer. Econ. Rev.*, vol. 102, no. 3, pp. 381–385, 2012.
- [59] W. Golden and P. Powell, "Towards a definition of flexibility: in search of the holy grail?," *Omega*, vol. 28, no. 4, pp. 373–384, 2000.
- [60] E. Lannoye, D. Flynn, and M. O'Malley, "Evaluation of power system flexibility," *IEEE Trans. Power Syst.*, vol. 27, no. 2, pp. 922–931, 2012.
- [61] J. Ma, V. Silva, R. Belhomme, D. S. Kirschen, and L. F. Ochoa, "Evaluating and planning flexibility in sustainable power systems," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PES)*, Vancouver, BC, Canada, 2013, pp. 1–11.
- [62] A. Papavasiliou and S. S. Oren, "Large-scale integration of deferrable demand and renewable energy sources," *IEEE Trans. Power Syst.*, vol. 29, no. 1, pp. 489–499, Jan. 2014.
- [63] I. A. Sajjad, G. Chicco, and R. Napoli "Demand flexibility time intervals for aggregate residential load patterns," *Power Tech*, 2015.
- [64] I. A. Sajjad, and G. Chicco, "Definitions of demand flexibility for aggregate residential loads," *IEEE Trans. Smart Grid*, vol. 7, no. 6, pp. 2633–2643, 2016.
- [65] R. Stamminger *et al.*, "Synergy potential of smart appliances," Fed. Ministry Environ., Nat. Conserv., Nuclear Safety, Bonn, Germany, Tech. Rep. Project Smart, 2008.

- [66] A. Safdarian, M. Fotuhi-Firuzabad, and M. Lehtonen, "Benefits of demand response on operation of distribution networks: A case study," *IEEE Syst. J.*, vol. 10, no. 1, pp. 189–197, Mar. 2016.
- [67] M. Alizadeh, A. Scaglione, A. Applebaum, G. Kesidis, and K. Levitt, "Reduced-order load models for large populations of flexible appliances," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 1758–1774, Jul. 2015.
- [68] VMware, "Apache flume and apache sqoop data ingestion to apache Hadoop clusters on vmware vsphere," VMware, CA, USA, White Paper, 2013.
- [69] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu., P. Wyckoff, and R. Murthy, Hive: A warehousing solution over a mapreduce framework," *Proc. Very Large Data Bases*, vol. 2, no. 2, pp. 1626-1629, Aug. 2009.
- [70] J. Li, "Design of real-time data analysis system based on Impala," *IEEE Workshop in Advanced Research and Technology in Industry Applications*, pp. 934-936, Sept. 2014.
- [71] Apache Mahout, *Apache*, [Online]. Available: <https://mahout.apache.org/docs/0.13.0/>
- [72] A. Bifet, and G. D-F. Morales, "Big data stream learning with SAMOA," IEEE Data Mining Workshop, pp. 1199-1202, Dec. 2014.
- [73] Tableau Software. Tableau. [Online]. Available: https://onlinehelp.tableau.com/current/pro/desktop/enus/help.htm#save_savework_packagedworkbooks.html
- [74] N. Serrano, G. Gallardo, and J. Hernantes, "Infrastructure as a service and cloud technologies," *Software, IEEE*, vol. 32, no. 2, pp. 30-36, 2015.
- [75] F. Fakhar, and M. A. Shibli, "Management of symmetric cryptographic keys in cloud based environment," *Advanced Communication Technology (ICACT)*, Jan. 2013.
- [76] Cloudera Inc. "Cloudera ODBC driver for impala", 2016, Available [Online]: <http://www.cloudera.com/documentation/other/connectors/impala-odbc/latest/Cloudera-ODBC-Driver-for-Impala-Install-Guide.pdf>
- [77] N. Serrano, G. Gallardo, and J. Hernantes, "Infrastructure as a service and cloud technologies," *Software, IEEE*, vol. 32, no. 2, pp. 30-36, 2015.
- [78] F. Fakhar, and M. A. Shibli, "Management of symmetric cryptographic keys in cloud based environment," *Advanced Communication Technology (ICACT)*, Jan. 2013.
- [79] Lichman, M. (2013). UCI Machine Learning Repository <http://archive.ics.uci.edu/ml>

- [80] Solar Radiation Research Laboratory (BMS), Available [Online]: http://www.nrel.gov/midc/srrl_bms
- [81] Kingspan wind, “KW3 Small Wind Turbines,” RAL 9005, datasheet.
- [82] A. A. Munshi, and Y. A.-R. I. Mohamed, “Big data framework for analytics in smart grids,” *Electric Power Systems Researcrh*, vol. 151, pp. 369-380, Oct. 2017.
- [83] SUNPOWER, “E20/435 Solar Panel,” SPR-435NE-WHT-D datasheet, 2011.
- [84] G. M. Masters, *Renewable and Efficient Electric Power Systems*, John Wiley & Son, pp. 505-531, 2004.
- [85] A. P. Robinson, P. T. Blythe, M. C. Bell, Y. Hubner, and G. A. Hill, “Analysis of electric vehicle driver recharging demand profiles and subsequent impacts on the carbon content of electric vehicle trips,” *Energy Policy*, vol. 61, pp. 337-348, 2013.
- [86] M. Alonso, H. Amaris, J. G. Germain, and J. M. Galan, “Optimal charging scheduling of electric vehicles in smart grids by heuristic algorithms,” *Energies*, vol. 7, no. 4, pp. 2449-2475, 2014.
- [87] Irish Social Science Data Archive (ISSDA), Available [Online]: www.ucd.ie/issda
- [88] R. Smith, “Assault on California power station raises alarm on potential for terrorism,” *Wall Street J.*, pp. 1–7, Feb. 2014.
- [89] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: cluster computing with working sets. *In Proc. HotCloud '10*, 2010.
- [90] M. Guller, *Big Data Analytics with Spark: A practitioner’s Guide to Using Spark for Large Scale Data Analysis*, Apress, 2015.
- [91] M. Armbrust, R. Xin, C. Lian, Y. Yuai, D. Liu, J. Bradley, X. Meng, T. Kaftan, M. Franklin, A. Ghodsi, and M. Zaharia. Spark SQL: Relational data processing in spark. *In ACM Special Interest Group on Management of Data*, 2015.
- [92] RapidMiner, “RapidMiner Radoop Documentation,” Apr. 2015. Available [Online]: <https://docs.rapidminer.com/downloads/rapidminer-radoop-manual.pdf>
- [93] Matlab, “Tall Arrays,” [Online]: https://www.mathworks.com/help/matlab/import_export/tall-arrays.html
- [94] Pecan Street Inc. Dataport (2015). <http://www.pecanstreet.org>

- [95] B. Malysiak-Mrozek, M. Stabla, and D. Mrozek, "Soft and declarative fishing of information in big data lake," in *IEEE Trans. on Fuzzy Systems*, vol. 26, no. 5, pp. 2732-2747, 2018.
- [96] P. Comon, "Independent component analysis: a new concept?," *Signal Processing*, pp. 287-314, 1994,.
- [97] C. Jutten, and J. Herault, "Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1-10, 1991.
- [98] EVObsession, (2015). Electric car charging 101 – types of charging, charging networks, apps and more!”, [Online]. Available: <http://evobsession.com/electric-car-charging-101-types-of-charging-apps-more/>
- [99] M. P. Wand, "Data-based choice of histogram bin width," *Stat. Comput. Graphics*, vol. 51, no. 1, pp. 59-64, 1997.
- [100] Synergy, The KaleidaGraph Guide to Curve Fitting, 2006, [Online]: <http://www.synergy.com/Tools/curvefitting.pdf>
- [101] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements," in *Proc. SIAM*, pp. 747-758, 2011.
- [102] R. Duda, P. E. Hatt, and D. G. Stork, *Pattern Classification*, Wiley, 2000.

Appendix A

Flume Agent Configuration

```
# Active Flume Components
FlumeDatLakeAgent1.sources = FDLsource
FlumeDatLakeAgent1.channels = FDLmemoryChannel
FlumeDatLakeAgent1.sinks = FDLHDFSsink

# Setting the source to spool directory where the file exists
FlumeDatLakeAgent1.sources.FDLsource.type = spooldir
FlumeDatLakeAgent1.sources.FDLsource.spoolDir = /usr/local/flume/smarmeters
FlumeDatLakeAgent1.sources.FDLsource.bind=10.142.0.8
FlumeDatLakeAgent1.sources.FDLsource.port=10010

# Setting the channel to memory
FlumeDatLakeAgent1.channels.FDLmemoryChannel.type = memory
# Max number of events stored in the memory channel
FlumeDatLakeAgent1.channels.FDLmemoryChannel.capacity = 1000000
FlumeDatLakeAgent1.channels.FDLmemoryChannel.batchSize = 1000000
FlumeDatLakeAgent1.channels.FDLmemoryChannel.transactioncapacity = 1000000

# Use a channel which buffers events in file
FlumeDatLakeAgent1.channels.FDLmemoryChannel.type = file

FlumeDatLakeAgent1.FDLmemoryChannel.checkpointDir = /usr/local/TempDLBuffer
FlumeDatLakeAgent1.channels.FDLmemoryChannel.dataDirs = /usr/local/TempDLBuffer2/Data2

# Setting the sink to HDFS
FlumeDatLakeAgent1.sinks.FDLHDFSsink.type = hdfs
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.path = hdfs://10.142.0.2/user/flume/smarmeters
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.fileType = DataStream

# Write format can be text or writable
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.writeFormat = Text
# use a single csv file at a time
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.maxOpenFiles = 10

FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.rollCount=10000
```

```
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.rollInterval=0
#Each file how many bytes
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.rollSize = 0
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.batchSize =1000

# rollover file based on max time of 1 min
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.rollInterval = 0
FlumeDatLakeAgent1.sinks.FDLHDFSsink.hdfs.idleTimeout = 60

# Connect source and sink with channel
FlumeDatLakeAgent1.sources.FDLsource.channels = FDLmemoryChannel
FlumeDatLakeAgent1.sinks.FDLHDFSsink.channel = FDLmemoryChannel
```

Appendix B

Commands to Configure Spark for Matlab and Read/Store into HDFS

`% defining the Spark cluster and properties.`

```
cluster = parallel.cluster.Hadoop(...  
'HadoopInstallFolder','/opt/cloudera/parcels/CDH/lib/hadoop', ...  
'SparkInstallFolder','/opt/cloudera/parcels/CDH/lib/spark/lib/spark-2.2.0-bin-hadoop2.6');  
cluster.SparkProperties('spark.driver.memory') = '4096m';  
cluster.SparkProperties('spark.executor.memory') = '4096m';  
cluster.SparkProperties('spark.yarn.executor.memoryOverhead')='2048';  
mapreducer(cluster);
```

`% setting up environment variables`

```
setenv('HADOOP_HOME','/opt/cloudera/parcels/CDH/lib/hadoop')  
setenv('HADOOP_PREFIX','/opt/cloudera/parcels/CDH/lib/hadoop')  
setenv('MATLAB_HADOOP_INSTALL','/opt/cloudera/parcels/CDH/lib/hadoop')
```

`% pointing to the HDFS location and defining data as tall table.`

```
ds = datastore('hdfs://master/user/LoadsTab'); %HDFS location  
tt = tall(ds);
```

`% storing into HDFS.`

```
formatOut = 'dd-mm-yy_HHMM'; %Timestamp  
command = ['hadoop fs -copyFromLocal ./loadsJan.csv /user/LoadsTab/Jan' datestr(now,formatOut)]; %Copy to HDFS  
status = system(command);
```

Appendix C

Preprocessing of ICA

Before applying the ICA algorithm on data, it is useful to do some preprocessing, to make the problem of ICA estimation simpler and better conditioned. The used preprocessing techniques are:

Centering:

A common and necessary preprocessing step is to center \mathbf{x} . This is achieved by subtracting its mean vector $m = E\{x\}$:

$$x_c = x - m \quad (\text{C.1})$$

This implies that \mathbf{s} is a zero-mean vector as well, by taking expectations on both sides of (5.1):

$$s = W(x_c + m) \quad (\text{C.2})$$

From this, all observation vectors are assumed to be centered, whereas, the mixing matrix remains unaffected after this centering step.

Whitening:

Another useful step in the preprocessing is to whiten the observed vector (\mathbf{x}). Whitening involves linearly transforming the observation vector \mathbf{x} , such that a new whitened vector \mathbf{x}_w that has components that are uncorrelated and have their variances equal unity is obtained:

$$E\{x_w x_w^T\} = I \quad (\text{C.3})$$

A simple method to perform the whitening transformation is to use the eigenvalue decomposition (EVD) of the covariance matrix:

$$E\{xx^T\} = EDE^T \quad (\text{C.4})$$

where E is the orthogonal matrix of eigenvectors of $E\{xx^T\}$ and D is the diagonal matrix of its eigenvalues $D = \text{diag}(d_1, \dots, d_n)$. The observation vector can be whitened by the following transformation:

$$x_w = ED^{-0.5}E^T x \quad (\text{C.5})$$

where the matrix $D^{-0.5}$ is computed by a simple component-wise operation:

$$D^{-0.5} = \text{diag}(d_1^{-0.5}, \dots, d_n^{-0.5}) \quad (\text{C.6})$$

Whitening transforms the mixing matrix into A_w . From (5.1) and (B.5):

$$x_w = ED^{-0.5}E^TAs = A_w s \quad (\text{C.7})$$

hence,

$$\begin{aligned} E\{x_w x_w^T\} &= A_w E\{s s^T\} A_w^T \\ &= A_w A_w^T \\ &= I \end{aligned} \quad (\text{C.8})$$

The main purpose of whitening is to reduce the number of parameters to be estimated. Instead of having to estimate the n^2 elements of the original matrix A , we only need to estimate the new orthogonal mixing matrix, where an orthogonal matrix has $n(n-1)/2$ degrees of freedom.

Appendix D

Modified F-Score

The advantage of using the modified F-score metric is that it involves all possible extraction outcomes for x , the algorithm extracted value, and x_0 , the ground truth value:

- The true negative (TN) is when $x=0$ and $x_0 = 0$.
- The false negative (FN) is when $x=0$ and $x_0 > 0$.
- The false positive (FP) is when $x>0$ and $x_0 = 0$.
- The accurate true positive (ATP) is when $x>0$ and $x_0 > 0$, and $\frac{|x-x_0|}{x_0} \leq \rho$.
- The inaccurate true positive (ITP) is when $x>0$ and $x_0 > 0$, and $\frac{|x-x_0|}{x_0} > \rho$.

where the error threshold $\rho = 0.2$.

The modified F-score not only measures the accuracy of the state of the EV charging, it also measures the accuracy of the extracted amplitude by involving the ATP (accurate amplitude estimation) and ITP (inaccurate amplitude estimation).

The modified F-score can be defined as the harmonic mean of the *precision* and *recall*, and are computed by:

$$Precision = \frac{ATP}{ATP + FP + ITP} \quad (D.1)$$

$$Recall = \frac{ATP}{ATP + FN + ITP} \quad (D.2)$$

$$F_{score} = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad (D.3)$$

Appendix E

K-means Clustering Algorithm

The K-means is one of the most popular partitioned clustering methods. It groups a set of N input data points into K clusters using an iterative procedure. The number of K clusters is a user-specified parameter. The average of all data points in a cluster is the representative data point (centroid). The main goal of K-means is to minimize the sum of square error over all K clusters.

The K-means clustering can be summarized by the following steps:

1. Initialize K data points randomly or on some prior knowledge, $C = [C_1, C_2, \dots, C_K]$.
2. Calculate the distances between each data point x and centroid C , assign each data point to the nearest centroid.

$$x_i \in C_w, \quad \text{if } \|x_i - C_w\| < \|x_i - C_j\| \\ \text{for } i = 1, \dots, N, j \neq w, \text{ and } j = 1, \dots, K \quad (\text{E.1})$$

3. Recalculate the centroid for each cluster.

$$C_k = \frac{1}{N_k} \sum_{x \in C_k} x \quad (\text{E.2})$$

where N_k is the number of data points in C_k .

Repeat steps 2 and 3; terminate when there is no change for each cluster.