Real-Time Embedded Low-Frequency Load Disaggregation

by

Stephen Makonin

BTech, British Columbia Institute of Technology, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2014

APPROVAL

Name:	Stephen Makonin
Degree:	Doctor of Philosophy (Computer Science)
Title of Thesis:	Real-Time Embedded Low-Frequency Load Disaggregation
Examining Committee:	Dr. Jim Delgrande Chair
	Dr. Fred Popowich, Professor, Computing Science, Senior Supervisor
	Dr. Verónica Dahl, Professor, Computing Science, Supervisor
	Dr. Lyn Bartram, Associate Professor, Interactive Arts + Technology, Supervisor
	Dr. Ivan V. Bajić, Associate Professor, Engineering Science, Supervisor
	Dr. Arthur (Ted) Kirkpatrick, Associate Professor, Computing Science, Internal Examiner
	Dr. José R. Martí, Professor, Electrical and Computer Engineering, The University of British Columbia, External Examiner
Date Approved:	August 5, 2014

Partial Copyright Licence

Abstract

Understanding how the electrical appliances and devices in a house consume power is an important factor that can allow occupants to make intelligent and informed decisions about conserving energy. This is often not an easy task. These electrical loads can turn ON and OFF either by the actions of occupants, or by automatic sensing and actuation (e.g. thermostat). Even if we could keep track of when loads turned ON and OFF, it is difficult to understand how much a load consumes at any given operational state because of the lack of proper measurement reporting within equipment manuals. Occupants could buy sensors that would help, but this comes at a high financial cost. Power utility companies around the world are now replacing old electro-mechanical meters with digital meters (smart meters) that have enhanced communication capabilities. These smart meters are essentially a free sensor that offer an opportunity to use computation to infer what loads are currently running in a house and to estimate how much each load is consuming, a process often referred to as load disaggregation.

This thesis presents a new load disaggregation algorithm (i.e. a disaggregator). This new disaggregator uses a super-state hidden Markov model and a new Viterbi algorithm variant which preserves dependencies between loads and can disaggregate multi-state loads, all while performing computationally efficient exact inference. Our sparse Viterbi algorithm can efficiently compute sparse matrices with a large number of super-states. Our disaggregator is the first of its kind to run in real-time on an inexpensive embedded processor using low sampling rates (e.g. per minute). Other contributions of the research include an analysis of electrical measurements, the release of a publicly available dataset, and a method for comprehensive accuracy testing and reporting.

Keywords: nonintrusive load monitoring; NILM; energy modelling; hidden Markov model; Viterbi algorithm; sustainability

To all the continuous power signals that I continuously discretized and quantized.

"Dans les champs de l'observation le hasard ne favorise que les esprits préparés."

"In the field of observation, chance favours only the prepared mind."

— Louis Pasteur, LECTURE, UNIVERSITY OF LILLE*, 1854*

Acknowledgments

First, I would like to thank the professors that help me along the way. To Fred Popowich my Senior Supervisor (SFU Computing Science), who believed in me and set me free to explore. To Verónica Dahl my Supervisor (SFU Computing Science), who provided valuable feedback on socioeconomic issues. To Lyn Bartram my Supervisor (SFU Interactive Arts + Technology), who showed me the importance of interaction and information visualization. To Ivan V. Bajić my Supervisor (SFU Engineering Science), whose expertise in signal processing help me develop my disaggregator. To Alan Mackworth my Depth Examiner (UBC Computer Science), who introduced me to computational sustainability through his course CPSC 530M. To José R. Martí my Thesis External Examiner (UBC Electrical and Computer Engineering) and Arthur (Ted) Kirkpatrick my Thesis Internal Examiner (SFU Computing Science), who provided valuable feedback on my thesis. To Jim Delgrande the Thesis Examination Chair (SFU Computing Science), who kept examination discussions on track. To Roger Alex Clapp and his PhD student Laura Guzman Flores (SFU Geography), who help me develop the Consumer Bill of Rights for Energy Conservation.

My research was partly supported by grants from: the Natural Sciences and Engineering Research Council of Canada (NSERC); and the Graphics, Animation, and New Media Network of Centres of Excellence of Canada (GRAND NCE). Finally, for the many scholarships and fellowships I was awarded, thanks to: Simon Fraser University (SFU), SFU Faculty of Applied Science, and SFU Graduate Student Society.

Last but not least, my personal heart felt thanks. To my Mom, who patiently and enthusiastically helped me proofread this thesis. To my Dad, who taught me how to fight through things and never give up. To Anna, for her love, support, and understanding. To Cyan, who's love and smile melted all the stress away, as only a child can do. To the practice of Yoga, for its ability to centre and heal the mind and the body. To Bob Gill my dear friend and BCIT colleague, who acted as a mentor and help me understand power.

Contents

List of Tables

List of Figures

List of Algorithms

PREFACE

This thesis and the research it describes is very much an interdisciplinary topic. Professors from different schools needed to be part of my supervisory committee. Computing Science (my main area), allowed me to learn about the analysis and design of machine learning algorithms, optimization and intelligent systems, and computational sustainability. Engineering Science allowed me to understand discrete random processes and probabilistic models. Interactive Arts and Technology allowed me to explore human-home interaction problems and made me aware of the need for design considerations that take into account the occupants of a home. There was also unofficial support from the School of Energy (at British Columbia Institute of Technology) which helped me understand power engineering and electronics fundamentals. It was important to understand load disaggregation from a transdisciplinary point of view.

This thesis is an original intellectual product of myself, Stephen Makonin, from research performed during 2010–2014, inclusively. There are a number of my peer-reviewed publications, of which I was the first author, that have influenced the compilation of this thesis: conference papers [1–10] and journal articles [11–14] .

On June 3 (2014) I presented my main contribution at the Second International Workshop on Non-Intrusive Load Monitoring (NILM) which is held once every two years. This was a one day workshop for all disaggregation researchers to gather, present findings, and network. There were over 90 participants from academia, government, and industry. Only 8 papers out of 23 submissions were accepted to present their work. There was a demonstration paper (previously published) and the remaining 14 papers were accepted as posters. Out of the 8 papers accepted for presentation, my paper [5] was 1 of only 3 papers that were accepted for algorithm advancement. I am honoured from this recognition I received by my research community elders and peers.

CHAPTER 1

INTRODUCTION

Load disaggregation is a topic studied by computational sustainability researchers who investigate algorithms that try to discern what electrical loads (i.e. appliances) are running within a physical area where power is supplied from the main power meter. Such physical areas can include communities, industrial sites, office towers / buildings, homes, and even within an appliance. When focusing on homes, this area of research is often referred to as nonintrusive (appliance) load monitoring (NILM or NIALM). By knowing what loads are running, when they are running, and how much power they are consuming, we can begin to make informed choices that can lead to a reduction in power consumption.

Load disaggregation research first began with a call in 1989 from the Electric Power Research Institute (EPRI) [15]. Shortly after, Sultanem [16] and then Hart [17] published their first research results. Most disaggregators (or load disaggregation algorithms) have a number of generalizable steps (see Figure 1.1). One of the first steps is to use *priors* (e.g. datasets, which we will discuss in Chapter 4) and a *pre-tuner* and *model builder* to build models of appliance power consumption. A *meter* is needed to measure the amount of aggregate power a house is currently consuming. Using the readings from a whole-house power *meter* (see Chapter 2) and appliance *models*, together with an *inference algorithm*, we can infer what state each appliance is in and use an *estimator* to estimate the amount of power it is consuming (see Chapter 3). Some disaggregators use an *active tuner* for tuning the appliance models continuously. The pre-tuner and active tuner are optional due to the choice of modelling type used (see Chapter 3 for more details).

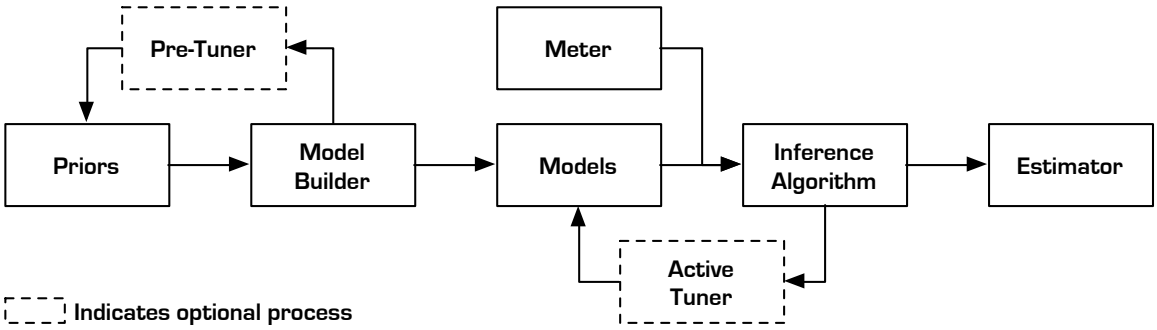


Figure 1.1: *A block diagram of a typical disaggregator. In an initial phase, priors are used to build models that can be pre-tuned. Once models are built, they along with aggregate meter data are used to infer what loads are on and how much they are consuming. These models can be actively tuned to improved accuracy.*

In this thesis we focus on contributions to load disaggregation in the home and argue while there is a focus on the home, the algorithms presented can be scaled up to a community level and down to the appliance level. We also discuss the importance of real-world solutions which need to be reflected in disaggregator design. We believe that load disaggregation should be a part of the smart meter and should only exist on the *home area network* side of the meter to alleviate any privacy concerns (see Figure 1.2). However, the disaggregator can run as separate equipment in the home. In either case no data would ever be sent to the power utility. We also believe the disaggregation data should be owned by the occupants and must be made available to the occupants so that they can make informed energy conservation decisions regardless of their socio-economic situation [2]. This *smarter* smart meter would be able to inform the occupants of a home as to what appliances are running and how much power they are consuming **without** sending any data to the power utility company. As such, we briefly discuss how load disaggregation is an essential part of the smart home, as well as any eco-feedback devices within the home.

In order for load disaggregation to be practical, it will need to run in real-time on commodity embedded hardware that is inexpensive and consumes little power. Inexpensive, to overcome the adoption barrier of cost and low power consumption, so that occupants spend as little money as possible to reduce the amount of their power utility bill. Such goals can create motivation for the *dumb* smart meter to become a *smarter* smart meter. Cost is a key consideration when utility companies need to roll-out millions of smart meters. Load disaggregation plays a key part in power utility company initiatives such as time-of-day usage tariffs and demand response. These initiatives are designed to offset the demand in

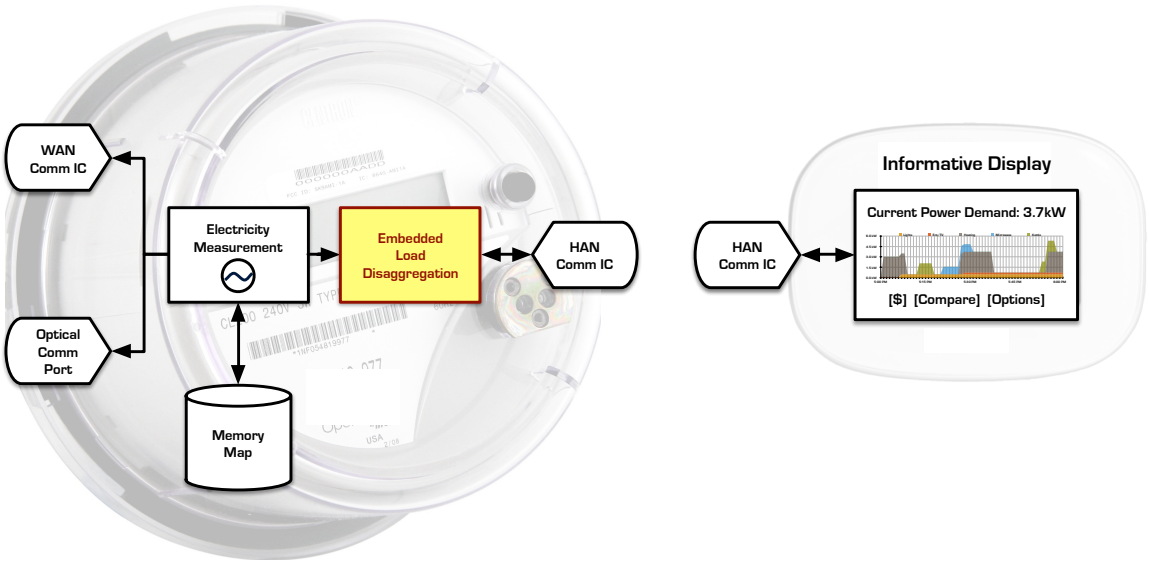


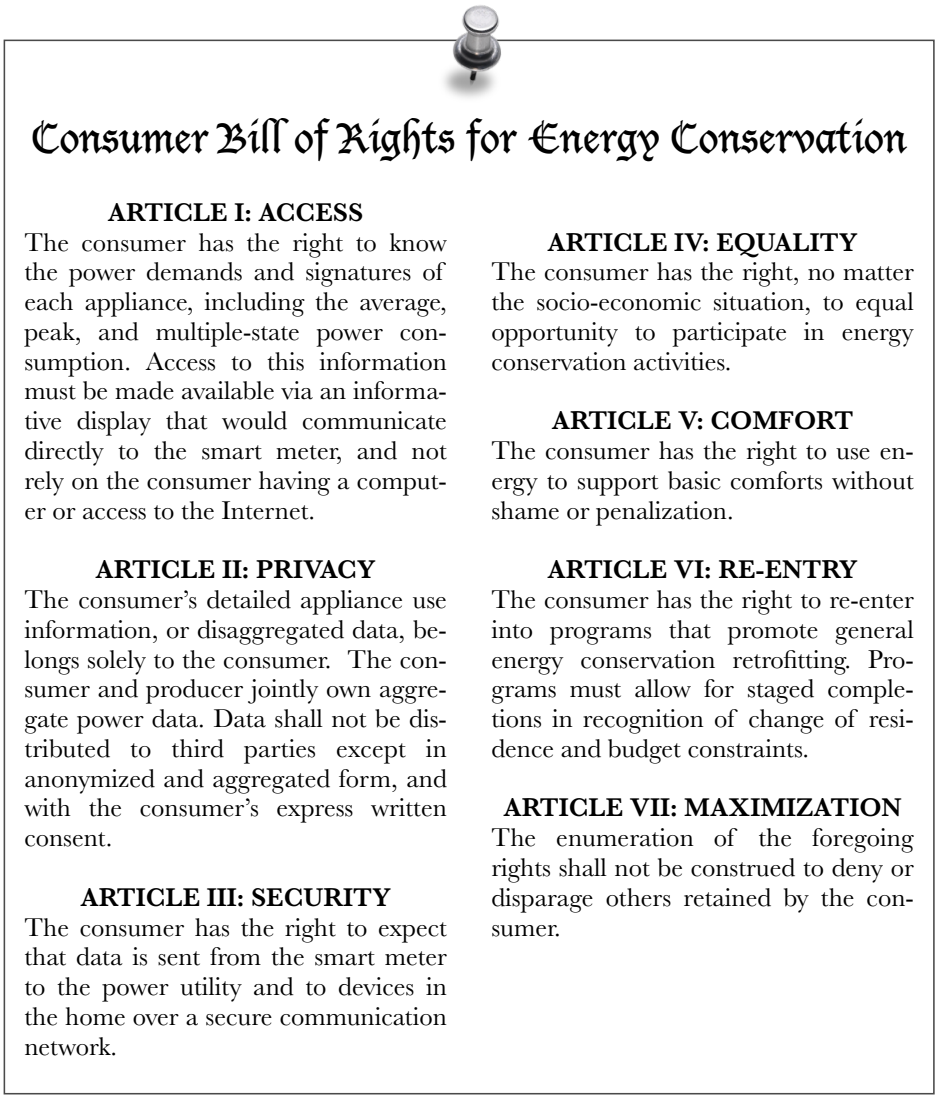
Figure 1.2: *Proposed physical system components: (left) the "smarter" smart meter, and (right) an informative display. Note that the Embedded Load Disaggregation integrated circuit (IC) module is placed on the home area network (HAN) side of the meter and only communicates with devices on the HAN side in order to maintain occupant privacy. At anytime no data would ever be sent to the power utility.*

power consumption at critical times to avoid power grid outages.

1.1 Motivation

Residential homes consume about 34% of the total power consumption in the USA and their consumption is projected to increase to 39% by 2030 [18]. Homeowners and occupants can play a part in the conservation solution. But, before this can happen, homeowners need to be informed about how they consume. Many studies [19–23] have shown that informed homeowners and occupants can and do reduce consumption (between 5% to 15%) when they are aware of their consumption behaviour. A recent study [24] showed that 80% of participants want to have access to disaggregation data (i.e. knowing how their appliances consume energy) and believed that every one should have access to this information. The study also shows that when load disaggregation information is made available to occupants, those occupants can reduce their energy consumption by an average of 14%. In fact, more research is showing that in order for occupants to reduce (on average) their energy consumption by more the 9.2%, real-time appliance specific consumption information is needed [18, 25].

Providing devices, such as plug-level meters, in a home can greatly improve our ability to conserve energy in an intelligent way, but such devices can be expensive. Occupants on low or fixed incomes will need to conserve more, as prices rise, to stay within their budgetary means. Many would find the cost of such equipment unaffordable. We recently wrote about the need for everyone to participate in energy conservation regardless of their socio-economic situation [2], where we proposed a *Consumer Bill of Rights for Energy Conservation* (Figure 1.3).



Consumer Bill of Rights for Energy Conservation

ARTICLE I: ACCESS
The consumer has the right to know the power demands and signatures of each appliance, including the average, peak, and multiple-state power consumption. Access to this information must be made available via an informative display that would communicate directly to the smart meter, and not rely on the consumer having a computer or access to the Internet.

ARTICLE II: PRIVACY
The consumer's detailed appliance use information, or disaggregated data, belongs solely to the consumer. The consumer and producer jointly own aggregate power data. Data shall not be distributed to third parties except in anonymized and aggregated form, and with the consumer's express written consent.

ARTICLE III: SECURITY
The consumer has the right to expect that data is sent from the smart meter to the power utility and to devices in the home over a secure communication network.

ARTICLE IV: EQUALITY
The consumer has the right, no matter the socio-economic situation, to equal opportunity to participate in energy conservation activities.

ARTICLE V: COMFORT
The consumer has the right to use energy to support basic comforts without shame or penalization.

ARTICLE VI: RE-ENTRY
The consumer has the right to re-enter into programs that promote general energy conservation retrofitting. Programs must allow for staged completions in recognition of change of residence and budget constraints.

ARTICLE VII: MAXIMIZATION
The enumeration of the foregoing rights shall not be construed to deny or disparage others retained by the consumer.

Figure 1.3: *The Consumer Bill of Rights for Energy Conservation.*

Access to rich information about appliance consumption, no matter the socio-economic situation of a household, is paramount for everyone to participate in energy conservation activities. This is because the dynamic nature of homes and occupants. We cannot begin to predict nor understand what type of energy conservation strategies different homes would use. We need only provide rich information about appliances and let the occupants make their own decisions based on their needs, opportunities, and comfort. So now we know having appliance-level consumption information is needed to have occupants achieve their maximum energy conservation potential. We need to understand key concepts that can see the success of load disaggregation as the pivotal driver for being the appliance-level information provider. From our title we take three key concepts that have motivated our research: real-time, embedded, and low-frequency. Additionally, other key concepts we need to review include: accuracy, the home area network, data privacy, informative display, data noise, deferrable actions and a load of loads.

Real-Time Real-time is the ability to disaggregate the power signal faster than the frequency it is sampled at. This also includes the concept that disaggregated data would be reported to the occupants in real-time so that the information being shown matches the activity of the house. Any lag in time between power measurement and informative feedback would cause problems for occupants in identifying what consumption events just occurred. Informing occupants of what appliances are ON and how much they are consuming after the fact causes a discontinuity in the understanding of how their actions can be linked with energy consumption. We will perform an experiment with this idea in Section 5.7.

Embedded Embedded is the ability to have the disaggregator run on low power, resource constrained, inexpensive commodity processors. This is important because any solution that requires a large amount of computation would in and of itself consume a large amount of energy while running. It may in fact cost more to run the disaggregator than the savings realized by its use to conserve energy. An embedded disaggregator is also important because we have argued that the disaggregator should exist in the smart meter so all occupants can receive information about how loads or appliances in their home consume energy. This information could then be used to make informed decisions about how best to conserve energy based on choice and preferences of the occupants. We will discuss this in more detail in Section 2.5 with a

supporting experiment in Section 5.7.

Low-Frequency Low-frequency is the ability to disaggregate at a sampling rate faster than the frequency of 1Hz or less. This is important because high-frequency metering (e.g. 15kHz) requires high performance sensors and a high data network bandwidth to send that information to informative displays and other devices in the home. High performance measurement sensors are expensive, which would be an adoption barrier by: (a) the utility having a disaggregator in the smart meter, or (b) having occupants buy an off-the-shelf disaggregator product. A high data network bandwidth is also an issue because of equipment cost and the potential network saturation due to a high volume of data being sent over the data network. We will discuss this in more detail in Chapter 2.

Accuracy The success of any disaggregator can be determined by a measure of its accuracy. However, by its very nature, a disaggregator must be measured for its accuracy performing different functions. The accuracy of a meter is often predetermined by the product we have purchased. The action of having the disaggregator classify at what state each load is in, needs to be measured separately from its function of estimating the rate of consumption. This is due to the fact classification and estimation are often two separate and distinct functions that the disaggregator executes. Different accuracy measurement techniques need to be used for each function. We will discuss this in more detail in Chapter 4.

Home Area Network (HAN) The HAN allows the smart meter and other devices to be able to communicate with each other in the confines of the home. Devices that exist outside the home cannot and should not be communicated with. Current wireless HAN standards, such as ZigBee [1], are considered low bit rate, meaning high volumes of data easily cause network saturation. Any disaggregation information being sent over such a network would need to be sent at a low-frequency rate (especially when the disaggregator is separate from the meter). We only introduce this concept as background information here.

Data Privacy Data Privacy is the assurance that disaggregation data is owned by the occupants of the home and is only communicated and stored within the devices in the

[1] Our research is not tied to ZigBee, is it communication protocol agnostic.

HAN. As depicted in Figure 1.2, the disaggregator should exist only on the HAN side of the smart meter. The power utility and equipment outside the HAN should not have access to disaggregation data. Again, we only introduce this concept as background information here.

Informative Display An informative display is a display unit with the capability to visualize disaggregated load data in various graphical forms. This display does not require the homeowner to have a computer or access to the Internet – this is important especially in the case of low-income households [2]. At the very least, the informative display must have the capacity to collect the smart meter data directly over the HAN to avoid data privacy issues.

Data Noise Data noise can be understood as unexpected or unaccounted for anomalies that can appear in the stream of data that an algorithm analyzes. This can take a number of forms when looking at disaggregation. There can be missing readings that leave gaps in a time series of data. There can be data streams that have timestamps that are out of sync. There can be corrupted data where data measurements within the reading are missing or measured wrongly due to sensor miscalculation or malfunction. Aside from miscalculation or malfunction, data can contain Gaussian noise due to small fluctuations in sensor/ADC (analog-to-digital converter) precision and/or the consumption of power by a load. Specifically for disaggregation, noise can be unmetered loads that create large unexpected patterns of energy consumption. Unless otherwise stated when we discuss noise (which we do throughout this thesis) we refer to the noise from unmetered loads.

Deferrable Actions There are two broad categories of actions that occupants can perform: deferrable and non-deferrable actions. Deferrable actions are those types of actions that occupants can postpone, actions that occupants do not necessarily need to perform now. For example, having the dishwasher run during periods of the day when the charge per kWh is less, resulting in a reduced power bill. Such loads are large consumers of power and more easily identifiable; for example, washing and drying clothes, washing the dishes with a dishwasher, baking with a wall oven, heating or cooling a house. We say they can be deferrable most of the time because there may be situations where such appliances need to be used to attain a certain level of occupancy comfort. For instance, on a very cold day it may be necessary for occupants to

turn up the heat to prevent the house from getting too cold. Non-deferrable actions are actions that cannot be delayed and could cause inconvenience and discomfort to occupants, and in some situation be unsafe. For instance, the occupant must turn the light ON to go downstairs if it is dark even though the price per kWh is high during that time of the day. As we have discussed in previous work [10], we believe that load disaggregation only be highly accurate to identify loads that have deferrable actions. Being that these loads are high consumers of power, this means that high accuracy is achievable as we will show in Chapter 5.

A Load of Loads Our final concept is *a load of loads*. The majority of modern loads/appliances are complex and multi-state consisting of sub-loads not just a *simple ON/OFF* behaviour[2]. Our philosophy is that a load has loads at any level–be it an appliance, room, home, or neighbourhood. So any disaggregator should be able to disaggregate complex and multi-state loads very accurately. Further, to support disaggregation at various levels, a disaggregator must be agnostic of low-frequency rates and measurement types (e.g. current, power, etc.). We will discuss this in more detail in Chapter 3.

1.2 Example Scenario

Consider an example scenario [8], describing the activity of an occupant in a studio suite over the course of one-hour (5:00pm – 6:00pm). Our homeowner lives in Vancouver (BC, Canada) in a small 38m^2 studio suite. The local power utility charges 10¢/kWh. It still gets cold outside so the heating occasionally turns on. Heating is provided by two electric baseboard heaters. In December, sunset is around 5:10pm, and it gets dark quickly. Before starting to cook dinner, she makes a cup of tea using her electric kettle. On TV, local news starts at 5:30pm for a half-hour and she usually eats dinner while watching. Our homeowner uses the microwave to prepare a hot meal. Towards the end of the local news, during a commercial break, she uses the electric kettle to boil more water.

Using the appliance information in Table 1.1, can we easily determine what loads turned ON/OFF and when, by examining Figure 1.4? This might be a task akin to balancing a cheque book, if done in a short period of time after the events have occurred. However, as

[2]Hart [17] identified four basic types: simple ON/OFF, finite-state, constantly on, and continuously variable.

Table 1.1: *Studio Suite Appliances*

Appliance	Power	Description
Lights	480 W	8 Incandescent 60W Bulbs
Ent/TV	250 W	Panasonic 50 Plasma TV
Heating	3.0 kW	2 Cadet 1500W Baseboard
Microwave	1.1 kW	Panasonic Convection
Kettle	1.6 kW	Cuisinart Cordless 1.7L

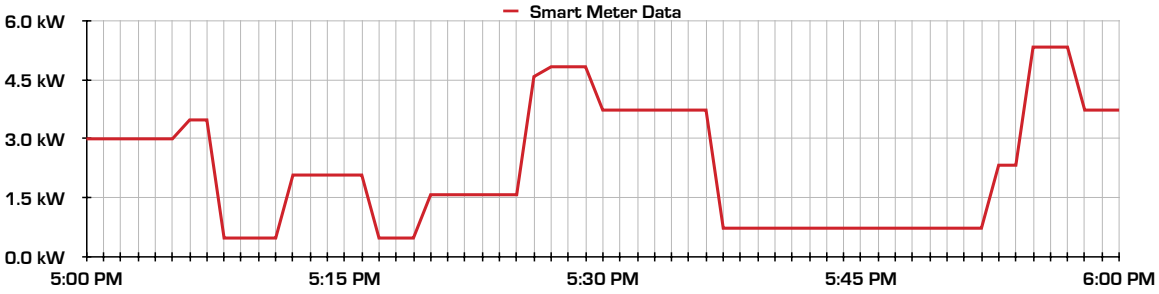


Figure 1.4: *This line chart is typical of the type of aggregate data that the power utility provides the customer via a web portal that the customer can login to using a web browser.*

time passes such a task becomes increasingly more difficult to impossible, especially if the above scenario is recalled from memory [26,27]. How could we solve this problem?

1.3 Our Contributions

We can solve the problem raised in the previous section computationally by using load disaggregation. We can create a disaggregator that uses low-frequency data and can run on an embedded processor (low power, computation, and storage) in real-time with a high degree of accuracy. A good disaggregator can fill in the blank area under the smart meter data line in Figure 1.4. With rich information about loads determined by our disaggregator a chart like in Figure 1.5 can then be created. Table 1.3 lists the load consumption details. Our research, which is addressed in this thesis, can do this and presents six significant contributions that further advance the field of disaggregation. These are listed below.

Contribution 1: We have examined the area of measurements, often overlooked by many disaggregation researchers. We wanted to examine the best way to measure and the best measurement to use for a disaggregator. One of the first investigations we

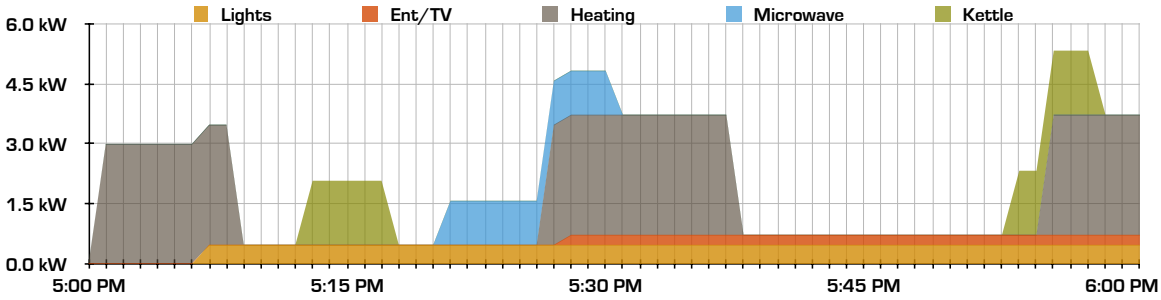


Figure 1.5: *This area line chart fills in the blank area under the aggregate line in Figure 1.4. This fill represents how the aggregate total at any given time is a summation of all the disaggregated loads. An area chart like this should be provided to the occupants on an informative display. This disaggregated data is stored and communicated privately within the home. See Table 1.3 for the exact data used.*

Table 1.2: *Power Usage Details Used for the Example Scenario (1/min sampling), Part 1*

Time Step	Mains	Lights	Ent/TV	Heating	Microwave	Kettle
5:00 PM	3000	0	0	3000	0	0
5:01 PM	3000	0	0	3000	0	0
5:02 PM	3000	0	0	3000	0	0
5:03 PM	3000	0	0	3000	0	0
5:04 PM	3000	0	0	3000	0	0
5:05 PM	3000	0	0	3000	0	0
5:06 PM	3480	480	0	3000	0	0
5:07 PM	3480	480	0	3000	0	0
5:08 PM	480	480	0	0	0	0
5:09 PM	480	480	0	0	0	0
5:10 PM	480	480	0	0	0	0
5:11 PM	480	480	0	0	0	0
5:12 PM	2080	480	0	0	0	1600
5:13 PM	2080	480	0	0	0	1600
5:14 PM	2080	480	0	0	0	1600
5:15 PM	2080	480	0	0	0	1600
5:16 PM	2080	480	0	0	0	1600
5:17 PM	480	480	0	0	0	0
5:18 PM	480	480	0	0	0	0
5:19 PM	480	480	0	0	0	0
5:20 PM	1580	480	0	0	1100	0
5:21 PM	1580	480	0	0	1100	0
5:22 PM	1580	480	0	0	1100	0
5:23 PM	1580	480	0	0	1100	0
5:24 PM	1580	480	0	0	1100	0

Table 1.3: *Power Usage Details Used for the Example Scenario (1/min sampling), Part 2*

Time Step	Mains	Lights	Ent/TV	Heating	Microwave	Kettle
5:25 PM	1580	480	0	0	1100	0
5:26 PM	4580	480	0	3000	1100	0
5:27 PM	4830	480	250	3000	1100	0
5:28 PM	4830	480	250	3000	1100	0
5:29 PM	4830	480	250	3000	1100	0
5:30 PM	3730	480	250	3000	0	0
5:31 PM	3730	480	250	3000	0	0
5:32 PM	3730	480	250	3000	0	0
5:33 PM	3730	480	250	3000	0	0
5:34 PM	3730	480	250	3000	0	0
5:35 PM	3730	480	250	3000	0	0
5:36 PM	3730	480	250	3000	0	0
5:37 PM	730	480	250	0	0	0
5:38 PM	730	480	250	0	0	0
5:39 PM	730	480	250	0	0	0
5:40 PM	730	480	250	0	0	0
5:41 PM	730	480	250	0	0	0
5:42 PM	730	480	250	0	0	0
5:43 PM	730	480	250	0	0	0
5:44 PM	730	480	250	0	0	0
5:45 PM	730	480	250	0	0	0
5:46 PM	730	480	250	0	0	0
5:47 PM	730	480	250	0	0	0
5:48 PM	730	480	250	0	0	0
5:49 PM	730	480	250	0	0	0
5:50 PM	730	480	250	0	0	0
5:51 PM	730	480	250	0	0	0
5:52 PM	730	480	250	0	0	0
5:53 PM	2330	480	250	0	0	1600
5:54 PM	2330	480	250	0	0	1600
5:55 PM	5330	480	250	3000	0	1600
5:56 PM	5330	480	250	3000	0	1600
5:57 PM	5330	480	250	3000	0	1600
5:58 PM	3730	480	250	3000	0	0
5:59 PM	3730	480	250	3000	0	0
6:00 PM	3730	480	250	3000	0	0

performed, as published in [3,7], was to analyze what electrical measurement is best for disaggregation, which we found to be current (I). This is mainly due to the fact that current is more of a stable measurement than power. However, we did not want our disaggregator to be limited to only using current (as we will mention below). In Chapter 2 we examine everything about measurements and show through a number of investigations supporting our claim.

Contribution 2: Another area we have examined and published about is the matrix sparsity found in hidden Markov models [5]. Others have in the past, as well. However, their disaggregators have implemented complex solutions to take advantage of sparsity which is not alway efficient. We present an efficient way to take advantage of sparsity in matrix storage and processing. We show how to do this by using a super-state hidden Markov model, which was previously dismissed because of state exponentiality. Our disaggregator presents new algorithms that are not just factorial variants. Part of our disaggregator is a new Viterbi algorithm variant, called *sparse Viterbi algorithm* that can efficiently process very large sparse matrices (>1,500,000 states [5]). Chapter 3 discusses in detail our disaggregator and provides investigations about efficiency claims.

Contribution 3: We have contributed back to the load disaggregation community by releasing a publicly available dataset, called AMPds. The creation of datasets takes a considerable amount of time, effort, and money, but is important for researchers who want to test their disaggregators but may not have the resources available to create their own dataset. Publicly available datasets are also important as they contribute to reproducibility by having a common way to compare results between different disaggregators. AMPds is now widely used in many different types of research around the world [3] and stands on its own as a valuable contribution. See Chapter 4, Section 4.4.1, for details about how AMPds was created.

Contribution 4: Yet another area we have examined is the practise of reporting how tests were run and what are the best accuracy metrics to use. Many research papers that are released pertaining to load disaggregation often leave out experimental testing details and/or they overstate accuracy claims. We have published a guide of best practises [14] where we introduced a comprehensive set of requirements needed to describe the reporting of experimental results in a paper. One such measure, the

percent-noisy measure (%-NM) defined in (4.9) reports how noisy the testing data was. Another measure, finite-state f-score (FS f-score) is a new type of f-score accuracy measure that allows the reporting of non-binary classification accuracies (i.e. the classification of multi-state loads). We detail this in Chapter 4.

Contribution 5: Our final and main contribution combines all previous contributions to deliver our disaggregator (μDisagg) [5,7]. To provide support to our claims, we test μDisagg using multiple, publicly available, datasets (Chapter 5) and report our accuracy results in detail (using our reporting method in Section 4.4.3). Our experimentation will also show that our discrete model's accuracy is equal to, if not better than, the accuracy of continuous models while performing disaggregation faster. Our specific sub claims for this contribution are as follows.

Contribution 5a: μDisagg is agnostic of low-frequency sampling rates and measurement types, e.i. can disaggregate different measurements and different speeds.

Contribution 5b: μDisagg is highly accurate at load state classification and load consumption estimation.

Contribution 5c: μDisagg is the first algorithm to run in real-time (e.g. Arm Cortex-M3) using low-frequency data on inexpensive commodity embedded processors.

Contribution 5d: μDisagg can disaggregate appliances with complex multi-state power signatures.

Contribution 5e: μDisagg is the first HMM solution that preserves dependencies between loads.

Contribution 5f: μDisagg can perform computationally efficient exact inference, while other methods only use approximate methods.

In Chapter 6, we finish with some concluding remarks about and discuss the limitations of our contributions, and we present possible future work.

Chapter 2
Metering

One of the first things we will look at is metering (**Contribution 1**). Not only is it fundamental to the beginning of the *live* disaggregation process, it is also fundamental for the collection of priors and for the creation of models. In order to be able to disaggregate, the disaggregator uses a power meter to measure the power on the line(s) that need monitoring. Load disaggregation is considered single-point sensing. Single-point sensing, as the name suggests, is to monitor power consumption for one point, usually the main power line that enters a home – the point where the power utility places a meter on the side of a house. There are a number of reasons why single-point sensing is attractive: (a) relatively low cost, (b) less invasive installation, and (c) a generalized solution. As smart meters are installed on homes, the single sensor needed for load disaggregation is present. There is no need to install additional costly sub-meters/sensors. There are two main sampling categories that also need to be considered: high-frequency (>1Hz) [28,29], and low-frequency ($\leqslant$1Hz) sampling. The type of frequency chosen has an impact on what measurement types we may want to use. This will be discussed in following sections.

We start off by giving a high-level review of measurement types (Section 2.1) and discuss multi-point sensing (Section 2.2) to complement the following sections. Disaggregators can fall under two general categories when looking at metering: those that use complex meter data (Section 2.3); and those that use basic data, like data recorded from a smart meter (Section 2.4). We then present our approach to metering (Section 2.5) which is supported by an investigation we performed that compared electrical current to real and

apparent power (Section 2.5.1). We finish with a worked example (Section 2.6) that reflects on the example scenario we presented in Section 1.2.

2.1 Measurement Basics

Often, as in our case, disaggregators use meters that measure the alternating current (AC) signal entering the home. There are many textbooks that can provide an in depth discussion of electronics [30] – we will attempt to provide a simplified introduction that suits our purpose. For metering, the most basic measurements are voltage, current, apparent power, and frequency. **Voltage** ($\triangle$V, measured in volts or V) is the potential difference between two points in a circuit, often referred to as tension or pressure. In North America, the potential difference between each of the phase lines (L1 or L2) and the neutral line entering the home is 120V. **Current** (I, measured in amperes or A) is the movement of electric charge that (in an AC system) periodically reverses direction. **Apparent power** (S, measured in volt-amperes or VA) is simply the product of voltage and current or V$\times$I. **Frequency** (f, measured in hertz or Hz) is the number of cycles per second that voltage cycles at. In North America, electricity is supplied to customers at 60Hz or 60 cycles per second. The time period between cycles can be calculated by $\frac{1}{f}$.

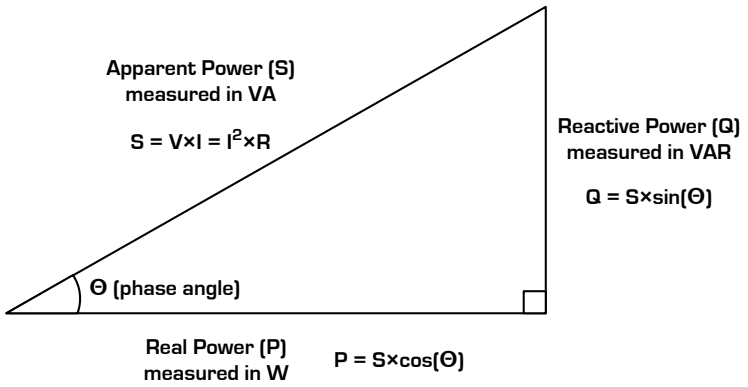


Figure 2.1: *The Power Triangle.*

There are intermediate measurements derived from basic measurements: real power, power factor, reactive power, and energy. The power triangle (Figure 2.1) shows the relation ship between the three different power measurements: apparent, real and reactive

power. **Real power** (P, measured in watts or W), sometimes known as active power or average power, is the net transfer of energy in one direction. This is different than apparent power (the gross transfer) because there is an additional amount of transfer needed to overcome the opposing energy transfer from reactive components (e.g. inductive motors) in a circuit. **Power factor** (PF) or $\cos(\Theta)$ is the ratio between real power and apparent power (or $\frac{P}{S}$) and Θ is the angle between voltage and current. The power factor is unity (1) when the voltage and current are in phase, and zero when the current leads or lags the voltage by $90\,^{\circ}$. Power factors are usually stated as *leading* or *lagging* to show the sign of the phase angle of current with respect to voltage. **Reactive power** (Q, measured in volt-ampere-reactive or VAR) is the rate at which power is stored and released back by components such as capacitors and inductors. While power is an instantaneous rate measurement, **energy** is the amount of power consumed over time and is often the measurement that appears on your power utility bill as kilowatt-hour or kWh. For example, 1kWh is equivalent to having a 1000W light bulb ON for 1 hour.

There are also advanced measurements which require signal analysis: harmonic distortion, electromagnetic interference, transients, and electrical characteristics. **Harmonic distortion** is caused by frequency components that cause unsmooth characteristics in the voltage's sine wave (for background information see [31, 32]). **Electromagnetic interference** (EMI) is caused by sources such as integrated circuits that introduce electrical currents on a power line causing a noisy current signal. **Transients** commonly occur as either a brief spike in power when a load is turned ON or a brief dip in power when a load is turned OFF before the power signal returns to a steady-state. The amplitude and duration of these spikes and dips are very different for each load and can be used to identify different loads, providing that the meter can sample at a high rate. Electrical characteristics can be seen as the metadata of measurements. For example, aggregate values (minimum, maximum, average) and eigenvalues. Eigenvalues (EIG) require an advanced understanding of power systems. As such, only Liang et al. [29, 33] has ever discussed them.

2.2 Multi-Point Sensing

Multi-point sensing is the ability to monitor power consumption from more than one location. This can be done by metering one or more breakers within a house power panel (commonly known as branch circuit power metering or BCPM, e.g. DENT PowerScout

18) or by having specific appliances plugged into their own plug-level meter (e.g. Insteon iMeter Solo). In most cases, large loads will have their own circuit breaker (e.g. furnace, clothes dryer, kitchen oven) and are ideal for BCPM. Smaller appliances that can share the same circuit breaker (usually 15A circuits e.g. fridge, dishwasher, toaster, electronics) are well suited to have plug-level meters.

We might ask ourselves the question: *why go through the trouble of load disaggregation when you could use plug-level meters or a BCPM?* There is a consensus among researchers [34–38] that the costs to purchase and install these additional meters make this option impractical for many occupants. Even with the use of multiple meters, not every load can have its own meter. You still have a single-point sensing problem, albeit on a smaller scale. For example, if you can only afford to monitor the circuit breakers, and the majority of breakers have more than two appliances serviced by them, you still need to disaggregate the load from the circuit breaker. Countering, one can argue that using partial multi-point sensing can mitigate scalability problems by reducing the number of loads that need to be disaggregated at one monitoring point, reducing the computational cost.

Given all this, multi-point sensing still plays a major role in disaggregation because it is needed for priors to use in accuracy testing (see Section 4.1). Researchers use BCPMs and plug-level meters to collect priors to build models specific to a home's appliances and loads. This is because, when comparing one home to another, each home has an almost unique set of different appliances. Most appliances do not operate in simple ON and OFF states. Appliances with different operational modes (multi-state) have complex power signatures and are harder to identify from the aggregate power measurement of the smart meter.

To move away from the collection of priors and to allow a disaggregator to be highly accurate, appliance manufacturers would need to provide more accurate and detailed information about their appliances. Standards organizations, such as International Organization for Standardization (ISO) and the Canadian Standards Association (CSA) can play a role in encouraging manufacturers to comply with certain policies that would provide for better appliance power signatures to the public. In Canada, EnerGuide [1] labels provide average power usage information about appliances. However, more detailed information

[1] see *http://www.nrcan.gc.ca/energy/products/energuide/12523*

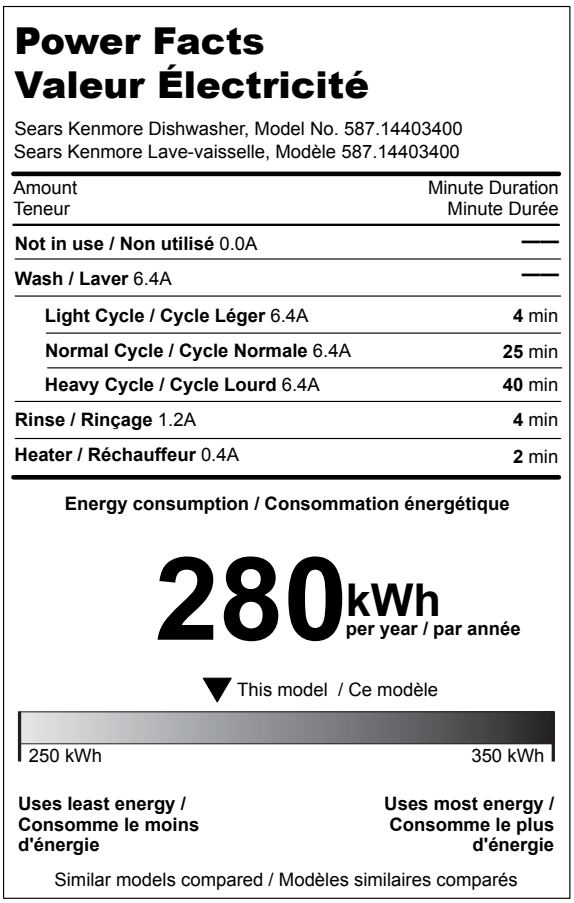


Figure 2.2: *Example power facts label that would appear on an appliance which would provide the necessary information for NILM to disaggregate this appliance from the whole-house power meter reading. This label can be seen as an extension of the existing EnerGuide label.*

such as the power usage of all the operational states and the expected or average duration of each operational state needs to be provided. This can be done in two ways: online download and printed label. For an online download (the most ideal way), manufacturers would upload appliance information to a government or standards organization database where the consumers could download the power signature information for the disaggregator to use. An alternate solution (for low-income households) would be the appliance manual, or a sticker on the appliance, to contain a label that lists all this information (see Figure 2.2 for an example [2]) and the user would be required to enter this information into the load monitoring system. This label has a highly structured format and it would be very easy to provide a smart phone app that could allow occupants to take a picture

of the sticker, and have the image converted into load state parameters that could then be uploaded to the occupant's smart meter or disaggregation device.

2.3 Complex Measurements

The earliest research into disaggregation looked at identifying loads by the real power (P) and reactive power (Q) draws. Hart [17] used real power (P) and reactive power (Q) known as $\triangle$P$\triangle$Q plot in addition to the appliance on/off duration at a sampling frequency of 1Hz. He found the dishwasher had "ramping periods" (synonym to continuously variable) which would need a very complex finite-state machine to model its full behaviour. He concluded that load disaggregation is not suitable for "small appliances, continuously variable appliances, and appliances which are always on". Norford et al. [28] used real power and reactive power while looking at transient waves for commercial buildings sampled in kHz-range (an exact number was not mentioned). They found that although they used reactive power in their transient detector, "real power was sufficient to assess many of the major aspects". They noted transient analysis is more advantageous for detecting equipment start-up and not equipment running at steady-state, but it comes at the cost of more computational power. Laughman et al. [39] extended the $\triangle$P$\triangle$Q plot by adding harmonics as a third dimension. Adding harmonics allowed them to distinguish loads that overlapped on $\triangle$P$\triangle$Q plots. They discussed the fact this type of transient analysis could lead to a form of equipment diagnostics, linking specific transients to specific equipment electrical and mechanical faults. Fisera et al. [40] used only real power and reactive power in conjunction with the existing building management system (BMS) sampling at 1Hz. They found that their method had exponential computational costs as more equipment was added. Computational cost was also burdened "with the increasing period of the control/status signals" provided by the BMS. They were, however, able to provide load disaggregation results online in near real-time.

Some researchers have focused on different measurement statistical properties that a load may exhibit when running. Berenguer et al. [41] used the electrical current startup signatures to detect appliances. They found the short impulse, when the appliance was turned on, provided a unique signature for each different appliance. They observed their system "produce[d] sensible information on the global activity of [a] person". Chang

et al. [42] used voltage and current measurements sampled at 15 kHz. Where other researchers found transient analysis was sufficient for load disaggregation, they found it necessary to "combine transient and steady-state signatures ... to improve recognition accuracy and computational speed". Lee et al. [43] used the different electrical characteristics of real power (e.g. raw, average, peak, etc.) measured from each appliance at a sampling rate of 0.2Hz. They developed "activity-appliance models". They believed these models "could help detect unattended appliances". Tsai et al. [36] used current electrical characteristic measurements (e.g. intensity, peak, average, etc.) on five appliances (fan, florescent light, radio, and microwave oven) at different sampling rates from 1Hz to 1MHz. They captured current waveforms as the appliances were being turned on at different voltage phase angles. This was done to create unique and repeatable profiles. They needed to reduce the sampling rate from every 1 microsecond to every 500 microseconds to "reduce the computational burden and memory requirements of the system". Figueiredo et al. [44, 45] used voltage, current, and power factor measurements at 1kHz. They concluded that simple load disaggregation methods had high accuracy results when performing steady-state signature matching. However, they needed to acquire more signature IDs to increase the accuracy and robustness.

Other researchers such as Gupta et al. [46] used EMI spectrum analysis (sampling at 100kHz) to detect appliances, and found a number of interesting results. Appliances that are the same make and model had very similar signatures and that the signature is mostly independent of the home. But, the placement of the EMI sensor within a home affects the signal which is a "function of the line inductance" between the appliance and the sensor. This means that two identical appliances have different EMI signatures depending on where they were plugged in on the power line. They were able to detect near simultaneous appliance events at 102 milliseconds apart. Their solution required specialized equipment that needed to be trained on every appliance in the home.

Metering these types of measurements often requires sampling at high-frequencies which requires specialized measurement equipment that in many cases researchers either have custom built (e.g. [41, 47–49]) or purchased [29]. The availability and cost of this type of equipment means that it is not feasible for use in the home.

2.4 Basic Measurements

With the advent of smart meters being installed on homes around the world, and the ability for smart meters to communicate consumption data, basic measurements have become a focal point for many researchers – in particular apparent power (S). The smart meter can be seen as a free sensor that should be used by any residential disaggregator. Unless otherwise noted, these researchers use a low-frequency sampling rate, which is what would be provided by a smart meter.

Early research was inspired by using a residential power meter to disaggregate, but these researchers deviated slightly in the development of their disaggregator. Baranski et al. [48, 50, 51] found the use of an optical sensor was an inexpensive option to read power measurements off a home's existing power meter. They used genetic algorithms that were able to create a state-machine of appliances that had high usage. The genetic algorithm could actively learn and discover new appliances that were either simple on/off or multi-state (with $\leqslant$5 states) without prior knowledge. Berges et al. [34] were also influenced by smart meters, but decided to use a high-frequency sampling rate (10kHz). They only presented results for one appliance (the refrigerator), no other appliances were monitored or tested. They found when the refrigerator went into defrost cycle there was significant error in detection. They identified the need for better signature capture and machine learning algorithms.

Recent research has actually used residential meter data with appropriate measurement types and sampling frequency. Kim et al. [35] were motivated to use power measurements because of the advent of smart meters. They found it difficult to disaggregate steady-states and needed to rely on appliance state changes. The accuracy of identifying appliances decreased as appliances were added, from one appliance (100%) to eight appliances (between 73–65%). They concluded that additional features (none mentioned) would need to be added to maintain higher accuracy as more appliances were added. Kolter et al. [52] were again influenced by smart meters. To be able to disaggregate with high accuracy they used complex unsupervised machine learning algorithms. Their disaggregator did have issues with distinguishing similar signatures. Zeifman [38] identified a number of real world constraints for disaggregators, smart meters being one of them. He states that his solution meets all the real world constraints identified except for *various appliance types* where only simple on/off appliances could be detected. Even though he states that the

real-time capabilities constraint was met, he only tested his system offline using MATLAB. Twenty-six days of data took two minutes of processing; therefore (he concluded) "real-time implementation is feasible". Parson et al. [53] used power measurements sampled once per minute. They were able to tune a general appliance model to a specific appliance make and model across six different homes.

These researchers only considered apparent power because of the realization that *smart meters* are being installed on homes. Having access to power readings at no cost is a convenience that cannot be overlooked, but this would mean there is now a limit to the measurement types and sampling rate. A reduction in measurement types used reduces the identification fidelity [39]. Low-frequency sampling should be considered when focused on disaggregation for the home because smart meters have limited communication capabilities. Smart meters can communicate readings, at most, every 1–5 seconds (0.2–1Hz) [54]. These rates are too infrequent for signature matching algorithms that run outside the smart meter, but more advanced machine learning algorithms can aid in uniquely identifying which appliances are being used.

2.5 Our Approach

Relying on high frequency load disaggregation, we believe, is not a viable option. Firstly, if load disaggregation [10] is to become a part of the smart meter, as we believe it should, then an embedded processor that can perform disaggregation must be inexpensive and the disaggregator must be efficient in terms of computation and storage usage. When utility companies roll out millions of smart meters, cost becomes a large consideration.

Secondly, data transmission rate as $\geqslant$1Hz will most likely cause network saturation due to the low bit rate of HANs like ZigBee[2]. Low frequency disaggregation can be just as accurate, if not more, than high frequency disaggregation even with the loss of unique power-on surge spikes – often missed when sampling at low frequencies. Our contribution (**Contribution 5**) is a low frequency disaggregator that is more accurate than current state of the art high frequency disaggregators. In Section 5.6 we will compare the accuracy of our disaggregator to the accuracy of other disaggregators.

Our approach is to create a model of the house and a disaggregator that is measurement

[2]From personal conversations/correspondence with BCHydro, the local power utility. Our research is not tied to ZigBee, is it communication protocol agnostic.

agnostic. Our goal is to have the ability to disaggregate with either *current* (I) or *apparent power* (S). Disaggregation using current is ideal for off-grid homes and buildings that use DC power; while apparent power may be best for home powered by AC. However, in Section 2.5.1 we discuss what is the best measurement to use – there is a clear advantage to using current over power for disaggregation. We believe that load disaggregation should be a part of the smart meter, where occupants can use this information to make more intelligent decisions about conserving energy.

We quickly found that meter manufacturers have closed, proprietary hardware that did not allow us to embed a disaggregator right into the meter. We envisioned two scenarios: (a) where occupants may have an existing power meter that could communicate over a serial connection; or (b) did not have a meter, and would want to purchase or build one. This is where the Arduino Power Meter Reader (see Section 2.5.3) and Precision Ammeter (see Section 2.5.4) projects fit in. These projects have hardware and software that is open source. There are a number of open source hardware and software development platforms available to build prototype systems. We chose the Arduino [3] platform because of its popularity, the support ecosystem, and commitment to open source.

There exists other open source development platforms which we evaluated for use. The Maple [4] from LeafLab is an ARM Cortex-M3 based board with a smaller footprint. However, since its release there has only been a small development community and in 2014 is has ceased to be actively developed on. The Teensy 3.0 [5] from PRJC was a Kickstarted funded project (ARM Cortex-M3 based, as well). It is more popular than the Maple and has a very small foot print. However, the lack of programming library documentation makes this platform difficult to work with. Additionally, the small footprint size does not allow for good separation between analog ground and digital ground which caused ADC readings to contain an excessive amount of jitter. These issues we experience in the development a multi-circuit ammeter array not discussed in this thesis [7]. At this point in our research the TI LaunchPad [6] hardware platform had just been released. We were not able to consider using this platform because the processors were computationally under powered, the lack of platform documentation, and the absence for a development community.

[3] see *http://www.arduino.cc*

[4] see *http://leaflabs.com/devices/maple/*

[5] see *https://www.pjrc.com/store/teensy3.html*

[6] see *http://www.ti.com/ww/en/launchpad/launchpad.html*

2.5.1 Measurement Reading Flux

With the detailed and long term information available in our AMPds dataset [3], we ran an experiment to compare the reading of current (I), apparent power (S), and real power (P) to see which measurement fluctuates more (**Contribution 1**). We introduced I, S, and P and their relationship to each other in Section 2.1. We first reported on similar investigations in the past [3,7]. Here are the findings of our investigation into measurement fluctuations.

Table 2.1: *Comparing Fluctuations in Distinct Readings*

ID	Load	dA	A	VA	W	Flux (VA/dA)	Flux (VA/A)
B1E	North Bedroom	14	3	95	86	6.8×	32×
B2E	Master & South Bedroom	19	3	186	177	9.8×	62×
BME	Basement Plugs & Lights	52	9	399	381	7.7×	44×
CDE	Clothes Dryer	88	21	639	632	7.3×	30×
CWE	Clothes Washer	123	13	982	745	8.0×	76×
DNE	Dining Room Plugs	9	1	99	55	11.0×	99×
DWE	Dishwasher	48	7	295	285	6.1×	42×
EBE	Electronics Workbench	17	3	113	109	6.6×	38×
EQE	Telco/Net Equipment	4	1	29	26	7.3×	29×
FGE	Kitchen Fridge	133	16	576	548	4.3×	36×
FRE	HVAC/Furnace	45	6	387	317	8.6×	65×
GRE	Detached Garage	72	13	136	126	1.9×	10×
HPE	Heat Pump	193	34	1338	1321	6.9×	39×
HTE	Instant Hot Water Unit	9	1	84	70	9.3×	84×
OFE	Home Office	73	9	434	429	5.9×	48×
OUE	Outside Plug	2	2	7	6	3.5×	4×
TVE	Entertainment/TV	43	6	433	416	10.1×	72×
UTE	Utility Room Plug	7	1	43	24	6.1×	43×
WOE	Wall Oven	97	18	645	630	6.6×	36×

Table 2.1 shows the result of an analysis we performed on 524,544 data points (per min readings) over 1 year (in AMPds). We found both apparent and real power readings had a high degree of fluctuation (as high as 99× for A or 11× for dA , see Table 2.1) compared to current. This is due, in part, to the meter using two sensor readings (current and voltage) that can both fluctuate independently to measure real power. With meters that measure

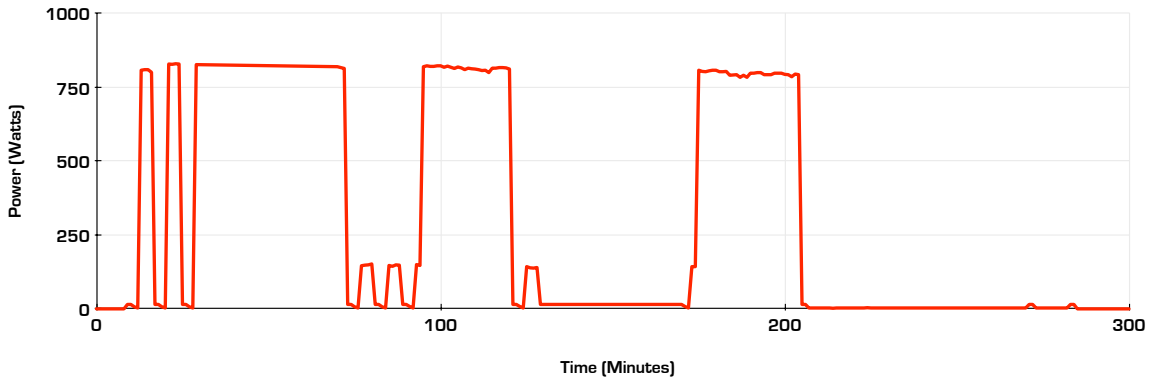
multiple circuits or breakers, current is measured on the same wire as the load while voltage is measured in one spot on the breaker power panel. We experienced a noticeable voltage drop when measuring voltage at the top of the breaker power panel versus the bottom. This means if the meter is measuring the voltage level at a single spot, the further away the current transformer (CT), the less accurate the voltage reading. This leads to a less accurate power reading when calculating the power associated to that CT. In addition, the resistiveness of the load changes due to other factors such as wire gauge and material used. In other words, there is again a voltage drop from the breaker as compared to the plug outlet. It is worth noting that current is not affected by these problems.

We concluded that using current would result in: (a) being better able to determine load states from historical data algorithmically, and (b) a higher classification accuracy score for the disaggregator. Figure 2.3 shows the results from Table 2.1 for the dishwasher. By examining the number of distinct current reads (Figure 2.3(b)) we were able to algorithmically determine the dishwasher had 4 finite-states. Figure 2.3(b) will form the basis for *probability mass functions* (PMF) which our disaggregator will use.

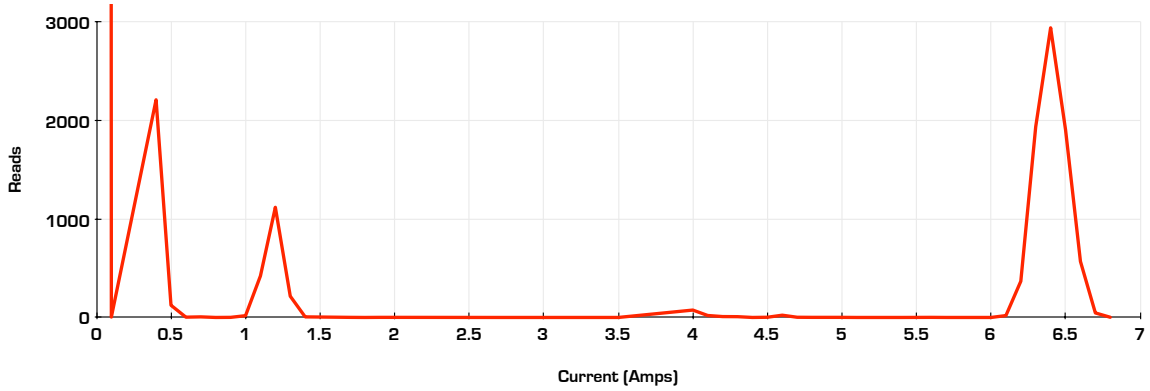
We noticed the outside plug (OUE) had very few fluctuations when compared with I, S, and P. This was due to the fact the outside plug only had one event where there was a 2.6A load ON for 2 minutes (at timestamp 1338438360). Even though there was no other instance of a load being ON on OUE, there were times when S was measured at 4 or 5VA even though were was a measurement of 0.0A. We are not sure why this might be the case, only that there may be erroneous current sensor readings – a value too low to be reported as anything other than 0.0A.

2.5.2 Switch Continuity Principle

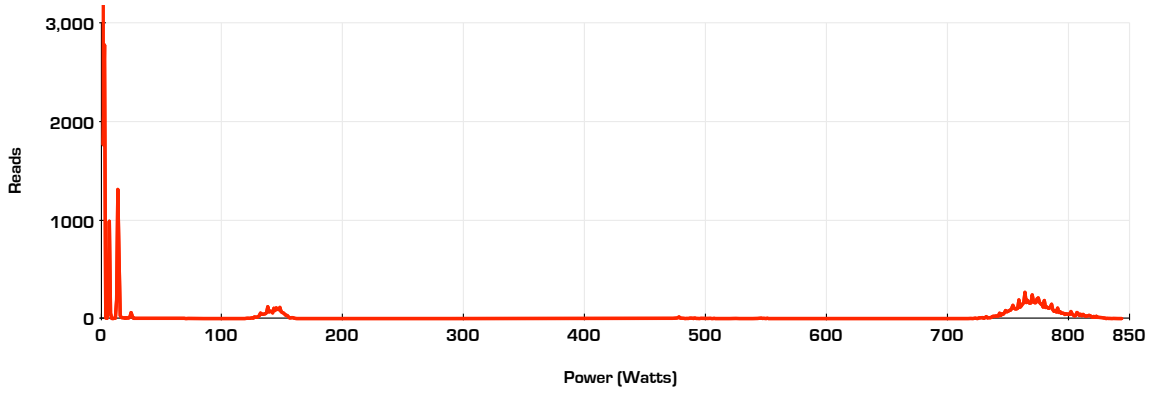
Hart's *switch continuity principle* [17] has been used by many NILM researchers. For example, Parson et al. [53] used this principal to actively tune generic appliance models. In Hart's tests, multiple loads switching states accounted for 4% of the reading collected at a sampling rate of 2–3 seconds. Since 1992, more homes have more energy efficient, multi-state appliances. We wanted to test this principle to see if it holds true on low-frequency sampling. We again examined the data in AMPds [3] in addition to the data in the REDD dataset [55]. Because AMPds contains various measurements, we were able to examine multiple measurements.



(a) 5 Hours of Real Power Consumption



(b) Distinct Current Measurements



(c) Distinct Real Power Measurements

Figure 2.3: *An examination of the dishwasher. Comparing current (b)* vs *power (c) there is 6× (using dA) as many different measurements for power as there are for current. This is due to fluctuations in voltage. Each spike in (b) can be seen as a distinct load/appliance state (4 states in the case of our dishwasher).*

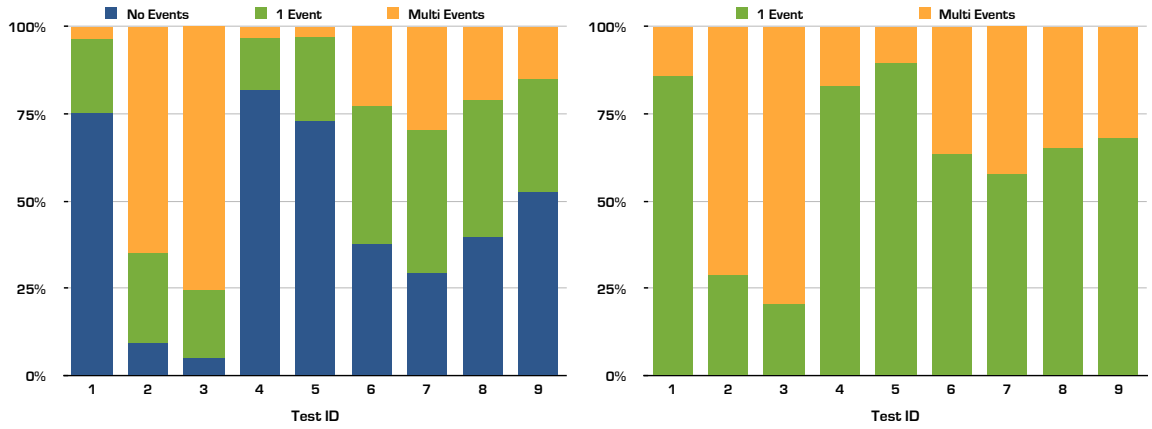


Figure 2.4: *Stacked bar charts of simultaneous load state changes: (left) results for Table 2.2, and (right) results without no-events data (i.e. when loads stayed in the same state). We removed no-events data because learning opportunities only occur during state changes. Comparing Tests 1, 2, and 3 (A, VA, and W from the AMPds dataset) we can observe that current measurements, again, are the better choice. The remaining tests show how the switch continuity principle cannot be relied upon using power readings in the REDD dataset.*

Table 2.2: *A Summary of Simultaneous Load State Changes*

Test	Dataset	Unit	Loads	States	Readings	No Events	1 Event	Multiple
1	AMPds	dA	19	42	524.5k	75.2%	21.3%	3.5%
2	AMPds	VA	19	342	524.5k	9.2%	25.9%	64.9%
3	AMPds	W	19	316	524.5k	5.1%	19.2%	75.7%
4	REDD House 1	VA	10	152	406.7k	81.7%	15.2%	3.1%
5	REDD House 2	VA	8	67	316.8k	72.8%	24.3%	2.9%
6	REDD House 3	VA	12	307	376.1k	37.3%	39.8%	22.9%
7	REDD House 4	VA	11	211	428.1k	29.4%	40.7%	29.9%
8	REDD House 5	VA	14	371	77.5k	39.4%	39.5%	21.2%
9	REDD House 6	VA	11	174	192.2k	52.4%	32.3%	15.3%

Table 2.2 and Figure 2.4(left) show the results of our tests. Tests 1–3 used AMPds with data sampled at 1 minute intervals, and tests 4–9 used REDD with data sampled at 3 second intervals. The States column is a summation of all the states of every load. Load states were determined using the algorithm in Sections 3.5.1 and 3.5.2. We got mixed results. Tests on the AMPds dataset show that using deci-Amperes with a lower number of load states echoes the results reported by Hart. Using apparent and real power (where readings fluctuate more) causes a larger number of load states and a larger number of simultaneous state switches. Tests on the REDD dataset are mixed with house 1 and 2 performing well, while the other houses did not. These results provide additional support for our decision

to use current rather than power for disaggregation.

Figure 2.4(right) shows the results if we ignore *no-events* data. We wanted to ignore data where no events occurred in our analysis because learning can only happen at the point where an event has occurred (i.e. load states change) as Parson et al. [53] has demonstrated with tuning generic appliance models using a difference factorial HMM with the switching continuity assumption. Learning opportunities only can be performed when only one load switches state and these results highlight the fact we cannot (in some cases) rely on this principle to actively learn new loads or actively tune existing known loads.

2.5.3 Arduino Power Meter Reader

This Arduino Power Meter Reader (APMR) was first published in [1] as an open source project [7]. In addition to Adruino, we chose Ubuntu Linux [8] and MySQL [9] for our database server, and Electric Imp for the in-home display. Our choices were based on five principles: (a) that the hardware and software platforms be open source and available to the public; (b) that there are no additional licensing fees or on going costs associated with using the platform for the hardware and software creator, vendor, or supplier; (c) that the cost of the hardware be relatively inexpensive; (d) that the hardware be readily available for purchase; and (e) software be easily accessible for download.

We looked at alternative operating system and database options. The above principles prevented us considering Microsoft Windows and SQL Server. An open source alternative to MySQL is PostgreSQL [10]. PostgreSQL is a high-performance, reliable database system which is in many respects seen as the open source alternative to Oracle. However from our experience, PostgreSQL is difficult to setup and maintain. There are other Linux distributions we could have used but none are as actively worked on and widely used than Ubuntu. Debian [11] then Fedora [12] would have our alternative choices if Ubuntu did not meet our needs.

This section discusses our prototype design in detail. Figure 2.5 shows a block diagram

[7] located at *https://github.com/smakonin/APMR*

[8] see *http://www.ubuntu.com*

[9] see *http://www.mysql.com*

[10] see *http://www.postgresql.org*

[11] see *https://www.debian.org*

[12] see *http://fedoraproject.org*

of and describes the overall system. Figure 2.6 shows our prototype (APMR 2.0), power meter, and other equipment used for our experiment. Our prototype communicated to a Schneider Electric PowerLogic ION6200 power meter which was monitoring the load of a computer workstation and LCD monitor. The APMR was set to read power in 1 minute intervals. Over a 2 week period both the APMR and IHD operated without fault. Prior to the current project, the APMR version 1.0 ran for 8 months reading 2 ION6200 meters once per minute without fault. Our testing confirms the stability of the APMR and IHD choices in an open source hardware platform, and our custom hardware and firmware.

Our prototype is based on the open source prototyping platform Arduino. The Arduino Mega 2560 board was used with a third generation Arduino Ethernet Shield and two custom built shields; shields are simply add-on boards that are stacked up onto the main board. Other Arduino main boards (e.g. Arduino Uno) could not be used due to having a small amount for flash memory which out program image size exceeded. The two custom shields that were built were the Real-Time Clock (RTC) Shield and the RS-485 Shield. The RTC Shield is used for exact timing purposes, recording readings every minute on the zero-second. The RS-485 Shield allows for communication with a Modbus enabled power meter (in our case the Schneider Electric PowerLogic ION6200). Both custom shields use inexpensive electronics and further savings can be realized by combining the two shields into one or by not using the RTC Shield. Figure 2.7 presents the schematics for both custom shields used with AMPR.

The firmware of our prototype was designed with a number of features that make it recoverable, scalable, and robust. These features are: the readings are recorded locally using an SD card, the SD card log files can be downloaded using a web browser, the settings are saved on EEPROM and changeable via a web browser, the readings are sent to a remote website/database server, the ability to sense when the network is down, and then sends all unsent readings when the network becomes available. Figure 2.8 provides a flowchart of the firmware design of the APMR which incorporates these features.

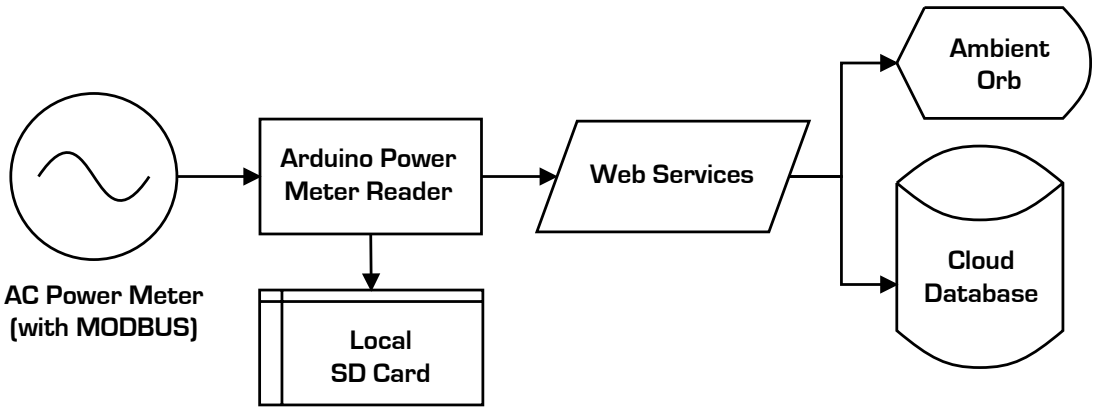


Figure 2.5: *Overall APMR prototype system block diagram. The original project [1] has an ambient orb which communicated wireless to APMR and the power utility. The ambient orb uses three colour LEDs (green, yellow, and red) as ambient indicators for low current demand, average current demand, and high current demand. Current demand was based on how much energy a home was currently consuming relative to its past consumption. The LEDs would pulse ON-OFF during peak times when energy costs (per kWh) were higher than normal.*

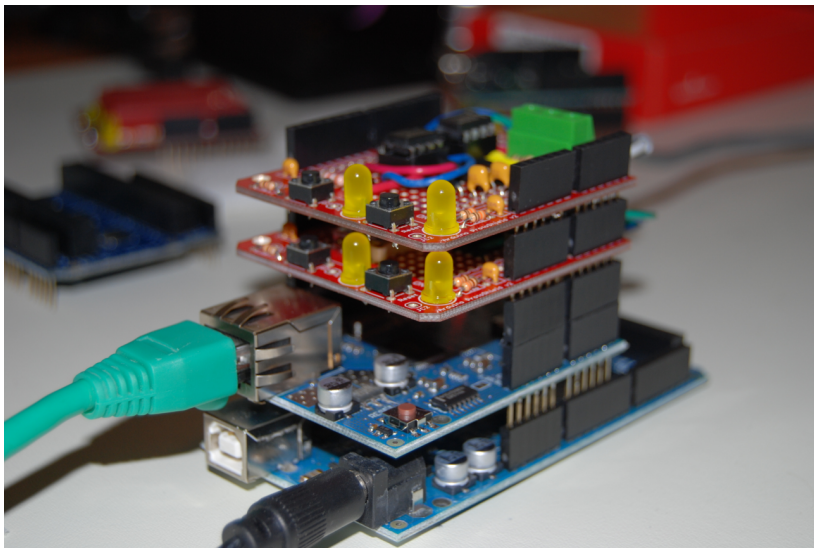


Figure 2.6: *The APMR prototype which can: communicate via RS-485/Modbus (top board) to a meter, keep track of date and time using a real-time clock (second board from top), and communicate data over wired TCP/IP (second board from bottom). It uses an Arduino Mega 2560 (bottom) as the main board.*

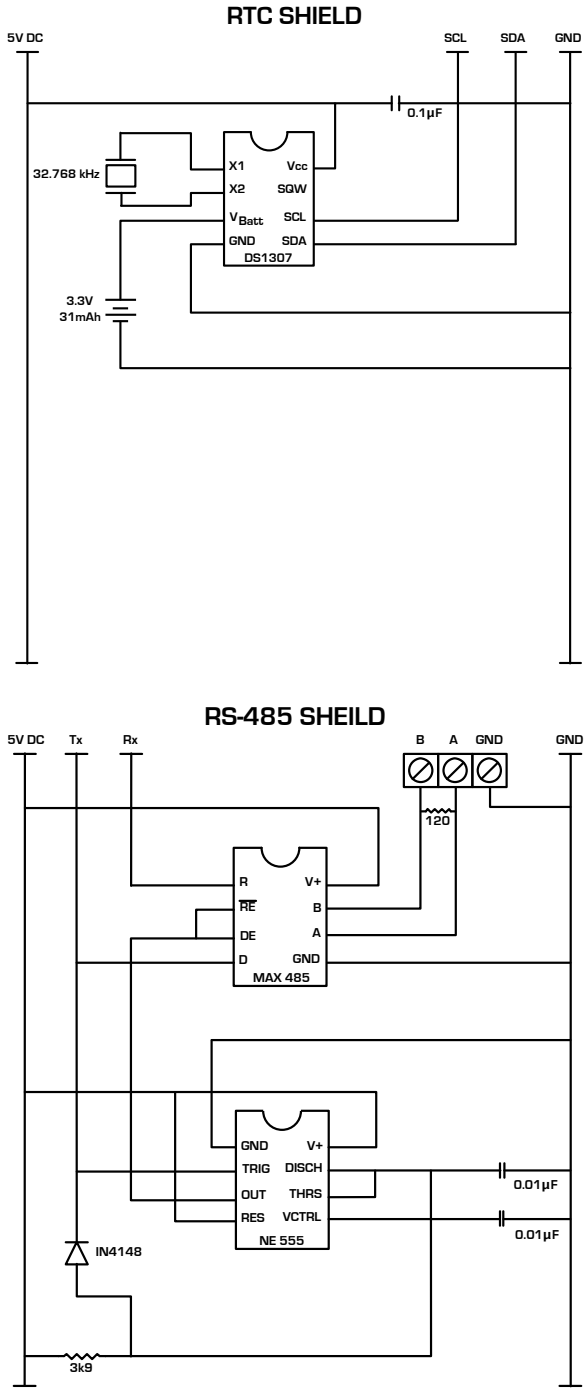


Figure 2.7: *RTC and RS-485 Shield Schematics for the custom shields used by APMR.*

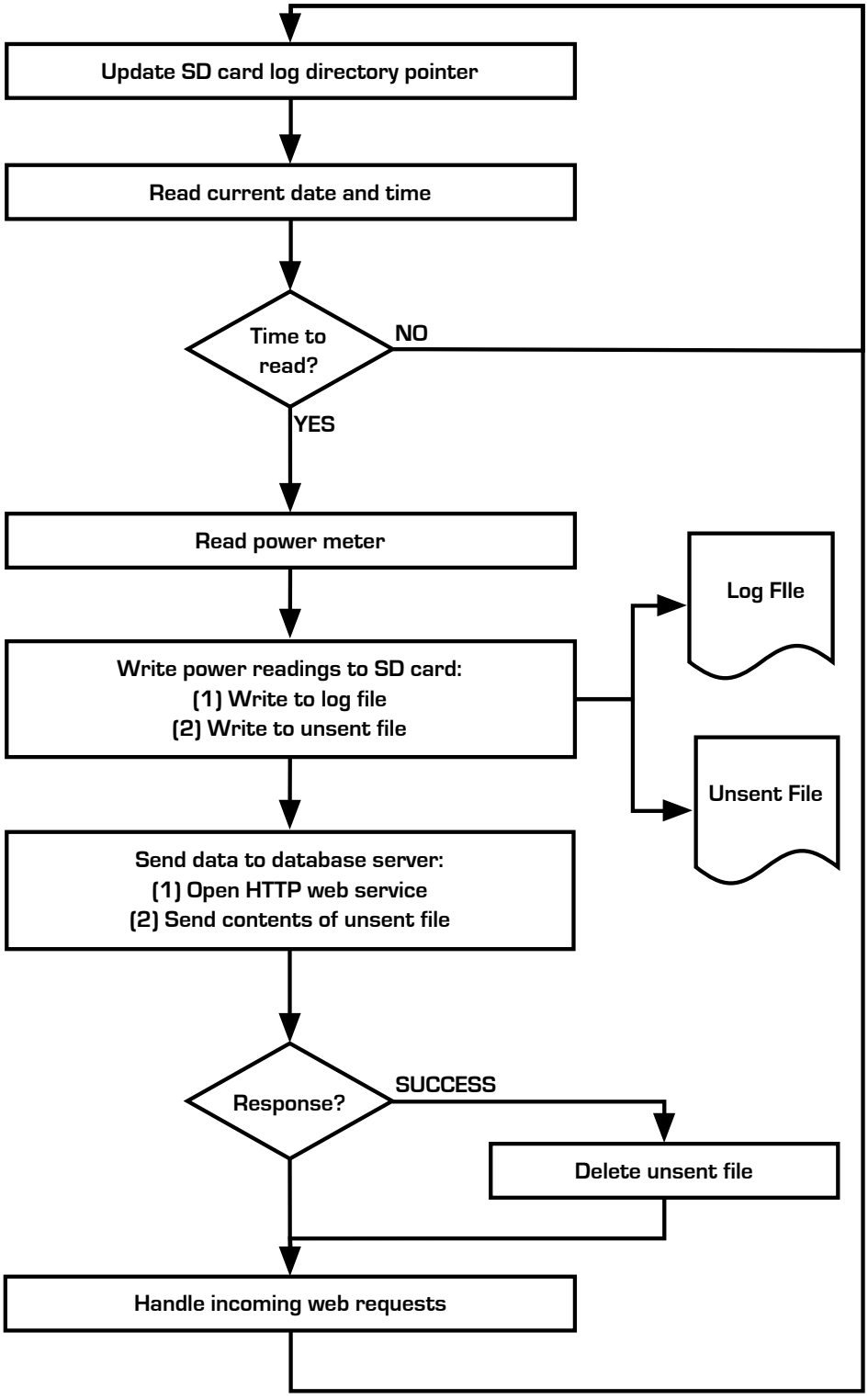


Figure 2.8: *APMR firmware design flowchart.*

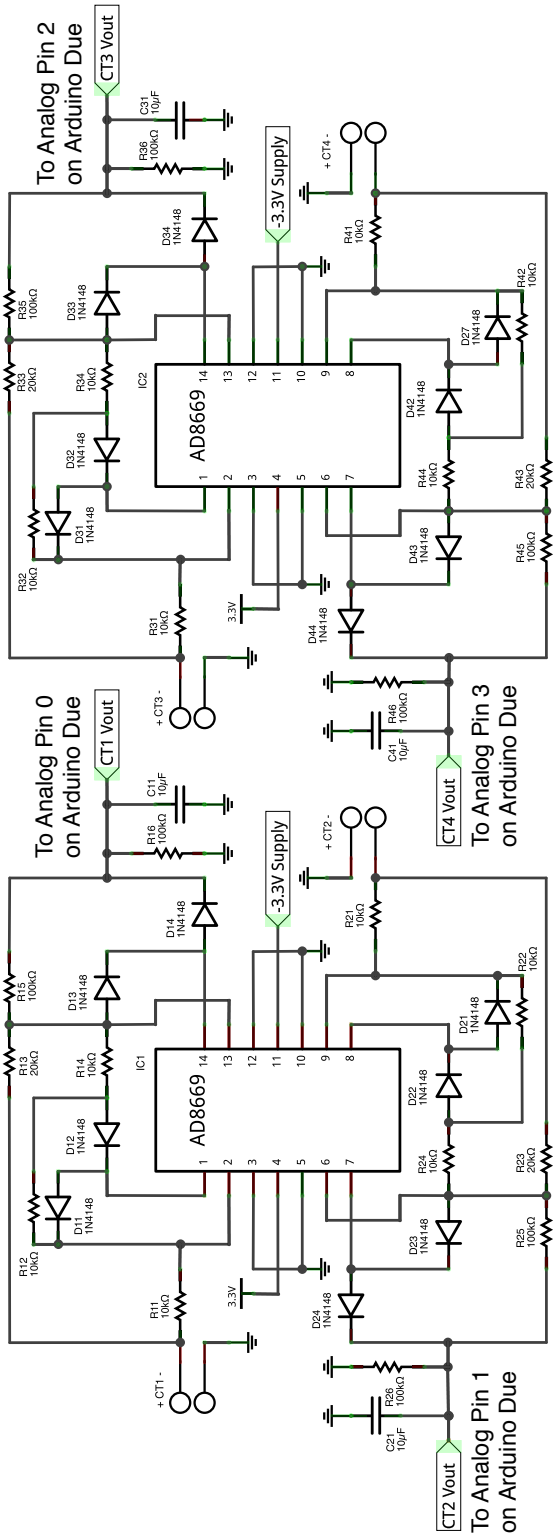


Figure 2.9: *Precision Ammeter op-amp circuit design schematic for the Arduino Due (Section 2.5.4). This circuit amplifies and inverts the negative portion of the AC signal (0–333 mV) generated on the secondary side of the current transformer (CT) to a smooth DC signal (0–3.3V). This DC signal is then passed to an analog pin on the Arduino Due where an ADC converts the signal into an integer value. Four CTs can be monitored using this op-amp circuit. Meaning, there are four identical sub-circuits – one for each CT/analog pin pair.*

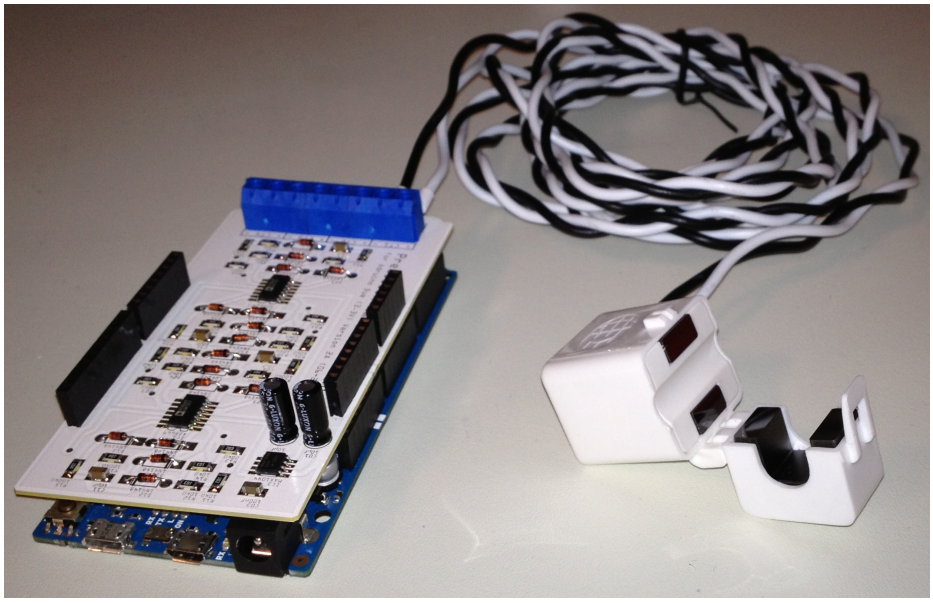


Figure 2.10: *Precision Ammeter prototype. The ammeter shield is seeded into the Arduino Due. Although only one split-core CT is connected (in this picture) via the screw terminals in the back, a maximum of 4 CTs can be connected. The ammeter shield used the op-amp circuit in Figure 2.9.*

2.5.4 Precision Ammeter

The Precision Ammeter [7] is an open source project [13]. The Precision Ammeter would serve two functions: a way to measure current, and a platform to run μDisagg. Figure 2.10 depicts and describes our prototype. We designed a custom ammeter shield to be used with the Arduino Due. A shield is a board that plugs in to the top of the Arduino using a number of pins. Each pin is assigned a function (such as analog in, digital I/O) that allows the shield to communicate with the Arduino.

We studied projects such as Open Energy Monitor [14] that sample both voltage and current waveforms using integral equations. These methods proved to be processing intensive for the microprocessors and thus we decided to design circuitry that would accomplish the same task, but without having the microprocessor extensively sample the waveforms (see Figure 2.9 for the schematic and Figure 2.10 for the final product picture).

With our choice of having a current transformer to sample the currents of a specific load, we obtained a voltage reading that would be directly fed into our *analog to digital*

[13] located at *https://github.com/smakonin/Ammeter*

[14] see *http://openenergymonitor.org/emon/*

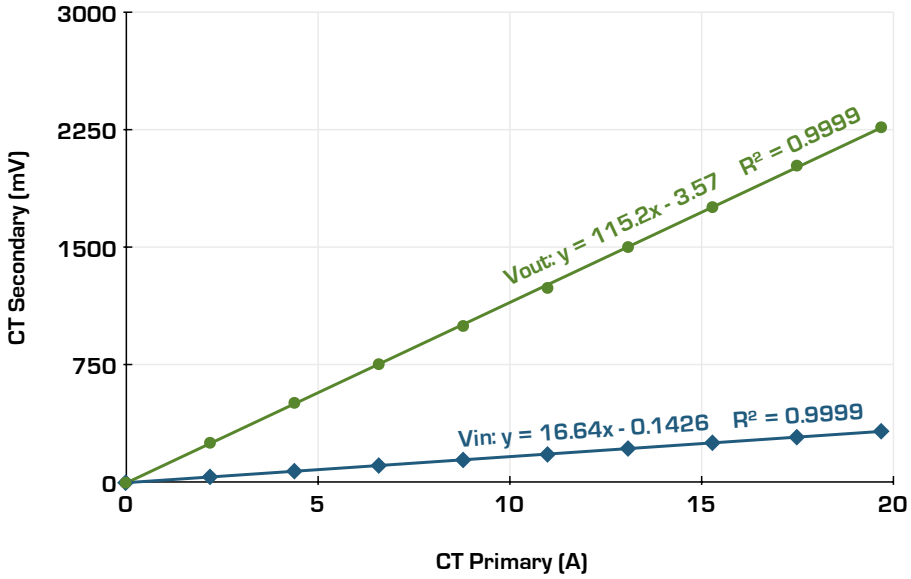


Figure 2.11: *Op-Amp circuit linearity test results. Using a 20A CT we tested the full CT secondary range (0–333mV).* Vin *is the voltage input of the op-amp circuit (also the CT secondary voltage) and* Vout *is the voltage output of the op-amp circuit that is used by the ADC. Loads created using 300W incandescent light bulbs.*

converter (ADC) on the Arduino Due board. From the output of the current transformer we obtained a voltage reading that would range from 0–333mV. This value is sinusoidal and thus we rectified this signal to obtain instantaneous values of current being fed through to the load.

The specifications of the Arduino Due board noted input ports are limited to a maximum input voltage of 3.3V to ensure no damage occurs to the board. The Arduino Due board allows its users to work with power supplies of 3.3V. Thus to power up our operational amplifiers (op-amps), we chose to use the 3.3V power supply along with the MAX1044 charge pump to create the -3.3V power supply. This allowed our op-amps to output a maximum of 3.3V and a minimum of -3.3V. If the op-amps saturates before reaching the minimum or maximum output voltages, the output would never actually reach 3.3V and the microprocessor's port maximum input voltage would never be exceeded. We use a gain of 5 ($= \frac{100\,\text{k}\Omega}{20\,\text{k}\Omega}$).

We performed a linearity test (see Figure 2.11) to verify the op-amp circuit we designed did not saturate when rectifying an amplifying the CT secondary range (0–333mV). The CTs we use (primary 200A, 100A, 50A, and 20A) all have the same secondary output range

(0–333mV). The 20A CT was used for testing due to limitations and safety. We found that due to different tolerances with resistors we had an actual gain of 6.9 not 5.

The firmware of our ammeter was designed to be simple and accurate when measuring current. To provide accurate uniform sampling of each CT, we used a timing interrupt running at 1kHz. We used sliding window averaging (the last 1000 samples) to smooth out any jittering and sudden spikes in current readings. An I^2C interface was also provided to allow other equipment to receive measurement readings.

2.6 Worked Example

We now describe how our approach would work with our studio suite example in Section 1.2. Ideally our occupant would have a smart meter installed in her suite. Once every minute disaggregation information about the loads in her suite would be updated on an informative display (Figure 2.12) located somewhere in her suite. The information displayed would be a list of loads. Each load in the list would indicate what state that load was in, the current power draw, the energy consumed since ON, and its cost. Loads that were OFF would not be listed. For example, at 5:20pm the lights would be listed as ON with a power draw of 480W with the energy consumed measured at 0.12kWh at a cost of 1.2¢, the microwave be listed as ON with a power draw of 1100W with the energy consumed measured at 0.02kWh at a cost of 0.2¢, and all other loads would not be listed because they were OFF.

There are a number of different ways informative displays can be designed. There are physical ambient forms like our *ambient orb* [1], to abstract visualizations [6, 9], to more traditional forms like we depict in Figure 2.12. Design choices provide a range in the type of feedback possibilities and have an impact on how occupants interact with their home and this is very much an open research topic in the area of human-computer interaction.

2.7 Summary

We having made the decision to use low-frequency measurement sampling. We have also developed prototype hardware for our disaggregator to run on. Next, we look at how various researchers have used models and inference algorithms to create their disaggregators. We will also introduce our disaggregator.

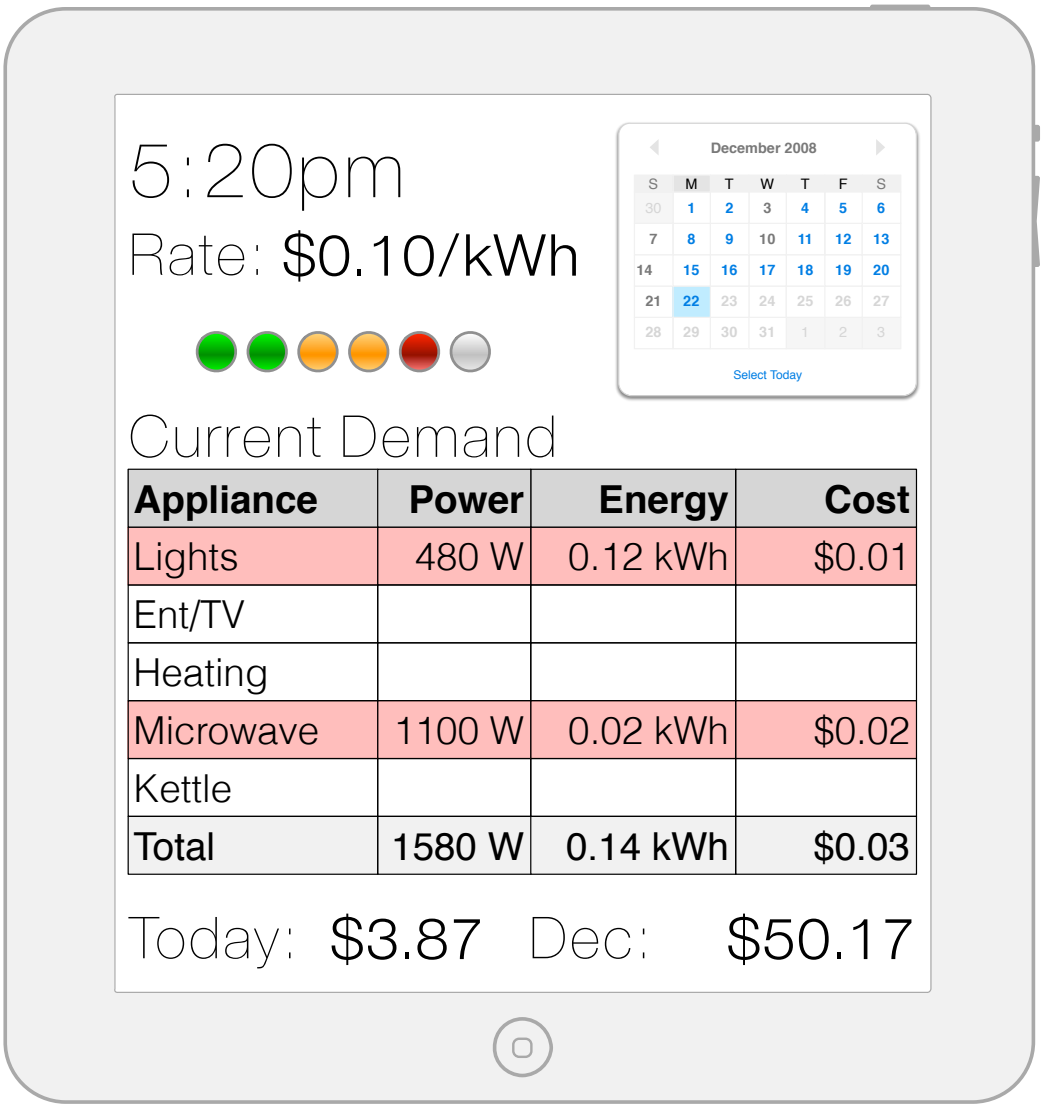


Figure 2.12: *A prototype informative display showing what the occupant might see as described in Section 2.6. The current date (on the calendar), time, and cost of electricity (top). Below is a six segment colour display of current demand versus average historical demand (green = below, orange = normal, red = high power usage). The current Demand table lists all tracked loads and highlights (in red) those that are ON. The total cost for today and the current month are shown at the bottom of the screen. The home button (bottom) would allow the occupant to return to this screen from other screens. Ideally there would be screens that display historical aggregate information by date range (day, week, month, year) or by specific load.*

CHAPTER 3

MODELS AND INFERENCE

Effective and efficient load disaggregation requires not only modelling of loads within a house, but also algorithms to infer the different states of each load. Modelling and inference problems are widely studied in computing science, engineering, artificial intelligence, and more specifically machine learning. These give us some good starting points for investigating the appropriateness of different modelling methods and inference algorithms. In this chapter, we will introduce some algorithms that have been used for load disaggregation and NILM. We investigate how we can build on existing methods to solve our load disaggregation problem.

Many researchers have published vastly different approaches to load disaggregation. Artificial Neural Networks (e.g. [36, 42]), Support Vector Machines (e.g. [44, 45, 52]), and Nearest Neighbour (e.g. [34, 36, 44–46, 56]) algorithms have been popular methods for disaggregation in the past. Artificial Neural Networks (ANN) [57, pp. 47–93] are easy to use. However, both the construction and training of ANNs is arbitrary and tuning can often result in the convergence on local maxima and overfitting. Support Vector Machines (SVM) [57, pp. 119–132] use *optimal line separation* between classifications and do not suffer from the same problems of ANNs[1] provided the dataset used is not large. Nearest Neighbour (k-NN) [57, pp. 183–186] is used to classify unlabelled data that is nearest to each

[1] As discussed in detail at *http://www.svms.org/anns.html* where a good discussion of SVM *vs* ANN is presented.

other (based on a distance function). However, this method can be memory intensive having large storage requirements and can be susceptible to the *curse of dimensionality*. Since 2011, methods that use hidden Markov model (HMM) [57, pp. 347–361] have become a focal point for most researchers [35,38,53,58–60]. HMM-based disaggregators are what we will focus on in this chapter.

HMMs are a natural fit for disaggregation because they have the ability to model time series data and represent the unobservable state of each load (Section 3.1). Amongst the researchers who have used HMMs, there has been a focus on using factorial HMMs [35, 53, 59, 60] (Section 3.2). Although there is a focus on factorial HMM (and variants of) in particular, there are other methods that are worth discussing: the Combined Load HMM [58] (Section 3.3) and Viterbi Algorithm with Sparse Transitions [38] (Section 3.4). Our approach to disaggregation does not make use of a factorial HMM (Sections 3.5). This is supported by an investigation in Section 3.5.3 which shows how matrix sparsity needs to be taken advantage of (**Contribution 2**). We finish with a worked example (Section 3.6) that reflects on the example scenario we presented in Chapter 1.

3.1 HMM Basics

Since individual appliance internal states are not directly observed in the total consumption rate reading, hidden Markov models (HMM) have been a natural choice for modelling the disaggregation process. HMMs are also a simple and efficient machine learning algorithm for modelling states over a length of time [57,61,62]. HMMs are built on the Markov assumption which states that Markov property is assumed to hold for a given model. The Markov property is a conditional probability distribution where the current state at t is only dependant on the previous state at $t-1$ and not on the sequence of states that preceded $t-1$. This means inference is computationally inexpensive to perform provided the number of states is not large.

A typical HMM can be defined as

$$\lambda = \{\mathbf{S}, \mathbf{O}, \mathbf{P}_0, \mathbf{A}, \mathbf{B}\}, \tag{3.1}$$

where $\mathbf{S}$ is a set of possible states, $\mathbf{O}$ is a set of possible observations, $\mathbf{P}_0$ is the initial state probabilities at time t_0, $\mathbf{A}$ is the transition matrix, and $\mathbf{B}$ is the emission matrix. We can define the total number of states as $K = |\mathbf{S}|$ and the total number of observations as $N = |\mathbf{O}|$.

The transition matrix $\mathbf{A}$ would then be defined as the probability for state transition from $t-1$ to t, with a $K \times K$ matrix where $\sum_i \mathbf{A}[i,j] = 1.0$. The emission matrix $\mathbf{B}$ would then be defined as the probability for seeing a particular observation at time t, with a $K \times N$ matrix where $\sum_j \mathbf{B}[j,n] = 1.0$. More formally we can say that

$$\mathbf{A}[i,j] = p(S_t = j | S_{t-1} = i)\,, \tag{3.2}$$

$$\mathbf{B}[j,n] = p(O_t = n | S_t = j)\,. \tag{3.3}$$

Although there are a number of algorithms for HMM decoding, such as the Viterbi algorithm [63], and its variants, the main practical obstacle to load disaggregation is the complexity (both space and time) involved in the process. To give an example, if each of the M loads has a total of K internal states, the total number of HMM states is K^M (assuming for simplicity that all loads have the same number of states); this exponential size of the state space has been a practical limitation to the wider application of disaggregation. With just 11 loads you can have over 1,000,000 super-states. We call these super-states because they are a combination of a number of separate load states. Super-states could also be seen as equivalent to the state the house is in or the *home state*. We will expand on the super-state idea in Section 3.5 as part of our approach to load disaggregation.

3.2 Factorial HMM and HSMM

To curtail the state exponentiality problem of HMMs, Kolter [59], Parsons [53], and Johnson [60], among others [64], have used factorial HMMs [65] (incl. semi-Markov) for disaggregation. Such factorial models lead to lower complexity. For instance, 8 two-state loads would have 2^8 or 256 states where factorial HMM (FHMM) would have 8 chains. Each load is a separate Markov chain that can evolve in parallel to the others. Each chain can represent either a simple ON/OFF or a multi-state load. We define simple ON/OFF loads as a *subset* of multi-state loads. Because each load is a separate chain, dependencies amongst separate loads is lost. Additionally, there is the added complexity of training these chains [66] and exact inference is not possible, approximation methods must be used [59].

Kim et al. [35] used a combination of four factorial HMM variants to provide an unsupervised learning technique. Factorial HMM (FHMM) was used in conjunction with a fractional hidden semi-Markov model (FHSMM), a conditional factorial HMM (CFHMM),

and a conditional factorial HSMM (CFHSMM). The FHMM modelled the loads and their states. The FHSMM extended the FHMM and modelled load-state durations. The CFHMM extended the FHMM and modelled time-of-day usage, load dependencies, and other sensors (no details mentioned). The CFHSMM combined the FHSMM and CFHMM to model load and additional feature dependencies with more accurate probability distributions. Parameter estimation was done using an expectation maximization (EM) algorithm where the M-setup update was done using Gibbs sampling [65] because of intractable inference [17]. Emission probabilities used Gaussian distributions which were prone to overfitting. They were not able to use the Viterbi algorithm to infer load states because of the intractability of CFHSMM and instead used *simulated annealing* (SA) [67]. They achieved classification accuracies of between 69%–98% (for 10 homes) using their M-fscore accuracy measure. Their results seem to suggest the accuracy of the disaggregator quickly decreases as more appliances were added for disaggregation. Such a disaggregator requires a high degree of computational power to disaggregate.

Kolter et al. [59] used a combination of two FHMM variants: additive FHMM and difference FHMM. Additive FHMM was used for finding the aggregate observed load. They also used a difference FHMM that assumed the switch continuity principle [17] to estimate state change by calculating the difference in the load from $t-1$ and t. For FHMMs, exact inference is not possible so they use an additive fractional approximate MAP (AFAMAP) algorithm with predicted mean output. They achieved an average precision accuracy of 87.2% and an average recall accuracy of 60.3% based on the classification of 7 appliances using two weeks of high-frequency data. Four of seven loads scored with moderate to high accuracy, but loads such as electronics scored very low. Due to signal feature loss in low-frequency data, this approach would produce lower accuracy when data sampling is $\leqslant$1Hz (see Section 2.5.2).

Parson et al. [53, 68] used a variant of the difference FHMM from Kolter et al. [59]. Parson proposed an EM training process that would train generic appliance models to specific (to the house) appliance models. Their disaggregator used an extension of the Viterbi that would run repeatedly, each time filtering out an appliance and subtracting the appliance's load from the aggregate total load. When all appliances were disaggregated processing stopped. The Viterbi algorithm [63] was extended to ignore small joint probability observations and all sequences with joint probability were evaluated. They tested their disaggregator on data sampled at one minute intervals. They reported a mean normalized error

in estimated load per day from as low as 38% to as high as 3469%. They also reported a root-mean-square error overall time periods in power estimation from as low as 84% to as high as 3107%. They have demonstrated that there is promise in being able to take general appliance models and actively tune them using aggregated data. However, these demonstrations have been done on a small number of cyclical type appliances such as fridges and freezers which require a training window of data (e.g. fridges can require up to a 200 minute window). The use of such length training windows is more suited to an algorithm that runs off-line on a server.

Recently, Johnson et al. [60] considered using the factorial variant of a hidden semi-Markov model (HSMM) [69, 70] because they provided a means of representing state durations in a load model. They introduced the idea of *changepoint detection* as a way to rule out observations that would not present a learning opportunity for active tuning or infer a state change. This allowed them to reduce the computational complexity. Although the authors claimed this was unsupervised learning, they were incorrect [68]. This was a supervised learning solution because they used labelled data to building the appliance models. Further, their models were specific to a given dataset (REDD). Rather than having their algorithm run on the entire dataset, they hand picked a number of specific segments for testing and evaluation – this does not constitute a real world scenario.

Factorial models reduce the state exponentiality problem of HMMs but at the expense of exact inference and load dependencies. While trying to avoid the complexity problems of using a common HMM, these authors have introduced new complexities that deal with model tuning approximate inference. It is difficult to see these factorial algorithms run in real-time on embedded hardware. To this point, Parson has used a cloud computing based solution [53] to perform algorithm execution, but this may raise data privacy and security concerns amongst some consumers.

3.3 Combined Load HMM

Zia et al. [58] built their models using the sub-metering data for each load. Each load was monitored for several days using low-frequency sampling at 0.1Hz. Once data was collected, the sub-meters were removed. They determined load states and created an HMM for each load by hand. The steady state signature is equated to load in a given state. Their model was trained using a case statement, if the load's signal fell within the bounds of

a particular case then the load was said to be in that state. This same data was used to create Gaussian distributions for the emission matrix. Two loads were then combined with each other to make a larger HMM. They then used the Viterbi algorithm to decode the observation sequence to find the most likely state sequence of the larger HMM. Pattern matching was then performed on the state sequence to identify which load was in what state.

Although Zia's paper is arguably one of the first papers to make use of HMMs for disaggregation, it is written in a way that makes the work difficult to reproduce and hard to compare with other disaggregators. A two load combinatorial search was performed until the observed total load was processed. It seems as though the total load is only ever the summation of the 2 loads being tested. Combining and testing for only two loads is not a realistic scenario. They do not report any accuracy results so it is hard to tell how successful their disaggregator is.

3.4 Viterbi Algorithm with Sparse Transitions

Zeifman [38,71] proposed *Viterbi Algorithm with Sparse Transitions* (VAST), a modified version of the Viterbi algorithm, on multiple transition matrices (each a triple of loads), but limited the number of load internal states to only two (ON or OFF). Such an approach would require the approximation of a many-state load (e.g., the dishwasher) to be a simple ON/OFF load [38]. He ordered the lists of appliances by power demand and created partitions that each had three and only three loads with overlapping Gaussian probability distributions. Each appliance had two neighbouring appliances that were modelled using a Markov Chain. He used this modification to reduce the computational cost (due to sparsity) that would have been produced by using a large transition matrix. For example, using appliance pairs results in 16 possible state transitions (per pair) that need to be computed; whereas, a triple of appliances (the original VAST algorithm [71]) resulted in 64 possible state transitions. The reduction in computational cost outweighed the reduction in appliance detection accuracy, as fewer appliances were being compared to each other.

Zeifman created probability distributions based on power and duration features for each load. He clustered negative power changes by ΔP and the hourly presence/absence of the appliance running. He clustered positive power changes by ΔP and time duration of the power spike. An ISODATA algorithm [72] was used to create the clusters. If clusters

were too close they were merged and if a cluster contained multiple appliances it was split. Clustering was no longer used after the initial distributions/clusters were created. This resulted in distributions of power and time of the appliances.

Zeifman reported the accuracy of VAST to an average of 90.2% on 9 (simple ON/OFF) loads using Kim's M-fscore accuracy measure [35]. The majority of modern appliances have a number of different operational states (not just ON/OFF) which severely limits what loads VAST could disaggregate. Zeifman used triples of loads to avoid sparsity, but it is worth noting that the three-load transitions matrices can still have zero-probability elements and that the sparsity in the emission matrix was not dealt with.

3.5 Our Approach

Our disaggregator (Figure 3.1) first analyzes the sub-metered data from load (priors) and creates a probability mass function for each (Section 3.5.1). Load states are then determined by quantizing the PMF further. Each of the load's states are then combined to create a super-state HMM (Section 3.5.2). The super-state HMM is very sparse, and matrices are compressed to take advantage of this (Section 3.5.3). Once the super-state HMM is built there is no need for further sub-metering. There is also no need for a pre-tuner. The act of model building creates a super-state HMM that is in a steady-state without the need of active tuning – a step using the forward-backward algorithm. We then take the last times observation and the current observation and use our sparse Viterbi algorithm (Section 3.5.4) to disaggregate the state of each load and estimate load consumption (Section 3.5.5). The compression technique we use provides for a computationally efficient way to perform exact inference in a way that is both space and time optimized to avoid the state exponentiality problem HMMs have (**Contribution 2**).

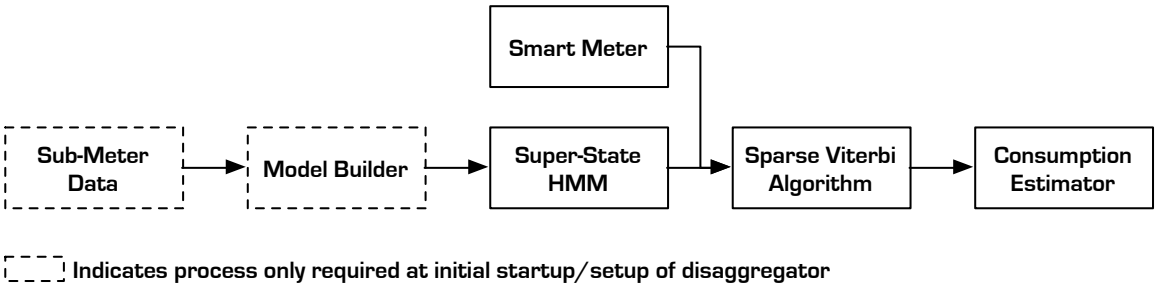


Figure 3.1: *Block diagram of our disaggregator.*

Our solution preserves load dependence information (**Contribution 5e**) and a way to perform exact inference in a computationally efficient manner (**Contribution 5f**), which is not possible when using factorial HMM or VAST. For example, the condition when a heat pump turns ON, the HVAC fan (on a separate breaker) increases its rotation speed, and does the reverse when the heat pump turns OFF (as we have observed in our dataset [3]). Unlike Zia [58], our disaggregator can determine load states automatically during model building, can combine all loads into one super-state HMM not just two loads, and does not require the added off-line pattern matching step. Kim et al. [35] has a number of issues, the least being their use of 4 FHMMs to disaggregate which would not be able to run on an embedded processor (**Contribution 5c**). Like Zeifman [38, 71] we take advantage of matrix sparsity (in a different way). However, unlike Zeifman we can disaggregate loads with complex multi-state power signatures – not just ON/OFF loads (**Contribution 5d**). In Chapter 5 we will show how our disaggregator can achieve a high degree for both load classification and consumption estimation (**Contribution 5b**) using different low-frequency sampling rates and different measurement (**Contribution 5a**).

3.5.1 Probability Mass Functions

It is uncommon for disaggregators to represent a load as a discrete distribution. Even though a power signal is continuous by nature, we can only ever measure and record its signal values discretely. As we discussed in Chapter 2, the rate at which we can sample can be very fast, given the right equipment, but nonetheless it is still discrete. All the researchers that have been cited in this chapter use Gaussian distributions which are continuous in nature. We take a different approach, and represent an appliance in a discrete distribution by using a *probability mass function* (PMF). Studying power systems, we can reflect on the nature of electrical systems and power. We know the following properties to hold true: (a) measurements of current draw are discrete as a limitation of the measuring device, (b) size of the breaker and/or the electrical limits of the load provide an upper bound on the measurements of current draw, and (c) zero is a lower bound on the measurements of current draw – current will never be negative. This forms our intuition for using PMFs.

Let there be M independent discrete random variables $Y_1, Y_2, \ldots, Y_M$, corresponding to current draws from M loads. Each Y_m is the current or power measurement of a metered

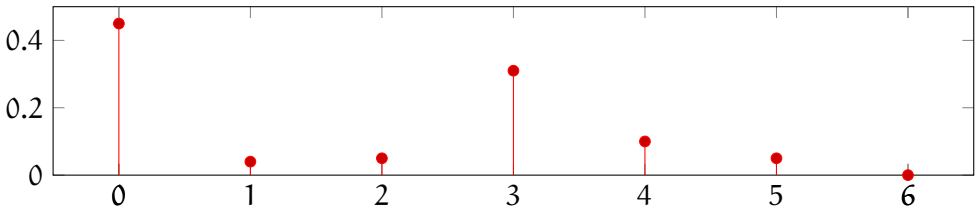


Figure 3.2: *A stem diagram of the example PMF.*

electric load with a PMF of $p_{Y_m}(n)$, where m is the load index $i \in \{1, 2, ..., M\}$, y is a number from a discrete set of possible measurements $y \in \{0, 1, ..., N_m\}$, and N_m is the upper bound imposed by the breaker that the m-th load is connected to. For example, with current measurements (in dA) on a 15A breaker, we would have $N_m = 150$. The PMF $p_{Y_m}(n)$ is defined as follows:

$$p_{Y_m}(n) = \begin{cases} \Pr[Y_m = n], & \text{if } n \in \{0, 1, \ldots, N_m\}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.4}$$

where $\Pr[Y_m = n]$ is the probability that the current/power draw of the m-th load is n. For example, using Table 3.2, if the PMF of Y_m is 2, then $\Pr[Y_m = 2] = 0.25$, so the probability of the m-th load drawing 2 dA (i.e., 0.2 A) is 0.25.

Table 3.1: *An Example PMF*

n	0	1	2	3	4	5	6
j	900	80	100	620	200	100	0
$p_{Y_m}(n)$	0.45	0.04	0.05	0.31	0.10	0.05	0.00
K_m	0	1					

The probability $\Pr[Y_m = n]$ is estimated from measurements over a sample period. For example, if over T measurements, the current draw n was recorded j times, then $\Pr[Y_m = n] = \frac{j}{T}$. During the sample period each load is metered at a consistent rate of one measurement per minute. This rate determines the time resolution at which the load disaggregation will be performed.

Peaks in the PMF $p_{Y_m}(n)$ are designated as probable load states $k \in \{0, 1, \ldots, K_m\}$, where $K_m + 1$ is the number of states for the m-th load. The states are assigned by quantizing the range of possible measurements $[0, N_m]$ (without gaps) so that each quantization bin contains one peak of the PMF. For example, in Figure 3.2 and Table 3.2, there are two

peaks in the PMF, so we would model this PMF by a two-state model, with states being indexed $\{0, 1\}$. The probability of each state is the total probability mass within its quantization bin.

3.5.2 Super-State HMM

We model a house with M loads as an HMM $\lambda = \{\mathbf{P}_0, \mathbf{A}, \mathbf{B}\}$ having a row-vector of initial prior probabilities $\mathbf{P}_0$ of length K, a $K \times K$ transition matrix $\mathbf{A}$, and a $K \times N$ emission matrix $\mathbf{B}$, where K is the number of whole-house states (or super-states), and N is the number of possible observations. If $t-1$ and t represent the previous and the current time instants, the entries of $\mathbf{A}$ and $\mathbf{B}$ are defined as

$$\mathbf{A}[i,j] = p(S_t = j | S_{t-1} = i) \quad , \quad \mathbf{B}[j,n] = p(y_t = n | S_t = j) \; ;$$

where S_t is the super-state at time t, and y_t the observation at time t.

The super-state is composed of the states of all M appliances: $S_t = (X_t^{(1)}, X_t^{(2)}, ..., X_t^{(M)})$, where random variable $X_t^{(m)}$ is the internal state of the m-th load at time t. For example, a dishwasher may have 4 internal states that consist of {OFF, WASH, RINSE, DRY}. The total number of super-states is $K = \prod_{m=1}^{M} K^{(m)}$ where $K^{(m)}$ is the number of internal states of the m-th appliance.

As the state of a load changes so too does its power or current draw. In this work we consider current draw as the observation. Let $y(\cdot)$ be the current draw of the corresponding internal appliance state, so that $y\left(x_t^{(m)}\right)$ is the current draw of the m-th appliance in state $x_t^{(m)}$. For notational convenience, we assume that the current values are non-negative integers, i.e., $y\left(x_t^{(m)}\right) \in \{0, 1, ..., N\}$; in practise, these would not necessarily be integers, but would still be constrained to a discrete set of possible readings of the current meter. The observed measurement at time t from the smart meter is the sum of the current draws of individual appliances:

$$y_t = \sum_{m=1}^{M} y\left(x_t^{(m)}\right) \; . \tag{3.5}$$

Model parameters, such as load state probabilities $p(X_t^{(m)} = x_t^{(m)})$ as well as conditional probabilities in $\mathbf{A}$ and $\mathbf{B}$, can be obtained from existing load disaggregation datasets (e.g. AMPds [3]). In general, even though the assumed full set of possible current draws is

$\{0, 1, ..., N\}$, all appliances have fewer than N states. To model this, for the m-th appliance, we quantize the set $\{0, 1, ..., N\}$ into $K^{(m)}$ bins such that the first bin contains 0 (and corresponds to the OFF state), while other bins are centred around the peaks of the empirical probability mass function (see Section 3.5.1) of the current draw

$$p_{Y_m}(n) = p\left(y\left(X_t^{(m)}\right) = n\right), n \in \{0, 1, ..., N\}, \tag{3.6}$$

obtained from the dataset. A peak in the PMF is identified when the slope on the left, $p_{Y_m}(n) - p_{Y_m}(n-1)$, is positive, the slope on the right, $p_{Y_m}(n+1) - p_{Y_m}(n)$, is negative, and $p_{Y_m}(n) > \epsilon$, where $\epsilon = 0.00021$ used to ensure that small peaks (noise) are not quantized as states. We chose the value of ϵ by observing the PMFs and finding that and spike that had <=110 out of (524544) occurrences was not a state.

A simple example of such quantization is given in Table 3.2, where $N = 6$, but only two states are identified after quantization, hence $K^{(m)} = 2$. These states are indexed by $k^{(m)} \in \{0, 1\}$ and are centred around the values $n = 0$ and $n = 3$. The quantized states are denoted $\widehat{X}_t^{(m)}$. The current draws of the quantized states are stored as a $K^{(m)}$-dimensional vector, denoted $\mathbf{y}_{peak}^{(m)}$, whose elements are the locations of the peaks in the original PMF. For the example in Table 3.2, $\mathbf{y}_{peak}^{(m)}[0] = 0$ and $\mathbf{y}_{peak}^{(m)}[1] = 3$. The probability of a given quantized state is simply the sum of probability masses in the corresponding bin, hence for the above example,

$$p\left(y\left(\widehat{X}_t^{(m)}\right) = 0\right) = p(k^{(m)} = 0) = 0.45$$

and

$$p\left(y\left(\widehat{X}_t^{(m)}\right) = 3\right) = p(k^{(m)} = 1) = 0.55\,.$$

Since $K^{(m)} \leqslant N$, it can be seen that state quantization will increase the sparsity of $\mathbf{A}$ and $\mathbf{B}$.

The super-state corresponding to quantized internal states is $\widehat{S}_t = (\widehat{X}_t^{(1)}, \widehat{X}_t^{(2)}, ..., \widehat{X}_t^{(M)})$. The quantized super-states can be indexed linearly in terms of the indices of the quantized internal states $k^{(1)}, k^{(2)}, ..., k^{(M)}$ as follows:

$$k = k^{(M)} + \sum_{m=1}^{M-1}\left(k^{(m)} \cdot \prod_{i=m+1}^{M} K^{(i)}\right). \tag{3.7}$$

Table 3.2: *An example of PMF and state quantization*

n	0	1	2	3	4	5
count(n)	900	80	100	620	200	100
$p_{Y_m}(n)$	0.45	0.04	0.05	0.31	0.10	0.05
$\mathbf{y}_{peak}^{(m)}$	0	3				
$k^{(m)}$	0	1				
$p(k^{(m)})$	0.45	0.55				

Based on (3.7), one can also extract individual load state indices $k^{(m)}$ from k by iteratively extracting the remainder of division of k by partial products $\prod_{i=1}^{m} K^{(i)}$, starting with $k^{(M)}$ (see Algorithm 3.1).

Algorithm 3.1 KTH-CARTESIAN(k)

```
s ← []                                  # init as empty vector
k' ← k                                  # assign the super-state

for m = 1 → M − 1 do                    # calc each load state
    divisor ← ∏_{i=m+1}^{M} K^(i)
    s.append(k'|divisor)
    k' ← k' mod divisor
end for
s.append(k')                            # add last load's state

return s                                # return load states
```

3.5.3 Exponentially Large but Sparse Matrices

As the number of loads to disaggregate increases, the number of super-states K grows exponentially (see Figure 3.3(a)), and so too do the dimensions of **A** and **B**. However, in the case of load disaggregation these matrices are very sparse. In fact, so sparse that using an HMM with full super-state space is a practical option. The theoretical sparsity of **B** is clear from its definition. Since for each specific super-state

$$s_t = (x_t^{(1)}, x_t^{(2)}, ..., x_t^{(M)}) \tag{3.8}$$

there is exactly one output

$$y_t = y\left(x_t^{(1)}\right) + y\left(x_t^{(2)}\right) + \cdots + y\left(x_t^{(M)}\right) , \tag{3.9}$$

each row of **B** should contain exactly one non-zero element (and that element is equal to 1). However, with quantization, this might not hold exactly, as the current draw may fluctuate slightly within a quantized state even if the super-state does not change, so we may observe several non-zero values in a given row of **B**, which of course should still sum up to 1. So the best-case sparsity of **B**, defined as the fraction of zero entries, can be calculated as

$$1 - \frac{K}{K \cdot N} = 1 - \frac{1}{N} . \tag{3.10}$$

The sparsity in **A** refers to the fact there are relatively few possible transitions from any given super-state. While this is not as obvious as in the case of **B**, it can be appreciated by realizing that multi-state appliances usually operate in *cycles* that determine the sequence of their states. For example, a possible state sequence for a dishwasher could be OFF $\rightarrow$ WASH $\rightarrow$ RINSE $\rightarrow$ DRY $\rightarrow$ OFF. Meanwhile, DRY $\rightarrow$ WASH $\rightarrow$ OFF would not make much sense.

To give a real-world example, we examined one year's worth of load data in the AMPds dataset [3] and found that **A** was 43.2% sparse while **B** was 97.3% sparse, when using two loads, clothes dryer (CDE) and kitchen fridge (FGE), each having 3 quantized internal states ($3^2 = 9$ super-states). The house had a 200A service, so $N = 200$ if whole Ampere measurements are used. In this scenario, the best-case sparsity for **B** would be $1 - \frac{1}{200} = 0.995$ or 99.5%.

Sparsity of **A** and **B** can be used to simplify computations involved in Viterbi-based state decoding, as will be described in the next section. In addition, storage requirements can be significantly reduced. To this end, we employ the *Harwell-Boeing sparse matrix format* [73] (Algorithm 3.2) to store **A** and **B** in compressed form. Figure 3.3(b) compares the amount of space needed for compressed and uncompressed **A** for disaggregating from 2 to 11 loads, based on the data from AMPds [3]. For 11 loads with 34,560 super-states, uncompressed **A** requires about 9.6 GB, while compressed **A** requires only 93.8kB. We also define a function to returning a list of tuples of all non-zero probability elements (see Algorithm 3.3) which will be used for optimizing the our sparse Viterbi algorithm. Figure 3.3(c) compares the amount of calculations saved when only non-zero probabilities are calculated.

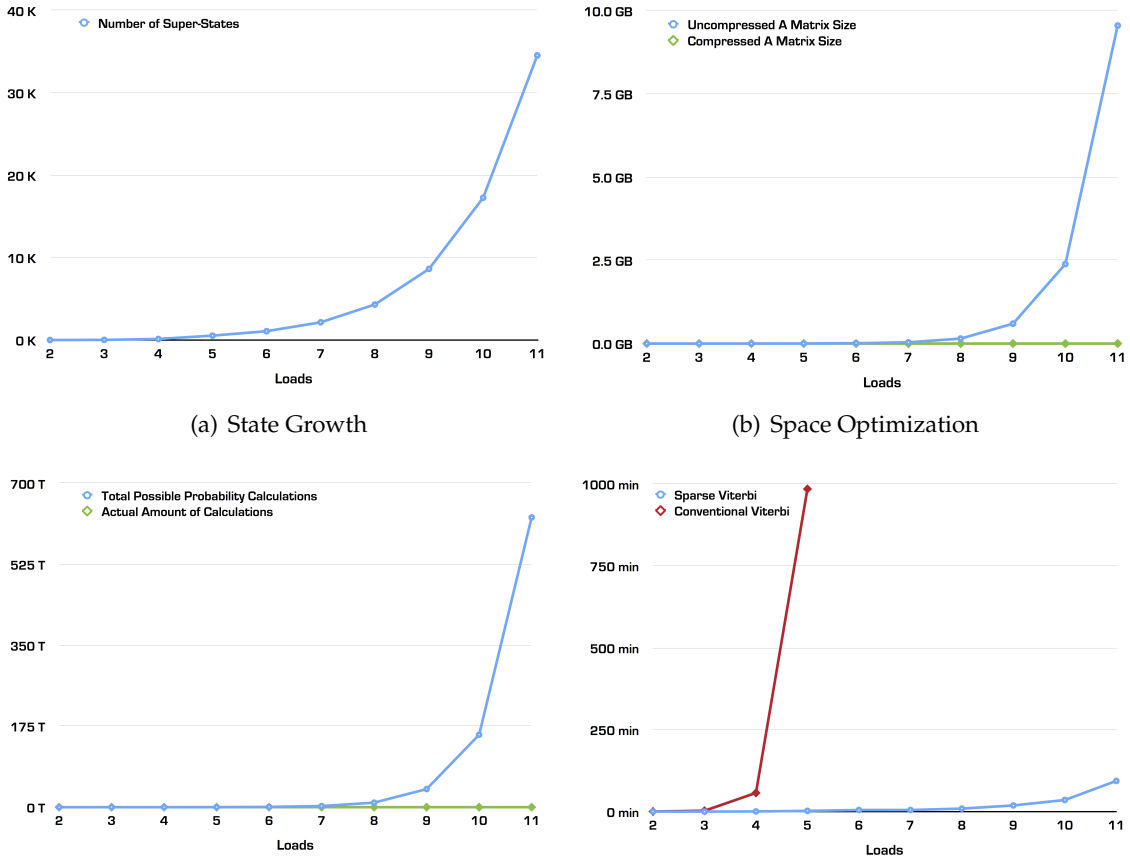
Algorithm 3.2 COMPRESS($\mathbf{M}$)

```
1:  # these vectors constitute our compressed matrix M
2:  val, row_idx ← []                                   # init vectors to empty
3:  col_ptr ← [1]                                       # init with initial value
4:
5:  for col = 1 → M[0].length() do                      # over all columns
6:      for row = 1 → M.length() do                     # over all rows
7:          if M[row, col] ≠ 0.0 then                   # non-zero prob only
8:              val.append(M[row, col])
9:              row_idx.append(row)
10:         end if
11:     end for
12:     col_ptr.append(row_idx.length() + 1)            # add next col pointer
13: end for
14:
15: return (val, row_idx, col_ptr)                      # return compressed M
```

Algorithm 3.3 COLUMN-VECTOR($\mathbf{M}$, col)

```
1:  v ← []                                              # init vector to empty
2:
3:  # start at the column pointer, loop adding the stored row and non-zero probability
4:  for i = M.col_ptr[col] → M.col_ptr[col + 1] − 1 do
5:      v.append((M.row_idx[i], M.val[i]))
6:  end for
7:
8:  return v                                            # return column vector
```

To show the benefits of taking advantage of sparsity, we used the AMPds. Our tests showed that with 2 loads, each having 3 states (9 super-states), for every 1 section of code execution (Algorithm 3.4, line 7) of sparse Viterbi, the conventional Viterbi algorithm would, on average, execute the same section 72.7 times. Figure 3.3(d) shows the runtime to disaggregate all readings (from 2 to 11 loads) by conventional and sparse Viterbi algorithms. The savings are considerable, especially for more than 4 loads (135 super-states). For example, for 2 loads, the execution time was reduced from 16.8 seconds to 11.4 seconds (32% faster) and for 11 loads (34,560 super-states) the execution time was 94 minutes. Tests were done on a MacBook Pro machine with a 2.6 GHz Intel Core i5 processor.



(a) State Growth

(b) Space Optimization

(c) Calculations Optimization

(d) Runtime Results

Figure 3.3: *Comparisons of optimization and runtime: (a) number of super-states (in thousands) as compared to the number of loads disaggregated; (b) the size (in GB) of compressed* vs *uncompressed matrix* **A**; *(c) a compression of calculations (in trillions) saved running our sparse Viterbi algorithm to disaggregate 524,544 readings; and (d) runtime (in minutes) comparison of conventional and sparse Viterbi algorithms when disaggregating 524,544 readings.*

3.5.4 Sparse Viterbi Algorithm

The standard Viterbi algorithm is well suited for largely populated matrices. However, when used with sparse matrices, there is an extensive amount of naive probability calculations involving zero-probability terms. Harwell-Boeing matrix compression not only reduces the amount of storage, but it also allows us to avoid calculating zero-probability terms. Taking advantage of matrix sparsity to avoid unnecessary calculations forms the basis of our algorithm called *sparse Viterbi algorithm* (see Algorithm 3.4), which is based on a greedy version of the Viterbi algorithm [57, pp. 352].

Let y_{t-1} and y_t be the total current measurements at times $t-1$ and t. The goal is to infer the quantized super-state $\widehat{s}_t$ (or, equivalently, its index k_t), from which we will determine the quantized internal states. This will be achieved by decoding the internal states' index from the super-state index using (3.7). These posterior probabilities are stored in vector $\mathbf{P}_{t-1}$ as part of *initialization* (Algorithm 3.4, lines 1–5), directly from [57].

$$\mathbf{P}_{t-1}[j] = \mathbf{P}_0[j] \cdot \mathbf{B}[j, y_{t-1}], j = 1, 2, ..., K\,. \tag{3.11}$$

The computation is reduced by only considering non-zero elements of $\mathbf{B}[j, y_{t-1}]$ in (5.4), to be clarified below. We now calculate the posterior probabilities for the current time period (the *recursion* step, Algorithm 3.4, lines 7–10)

$$\mathbf{P}_t[j] = \max_{i=1}^{K}(\mathbf{P}_{t-1}[i] \cdot \mathbf{A}[i, j] \cdot \mathbf{B}[j, y_t]), j = 1, 2, ..., K\,. \tag{3.12}$$

We *terminate* (Algorithm 3.4, line 12) to find the most likely current super-state index

$$k_t = \arg\max(\mathbf{P}_t)\,. \tag{3.13}$$

This algorithm is called each time we need to disaggregate a reading, using a *sliding window* of observations. For example, we disaggregate $t = \{1, 2\}$, then $t = \{2, 3\}$, $t = \{3, 4\}$, and so on. Disaggregation only begins when the first 2 observations are received from the meter. For our purposes we are not interested in the prediction of the super-state from $t-1$ (the *backtracking* step). Once the k_t is determined feedback is sent to the occupant – this makes it final, no turning back time. Note that we conducted additional experiments using $\mathbf{P}_t$ rather than the $\mathbf{P}_0$ suggested by Marsland in (5.4) for subsequent time periods, but there was a

degradation in accuracy results. We had hoped that using $\mathbf{P}_t$ would result in better accuracy, but it turned out the initial values of $\mathbf{P}_0$ were more reliable. Again, zero-probability terms are avoided in the calculation. The zero-probability terms are effectively ignored due to the Harwell-Boeing matrix compression method, which only stores non-zero elements of the corresponding matrices. A call to the function COLUMN-VECTOR() (Algorithm 3.4, lines 2, 5, and 6) only returns non-zero elements, which removes zero probability terms from the calculations.

Algorithm 3.4 DISCRETE-SPARSE-VITERBI($K, \mathbf{P}_0, \mathbf{A}, \mathbf{B}, y_{t-1}, y_t$)

```
1:  P_{t-1}[k], P_t[k] ← 0.0, k = 1, 2, ..., K                  # step 1: initialization
2:
3:  for (j, p_b) ∈ COLUMN-VECTOR(B, y_{t-1}) do
4:      P_{t-1}[j] ← P_0[j] · p_b
5:  end for
6:
7:  for (j, p_b) ∈ COLUMN-VECTOR(B, y_t) do                     # step 2: recursion
8:      (i, p_a)[] ← COLUMN-VECTOR(A, j)
9:      P_t[j] ← max_{(i,p_a)}(P_{t-1}[i] · p_a · p_b)
10: end for
11:
12: return arg max(P_t)                                         # step 3: terminate
```

3.5.5 Load Consumption Estimation

If we know the current/power draws of each appliance $y_t^{(m)}$, we could sum these levels

$$y_t \equiv \sum_{m=1}^{M} y_t^{(m)} , \tag{3.14}$$

and the total should equal the aggregate reading from the smart meter y_t. However, the consumption amount of each load is hidden because the state of each load is hidden. We can estimate the amount of consumption draw for each load by $y(\hat{x}_t^{(m)})$. The current draw of the decoded quantized state $\hat{x}_t^{(m)}$ is assumed to be the location of the corresponding PMF peak, i.e., $y(\hat{x}_t^{(m)}) = \mathbf{y}_{peak}^{(m)}[k_t^{(m)}]$, where $k_t^{(m)}$ is the index of the decoded quantized internal state of appliance m at time t. To find the estimate of the whole-house we simply

sum the load estimates

$$\hat{y}_t = \sum_{m=1}^{M} y(\hat{x}_t^{(m)}) = \sum_{m=1}^{M} \mathbf{y}_{peak}^{(m)}[k_t^{(m)}] \,. \tag{3.15}$$

As we will demonstrate with experiments in Chapter 5, this simple method is quite accurate.

3.6 Worked Example

We now describe how our approach would work with our studio suite example in Section 1.2. The data for our example scenario (Table 1.3) ends at 6:00pm. For illustration, let us say that at 6:01pm the smart meter reports an aggregate power reading of 3.48kW. We want to disaggregate this with our disaggregator. First we would need to build a model of the studio suite. This would involve using our priors (i.e. the data from Table 1.3) to build a PMF for each load (Table 5.4), a super-state HMM for the entire studio suite was: $T = 61$, $M = 5$, $N = 15000$ or 15kW (120V $\times$ 125A service), $K = 2^5 = 32$ super-states (since each load only has 2 states),

$\mathbf{P}_0 = [0, 0, 0, 0, 0.098, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.115, 0.082, 0.098, 0, 0.033, 0, 0.016, 0, 0.262, 0.033, 0, 0, 0.164, 0.049, 0.049, 0]$,
$\mathbf{A} = \{$ val $= [$ 0.857, 0.714, 0.2, 0.5, 0.143, 0.8, 0.143, 0.833, 0.5, 0.143, 0.167, 0.938, 0.111, 0.063, 0.5, 0.889, 0.333, 0.333, 0.5, 0.667, 0.667, 1 $]$, row_idx $= [$ 4, 16, 17, 20, 16, 17, 16, 18, 20, 4, 18, 24, 28, 24, 25, 28, 29, 30, 25, 29, 30, 22 $]$, col_ptr $= [$ 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 4, 6, 8, 8, 10, 10, 11, 11, 13, 15, 15, 15, 18, 20, 22, 22 $]$ $\}$ (a full matrix would be 97.85% sparse), and
$\mathbf{B} = \{$ val $= [$ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 $]$, row_idx $= [$ 0, 8, 16, 24, 2, 10, 18, 1, 26, 9, 17, 25, 3, 11, 4, 19, 12, 27, 20, 28, 6, 14, 22, 5, 30, 13, 21, 29, 7, 15, 23, 31 $]$, col_ptr $= [$ 0_1, 1_{250}, 2_{230}, 3_{250}, 4_{370}, 5_{250}, 6_{230}, 7_{20}, 8_{230}, 9_{20}, 10_{230}, 11_{250}, 12_{370}, 13_{250}, 14_{50}, 15_{180}, 16_{70}, 17_{180}, 18_{50}, 19_{250}, 20_{370}, 21_{250}, 22_{230}, 23_{20}, 24_{230}, 25_{20}, 26_{230}, 27_{250}, 28_{370}, 29_{250}, 30_{230}, 31_{250}, 32_{8571} $]$ $\}$ (*subscript values indicate number of times the number occurs*, a full matrix would be 99.99% sparse).

We can now perform inference and consumption estimation using our sparse Viterbi algorithm (Algorithm 3.4). We know that the observation at $t-1$ is $y_{t-1} = 3730$ and the

Table 3.3: *PMF and state quantizations for loads in the studio suite example*

(a) Lights, $m = 1$

n	0	...	480	...	15000
count(n)	6	0	55	0	0
$p_{Y_m}(n)$	0.10	0.00	0.90	0.00	0.00
$\mathbf{y}_{peak}^{(m)}$	0	480			
$k^{(m)}$	0	1			
$p(k^{(m)})$	0.10	0.90			

(b) Ent/TV, $m = 2$

n	0	...	250	...	15000
count(n)	27	0	34	0	0
$p_{Y_m}(n)$	0.44	0.00	0.56	0.00	0.00
$\mathbf{y}_{peak}^{(m)}$	0	250			
$k^{(m)}$	0	1			
$p(k^{(m)})$	0.44	0.56			

(c) Heating, $m = 3$

n	0	...	3000	...	15000
count(n)	36	0	25	0	0
$p_{Y_m}(n)$	0.59	0.00	0.41	0.00	0.00
$\mathbf{y}_{peak}^{(m)}$	0	3000			
$k^{(m)}$	0	1			
$p(k^{(m)})$	0.59	0.41			

(d) Microwave, $m = 4$

n	0	...	1100	...	15000
count(n)	51	0	10	0	0
$p_{Y_m}(n)$	0.84	0.00	0.16	0.00	0.00
$\mathbf{y}_{peak}^{(m)}$	0	1100			
$k^{(m)}$	0	1			
$p(k^{(m)})$	0.84	0.16			

(e) Kettle, $m = 5$

n	0	...	1600	...	15000
count(n)	51	0	10	0	0
$p_{Y_m}(n)$	0.84	0.00	0.16	0.00	0.00
$\mathbf{y}_{peak}^{(m)}$	0	1600			
$k^{(m)}$	0	1			
$p(k^{(m)})$	0.84	0.16			

current time observation is $y_t = 730$. We would call

$$\text{DISCRETE-SPARSE-VITERBI}(K, \mathbf{P}_0, \mathbf{A}, \mathbf{B}, 3730, 730).$$

The following computational steps would occur:

`Line 1:` previous time probability vector $\mathbf{P}_{t-1}$ of length K initialized to 0.0

`Line 1:` current time probability vector $\mathbf{P}_t$ of length K initialized to 0.0

`Line 3:` call to COLUMN-VECTOR($\mathbf{B}$, 3730) returns [(28, 1.0)]

`Line 4:` $\mathbf{P}_{t-1}[28] = \mathbf{P}_0[28] \cdot 1.0 = 0.164 \cdot 1.0 = 0.164$

`Line 4:` $\mathbf{P}_{t-1} = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.164, 0, 0, 0]$

`Line 7:` call to COLUMN-VECTOR($\mathbf{B}$, 730) returns [(24, 1.0)]

Line 8: call to COLUMN-VECTOR($\mathbf{A}, 24$) returns $[(28, 0.111)]$

Line 9: $\mathbf{P}_t[24] = \max([0, 0.018]) = 0.018$

Line 9: $\mathbf{P}_t = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.018, 0, 0, 0, 0, 0, 0, 0]$

Line 12: $k_t = \arg\max(\mathbf{P}_t) = 24$, which is the super-state or the state the studio suite is in

Line 12: $k_t = 24 \equiv \widehat{S}_t = [1, 1, 0, 0, 0]$, meaning that only the loads, lights and TV, are ON

We know that $k_t \equiv \widehat{S}_t$ because a call to KTH-CARTESIAN(24) (Algorithm 3.1) returns a vector of load states $[1, 1, 0, 0, 0]$ of length M which shows the lights and TV are both in state 1 or ON and the heating, microwave, and kettle are in state 0 or OFF. Knowing the load states allows us to now estimate the power consumption rate of each load, and indeed the whole studio suite. We do this by a simple summation, as defined in (3.15),

$$\sum_m \mathbf{y}^{(m)}_{peak}[k^{(m)}_t] = 480 + 250 + 0 + 0 + 0 = 730\,.$$

The consumption estimate is 730W, where the lights are consuming 480W, the TV is consuming 250W, and all other loads are OFF with 0W consumption.

3.7 Summary

We have presented our disaggregator which is our main contribution (**Contribution 5**) to this thesis. Our disaggregator has many advantages over existing disaggregators, mainly its ability to perform exact inference and preserve load dependencies. In the next Chapter we review existing accuracy measures and we explore how to test the accuracies of disaggregators in a consistent and comprehensive way.

Chapter 4

Accuracy

Trying to disaggregate loads from a single monitoring point is a difficult problem when it comes to accuracy. Creating accurate models of consumption relies on good priors (i.e. datasets) and a good way to measure the accuracy of the disaggregator. Load disaggregation is a fairly new research area and there is no accepted way nor accepted best practises to report algorithm accuracies. A review of disaggregator research has led us and others [35, 74] to the conclusion that there is no consistent way to measure performance accuracy. Any disaggregator performs two main functions: the classification of what load is running in what state, and the estimation of how much that load is consuming. Classification and estimation are often two separate and distinct functions that the disaggregator executes. This means different accuracy measures need to be considered, ones that best measure each function. In other words, the accuracy of classification and estimation functions need to be measured differently.

In recent years a number of datasets have been publicly released (including our own [3]) so that researchers can test the accuracy of their disaggregators (Section 4.1, **Contribution 3**). We review what types of accuracy measures, common to other research areas, that disaggregation researchers have used, such as basic accuracy and f-score (Section 4.2). We then review new accuracy measures created specifically for load disaggregation (Section 4.3). We critically discuss problems that exist with all these accuracy measures and then present our approach to accuracy reporting. We present a comprehensive approach (Section 4.4) supported by a worked example (Section 4.5). We show how any one single

accuracy measure can report high accuracies that may be misleading (**Contribution 4**).

4.1 Public Datasets

There are existing datasets for load disaggregation researchers to use – each with significant limitations. These datasets generally provide only a power measurement for the whole-house and/or multiple house loads. There are only a handful of datasets due to the costs involved with equipment purchase and installation. We briefly discuss some of the more popular datasets that exist as of writing this thesis.

The MIT Reference Energy Disaggregation Data Set or REDD [55] supplies high and low frequency readings specifically for residential load disaggregation for a short period of time (from a few weeks to a few months). Zeifmann [38] found that whole-house measurements were provided in *apparent power* and individual circuits were measured in real power. Consequently, the sum of individual circuits did not equal the whole-house. The CMU *Building-Level fUlly labeled Electricity Disaggregation* dataset or BLUED [75] contains high frequency readings of a single family home with a list of appliance events, but only for one week. The UMASS Smart* Home Data Set [76] contains high and low frequency readings, but is not specifically designed for NILM evaluation. The Tracebase dataset [77] contains appliance power traces sampled at intervals of one second. The Indian Dataset for Ambient Water and Energy (iAWE) [78] contains the data of 33 houses sampled at 1Hz. The UK Domestic Appliance-Level Electricity (UK-DALE) [79] which is formatted like REDD and has whole-house readings sampled at 16kHz and appliances sampled at $\frac{1}{6}$Hz. There are organizations that provide datasets for their customers. For example, Green Button has a number of sample datasets publicly available from their website [1]. The Plugwise dataset was used by Kolter [52] but is a private dataset only available upon written request to the company.

Recently there has been research devoted to the unification of dataset descriptions and the creation of a common metadata schema [80]. The main driving force for this is to have a standard way to import different datasets that have vastly different file structures. This was an addition to Batra et al. [81] releasing an open source NILM toolkit called NILMTK [2]. NILMTK can upload a number of different datasets into a common Python data structure.

[1] see *http://www.greenbuttondata.org/greendevelop.aspx*

[2] see *http://nilmtk.github.io*

It also provides statistical functions that filter out dataset errors and allow for data resampling. Two disaggregators are also included: a combinatorial optimizer and a factorial HMM.

4.2 Common Methods

The most basic accuracy measure used by a majority of load disaggregation researchers is defined as

$$\text{Acc.} = \frac{\text{correct matches}}{\text{total possible matches}} . \tag{4.1}$$

Tsai et al. [36] used this accuracy measure by employing correct signals matched (no further details), and Chang et al. [42] used recognition accuracy on both their training and the testing results. Both reported high accuracies of $\geqslant$95% and 100%, respectively. These numbers can be misleading because they do not measure the classification's performance [82, 83]. For example, if a fridge is running only 10% of the time and a disaggregator (100% of that time) says the fridge was not running it would have a measured accuracy of 90%. Kim et al. [35] points out that accuracy results are "very skewed because using an appliance is a relatively rare event .... appliances [that] are off will achieve high accuracy". Better accuracy performance measures need to be considered.

F-score (f-measure or F_1 score) is the harmonic mean of *precision* and *recall*:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} , \tag{4.2}$$

where precision is $tp \div (tp + fp)$ and recall is $tp \div (tp + fn)$, where tp are true-positive results, fp are false-positive results, and fn are false-negative results. This form of accuracy measure is often found in information retrieval and text/document classification. Figueiredo et al. [44,45] used f-score to measure their disaggregator based on 50 samples of data for each appliance. Berges et al. [34] used f-score for both training and testing of their disaggregator comparing 5.5 days of consumption where the disaggregator predictions of an appliance was compared to the corresponding plug-level meter readings. Kim et al. [35] argued that f-score measures binary classifier outcomes and power signals are not binary. This suggests that a better accuracy performance measure needs to be considered.

Around the same time, Zeifman et al. [74] showed there are two types of errors that need accuracy measures: false detection (Type I) and missed detection (Type II). He argued that the *receiver operating characteristic* (ROC) curve (see [82]) depicts the "trade-off" between the *specificity* (Type I) and *sensitivity* (Type II) of those errors and this "trade-off" needs to be assessed. Specificity is $\mathtt{tn} \div (\mathtt{tn} + \mathtt{fp})$ and sensitivity is the equivalent of recall. ROC uses specificity and sensitivity to evaluate the performance of an algorithm on one type of classification [83]. Later, Zeifman [38] abandons the ROC argument in favour of the *modified f-score* measure (discussed in the next section).

4.3 New Methods

Kim et al. [35] presented *modified f-score* (M-fscore) which they argued was more suited for measuring the accuracy performance of disaggregators. They argue that we should not only measure the accuracy of the classification of the state of the appliance, additionally we should measure the accuracy of the predicted appliance consumption. To measure the accuracy of state classification we define the binary classification as such:

$$\mathtt{label} = \begin{cases} \mathtt{positive}, & \text{if } \mathtt{power} > 0\,, \\ \mathtt{negative}, & \text{otherwise}\,. \end{cases} \tag{4.3}$$

To measure the accuracy of the predicted appliance consumption, $\mathtt{tp}$ is divided into 2 measurements: accurate true-positives ($\mathtt{atp}$), and inaccurate true-positives ($\mathtt{itp}$)– predicted power consumption that is significantly different from the ground truth would be labelled as $\mathtt{itp}$. The prediction is defined as follows:

$$\mathtt{prediction} = \begin{cases} \mathtt{tn}, & \text{if } \hat{c} = 0 \wedge c = 0\,, \\ \mathtt{fn}, & \text{if } \hat{c} = 0 \wedge c > 0\,, \\ \mathtt{fp}, & \text{if } \hat{c} > 0 \wedge c = 0\,, \\ \mathtt{atp}, & \text{if } \hat{c} > 0 \wedge c > 0 \wedge \delta \leqslant \rho\,, \\ \mathtt{itp}, & \text{if } \hat{c} > 0 \wedge c > 0 \wedge \delta > \rho, \end{cases} \tag{4.4}$$

where δ is the error ($\frac{\hat{c}-c}{c}$), ρ is the error threshold, c in the ground truth consumption, and

$\hat{c}$ is the predicted consumption. Precision is redefined as

$$\text{precision} = \frac{\text{atp}}{\text{atp} + \text{itp} + \text{fp}}\,, \tag{4.5}$$

and recall is redefined as

$$\text{recall} = \frac{\text{atp}}{\text{atp} + \text{itp} + \text{fn}}\,. \tag{4.6}$$

M-fscore is now the harmonic mean of the redefined precision and redefined recall.

Accuracies based on power estimation also need to be reported to show how accurately the disaggregator can estimate how much power is being consumed compared to actual (ground truth) consumption. This is important because systems that disaggregate need to report to occupants what portion of a power bill can be attributed to each appliance. Additionally, when dealing with time-of-use billing (charging more per kWh at peak times), occupants need to know how much might have been saved if certain appliances (e.g. a clothes dryer) were not used during the peak period.

A method used to report estimation accuracies was developed by Kolter and Johnson [55]. They provide the equation

$$\text{Est.Acc.} = 1 - \frac{\sum_{m=1}^{M}\sum_{t=1}^{T}|\hat{y}_t^{(m)} - y_t^{(m)}|}{2\cdot\sum_{m=1}^{M}\sum_{t=1}^{T} y_t^{(m)}} \tag{4.7}$$

where T is the time sequence length, M is the number of appliances, $\hat{y}_t^{(m)}$ is the estimated power consumed at time t for appliance m, and $y_t^{(m)}$ is the ground truth power consumed at time t for appliance m.

4.4 Our Approach

We have released our own dataset which focuses on low-frequency data that has clean values. Other datasets (e.g. REDD) have a high number of erroneous entries which each researcher has been left to clean up using different ways. This makes it hard to compare the results of different researchers that use REDD – their results may be skewed based on the way they scrubbed the data.

Basic accuracy measure is not suited for disaggregation and reports artificially high accuracies because of the rare nature of most loads running. F-score is a very widely used classification accuracy measure, but for binary classifications. With the majority of loads running in different states, f-score as it stands is not a good measure for disaggregation. M-fscore was derived to take into account the non-binary nature of disaggregation, but it combines classifications and estimation accuracy. While this gives a good overall score, it does not provide detailed diagnostic information of how accurate the classification task and the estimation task are on their own. However, we need to consider other issues.

4.4.1 Almanac of Minutely Power dataset

We have also released our own dataset *Almanac of Minutely Power dataset* or AMPds [3] (**Contribution 3**) [3]. Modern home appliances now have embedded electronics that allow for different modes of operation creating complex behaviours. The real challenge for disaggregators is the need to detect these complex, multi-state appliances and loads. To that end, we have created AMPds which contains complex, multi-state loads. Our dataset is a record of energy consumption of a single house using 21 sub-meters for an entire year (from April 1, 2012 to March 31, 2013) at one minute read intervals. An additional year's worth of data is planned for release. We chose a one minute interval due to concerns over data communication network saturation, but this comes at a cost of loss of fidelity (i.e. missing power measurement spikes that could help identify loads more easily) [53]. We monitored a house built in 1955 in the greater Vancouver region in British Columbia, which underwent major renovations in 2005 and 2006–receiving a Canadian Government EnerGuide [4] rating of 82% which is classified as an *energy-efficient house* (in the range of 80–90%).

Using branch circuit power metering (BCPM, see Figure 4.1) we metered 21 breakers from the house power panel. The two BCMP units were queried once per minute by an industrial data acquisition server (see Figure 4.2(a)). Table 4.1 lists the BCPM measurements captured.

For natural gas metering there were two meters: the whole-house meter (WHG) and the gas furnace meter (FRG). For water metering there were also two meters: the whole-house meter (WHW) and the hot water meter (HTW). Figure 4.3 shows the pulse meters

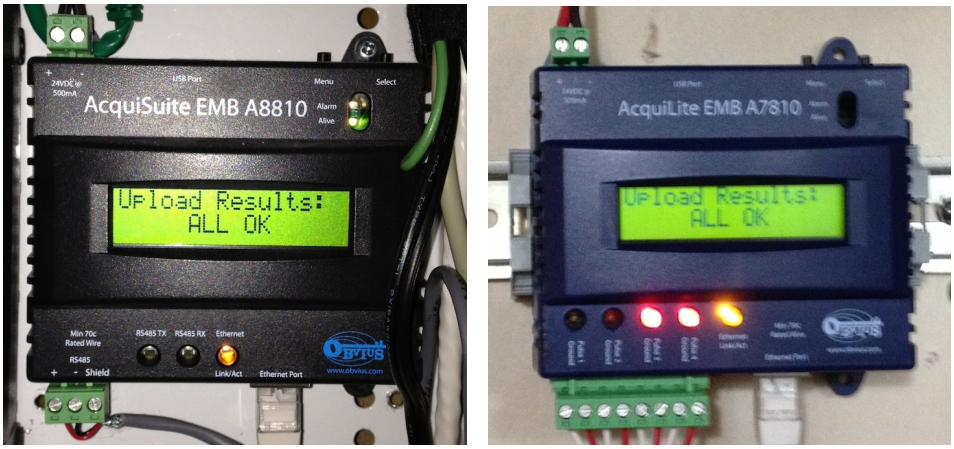
[3] dataset can be download from *http://ampds.org*

[4] see *https://www.nrcan.gc.ca/energy/efficiency/housing/new-homes/5035*

Figure 4.1: *Two DENT PowerScout 18 units metering 24 loads at the electrical circuit breaker panel. Measurements are read over a RS-485/Modbus communication link by data acquisition equipment (see Figure 4.2(a)).*

used. Table 4.2 lists the pulse measurements captured by the data acquisition server (see Figure 4.2(b)) for both natural gas and water consumption.

The data acquisition servers (Figure 4.2) push data captured to a remote, off-site MySQL server via an HTTP POST. When creating the AMPds *comma separated value* (CSV) files, we first cleaned the dataset by removing incomplete captures (i.e. some sub-meters had data missing for different timestamps), resulting in the removal of 1,054 rows. The dataset contains 524,544 valid readings per sub-meter. For the power metering we added a meter labelled UNE (or unmetered loads) as a *soft-meter* that is calculated by subtracting the sum of the sub-meters from the WHE whole-house power meter. Four sub-meters (the gas cooktop sub-meter, the microwave sub-meter, and the partial lights sub-meter) were removed from the dataset for they did not have enough activity.



(a) Power Data Acquisition (b) Gas and Water Data Acquisition

Figure 4.2: *Data acquisition units: (a) is an Obvius AcuiSuite EMB A8810 for communicating via Modbus to the 2 branch circuit power meters (BCPM), and (b) is an Obvius AcuiLite EMB A7810 for recording pulses from the natural gas and water meters. These units have a maximum read rate of once per minute.*

4.4.2 Ground Truth and Bias

Disaggregation researchers need to describe in detail the data they are using to build models, to train them, and to test those models on their algorithms. If they are using data from publicly available datasets such as AMPds [3] and REDD [55], they need to discuss the method used to clean the data. For instance, how they dealt with incomplete, mismatched, or erroneous data. These specifications form part of **Contribution 4**.

There also needs to be a clear statement on whether the testing included *noise* or was *denoised*. Noise can be defined as the amount of power remaining once all the power reading from each load has been subtracted from the whole-house power reading:

$$\text{noise} = y_t - \sum_{m=1}^{M} y_t^{(m)}, \tag{4.8}$$

where y_t is the total ground truth or observed value at time t, M is the number of appliances, $\hat{y}_t^{(m)}$ is the estimated power consumed at time t for appliance m, and $y_t^{(m)}$ is the ground truth power consumed. In denoised data, the whole-house power reading is equal to the summation of all appliance power readings – which we often refer to as the *unmetered* load or appliance. Using denoised data will cause higher accuracies to be reported but does not reflect a real-world application. Further, what needs to be reported is

(a) Gas Main Meter

(b) Gas Furnace Meter

(c) Water Main Meter

(d) Hot Water Meter

Figure 4.3: *Pulse meters for natural gas (a), (b) and water (c), (d); (a) is an Elster AC250 gas meter, (b) is an Elster BK-G4 gas meter, and (c) and (d) are Elster/Kent V100 water meters. Measurements are electrical pulses that are read by data acquisition equipment (see Figure 4.2(b)). Each pulse represents a quantity on consumption.*

Table 4.1: *Electricity Measurements Captured*

Column	Description	Units
0	Unix Timestamp (since Epoch)	s
1	Voltage (V)	V
2	Current (I)	A
3	Frequency (f)	Hz
4	Displacement Power Factor (DPF)	ratio
5	Apparent Power Factor (APF)	ratio
6	Real Power (P)	W
7	Real Energy (Pt)	Wh
8	Reactive Power (Q)	VAR
9	Reactive Energy (Qt)	VARh
10	Apparent Power (S)	VA
11	Apparent Energy (St)	VAh

Table 4.2: *Natural Gas & Water Measurements Captured*

Column	Description	Gas Units	Water Units
0	Unix Timestamp (since Epoch)	s	s
1	Pulse Counter	dm^3	L
2	Average Rate	dm^3/h	L/min
3	Instantaneous Rate	dm^3/h	L/min

the percent-noisy of each test. This *percent-noisy measure* (%-NM) would be calculated on the ground load data as such:

$$\%\text{-NM} = \frac{\sum_{t=1}^{T} |y_t - \sum_{m=1}^{M} y_t^{(m)}|}{\sum_{t=1}^{T} y_t} \tag{4.9}$$

where y_t is the aggregate observed current/power amount at time t and $y_t^{(m)}$ is the ground truth current/power amount for each appliance m to be disaggregated. For example, a denoised test would result in 0%; whereas, a %-NM of 0.40 would mean that 40% of the aggregate observed current/power for the whole test was noise.

Researchers should use standard methods to minimize any effects of bias. Bias occurs when data used for training is also used for testing which would result in the reporting of higher accuracies. A well accepted method used by the data mining community to avoid

bias is *10-fold cross-validation* [84, pp. 109]. This simple method splits the ground truth data into 10 subsets of size $\frac{n}{10}$. Disaggregators can be trained on 9 of the subsets and accuracy testing is performed on the excluded subset. This is repeated $10\times$ (each time a different subset is used for testing) and the mean accuracy is then calculated and reported. We report the dataset we use as ground truth and any modifications that were made. This could include: modifications such as down-sampling, the cleaning of erroneous data, what loads were used, and how the aggregate house reading modified, if it was. Modifications to aggregate readings may be done to denoise the data and exclude the consumption amount from the loads not disaggregated. We also have chosen to use *10-fold cross-validation* to minimize bias in our accuracy results.

4.4.3 Accuracy Measures

Kim *et al.* [35] approach combined load state classification and power estimation accuracies even though in many instances classification and estimation are two distinct functions of disaggregation. Combining classification and estimation hides important diagnostic information as to what parts of disaggregation has low accuracy. Furthermore, these accuracies require a specific type of accuracy measure that is suited for that function, which can lead to better diagnostic and performance information.

For classification accuracy measure we use f-score as defined in (4.2). However, we repurpose and modify Kim's [35] M-fscore to measure the classification of loads states only, which we call *finite-state f-score* (FS f-score). FS f-score (**Contribution 4**) uses precision as defined in (4.5) and recall as defined in (4.6). We redefine atp (accurate true-positives) and itp (inaccurate true-positives) as penalization for inaccurate classification of load state and not penalization for inaccurate power estimation. If fp are the false-positives (predicted but not real) classifications, and fn are the false-negatives (real but not predicted) classifications, then the binary classification of tp (true-positives) would be $tp = atp + itp$. We now can penalize the classification score based on how far the predicted state is from the ground truth state. To calculate itp, we use

$$itp = \frac{|\hat{x}_t^{(m)} - x_t^{(m)}|}{K^{(m)}}, \tag{4.10}$$

where $\hat{x}_t^{(m)}$ is the estimated state from appliance m at time t, $x_t^{(m)}$is the ground truth state,

and $K^{(m)}$ is the number of states for appliance m. To determine the atp we use the equation

$$atp = 1 - itp \, . \tag{4.11}$$

Tracking atp, itp, fp, and fn for each appliance would allow for appliance specific reporting. A summation over all loads M on each of atp, itp, fp, and fn would allow for overall classification accuracy reporting. We report both overall and load specific classification accuracy scores.

As we discussed before, accuracies based on power estimation also need to be reported (both overall and load specific). The method we use to report estimation accuracies is defined in (4.12). This method allows for overall estimation accuracy reporting. By eliminating the two summations over M we can then report estimations for each appliance

$$Est.Acc.^{(m)} = 1 - \frac{\sum_{t=1}^{T} |\hat{y}_t^{(m)} - y_t^{(m)}|}{2 \cdot \sum_{t=1}^{T} y_t^{(m)}} \, . \tag{4.12}$$

Both classification accuracy and estimation accuracy need to be reported in overall scores and in appliance specific scores. Reporting how each appliance scores is important for identifying strengths and weaknesses of different disaggregators. With this more detailed accuracy information, one could imagine a system that could select different algorithms depending on the context (including specific history) of the disaggregation task. Finally, although more detailed information has its advantages, reporting specific scores for appliance states is not necessary because different makes/models of appliances will have a different number for states at different power levels.

4.5 Worked Example

We investigated how basic accuracy can be misleading, by reporting high confidence numbers that do not accurately reflect inaccuracies, at predicting rare events. For many loads changing state is rare. We also show why M-fscore, which combines classification and estimation, is not a detailed enough measure. We used AMPds data for this test. Current draw (I) values were rounded up to the nearest whole-Ampere and 10-fold cross-validation was used on the entire 1 year of data. The whole-house current draw measurement was denoised so that it equalled the summation of the current draw from the 11 loads chosen for

disaggregation. The results are listed in Table 4.3.

Table 4.3: *Sample Accuracy Results For Disaggregating Using Whole-Ampere Measurements*

ID	Load	Accuracy	Precision	Recall	FS F-Score	Estimation
	Overall Score	97.3%	90.4%	92.8%	91.6%	99.4%
BME	Basement	96.3%	85.5%	85.2%	85.3%	96.6%
CDE	Clothes Dryer	99.3%	93.7%	68.3%	79.0%	97.1%
CWE	Clothes Washer	97.9%	35.6%	7.0%	11.6%	60.1%
DWE	Dishwasher	98.8%	71.7%	74.2%	72.9%	88.7%
FGE	Kitchen Fridge	88.2%	79.1%	91.6%	84.9%	93.6%
FRE	HVAC/Furnace	99.8%	99.9%	99.9%	99.9%	98.8%
GRE	Detached Garage	99.9%	1.3%	0.3%	0.5%	54.0%
HPE	Heat Pump	99.7%	98.4%	97.3%	97.8%	99.8%
OFE	Home Office	94.7%	29.6%	23.4%	26.1%	87.8%
TVE	Entertainment/TV	95.4%	70.4%	86.8%	77.7%	95.5%
WOE	Wall Oven	99.8%	75.0%	61.3%	67.5%	96.6%

Table 4.3 reports basic accuracy score as being far better than FS f-score. This is most noted for the results of the GRE load. Overall, the FS f-score and estimation of our test scores high, but this masks the fact that loads: CWE, GRE, and OFE did not score well at all. As well, the OFE load shows how there can be a high estimation score but a very low FS f-score – which means reporting only estimation hides low classification scores.

4.6 Summary

We have introduced our own dataset (AMPds) and have critically discussed problems that exist with all accuracy measures. We then presented our approach to accuracy reporting, which we believe is a comprehensive approach which was followed by a worked example. This chapter was critical in laying the ground work for the next chapter that reports the results of different experiments we tested on our disaggregator.

CHAPTER 5

EXPERIMENTS

In this chapter we will show how the implementation of our disaggregator (μDisagg) meets all our claims listed in **Contribution 5**. We start by defining a number of terms that we will use consistently throughout this chapter when reporting results (Section 5.1). Next, we discuss how we performed our experimental tests and the testing procedure we used (Section 5.2). Then, we discuss how we modified our disaggregator to use continuous emissions to compare against discrete emissions (Section 5.3). These sections set the foundation for our experimental test.

We want to show there is indeed dependency amongst loads and in some cases the correlation is quite high. In Section 5.4 we use correlation coefficient tests to show load dependencies (**Contribution 5e**) which, along with exact inference (**Contribution 5f**), μDisagg uses. How μDisagg uses load dependencies and exact inference was discussed in Chapter 3. In Sections 5.5 and 5.6 we show how μDisagg can disaggregate using different low-frequency sampling rates and different measurements (**Contribution 5a**) while achieving a high degree for both load classification and consumption estimation (**Contribution 5b**) for multi-state appliances (**Contribution 5d**). In Section 5.7 we use the Arduino Due to test the ability of μDisagg to run on an embedded processor in real-time (**Contribution 5c**).

5.1 Nomenclature

There are a number of standard terms we use when reporting experimental results in our findings tables. We now explain their meaning here.

Test Type Each test ran on the same dataset and with the same loads, but was configured slightly differently. These different test configurations were: Discrete Denoised, Discrete Noisy, Discrete Model, Continuous Noisy, and Continuous Model. Below is an explanation of each type of configuration.

Discrete A discrete configuration used an emission matrix that was generated with the algorithms discussed in Section 3.5.2. The emission matrix column index was a quantized value where $y \in [0, N]$. For example, if $y = 5.6$ then the column index for the emission matrix would be column 56 (in dA precision).

Continuous A continuous configuration used normal distributions for emission probabilities. One normal distribution per super-state was fitted, creating an emission vector. See Section 5.3 for further details.

Denoised A denoised configuration removes any noise (as defined in (4.8)) from the aggregate reading so that the aggregate observed value y is equal to the sum of the hidden values $y^{(m)}$ (in ground truth) of each load m of M loads in our model: $y = \sum_{m=1}^{M} y^{(m)}$.

Noisy A noisy configuration does not remove the noise in the aggregate observed value y defined in (4.8), nor does it try to model the noise as a load. To us, this represents a more realistic configuration to test against. We will discuss this more in Section 5.5.

Model A model configuration treats the noise in ground truth (as defined in (4.8)) as a load we label *unmetered*. A PMF is created for the unmetered load and it is quantized into states. As with other loads, it was integrated as part of a super-state. This is why, for this test, there was one more load compared to the equivalent denoised or noisy test. It is harder to model noise because it can have many more states than a real load. For our tests we needed to constrain the amount of states to eight to maintain computational efficiency in both space and time.

$K^{(M)}, K, M, T$ **Variables** We report on a number of model variables: $K^{(M)}$ is the average number of states per load, K is the number of super-states in our model, M is the number of loads in our model, and T is the number of readings that were disaggregated per fold.

Rate Here we report the rate/speed (in seconds/sample) of μDisagg to disaggregate one aggregate observed value y.

Unseen Here we report the number of observations, when inference ran, that resulted in a zero-probability. This happens when inference does not find the appropriate matrix element because an observation or transition had not been present in the training during the building of the **A** and **B** matrices.

Noise Here we report our percent-noisy measure (%-NM) defined in (4.9).

Acc Here we report the basic accuracy measure as defined in (4.1). Although this measure is not the best way to report accuracy, we include it because a majority of disaggregation researchers use this measure and we want to show how inaccurate it can be as compared to other accuracy measures.

Precision, Recall, F-Score Here we report our Finite State version of these normally binary measures which we discussed in Section 4.4.3.

Est Acc Here we report the consumption estimation accuracy measure defined in (4.12).

5.2 Experimental Setup

To support our claims in **Contribution 5**, we chose two low-frequency datasets: AMPds [3] and REDD [55]. Both datasets differ in terms of low-frequency sampling rates (per minute *vs* per 3-seconds) and measure types (current *vs* apparent power). A prototype of μDisagg was first coded in Python 3.4. We chose Python because of list manipulation capabilities and its ease in creating rapid prototypes that are easily translated into C (at a later point). Our prototype ran as a single threaded process and would disaggregate the specified loads in all the readings of a given dataset. The final version was coded in C and ran on our Precision Ammeter (Section 2.5.4) which we will discuss in Section 5.7. All tests, with the exception of the embedded testing in Section 5.7, ran on a Mac Pro (Late 2013 model) with

a 3.5GHz 6-core Intel Xeon E5 processor and 64GB of memory. To mitigate bias, each test used 10-fold cross-validation.

Algorithm 5.1 shows the structure and main logic of the program used for running tests on our disaggregator. Initially we would need to specify a list of loads we want to disaggregate (line 1). Then, we pick an option on how to handle noise (line 2) and a way to model emissions (line 3). We set the denoise flag based on the noise option (lines 5–8). If we want to model noise, based on the noise option, then add a load called unmetered (lines 10–12). Then, we set the number of loads (line 13). Next, we read in the desired dataset file (line 15) and split it into 10 partitions (line 16), one for each fold. We begin looping once for each fold (line 17). Then, we assign the next fold's test data (lines 18) and the training data making sure the test data is not part of the training data (line 19). We create a PMF for each load (line 21). Next, we quantize the PMF of each load into a set of load states and find their peak values (line 22). Finally, we create the super-state HMM models based on the training data, previous calculated data, and emissions option (line 24). The training, or model building, phase is complete for this fold.

We start testing by setting the previous observation value from the first reading in the test data (lines 26) and begin looping through each reading as a test (line 27). We then set the current observed value from the test data (line 28). Next, we call the appropriate sparse Viterbi algorithm based on the emissions option and return the inferred super-state (lines 29–35). We determine the state of each load from the super state (line 36) and estimate what the observed value is (line 37). Then, we evaluate the test results against ground truth (line 38). Our final test step is to assign the current observation as the previous observation (line 39). We continue looping until all test data in all folds are processed.

5.3 Continuous Emissions

Previously we built a discrete emission matrix $\mathbf{B}[j, n]$ based on priors – what we call the *discrete version* of our disaggregator. We had concerns that any unseen observation would cause inaccurate predictions which would result in the return of zero-probabilities. We wanted to see what effect the inability of handling unseen observations would have on the accuracy of our discrete model. As such, we modified our existing super-state HMM and sparse Viterbi algorithm to allow for continuous emissions for comparative testing purposes. Doing so allowed unseen observations to have a non-zero probability. This

Algorithm 5.1 TESTING-PROCEDURE :

```
load_ids ← ['Fridge', 'Oven', 'Heater']                          # for example
noise ← 'no' or 'yes' or 'model'                                 # gets one of these values
emissions ← 'discrete' or 'continuous'                           # gets one of these values

denoise ← false
if noise = 'yes' then
    denoise ← true                                               # set to remove all noise
end if

if noise = 'model' then
    load_ids ← load_ids.append('Unmetered')                      # add noise as a load
end if
M ←len(load_ids)                                                 # the number of loads

data ← load_dataset('AMPds.csv', denoise)                        # for example
data ← split(data, 10)                                           # partition into 10-folds
for i = 1 → 10 do
    test ← data[i]                                               # set the test data
    train ← data − test                                          # remove the test data

    (PMF, N) ← create_pmfs(train, M, N)                          # defined in Section 3.5.1
    (K, K^(m), y_peak^(m), y_bound^(m)) ← quantize_pmfs(PMF)      # determine load states
                                                                 # defined in Section 3.5.2
    (P_0, A, B) ← create_sshmm(train, M, N, K, K^(m), y_peak^(m), y_bound^(m), emissions)

    y_0 ← test[0]                                                # the aggregate y only
    for j = 1 → len(test) do
        y_1 ← test[j]                                            # the aggregate y only
        if emissions = 'discrete' then
                                                                 # see Algorithm 3.4
            k ← DISCRETE-SPARSE-VITERBI(K, P_0, A, B, y_0, y_1)
        else
                                                                 # see Algorithm 5.2
            k ← CONTINUOUS-SPARSE-VITERBI(K, P_0, A, B, y_0, y_1)
        end if
        s ← KTH-CARTESIAN(K^(m), k)                              # see Algorithm 3.1
        y_est ← estimate_consumption(y_peak^(m), k)              # defined in (3.14)
        eval_accuracy(test[j], k, s, y_est)                      # defined in Section 4.4.3
        y_0 ← y_1
    end for
end for
```

version we term as our *continuous version*.

We say that the probability of super-state S_t at time t given an observation y_t is distributed normally $p(S_t = j|y_t) \sim \mathcal{N}(\mu, \sigma^2)$, where the normal (Gaussian) distribution $\mathcal{N}$ is defined by mean μ and variance σ^2. Instead of populating our emission matrix during model building, we now fit $\mathcal{N}$ by calculating the recursive mean [85]

$$\mu_t = \frac{t-1}{t}\mu_{t-1} + \frac{1}{t}y_t \,, \tag{5.1}$$

and recursive variance [85]

$$\sigma_t^2 = \frac{t-1}{t}\sigma_{t-1}^2 + \frac{1}{t-1}(y_t - \mu_t)^2 \,, \tag{5.2}$$

for each $t \in \{0, 1, ..., T\}$ in our priors. While μ_t and σ_t^2 are intermediate values, once all prior data has been processed, they are noted as a final μ and a final σ^2. We populate an emission vector $\mathbf{B}$ with the tuple of final values (μ, σ^2). We fit one $\mathcal{N}$ per super-state. Our continuous sparse Viterbi algorithm then uses the normal distribution function [86, pp. 89]

$$p(S_t = j|y_t) = g(y; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{y-\mu}{\sigma})^2} \tag{5.3}$$

to calculate the emission probabilities of super-state S_t given the observed current/power draw y_t at time t. Further, we can now use decimal numbers as observations. In our discrete version we needed to convert the decimal number into an integer (e.g. 1.2A $\rightarrow$ 12dA) that would be used as a column index for our emission matrix $\mathbf{B}$. There is the further benefit of having a reduction in the amount of storage space required for $\mathbf{B}$. However, this has increased the amount of computational time needed to perform each disaggregation (see the Rate column of test results tables).

Algorithm 5.2 is the version of our discrete sparse Viterbi algorithm (described in Section 3.5.4) that uses continuous emissions. These posterior probabilities are stored in vector $\mathbf{P}_{t-1}$ as part of *initialization* (Algorithm 5.2, lines 1–5).

$$\mathbf{P}_{t-1}[j] = \mathbf{P}_0[j] \cdot g(y_{t-1}; \mathbf{B}[j].\mu, \mathbf{B}[j].\sigma^2), j = 1, 2, ..., K \,. \tag{5.4}$$

Unlike our discrete algorithm, computation is not reduced because there are no fixed matrix elements with probability of zero. We now calculate the posterior probabilities for the current time period (the *recursion* step, Algorithm 5.2, lines 7–10)

$$\mathbf{P}_t[j] = \max_{i=1}^{K}(\mathbf{P}_{t-1}[i] \cdot \mathbf{A}[i,j] \cdot g(y_t; \mathbf{B}[j].\mu, \mathbf{B}[j].\sigma^2)), j = 1, 2, ..., K\,. \tag{5.5}$$

We *terminate* (Algorithm 5.2, line 12) to find the most likely current super-state index

$$k_t = \arg\max(\mathbf{P}_t)\,. \tag{5.6}$$

This algorithm is programmatically called the same way as our discrete version. Each time we need to disaggregate, we use a sliding window of observations $t-1$ and t.

Algorithm 5.2 CONTINUOUS-SPARSE-VITERBI$(K, \mathbf{P}_0, \mathbf{A}, \mathbf{B}, y_{t-1}, y_t)$

```
1:  P_{t-1}[k], P_t[k] ← 0.0, k = 1, 2, ..., K                    # step 1: initialization
2:
3:  for j = 1 → K do
4:      P_{t-1}[j] ← P_0[j] · g(y_{t-1}; B[j].μ, B[j].σ^2)
5:  end for
6:
7:  for j = 1 → K do                                               # step 2: recursion
8:      (i, p_a)[] ← COLUMN-VECTOR(A, j)
9:      P_t[j] ← max_{(i,p_a)}(P_{t-1}[i] · p_a · g(y_t; B[j].μ, B[j].σ^2))
10: end for
11:
12: return arg max(P_t)                                            # step 3: terminate
```

5.4 Load Dependency

We ran a number of correlation coefficient tests on the loads we would use for testing in both the AMPds [3] and REDD [55] datasets. Such a discussion is usually avoided as researchers using factorial-based models note load dependency is lost. To begin our correlation tests, we needed to normalize ([0,1]) the reading, in the dataset, of each load we wanted to test in each dataset

$$y_t^{'} = \frac{y_t}{\sum_{i=1}^{T} y_i}, j \in 1, 2, ..., T\,, \tag{5.7}$$

where y'_t is the normalized value, y_t, y_i are the measured readings at time t, and T is the total number of readings. We used the `corrcoef()` function [1] from the Python NumPy library to calculate the correlation coefficients. Each call returns a matrix which we present in various tables. We determine the strength of a correlation below from Table 5.1 [87].

Table 5.1: *Correlation Coefficient Rule of Thumb*

Positive Coefficient	Negative Coefficient	Strength of Relationship
0.5 to 1.0	-1.0 to -0.5	Strong
0.3 to 0.5	-0.5 to -0.3	Moderate
0.1 to 0.3	-0.3 to -0.1	Weak
0.0 to 0.1	-0.1 to 0.0	Very Weak
0.0	0.0	None

When examining AMPds, we first looked at loads we would use for accuracy tests in Section 5.5. There is a weak correlation (0.28) between the heat pump and the HVAC system (Table 5.4(a), page 86) which confirms our observation made earlier in Section 3.5 but it was not as strong as we thought it might be. If we examine all the sub-metered loads in AMPds (Table 5.5, page 87), we find other interesting load dependencies. There is a strong correlation (0.70) between the basement plugs and lights and the entertainment equipment (TV, DVD, etc). This would be the case because at night time the basement rec room is used to watch TV. There is also a strong correlation (0.52) between the utility room plug and the electronics workbench, but we are not certain why. There are a number of other weak correlations that we have not mentioned, but may warrant further investigation for using correlations to infer occupant activity and augment previous research we performed on occupant activity [4, 13].

When examining REDD we ran correlation tests on the four houses we would use for accuracy testing (House 1, 2, 3, and 6) in Section 5.6. In House 1, there is a strong correlation (0.73) between the heater and the microwave (Table 5.4(b), page 86). House 2 and House 3 had at best very weak correlations. The highest correlation in House 2 was 0.10 between the lights and the microwave (Table 5.4(c), page 86). The highest correlation in House 3 was also 0.10 but between the lights and the fridge (Table 5.4(d), page 86). House 6 had a weak negative correlation (-0.21) between the fridge and the dishwasher and a weak positive correlation (0.16) between the lights and the heater. One interesting find was in

[1] see *http://docs.scipy.org/doc/numpy/reference/generated/numpy.corrcoef.html*

House 4, where there was a strong correlation (0.81) between the air conditioning and stove, and a moderate correlation (0.35) between the air conditioning and clothes dryer. This would suggest the use of the stove and clothes dryer created enough heat to have the air conditioning turn ON in the home.

The results of these load dependancy tests confirm some loads have correlation with others. Super-states take advantage of this by having each super-state as a set of load states. The combinatorial nature of the super-state means dependancies are preserved as the HMM is being built. Factorial models and those that treat loads as independent HMMs or Markov chains cannot preserve these load dependancies like a super-state HMM can. We expect our disaggregator will have a clear advantage over other disaggregators because we preserve load dependancies. In Section 5.6 we will test our super-state HMM against the published results of two other factorial models.

5.5 Testing Deferrable Loads

Our previous discussions around deferrable actions (Section 1.1) had large appliances, such as a clothes dryer, run when the cost of electricity was cheaper, called off-peak hours. We argued it is important to be able to disaggregate these types of loads. This should be the focus of a disaggregator because the deferral of running such loads has a direct benefit to the occupants (saving money) and may have a direct benefit to the power grid as a whole. Deferring large loads from running during peak times of usage can ensure power grid stability and avoid grid brownouts. This idea is often discussed as *peak shaving* [88]. The benefit of having a disaggregator only disaggregate deferrable loads is: (a) they can be more easily identified due to their large consumption – better accuracy; and (b) they can run faster as there are less loads to disaggregate. However, the disaggregator must be robust enough to handle a large percentage of noise. This is because the noise would contain many other loads that would be running in the house. Contrarily, robustness against noise has a long term benefit because the load profile/makeup of a house could change without affecting the disaggregator's ability to identify these deferrable loads. We are very much interested in evaluating how μDisagg would perform in a situation such as this.

To test the accuracies for disaggregating deferrable loads we used AMPds [3]. The deferrable loads we chose were the clothes dryer, the dishwasher, the HVAC fan, the heat pump, and the kitchen wall oven. We ran all five test types (Discrete Denoised, Discrete

Noisy, Discrete Model, Continuous Noisy, and Continuous Model) on the entire one year worth of data (524,544 readings). The readings within APMds do not contain errors so there was no need to clean or convert the data. We ran tests using per minute sampling (minute tests) and per hour sampling (hour tests). For the per minute sampling tests we used Ampere measurements (at dA or 0.1 precision). When summarizing, we ignored reporting numbers for the Discrete Denoised tests as they did not represent a realistic scenario. We reported Discrete Denoised in the results tables for comparison as to what the ideal result would be.

Table 5.6 (page 88) shows the overall results for our minute tests. All five test types scored very high with the lowest score being 91.6% for recall for the Continuous Model. Looking at the specific load results for the noisy and model tests in Table 5.7 (page 88), we noticed that in all four tests the accuracy results for the dishwasher scored very low, while the other loads scored quite high. This is mainly due to having two of its load states (at peaks 0.4A and 1.2A) similar to other loads (2 loads and 3 loads respectively). In the model tests, the unmetered load scored low because we restricted the amount of states to a maximum of 8. We compared the percentage of total consumption for each load of the noisy and model tests with that of ground truth (Figure 5.2, page 89). In this test, the discrete test faired better than the continuous tests.

We also wanted to test how well our disaggregator would disaggregate very low-frequency sampling data – the hour tests. In these tests we used Watt-hour measurements (at daWh or Wh÷10 precision). We accumulated the Watt-hours for each hour over the entire year for a total of 17,520 readings. Table 5.8 (page 90) shows the overall results of the hour tests. The Discrete Noisy test performed with high accuracy (>85%) followed closely by the Discrete Model test. The continuous test did not fair as well scoring in the mid-70%. Examining the load specific scores in Table 5.9 (page 90), we noticed that the dishwasher, again, scored low. However, we also found the oven scored even lower often having a FS f-score of 0%. There were misclassifications that occurred because of the high level of noise and the infrequent, nonuniform use of the oven. We were surprised our disaggregator scored as well as it did after reading Kolter's [52] published results for hourly disaggregation. His discriminative disaggregator achieved an estimation accuracy of 55.1% which is 20.4% lower than our worst performing test, Continuous Model. Kolter used a different dataset (Plugwise) which contained a large number of houses, so it is hard to do a direct comparison of his results versus ours.

5.6 Comparing Disaggregators

To compare the accuracy of μDisagg to that of others [55,60] we used the REDD dataset [55] (a smaller dataset compared to AMPds, but data sampled every 3 seconds) and used the same estimation accuracy measure defined in (4.12). We wanted to compare our results against the Kolter 2011 [55] results and the Johnson 2013 [60] results. Kolter and Johnson [55] initially reported accuracies on Houses 1, 2, 3, 4, and 6 using a disaggregator that used supervised learning. Johnson et al. [60] reported accuracies on Houses 1, 2, 3, and 6 using a disaggregator that they claimed used unsupervised learning. However, their claim of unsupervised learning was incorrect. They used a supervised learning solution because they were using labelled data to building the appliance models. This is also the opinion of Parson [68]. The REDD dataset has high-frequency and a low-frequency sampling data. We chose to use the low-frequency sampling data which was in apparent power measurements. The whole-house aggregate power meter sampled at 1Hz (per second), opposed to the load sub-meters that sampled a $\frac{1}{3}$Hz (3-second intervals). This meant we needed to clean and convert the data for our tests. For our tests we down sampled the aggregate data from 1Hz to $\frac{1}{3}$Hz by discarding the between readings after matching the timestamps. There was the additional problem where the timestamps in some of the readings were out of sync because two different metering systems were used (one for aggregate and one for loads). To correct mismatches we had our conversion program scan the aggregate data records forward and backward until the aggregate reading matched the sum of all the load readings. There were a few instances where there was negative noise, as defined in (4.8). To correct this, we added the difference to the aggregate reading so that the noise cancelation would read zero.

Table 5.2: *Comparing REDD Accuracy Estimations with Other Published Results*

REDD	Kolter 2011 [55]	Johnson 2013 [60]	Discrete Denoised	Discrete Noisy	Discrete Model	Continuous Noisy	Continuous Model
House 1	46.6%	82.1%	99.5%	95.6%	99.3%	85.7%	89.0%
House 2	50.8%	84.8%	99.7%	94.8%	99.0%	90.2%	89.9%
House 3	33.3%	81.5%	98.4%	90.6%	97.5%	83.9%	85.1%
House 6	55.7%	77.7%	97.5%	98.4%	99.7%	98.1%	98.4%
Average	46.6%	81.5%	98.8%	94.9%	98.9%	89.5%	90.6%
Gain wrt. [55]	—	+34.9%	+52.2%	+48.3%	+52.3%	+42.9%	+44.0%
Gain wrt. [60]	-34.9%	—	+17.3%	+13.3%	+17.3%	+7.9%	+9.1%

We used the same five tests previously described in this chapter. We selected the same five loads to disaggregate as Johnson did: refrigerator, lighting, dishwasher, microwave, and furnace. We compared our results with the other two in Table 5.2. Note the other two papers did not report load specific accuracies, so we cannot compare those. While the Johnson 2013 results were significantly better than the Kolter 2011 results, the results from all 5 μDisagg tests were significantly better than the Johnson 2013 results. Our discrete noisy results were better than Johnson 2013 by 13.3%, and better than Kolter 2011 by 48.3%.

We report the other overall metrics and accuracy measures for House 1 in Table 5.10 (page 91), House 2 in Table 5.11 (page 91), House 3 in Table 5.12 (page 91), House 4 in Table 5.13 (page 92), House 5 in Table 5.14 (page 92), and House 6 in Table 5.15 (page 92). We clearly show that the majority of accuracy measures score high, with a few exceptions. The Discrete Noisy test for House 1, 2, and 3 has FS f-scores below 80%: 78.1%, 77.9%, and 65.0% respectively. The Continuous Noisy test for House 6 had an FS f-score of 72.6%. The four tested (excluding denoised) performed quite well with large amounts of noise. The Discrete Model test on House 1 had the largest amount of super-states at 20,480.

5.7 Testing Real-Time Embedded

Our prototype μDisagg, originally written in Python, was converted and optimized in C. A python script was created that would model data generated from a dataset and convert it to C data structures. As previously discussed in Section 2.5.4, we developed a hardware prototype ammeter that would monitor the home's current draw and then run μDisagg. Our ammeter uses the Arduino Due [2] as the embedded platform for μDisagg to execute on. The Arduino Due has an Arm Cortex-M3 processor which is a 32-bit ARM core microcontroller with a clock speed of 84Mhz. For memory, there is 96kB of SRAM and 512kB of flash memory. Flash memory is used to store the program image (in our case μDisagg). At the time of writing this thesis, the embedded processor (Atmel SAM3X8E) could be purchased for USD$12.00, which contains the essential RAM and flash needed to run μDisagg.

Table 5.3 reports metrics that support our claims in **Contribution 5c**. The **Build Time** column reports the total amount of time (in seconds) it took to build the C data structures. The **Image Size** column reports the binary image size (in kilobytes) of μDisagg. The **%**

[2] see *http://arduino.cc/en/Main/arduinoBoardDue*

of ROM column reports the percentage of flash memory used to store μDisagg on the Arduino Due. The **Burn Time** column reports the total amount of time (minutes:seconds) it took to upload and program the Arduino Due with the μDisagg image. The **Disagg Rate** column reports the elapsed time (in microseconds) for μDisagg to disaggregate one aggregate reading.

Table 5.3: *AMPds Tests from Section 5.5 Used for Running μDisagg on an Arduino Due*

Test Type	Loads	Build Time	Image Size	% of ROM	Burn Time	Disagg Rate
Discrete Denoised	5	23.9s	100.6kB	19%	0:44	10μs
Discrete Noisy	5	23.7s	162.4kB	30%	1:07	20μs
Discrete Model	6	26.5s	794.0kB	—	—	—
Continuous Noisy	5	27.4s	87.0kB	16%	0:39	440μs
Continuous Model	6	29.5s	494.8kB	94%	3:38	>60s

We could not perform the Discrete Model test because the image size was larger than the available flash memory by about 282kB. We could not complete the Continuous Model test because it never returned a result – taking too long to execute. Once the processing time exceeded 1 minute we stopped the test as it made no sense to have the execution time exceed the sampling rate which was per minute. For the other three tests (Discrete Denoised, Discrete Noisy, and Continuous Noisy), we can see μDisagg is able to perform disaggregation on a low cost, commodity, embedded processor in real-time. We also observed the processing time for discrete models is far less than continuous models (22×) but at the cost of requiring more storage (1.9×). When μDisagg was complied and ran on our Mac workstation, all five tests have a rate under 1 millisecond. It is interesting to note that in C, model space was in the 100s of kB, and in Python, model space was usually in the 100s of MB and in some cases 1–2 GB – we anecdotally noted about 2000–3000× more space was required. We concluded modelling noise is not the best option for an embedded environment when a high amount of states is used. In our test, modelling noise produced a load with eight states. The amount of model data added by modelling noise exceeds the limitations of embedded processors. We would recommend constraining the amount of states per load to 4 or 5.

5.8 Analysis and Review

We have shown, through experimentation, all our claims in **Contribution 5** have been met. Zeifman [38] previously identified six key requirements he believed needed to be met in order for load disaggregation to be solved. The six requirements are: feature selection, accuracy, no training, near real-time capabilities, scalability, and various appliance types. We review how well we have met these requirements.

Feature Selection Feature selection constrains the measurements that suitable disaggregators use to those measured by a typical smart meters: voltage, current, frequency, apparent power, and apparent energy. There is also a restriction in sampling frequency as smart meters only report readings at a low-frequency. μDisagg has met this requirement with the ability to use different measurements reported by the smart meter and low-frequency sampling. We have shown that μDisagg can disaggregate accurately at even lower sampling frequencies: per minute and per hour.

Accuracy Accuracy requires suitable disaggregators to have a minimal accuracy measure score of 80%. In Figure 5.1, μDisagg has met this requirement in overall accuracy measure for the majority of tests ran.

No Training No training constrains suitable disaggregators to those that do "not involve significant occupant efforts" [38] to train and accurately disaggregate loads. In the development of μDisagg we have tried to minimize the involvement of occupants. We have restricted the training effort needed to just initial model building without pre-tuning.

Near Real-Time Near real-time capabilities means suitable disaggregators must run online and respond to events as they happen. This means the algorithm must be robust and efficient. μDisagg has met this requirement and can disaggregate in under 1 millisecond.

Scalability Scalability constrains suitable disaggregators to those that do not require additional processing time and/or hardware to account for the identification of new appliances. This is still very much an open research question and μDisagg currently cannot do this. However, it can disaggregate accurately with high levels of noise so as new appliances are added or removed this will have minimal impact on the current

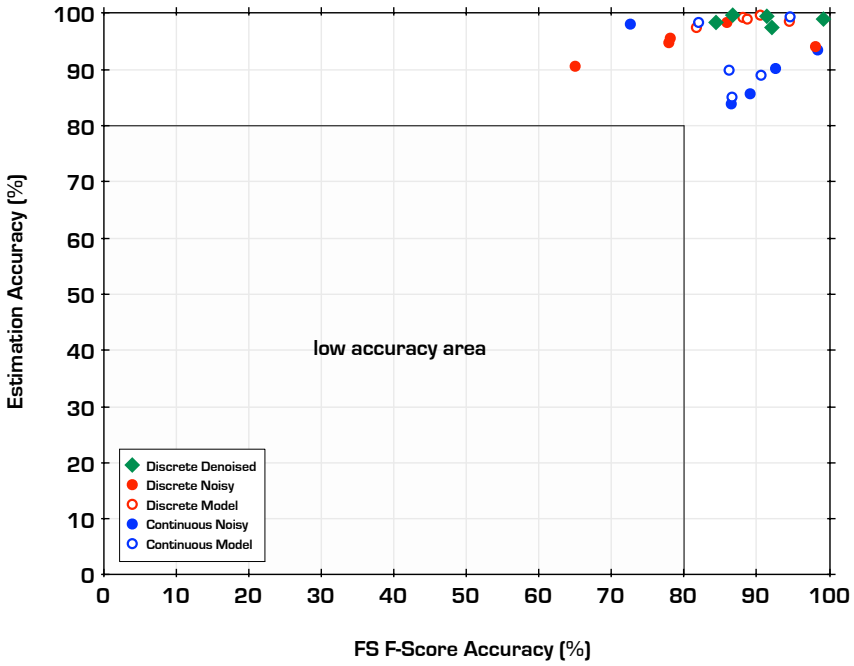


Figure 5.1: *Compare tests scores for FS f-score and estimation accuracies.*

appliances μDisagg is disaggregating (assuming those appliances do not change). One observation is the house used to create AMPds did not have any major deferrable load changes in the 10 years, after a major renovation was done.

Various Appliance Types Various appliance types [3] must be handled/detected by the disaggregators. To simplify things, μDisagg only uses one appliance type, multi-state (or finite state). Simple ON/OFF and constantly on are simply special cases of multi-state. Continuously variable appliances are usually small devices (e.g. Dremel tool with variable speed motor) that do not constitute a deferrable load. For larger appliances such as a front load washer, with variable speed drum, we create a state for this operation. Consumption estimation is based on the peak $\mathbf{y}^{(\mathrm{m})}_{\mathrm{peak}}$ which seems to be sufficient for estimation purposes. However, we do not directly address continuously variable appliances with complex variable signatures.

In our final chapter we conclude our thesis with a discussion on significance, limitations, and future directions.

[3] e.g. Hart's [17] four appliance types: simple on/off, finite state, constantly on, and continuously variable

Table 5.4: *Correlation Coefficient Amongst Loads for Each Test as Discussed in Section 5.4*

(a) Correlation Coefficient for Loads in AMPds

	Dryer	**Dishwasher**	**HVAC**	**Heat Pump**	**Oven**
Dryer	1.00	0.02	-0.01	-0.00	0.00
Dishwasher	0.02	1.00	0.00	-0.01	0.01
HVAC	-0.01	0.00	1.00	0.28	-0.01
Heat Pump	-0.00	-0.01	0.28	1.00	-0.01
Oven	0.00	0.01	-0.01	-0.01	1.00

(b) Correlation Coefficient for Loads in REDD House 1

	Fridge	**Dishwasher**	**Lights**	**Microwave**	**Heater**
Fridge	1.00	0.01	0.02	0.04	0.02
Dishwasher	0.01	1.00	0.03	-0.02	-0.02
Lights	0.02	0.03	1.00	0.12	0.13
Microwave	0.04	-0.02	0.12	1.00	0.73
Heater	0.02	-0.02	0.13	0.73	1.00

(c) Correlation Coefficient for Loads in REDD House 2

	Lights	**Microwave**	**Fridge**	**Dishwasher**
Lights	1.00	0.10	0.04	-0.01
Microwave	0.10	1.00	0.02	-0.01
Fridge	0.04	0.02	1.00	-0.01
Dishwasher	-0.01	-0.01	-0.01	1.00

(d) Correlation Coefficient for Loads in REDD House 3

	Lights	**Fridge**	**Dishwasher**	**Microwave**
Lights	1.00	0.10	0.05	0.06
Fridge	0.10	1.00	0.01	0.05
Dishwasher	0.05	0.01	1.00	-0.00
Microwave	0.06	0.05	-0.00	1.00

(e) Correlation Coefficient for Loads in REDD House 6

	Fridge	**Dishwasher**	**Heater**	**Lights**
Fridge	1.00	-0.21	0.07	-0.01
Dishwasher	-0.21	1.00	0.00	0.02
Heater	0.07	0.00	1.00	0.16
Lights	-0.01	0.02	0.16	1.00

Table 5.5: *Correlation Coefficient for Loads Dependence for All Loads in AMPds as Discussed in Section 5.4*

	B1E	B2E	BME	CDE	CWE	DNE	DWE	EBE	EQE	FGE	FRE	GRE	HPE	HTE	OFE	TVE	UTE	WOE
B1E	1.00	0.08	-0.00	0.02	0.01	0.00	0.05	0.03	0.00	0.01	-0.00	-0.00	-0.01	0.05	0.03	-0.02	0.01	0.00
B2E	0.08	1.00	0.16	0.05	0.05	0.01	0.07	0.16	0.01	0.01	0.01	0.00	-0.00	0.07	0.17	0.11	0.07	-0.00
BME	-0.00	0.16	1.00	0.03	0.01	0.05	0.07	-0.03	0.00	0.01	0.01	-0.00	-0.00	0.08	0.05	0.70	-0.04	0.01
CDE	0.02	0.05	0.03	1.00	0.05	0.02	0.02	0.01	0.00	0.01	-0.01	-0.00	-0.00	0.04	0.03	0.01	-0.01	0.00
CWE	0.01	0.05	0.01	0.05	1.00	0.00	0.02	-0.00	0.00	0.01	-0.01	0.01	-0.01	0.02	0.02	-0.00	-0.01	-0.00
DNE	0.00	0.01	0.05	0.02	0.00	1.00	0.02	-0.10	0.00	0.01	-0.02	-0.00	-0.00	0.01	-0.01	0.09	-0.03	-0.01
DWE	0.05	0.07	0.07	0.02	0.02	0.02	1.00	0.00	0.00	0.01	0.00	-0.00	-0.01	0.07	0.03	0.06	-0.00	0.01
EBE	0.03	0.16	-0.03	0.01	-0.00	-0.10	0.00	1.00	0.00	-0.02	0.13	0.00	0.07	0.02	0.19	-0.27	0.52	-0.01
EQE	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.01	0.00	0.01	-0.00	0.01	0.00	0.01	-0.00
FGE	0.01	0.01	0.01	0.01	0.01	0.01	0.01	-0.02	0.00	1.00	0.00	0.01	-0.02	0.02	-0.00	0.02	0.01	0.02
FRE	-0.00	0.01	0.01	-0.01	-0.01	-0.02	0.00	0.13	0.01	0.00	1.00	-0.00	0.28	0.01	0.06	-0.03	0.12	-0.01
GRE	-0.00	0.00	-0.00	-0.00	0.01	-0.00	-0.00	0.00	0.00	0.01	-0.00	1.00	-0.00	0.01	0.01	-0.00	0.01	0.00
HPE	-0.01	-0.00	-0.00	-0.00	-0.01	-0.00	-0.01	0.07	0.01	-0.02	0.28	-0.00	1.00	-0.00	0.03	-0.02	-0.02	-0.01
HTE	0.05	0.07	0.08	0.04	0.02	0.01	0.07	0.02	-0.00	0.02	0.01	0.01	-0.00	1.00	0.04	0.03	0.01	0.02
OFE	0.03	0.17	0.05	0.03	0.02	-0.01	0.03	0.19	0.01	-0.00	0.06	0.01	0.03	0.04	1.00	-0.06	0.10	0.00
TVE	-0.02	0.11	0.70	0.01	-0.00	0.09	0.06	-0.27	0.00	0.02	-0.03	-0.00	-0.02	0.03	-0.06	1.00	-0.16	0.00
UTE	0.01	0.07	-0.04	-0.01	-0.01	-0.03	-0.00	0.52	0.01	0.01	0.12	0.01	-0.02	0.01	0.10	-0.16	1.00	-0.00
WOE	0.00	-0.00	0.01	0.00	-0.00	-0.01	0.01	-0.01	-0.00	0.02	-0.01	0.00	-0.01	0.02	0.00	0.00	-0.00	1.00

B1E is the North bedroom plugs and lights. **B2E** is the master and South bedroom plugs and lights. **BME** is the basement plugs and lights. **CDE** is the clothes dryer. **CWE** is the clothes washer. **DNE** is the dining room plugs. **DWE** is the dishwasher. **EBE** is the electronics workbench. **EQE** is the security and network equipment. **FGE** is the kitchen fridge. **FRE** is the HVAC fan and thermostat. **GRE** is the detached garage door, plugs, and lights. **HPE** is the heat pump. **HTE** is the instant hot water unit. **OFE** is the home office plugs and lights. **TVE** is the entertainment, TV, DVD, and amplifier. **UTE** is the utility room plug. **WOE** is the kitchen wall oven.

Table 5.6: *A Summary of Experimental Test Results Using AMPds (see Section 5.1 for more details). Data is sampled at per minute intervals using Ampere (A) measurements at dA (0.1) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	4.4	1600	5	52454	0.000528	536	0.0%	99.2%	99.2%	99.2%	99.2%	99.0%
Discrete Noisy	4.4	1600	5	52454	0.000563	188	60.6%	98.0%	98.0%	**98.1%**	98.1%	94.1%
Discrete Model	5.0	12800	6	52454	0.003529	258	53.4%	93.6%	95.8%	93.1%	94.5%	98.6%
Continuous Noisy	4.4	1600	5	52454	0.016491	0	60.6%	**98.4%**	**98.7%**	**98.1%**	**98.4%**	93.5%
Continuous Model	5.0	12800	6	52454	0.115203	0	53.4%	93.5%	97.8%	91.6%	94.6%	**99.4%**

Table 5.7: *Specific Load per Minute Test Results Using AMPds*

(a) Discrete Noisy Test (dA, per minute)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	99.7%	99.8%	99.7%	99.7%	100.0%
Dishwasher	95.3%	15.6%	17.0%	16.3%	73.2%
Heat Pump	97.1%	97.1%	97.1%	97.1%	86.7%
HVAC	98.1%	98.1%	98.1%	98.1%	99.4%
Oven	99.6%	99.6%	99.6%	99.6%	84.5%

(b) Continuous Noisy Test (dA, per minute)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	99.7%	99.7%	99.7%	99.7%	96.8%
Dishwasher	97.1%	7.7%	0.5%	0.9%	54.5%
Heat Pump	97.1%	97.1%	97.1%	97.1%	80.9%
HVAC	98.5%	98.5%	98.5%	98.5%	99.6%
Oven	99.7%	99.7%	99.7%	99.7%	84.1%

(c) Discrete Model Test (dA, per minute)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	99.7%	99.8%	99.7%	99.7%	99.2%
Dishwasher	96.4%	8.9%	3.5%	5.0%	76.2%
Heat Pump	97.1%	97.1%	97.1%	97.1%	88.2%
HVAC	97.8%	97.9%	97.8%	97.8%	99.3%
Oven	99.6%	99.7%	99.6%	99.6%	84.1%
Unmetered	71.2%	50.1%	34.0%	40.5%	88.8%

(d) Continuous Model Test (dA, per minute)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	99.8%	99.8%	99.8%	99.8%	97.6%
Dishwasher	97.2%	12.5%	0.4%	0.7%	51.7%
Heat Pump	97.2%	97.2%	97.2%	97.2%	82.8%
HVAC	98.4%	98.4%	98.4%	98.4%	99.9%
Oven	99.8%	99.8%	99.8%	99.8%	83.3%
Unmetered	68.8%	49.2%	11.7%	18.9%	65.3%

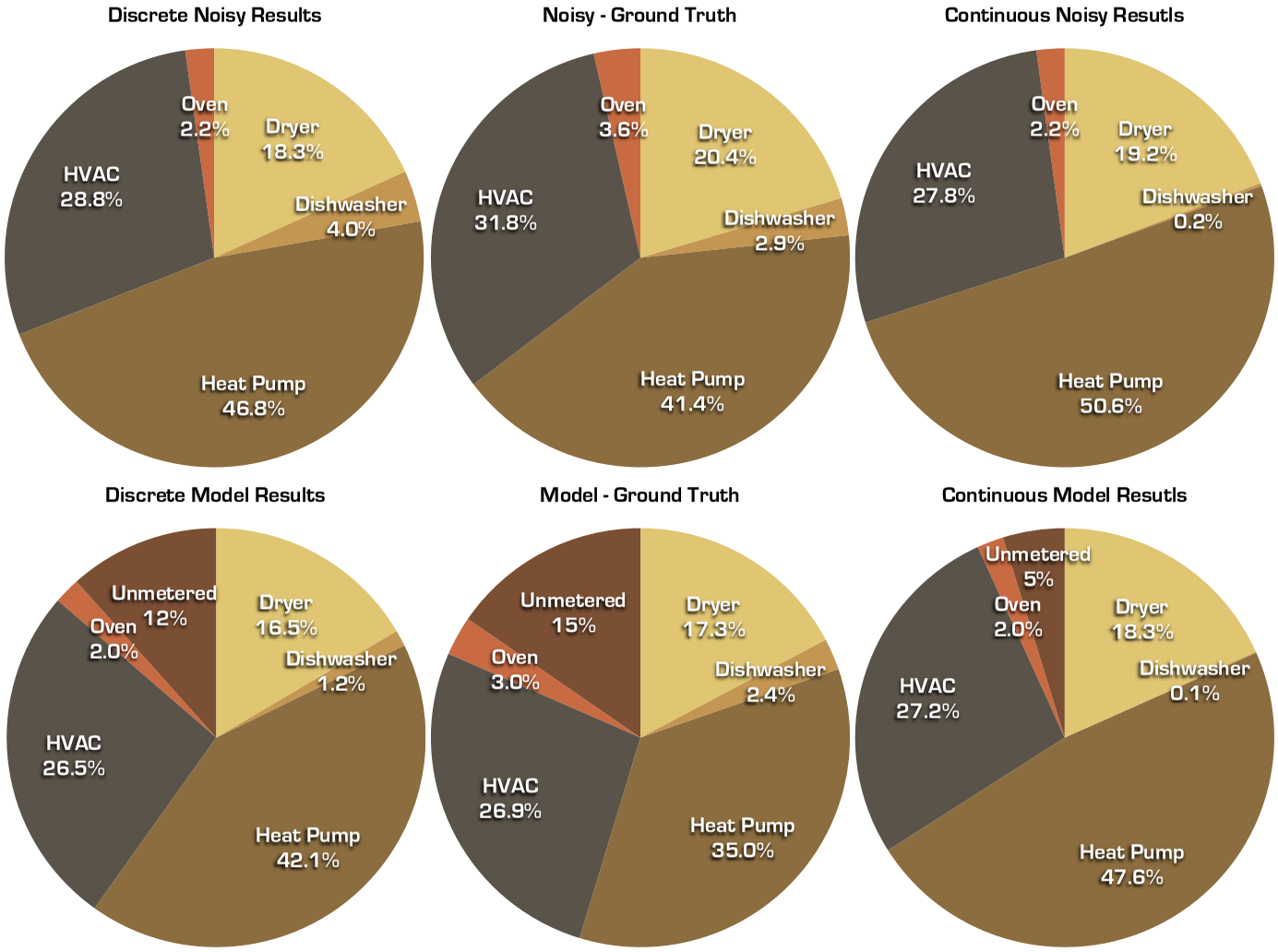


Figure 5.2: *The percentage of overall consumption for each load. The results of the noisy tests and the model tests are compared with their respective ground truths. Percentages are determined from the estimation accuracy calculations.*

Table 5.8: *A Summary of Experimental Test Results Using AMPds (see Section 5.1 for more details). Data is sampled at per hour intervals using Watt-hour (Wh) measurements at daWh (Wh÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	6.2	8400	5	1751	0.001257	159	0.0%	96.7%	97.0%	93.5%	95.2%	93.7%
Discrete Noisy	6.2	8400	5	1751	0.001265	148	68.2%	**94.6%**	**92.0%**	**89.3%**	**90.6%**	86.2%
Discrete Model	6.5	67200	6	1751	0.009232	180	51.4%	91.7%	87.5%	86.4%	87.0%	**88.0%**
Continuous Noisy	5.8	4704	5	1751	0.023841	0	67.4%	84.4%	79.3%	69.1%	73.9%	77.4%
Continuous Model	6.2	37632	6	1751	0.276905	0	50.2%	82.9%	78.6%	74.0%	76.2%	75.5%

Table 5.9: *Specific Load per Hour Test Results Using AMPds*

(a) Discrete Noisy Test (Wh, per hour)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	95.1%	34.1%	21.1%	26.1%	92.0%
Dishwasher	94.1%	12.9%	4.9%	7.1%	70.1%
Heat Pump	87.0%	87.8%	87.7%	87.7%	79.0%
HVAC	98.9%	99.7%	98.9%	99.3%	97.5%
Oven	97.8%	0.0%	0.0%	0.0%	67.9%

(b) Continuous Noisy Test (Wh, per hour)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	70.5%	43.8%	6.9%	11.9%	51.1%
Dishwasher	93.5%	5.0%	1.1%	1.7%	53.4%
Heat Pump	93.6%	93.7%	93.8%	93.8%	72.5%
HVAC	68.1%	68.1%	68.1%	68.1%	99.0%
Oven	96.5%	0.0%	0.0%	0.0%	57.7%

(c) Discrete Model Test (Wh, per hour)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	94.6%	28.6%	22.5%	25.2%	99.8%
Dishwasher	92.4%	10.5%	8.5%	9.4%	92.0%
HPE	88.2%	89.3%	88.9%	89.1%	71.8%
Heat Pump	98.6%	99.7%	98.6%	99.1%	97.5%
Oven	96.9%	0.0%	0.0%	0.0%	87.0%
Unmetered	79.2%	80.0%	79.2%	79.6%	92.9%

(d) Continuous Model Test (Wh, per hour)

Load	Acc	Precision	Recall	F-Score	Est Acc
Dryer	66.4%	30.1%	12.6%	17.7%	51.2%
Dishwasher	91.9%	5.8%	3.2%	4.1%	57.2%
Heat Pump	91.5%	92.6%	91.7%	92.1%	72.0%
HVAC	68.3%	68.3%	68.3%	68.3%	99.6%
Oven	93.8%	1.5%	2.2%	1.8%	65.2%
Unmetered	85.7%	85.9%	85.7%	85.8%	72.1%

Table 5.10: *A Summary of Experimental Test Results Using REDD House 1 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	5.4	2560	5	40674	0.000504	356	0.0%	96.7%	91.3%	91.5%	91.4%	99.5%
Discrete Noisy	5.4	2560	5	40674	0.000528	1620	56.7%	91.9%	75.9%	80.4%	78.1%	95.6%
Discrete Model	5.8	20480	6	40674	0.003415	1911	33.5%	**93.0%**	87.2%	89.0%	88.1%	**99.3%**
Continuous Noisy	5.6	4608	5	40674	0.069061	0	53.7%	91.5%	**88.8%**	**89.4%**	**89.1%**	85.7%
Continuous Model	5.8	32256	6	40674	0.401160	0	29.7%	92.4%	90.4%	90.7%	90.6%	89.0%

Table 5.11: *A Summary of Experimental Test Results Using REDD House 2 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	5.0	540	4	31684	0.000232	362	0.0%	95.0%	85.5%	88.0%	86.7%	99.7%
Discrete Noisy	5.0	540	4	31684	0.000249	147	49.8%	91.6%	75.8%	80.0%	77.9%	94.8%
Discrete Model	5.6	4320	5	31684	0.001421	210	16.5%	93.0%	88.8%	88.7%	88.7%	**99.0%**
Continuous Noisy	6.5	1600	4	31684	0.012959	0	43.3%	**93.3%**	**93.6%**	**91.5%**	**92.6%**	90.2%
Continuous Model	6.6	11200	5	31684	0.097394	0	8.8%	87.8%	86.7%	85.7%	86.2%	89.9%

Table 5.12: *A Summary of Experimental Test Results Using REDD House 3 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	5.8	960	4	37615	0.000199	693	0.0%	94.2%	82.5%	86.4%	84.4%	98.4%
Discrete Noisy	5.8	960	4	37615	0.000216	1168	66.3%	86.3%	62.1%	68.2%	65.0%	90.6%
Discrete Model	5.4	3840	5	37615	0.000859	1235	55.4%	**88.3%**	80.5%	82.9%	81.7%	**97.5%**
Continuous Noisy	7.0	2240	4	37615	0.011424	0	65.0%	81.9%	87.2%	**85.9%**	86.5%	83.9%
Continuous Model	7.2	17920	5	37615	0.088718	0	53.2%	82.6%	**87.5%**	85.8%	**86.6%**	85.1%

Table 5.13: *A Summary of Experimental Test Results Using REDD House 4 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	5.5	30	2	42807	0.000034	413	0.0%	98.5%	87.9%	93.4%	90.6%	98.2%
Discrete Noisy	5.5	30	2	42807	0.000046	6681	91.2%	91.5%	46.0%	52.1%	48.9%	86.3%
Discrete Model	5.3	150	3	42807	0.000098	6706	72.7%	**93.3%**	**90.3%**	**88.5%**	**89.4%**	**96.4%**
Continuous Noisy	5.5	24	2	42807	0.000307	0	91.0%	81.2%	33.4%	18.1%	23.5%	70.8%
Continuous Model	6.3	192	3	42807	0.003020	0	71.5%	86.2%	86.4%	75.1%	80.4%	85.1%

Table 5.14: *A Summary of Experimental Test Results Using REDD House 5 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	5.8	5040	5	7745	0.001110	2254	0.0%	93.8%	88.8%	86.8%	87.8%	91.8%
Discrete Noisy	5.8	5040	5	7745	0.001057	4066	57.3%	90.5%	81.5%	79.9%	80.7%	92.4%
Discrete Model	6.2	40320	6	7745	0.007935	4142	25.9%	**90.6%**	**87.5%**	84.0%	85.7%	87.1%
Continuous Noisy	5.6	4480	5	7745	0.039635	0	55.9%	81.0%	81.7%	78.9%	80.3%	**93.5%**
Continuous Model	6.0	35840	6	7745	0.321872	0	24.0%	86.5%	86.7%	**85.8%**	**86.2%**	82.5%

Table 5.15: *A Summary of Experimental Test Results Using REDD House 6 (see Section 5.1 for more details). Data is sampled at 3 second intervals using apparent power (VA). Discrete tests use measurements at daVA (VA÷10) precision.*

Test Type	$K^{(m)}$	K	M	T	Rate (s)	Unseen	Noise	Acc	Precision	Recall	F-Score	Est Acc
Discrete Denoised	4.0	105	4	19219	0.000075	188	0.0%	93.2%	92.9%	91.4%	92.1%	97.5%
Discrete Noisy	4.0	105	4	19219	0.000088	999	59.2%	88.0%	84.6%	87.1%	85.9%	98.4%
Discrete Model	4.0	420	5	19219	0.000162	1076	44.3%	**90.2%**	**89.8%**	**91.2%**	**90.5%**	**99.7%**
Continuous Noisy	6.0	1008	4	19219	0.011270	0	56.7%	73.1%	71.4%	73.9%	72.6%	98.1%
Continuous Model	6.4	8064	5	19219	0.075146	0	40.9%	81.3%	82.5%	81.4%	82.0%	98.4%

Chapter 6

Conclusions

Our disaggregator was designed to run in real-time on an embedded processor using low-frequency sampling data in an effort to show load disaggregation is indeed a viable method for enabling occupants to understand how their home consumes energy. This understanding would allow occupants to make intelligent, informed decisions on how they conserve the use of energy, which by all accounts is a very personal and dynamic decision making process. Personal and dynamic because the goals of the occupants involve many factors such as: home characteristics, occupant comfort levels, and budgetary constraints (to name a few).

In this thesis, we have presented work that makes contributions to many aspects of disaggregation research. Some, like AMPds (Section 4.4.1) go beyond contributing to disaggregation research; they contribute to other fields of research such as: computational sustainability, energy modelling, smart homes, eco-feedback, and ambient assisted living. Our main contribution is the original work of a new disaggregation algorithm called the sparse Viterbi algorithm. The sparse Viterbi algorithm uses efficient sparse matrix processing on a super-state hidden Markov model with a large number of states. First, we conclude with a synthesis of the varying aspects raised throughout our thesis and discuss their significance (Section 6.1). Second, we discuss the limitations of the work we have presented (Section 6.2), which provides opportunity for our third and final discussion of future directions (Section 6.3).

6.1 Significance

Oft-times, the point of realization and convergence comes from the many investigations and experiments conducted to expand one's understanding. In this thesis, we presented a number of investigations which converged to our final disaggregator design, μDisagg. Each investigation we performed built on the knowledge of the previous investigations. The design of μDisagg was birthed through inclusive examination of these investigations, which included a healthy dose of retrospection.

Inspired by philanthropic and humanitarian means, we discussed social-economic issues around energy conservation. We asked ourselves a number of questions. What if the occupant does not have a comupter or access to the Internet? What if the occupant cannot afford it? Does this mean they do not have a right to take part in energy conservation? Do they not have the right to access information that can help them conserve and save money? As the cost of electricity rises, how would low-income families cope without the tools needed to help conserve energy? While our research is not to answer these questions, these questions set our frame of mind and drive the goals of our research. To us this means our disaggregator needs to be able to disaggregate on an embedded processor and run in real-time in the home. In response, we developed a Consumer Bill of Rights for Energy Conservation (Figure 1.3) and designed a new appliance label standard (Section 2.2 and Figure 2.2) that would motivate the need for disaggregation as a tool for everyone to use.

Power utility companies are in a conflict of interest trying to provide tools that help occupants conserve energy. This conflict stems from the fact that utility companies have an objective (and an obligation to their shareholders) to make money. Providing tools that conserve energy and help occupants save money would mean that utility companies would be making less money – an action contrary to these objectives and obligations. Given that it may be up to the occupants to proactively participate in energy conservation, the tools provided need to be safe and trustworthy. Safety limited the types of measurements a disaggregator could use to Amperes – it is not safe to measure voltage to compute the power measurement. Trust can be achieved by providing an open source solution and for that solution to mitigate privacy concerns. An open source solution incentivizes occupants to take control of their energy conservation goals with the trust of using an open hardware and software platform. This, in effect, limited us to using algorithms that run online in an embedded environment that is computationally constrained by time and space. This

means our disaggregator must use models that are space efficient, yet rich in information. Additionally, our disaggregator must use inference algorithms that are time efficient, yet powerful enough to enumerate over all possible super-states without approximation. Our investigations into measurement fluctuations, load state switching, and metering hardware were proof of the viability of providing an open hardware platform.

We now have an open hardware platform (containing an inexpensive commodity embedded microcontroller) to run our disaggregator – the origin for the name μDisagg. The use of empirical PMFs and a state quantization algorithm means we could determine the states of a load given prior knowledge, we could also constrain the number of states (if desired). Conventional HMMs, where the super-states are enumerations of the combinations of different load states, were easily dismissed for their exponential complexity problems. However, when we investigated sparsity we found the matrices in a conventional HMM were so sparse that simple matrix compression would allow for a very large amount of super-states. The computational space was significantly reduced. By compressing the matrices in column format there was the additional benefit of having an inference algorithm only calculate non-zero probabilities. This significantly reduced the amount of computational time needed. The use of empirical PMFs and a conventional HMM allowed our super-state HMM to be built algorithmically. There is no need for manual configurations/interventions or pre-tuning, unlike the factorial models we have reviewed. In addition to this, unlike factorial models, our model preserves load dependencies and can perform exact inference resulting in better accuracies.

There has been a lack of available testing data, although this has improved recently. By creating our own dataset (AMPds) we were able to control the quality of data, the types of measurements collected, and the sampling frequency (Section 4.4.1). The release of AMPds to the public has given other researchers the means to reproduce our work and verify our findings [3]. AMPds has also contributed to the community, as a whole, by providing our data for other researchers to use in their experimentations. By collecting our own data, on a house which we have intrinsic knowledge of, we gained unprecedented insights into the understanding of the data collected. Additional, we did evaluated μDisagg on another researcher's dataset. These insights guided us in development of μDisagg and aided in the analysis of our experimental results. For instance, having knowledge about the house that is beyond the dataset gave us the intuition about load dependancy.

Measuring accuracy is an area without an accepted standard for disaggregation research even though is very important to prove the effectiveness of a disaggregator. With our insights in data, we proposed a comprehensive guide for reporting how tests were run and what accuracy metrics are the best to use. By doing so, we were better able to perform diagnostics that allows us to identify problems with our disaggregator. For example, we performed a number of revisions on our state quantization algorithm, based on the difference in classification accuracies versus estimation accuracies. The subsequent changes to parts of our algorithm improved our accuracy results to what we have reported in this thesis. Our proposed comprehensive guide also contributes to the disaggregation research community as a whole, providing researchers with an improved way to report experimental setup and results – further contributing to reproducibility.

Our experiments are the result of a culmination of our previous discussions. We were able to setup specific tests with a pragmatic approach. These tests demonstrated how μDisagg could perform with the ultimate goal of energy conservation in mind. Unlike many researchers, we compared our disaggregator results with other published works. We ran tests on the entire dataset, not a hand-picked subset. Our experiments also demonstrated how μDisagg performed on real house data and in an embedded environment. This is important because disaggregators will eventually run in the real world as a product.

The hope of any researcher who devotes their time and energy to the research of load disaggregation does so with the motivation of providing solutions that can aid in energy conservation. However, this point is rather lost when said researchers design algorithms that require the use of Matlab on a workstation. Yes, there is the argument that says we can run intensive algorithms in the *cloud*, but such a proposition begs those occupants, who want to participate in energy conservation, to trust a company with the privacy of their deeply personal and private data. This is not a viable solution for those who do not want to sell their privacy. A holistic solution deserves the trust of its users and that means a device that runs load disaggregation from within the home on an embedded system where the consumption data stays in the home under the ownership of its occupants. Our research provides a number of significant contributions that are applied rather than being theoretical in nature. Our goal has always been to have research contribute to practical solutions that can see load disaggregation being used in a private and secure setting within a device that runs within a home and consumes as little energy as possible. In this thesis, we have designed an algorithm that can do this. This algorithm can run on inexpensive

commodity embedded processors in real-time. This means there is the potential for an inexpensive, open-source product [1] that can run within the home which occupants can trust. To us this is significant.

6.2 Limitations

Although we believe there is significance in our research, there are a some of limitations to our work that do exist. In Section 5.8 we compared μDisagg to Zeifman's [38] six key requirements for disaggregation to be solved. We have confidently met four of them: feature selection, accuracy, near real-time capabilities, and various appliance types. The two we have not confidently met are no training and scalability. The idea of no training may be better rephrased as *minimal setup* which could be handled in two ways: the collection of priors or the active tuning of general load models. In this thesis, we have chosen the collection of priors which provides more accurate results from the start. Active tuning can only provide accurate results once the general models are tuned properly to the specific loads in the house which could take an arbitrary amount of time. How will the occupants know when this active tuning period has completed and that the disaggregation results can now be trusted as accurate?

Zeifman's idea of scalability is very much still an open research question. It is difficult to identify new loads without devoting some computational time to the act of finding them. This also assumes we want to disaggregate every load a house has. Disaggregation is very much a problem that is computationally exponential in both time and space. This means we can mitigate some effects of exponentiality with different strategies, such as our sparsity optimization or the use of factorial models. We need to disaggregate loads that will allow for the end goal of energy conservation and design robust algorithms that handle large amounts of noise (unmetered loads) like μDisagg can. If, for instance, an occupant wants to disaggregate lights, then it may be better to use home automation systems to do this. Not only can they tell the occupant whether the light are ON or OFF, that also gives the occupant remote control over them. New LED lights are being manufactured that self monitor their energy consumption. Automation is a far more advanced method for handling lights. In the future, a disaggregator could use this information to denoise the

[1] which will soon be released, see *http://udisagg.com* and *https://github.com/smakonin/uDisagg*

whole-house reading and provide more accurate disaggregation results.

In Chapter 5 we discussed encountering observations that were not represented in our model due to them not being represented in priors. Using discrete emissions caused legitimate concerns about encountering these unseen observations. When this happens, it causes inaccurate results. If we move to continuous emissions these concerns are alleviated, but at the cost of computational time. All the efficiencies gained from sparsity in an emissions matrix are gone. With continuous emissions, each super-state in the emission vector needs to be processed, whereas with the discrete model only those with non-zero probability were processed. So is model stability worth the sacrifice of increased computational time? If we want to generalize one model over many houses, then yes. However, if we are only interested in a single house and we have enough data, then no. If we have the data we should make use of it.

One of our main goals was to achieve high accuracy scores. We believe disaggregators with active tuning (unsupervised learning) may not work best for occupants. It is our opinion the disaggregator would take too long to learn what loads are in a house. This could cause occupants to lose confidence in the disaggregator's ability to work properly. However, these problems provide direction and motivation for future work. One thing to note, the house used for AMPds has not had any major appliances/loads change since its renovation in 2005 (now nearly 10 years). This would mean only a one-time calibration of μDisagg would be needed.

μDisagg can disaggregate a model with a large amount of super-states. However, there is still the exponential limitation in time and space. We have only pushed back the exponential problem through optimization, but have we pushed it back far enough? No, if you want to try to disaggregate every load in a home. Yes, if you want to disaggregate deferrable loads that affect the amount owing on a power utility bill. We believe the main goal for load disaggregation is to help occupants conserve energy through smarter use of loads that consume a lot of energy. For example, occupants running large consuming deferrable loads when the per kWh is low, instead of when it is high, would see a savings on their power utility bill. Nonetheless, it is still valuable to investigate other ways to mitigate exponential limitations so more loads could be disaggregated.

6.3 Future

Our work provides a solid foundation for continuing research in a number of different areas. One of the high priority items we would like to look at is active tuning, where a general model can be tuned to a specific house. This would address the limitations we described in the previous section. As we discussed earlier, Parson [68] has begun to look at this with the disaggregation of fridges and freezers. We believe there are a number of different approaches that need to be explored. For instance, exploring different ways to perform the distribution curve fitting of multi-state loads. Additionally, active tuning needs to be studied, using a variety of different load types and datasets. Active tuning can also be seen as unsupervised learning, if the general load models were not derived from the datasets used for testing.

The use of state durations is also an area we would like to further investigate. State durations represent the probability that a load will stay in a given state for a given amount of time. This could help improve the classification of loads that operate with similar current or power draws by adding further uniqueness. We initially performed some experiments but were unable to get any useful results. There were a number of super-states that had duration lengths but these duration lengths were not uniform. We attempted to use an explicit duration model by calculating the mean duration of the super-state. This only decreased accuracy testing results by 10%–50%, depending on the load. This may be due to the use of super-states. State duration at the load state level might work best.

The idea of multi-fuel loads needs to be explored. Multi-fuel loads consume more than one type of resource. For example, an instant hot water tank can consume cold water to make hot water, natural gas to heat the water, and electricity to power the onboard system and ignite the gas flame. This type of disaggregation would further help in the classification of loads and may be an alternative to using state durations.

Interfacing with other common household sensors is an area often discussed but rarely investigated. There are two important sensors in the home in addition to the smart meter: the security system, and the thermostat or HVAC Systems. Most homes have these two sensors and they can provide invaluable sensor data that can augment power signal data. We believe having the three sensors working in conjunction can provide for dynamic and adaptive home automation.

Higher-level disaggregation, at a neighbourhood-level, is of interest. Being able to disaggregate one house from another is important to understanding how energy is consumed within a neighbourhood. Benefits, such as predicting energy consumption, can allow forecasting and grid brownout avoidance. Other benefits can be realized such as electricity theft and inefficiencies due to equipment failure without the need of installing a vast number of sensors. It is important to note that forecasting can also be at the home-level allowing occupants to better time when deferrable loads can run avoiding running during peak hours when power costs more, and in some cases that could be automated.

Occupant engagement (including automation) is an area becoming increasingly important to study. Measuring the effectiveness of supplying disaggregation information to occupants and how that information is conveyed needs to be studied. Bartram has done extensive work on occupant-home interaction by investigating and designing a framework for such devices [89]. Bartram has continued to look at issues affecting the ability of homeowners to conserve energy [90, 91]. Her research reiterates the difficulties for systems to communicate to occupants how their house is performing. There is even a question as to what data needs to be communicated, on what medium, and in what form. This could results in the occupants losing interest in using tools to help with conservation efforts [92]. Part of making this technology relevant is the study of how automation can help occupant perform tasks automatically with being too intrusive [11].

6.4 Closure

In spite of the proactive discounting of a model/algorithm because of its theoretical limitations, promising models/algorithms often get overlooked or simply argued away. While the theoretical limitation of exponentiality remains, through the deep analysis of data, modifications can be made to these models and algorithms that allow for use in practice. We have demonstrated this through the design of our super-state HMM and sparse Viterbi algorithm. In addition to this contribution, we have also contributed to the research community, as a whole, through the release of a dataset and the proposal of a unified approach to accuracy reporting. More importantly, if μDisagg was a product installed in our studio suite example (Section 1.2), as with any home, the occupant would be able to understand how her home consumes energy, how she can save money, and how she could help the environment.

Bibliography

[1] S. Makonin, F. Popowich, T. Moon, and B. Gill, "Inspiring Energy Conservation Through Open Source Power Monitoring and In-Home Display," in *2013 IEEE Power and Energy Society General Meeting*, 2013. xiv, xviii, 28, 30, 36

[2] S. Makonin, L. Guzman Flores, R. Gill, R. A. Clapp, L. Bartram, and B. Gill, "A Consumer Bill of Rights for Energy Conservation," in *International Humanitarian Technology Conference (IHTC), 2014 IEEE Canada*, 2014. xviii, 2, 4, 7, 18

[3] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic, "AMPds: A Public Dataset for Load Disaggregation and Eco-Feedback Research," in *Electrical Power and Energy Conference (EPEC), 2013 IEEE*, 2013. xviii, 12, 24, 25, 45, 47, 50, 58, 63, 65, 73, 77, 79, 95

[4] S. Makonin and F. Popowich, "An intelligent agent for determining home occupancy using power monitors and light sensors," in *Toward Useful Services for Elderly and People with Disabilities*, ser. LNCS. Springer Berlin Heidelberg, 2011, vol. 6719, pp. 236–240. xviii, 78

[5] S. Makonin, I. V. Bajic, and F. Popowich, "Efficient Sparse Matrix Processing for Non-intrusive Load Monitoring (NILM)," in *2nd International Workshop on Non-Intrusive Load Monitoring*, 2014. xviii, 12, 13

[6] S. Makonin, P. Pasquier, and L. Bartram, "Elements of Consumption: An abstract visualization of household consumption," in *Smart Graphics*, ser. LNSC. Springer Berlin Heidelberg, 2011, vol. 6815, pp. 194–198. xviii, 36

[7] S. Makonin, W. Sung, R. Dela Cruz, B. Yarrow, B. Gill, F. Popowich, and I. V. Bajic, "Inspiring energy conservation through open source metering hardware and embedded

real-time load disaggregation," in *Asia-Pacific Power and Energy Engineering Conference (APPEEC), 5th IEEE PES*, 2013. xviii, 12, 13, 23, 24, 34

[8] S. Makonin, "Nonintrusive Load Monitoring (NILM): What an algorithm can tell you about your energy consumption," in *Poster Session at IEEE Vancouver Section Annual General Meeting*, 2014. xviii, 8

[9] S. Makonin, M. Kashani, and L. Bartram, "The Affect of Lifestyle Factors on Eco-Visualization Design," in *Computer Graphics International (CGI)*, 2012. xviii, 36

[10] S. Makonin, F. Popowich, and B. Gill, "The Cognitive Power Meter: Looking Beyond the Smart Meter," in *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on*, 2013. xviii, 8, 22

[11] S. Makonin, L. Bartram, and F. Popowich, "A Smarter Smart Home: Case Studies of Ambient Intelligence," *Pervasive Computing, IEEE*, vol. 12, no. 1, pp. 58–66, 2013. xviii, 100

[12] S. Makonin, I. V. Bajic, and F. Popowich, "Efficient Online Load Disaggregation Using a Sparse Viterbi Algorithm," *IEEE Signal Processing Letters*, [in submission]. xviii

[13] S. Makonin and F. Popowich, "Home Occupancy Agent: Occupancy and Sleep Detection," *GSTF Journal on Computing*, vol. 2, no. 1, pp. 182–186, 2012. xviii, 78

[14] S. Makonin and F. Popowich, "Nonintrusive Load Monitoring (NILM) Performance Evaluation: A unified approach for accuracy reporting," *Energy Efficiency*, [in revision]. xviii, 12

[15] H. Kitching, R. Abbott, and S. Hadden, "Requirements for an advanced utility load monitoring system," Electric Power Research Inst., Palo Alto, CA (USA); New England Power Service Co., Westborough, MA (USA); Plexus Research, Inc., Acton, MA (USA), Tech. Rep., 1989. 1

[16] F. Sultanem, "Using appliance signatures for monitoring residential loads at meter panel level," *Power Delivery, IEEE Transactions on*, vol. 6, no. 4, pp. 1380–1385, 1991. 1

[17] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, 1992. 1, 8, 19, 25, 41, 85

[18] K. Ehrhardt-Martinez, K. A. Donnelly, S. Laitner *et al.*, "Advanced metering initiatives and residential feedback programs: a meta-review for household electricity-saving opportunities." American Council for an Energy-Efficient Economy Washington, DC, 2010. 3

[19] S. Darby, "The effectiveness of feedback on energy consumption," *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, vol. 486, 2006. 3

[20] Electric Power Research Institute, "Residential electricity use feedback: A research synthesis and economic framework," *Retrieved October*, vol. 26, pp. 1–126, 2009. 3

[21] J. Granderson, "Preliminary findings from an analysis of building energy information system technologies," *Lawrence Berkeley National Laboratory*, 2009. 3

[22] N. Motegi, M. Piette, S. Kinney, and J. Dewey, "Case studies of energy information systems and related technology: operational practices, costs, and benefits," *Lawrence Berkeley National Laboratory*, 2003. 3

[23] D. Parker, D. Hoak, A. Meier, and R. Brown, "How much energy are we using? potential of residential energy demand feedback devices," *Florida Solar Energy Center, American Council for an Energy Efficient Economy. Ansilomar, Ca.(Sep 28, 2010)*, 2006. 3

[24] P. Chakravarty and A. Gupta, "Impact of energy disaggregation on consumer behavior," in *Behavior, Energy and Climate Change (BECC)*, 2013. 3

[25] K. Carrie Armel, A. Gupta, G. Shrimali, and A. Albert, "Is disaggregation the holy grail of energy efficiency? the case of electricity," *Energy Policy*, vol. 52, pp. 213–234, 2013. 3

[26] A. Gorin and A. Stone, "Recall biases and cognitive errors in retrospective self-reports: A call for momentary assessments," *Handbook of health psychology*, vol. 23, pp. 405–413, 2001. 9

[27] A. A. Stone, S. Shiffman, J. E. Schwartz, J. E. Broderick, and M. R. Hufford, "Patient non-compliance with paper diaries," *Bmj*, vol. 324, no. 7347, pp. 1193–1194, 2002. 9

[28] L. K. Norford and S. B. Leeb, "Non-intrusive electrical load monitoring in commercial buildings based on steady-state and transient load-detection algorithms," *Energy and Buildings*, vol. 24, no. 1, pp. 51 – 64, 1996. 14, 19

[29] J. Liang, S. Ng, G. Kendall, and J. Cheng, "Load signature study part i: Basic concept, structure, and methodology," *Power Delivery, IEEE Transactions on*, vol. 25, no. 2, pp. 551 –560, 2010. 14, 16, 20

[30] C. D. Simpson, *Principles of electronics*. Prentice Hall, 1996. 15

[31] K. Lee, S. Leeb, L. Norford, P. Armstrong, J. Holloway, and S. Shaw, "Estimation of variable-speed-drive power consumption from harmonic content," *Energy Conversion, IEEE Transactions on*, vol. 20, no. 3, pp. 566 – 574, 2005. 16

[32] W. Wichakool, A.-T. Avestruz, R. Cox, and S. Leeb, "Modeling and estimating current harmonics of variable electronic loads," *Power Electronics, IEEE Transactions on*, vol. 24, no. 12, pp. 2803 –2811, 2009. 16

[33] J. Liang, S. Ng, G. Kendall, and J. Cheng, "Load signature study part ii: Disaggregation framework, simulation, and applications," *Power Delivery, IEEE Transactions on*, vol. 25, no. 2, pp. 561 –569, 2010. 16

[34] 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. 17, 21, 38, 60

[35] H. Kim, M. Marwah, M. F. Arlitt, G. Lyon, and J. Han, "Unsupervised disaggregation of low frequency power measurements." in *SDM*, vol. 11. SIAM, 2011, pp. 747–758. 17, 21, 39, 40, 44, 45, 58, 60, 61, 68

[36] 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, no. 0, pp. 55–73, 2012. 17, 20, 38, 60

[37] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds, and S. Patel, "Disaggregated end-use energy sensing for the smart grid," *Pervasive Computing, IEEE*, vol. 10, no. 1, pp. 28 –39, 2011. 17

[38] M. Zeifman, "Disaggregation of home energy display data using probabilistic approach," *Consumer Electronics, IEEE Transactions on*, vol. 58, no. 1, pp. 23–31, 2012. 17, 21, 39, 43, 45, 59, 61, 84, 97

[39] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, and P. Armstrong, "Power signature analysis," *Power and Energy Magazine, IEEE*, vol. 1, no. 2, pp. 56 – 63, 2003. 19, 22

[40] R. Fisera and K. Macek, "Virtual sub-metering via combined classifiers," in *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, vol. 1, 2011, pp. 126 –131. 19

[41] M. Berenguer, M. Giordani, F. Giraud-By, and N. Noury, "Automatic detection of activities of daily living from detecting and classifying electrical events on the residential power line," in *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*. IEEE, 2008, pp. 29–32. 19, 20

[42] 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 *2010 14th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2010, pp. 27–32. 20, 38, 60

[43] S. Lee, G. Lin, W. Jih, and J. Hsu, "Appliance recognition and unattended appliance detection for energy conservation," in *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 20

[44] M. Figueiredo, A. de Almeida, and B. Ribeiro, "An experimental study on electrical signature identification of non-intrusive load monitoring (nilm) systems," *Adaptive and Natural Computing Algorithms*, pp. 31–40, 2011. 20, 38, 60

[45] M. Figueiredo, A. de Almeida, and B. Ribeiro, "Home electrical signal disaggregation for non-intrusive load monitoring (nilm) systems," *Neurocomputing*, vol. 96, no. 0, pp. 66–73, 2012. 20, 38, 60

[46] S. Gupta, M. Reynolds, and S. Patel, "Electrisense: single-point sensing using emi for electrical event detection and classification in the home," in *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM, 2010, pp. 139–148. 20, 38

[47] S. Leeb, S. Shaw, and J. Kirtley, J.L., "Transient event detection in spectral envelope estimates for nonintrusive load monitoring," *Power Delivery, IEEE Transactions on*, vol. 10, no. 3, pp. 1200 –1210, 1995. 20

[48] M. Baranski and J. Voss, "Nonintrusive appliance load monitoring based on an optical sensor," in *Power Tech Conference Proceedings, 2003 IEEE Bologna*, vol. 4, 2003, p. 8 pp. Vol.4. 20, 21

[49] S. Patel, T. Robertson, J. Kientz, M. Reynolds, and G. Abowd, "At the flick of a switch: Detecting and classifying unique electrical events on the residential power line," in *Proceedings of the 9th international conference on Ubiquitous computing*. Springer-Verlag, 2007, pp. 271–288. 20

[50] M. Baranski and J. Voss, "Detecting patterns of appliances from total load data using a dynamic programming approach," in *Data Mining, 2004. ICDM '04. Fourth IEEE International Conference on*, 2004, pp. 327 – 330. 21

[51] M. Baranski and J. Voss, "Genetic algorithm for pattern detection in nialm systems," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, 2004, pp. 3462 – 3468 vol.4. 21

[52] J. Kolter, S. Batra, and A. Ng, "Energy disaggregation via discriminative sparse coding," in *Proc. Neural Information Processing Systems*, 2010. 21, 38, 59, 80

[53] O. Parson, S. Ghosh, M. Weal, and A. Rogers, "Non-intrusive load monitoring using prior models of general appliance types," in *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, 2012. 22, 25, 28, 39, 40, 41, 42, 63

[54] ZigBee Alliance, "Zigbee smart energy profile specification," *Zigbee Doc. 075356r16ZB, rev 16*, 2011. 22

[55] J. Kolter and M. Johnson, "Redd: A public data set for energy disaggregation research," in *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, 2011. 25, 59, 62, 65, 73, 77, 81

[56] M. E. Berges, E. Goldman, H. Matthews, and L. Soibelman, "Learning systems for electric consumption of buildings," in *ASCI International Workshop on Computing in Civil Engineering*, 2009. 38

[57] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman & Hall/CRC, 2009. 38, 39, 53

[58] T. Zia, D. Bruckner, and A. Zaidi, "A hidden markov model based procedure for identifying household electric loads," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, Nov 2011, pp. 3218–3223. 39, 42, 45

[59] J. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," *Journal of Machine Learning Research - Proceedings Track*, vol. 22, pp. 1472–1482, 2012. 39, 40, 41

[60] M. J. Johnson and A. S. Willsky, "Bayesian nonparametric hidden semi-markov models," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 673–701, 2013. 39, 40, 42, 81

[61] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4 –16, 1986. 39

[62] Z. Ghahramani, "An introduction to hidden markov models and bayesian networks," *IJPRAI*, vol. 15, no. 1, pp. 9–42, 2001. 39

[63] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Information Theory*, vol. 13, no. 2, pp. 260–269, 1967. 40, 41

[64] Z. Guo, Z. J. Wang, and A. Kashani, "Home appliance load modeling from aggregated smart meter data," *IEEE Transactions On Power Systems*, [in submission]. 40

[65] Z. Ghahramani and M. Jordan, "Factorial hidden markov models," *Machine learning*, vol. 29, no. 2, pp. 245–273, 1997. 40, 41

[66] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. springer New York, 2006, vol. 1. 40

[67] R. Kohlenberg, T. Phillips, and W. Proctor, "A behavioral analysis of peaking in residential electrical-energy consumers1," *Journal of Applied Behavior Analysis*, vol. 9, no. 1, pp. 13–18, 1976. 41

[68] O. Parson, "Unsupervised Training Methods for Non-intrusive Appliance Load Monitoring from Smart Meter Data," Ph.D. dissertation, University of Southampton, Electronics and Computer Science, 2014. 41, 42, 81, 99

[69] V. Barbu and N. Limnios, "Hidden semi-markov model and estimation," in *Semi-Markov Chains and Hidden Semi-Markov Models toward Applications*, ser. Lecture Notes in Statistics. Springer New York, 2008, vol. 191, pp. 1–48. 42

[70] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 12 1966. 42

[71] M. Zeifman and K. Roth, "Viterbi algorithm with sparse transitions (vast) for non-intrusive load monitoring," in *Computational Intelligence Applications In Smart Grid (CIASG), 2011 IEEE Symposium on*, 2011, pp. 1–8. 43, 45

[72] J. Tou and R. Gonzalez, *Pattern recognition principles*. New York, Addison-Wesley, 1974. 43

[73] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Sparse matrix test problems," *ACM Transactions on Mathematical Software (TOMS)*, vol. 15, no. 1, pp. 1–14, 1989. 50

[74] M. Zeifman and K. Roth, "Nonintrusive appliance load monitoring: Review and outlook," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, pp. 76–84, 2011. 58, 61

[75] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "BLUED: a fully labeled public dataset for Event-Based Non-Intrusive load monitoring research," in *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*, Beijing, China, 2012. 59

[76] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An open data set and tools for enabling research in sustainable homes," in *2012 Workshop on Data Mining Applications in Sustainability (SustKDD 2012)*, 2012. 59

[77] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz, "On the accuracy of appliance identification based on distributed load metering data," in *2nd IFIP Conference on Sustainable Internet and ICT for Sustainability (SustainIT)*, 2012, pp. 1–9. 59

[78] N. Batra, M. Gulati, A. Singh, and M. B. Srivastava, "It's different: Insights into home energy consumption in india," in *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. ACM, 2013, pp. 1–8. 59

[79] J. Kelly and W. Knottenbelt, "'UK-DALE': A dataset recording UK Domestic Appliance-Level Electricity demand and whole-house demand," *ArXiv e-prints*, Apr. 2014. 59

[80] J. Kelly and W. Knottenbelt, "Metadata for Energy Disaggregation," *ArXiv e-prints*, Mar. 2014. 59

[81] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh, and M. Srivastava, "NILMTK: An Open Source Toolkit for Non-intrusive Load Monitoring," in *Fifth International Conference on Future Energy Systems (ACM e-Energy)*, 2014. 59

[82] C. Metz, "Basic principles of roc analysis," in *Seminars in nuclear medicine*, vol. 8, no. 4. Elsevier, 1978, pp. 283–298. 60, 61

[83] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *AI 2006: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, vol. 4304, pp. 1015–1021. 60, 61

[84] H. Liu and H. Motoda, *Feature selection for knowledge discovery and data mining*. Springer, 1998. 68

[85] K. Teknomo, "Recursive average and variance," in *http://people.revoledu.com/kardi/tutorial/RecursiveStatistic/index.html*, 2006. 76

[86] H. Stark and J. W. Woods, *Probability, Statistics, and Random Processes for Engineers*, 4th ed. Pearson, 2012. 76

[87] Explorable.com, "Statistical correlation," in *https://explorable.com/statistical-correlation*, 2009. 78

[88] B. Roberts, "Shaving load peaks from the substation," *POWER Magazine. [Online]. Available: www.powermag.com*, 2006. 79

[89] J. Rodgers and L. Bartram, "Visualizing residential resource use: A framework for design," in *InfoVis*, 2010. 100

[90] L. Bartram and R. Woodbury, "Smart homes or smart occupants? reframing computational design models for the green home," in *2011 AAAI Spring Symposium Series*, 2011. 100

[91] L. Bartram, J. Rodgers, and R. Woodbury, "Smart homes or smart occupants? supporting aware living in the home," *Human-Computer Interaction–INTERACT 2011*, pp. 52–64, 2011. 100

[92] F. Quintal, L. Pereira, and N. Nunes, "A long-term study of energy eco-feedback using non-intrusive load monitoring," *Persuasive Technology*, pp. 49–52, 2012. 100