Delft University of Technology
Master's Thesis in Embedded Systems

Location-aware Energy Disaggregation in Smart Homes

Antonio Reyes Lúa

embedded software

TU Delft Delft University of Technology

Location-aware Energy Disaggregation in Smart Homes

Master's Thesis in Embedded Systems

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Antonio Reyes Lúa
a.reyeslua@student.tudelft.nl

31st August 2015

Author
Antonio Reyes Lúa (a.reyeslua@student.tudelft.nl)
Title
Location-aware Energy Disaggregation in Smart Homes
MSc presentation
31st August 2015

Graduation Committee

Prof. Dr. K. G. Langendoen (Chair)	TU Delft
Dr. Venkatesha Prasad R (Supervisor)	TU Delft
A.U.N. Srirangam Narashiman (Daily Supervisor)	TU Delft
Dr. ir. Alexandru Iosup (External Member)	TU Delft

Abstract

Providing detailed appliance level energy consumption may lead consumers to understand their usage behavior and encourage them to optimize the energy usage. Non-intrusive load monitoring (NILM) or energy disaggregation aims to estimate appliance level energy consumption from aggregate consumption data of households. Hitherto, proposed NILM algorithms are either centralized or require high performance systems to derive appliance level data, owing to the computational complexity associated. This approach raises several issues related to scalability and privacy of consumer's data.

In this thesis, we present the *NILM-Loc Framework* that utilizes occupancy of users to derive accurate appliance level usage information. NILM-Loc framework limits the appliances considered for disaggregation based on the current location of the occupants. Thus, it can provide real-time feedback on appliance level energy consumption and run on an embedded system locally at the household. We propose several accuracy metrics to study the performance of NILM-Loc. To test its robustness, we empirically evaluated it across multiple publicly available datasets. NILM-Loc has significantly higher energy disaggregation accuracy while exponentially reducing the computational complexity. NILM-Loc presents accuracy improvements up to 30% better than other traditional methods. It reaches an accuracy of 89% for the evaluated datasets.

We also detail a case study for the use of the fine-grained appliance-level energy information obtained. We present a load scheduler that minimizes cost and discomfort based on hourly day-ahead pricing. The proposed Demand Response (DR) system ensures user discomfort is minimzed by abstracting patterns from past user behavior and incorporating them to the designed cost-optimal schedule.

"Do not go where the path may lead, go instead where there is no path and leave a trail." – Ralph Waldo Emerson

Preface

This thesis presents the work I have done in the Embedded Software group towards obtaining my master degree in Embedded Systems. Since an early age I have always been interested in measuring things and being able to quantify them to, hopefully, being able to do something with that information. However, it was not always clear what exactly certain data might be useful for. For example, as a child I knew how many steps where required to go from one classroom to the next one, or how many minutes in average it took to go from my house to school depending on the route we took and the exact minute we left home. It was then that I realized the importance of sensors and embedded systems, even though back then I did not named them in that way. I figured out that, in order to do that properly, I needed to have small and convenient "gadgets" that helped me in such tasks. Moreover, I also have always been drawn into the concept of smart homes and the idea of how they can potentially improve our day-to-day life. Therefore, choosing a research project that involved measuring energy consumption with "smart gadgets" in a residential context to obtain relevant insights that can help us improve certain aspects of our lives seemed like a perfect fit.

I would like to thank a number of people for their help. First of all, I wish to express my most sincere gratitude and appreciation to Akshay, my supervisor during this last year, from whom I was able to learn every day and who always provided me invaluable guidance and support. I would also like to thank everyone in the Embedded Software group who directly or indirectly supported me during the duration of this project. Furthermore, I want to let my friends and colleagues know how much I cherish their company and input. Life is a journey and this one would not have been the same without them. Finally, my family. Their everlasting love and never-ending support have made me the person who I am today and continue shaping me into the one I will become tomorrow.

Antonio Reyes Lúa

Delft, The Netherlands
31st August 2015

Contents

Chapter 1

Introduction

Worldwide total energy consumption in residential and commercial buildings is estimated to be 30-40% of generation [2] and is expected to rise due to increased use of appliances and electronic devices. A significant part of this could be reduced with better real-time information on appliance level consumption statistics. With this information, users can be encouraged to change their behavior to save 5-15% of electricity usage [3, 5]. Recently, smart homes have brought the promise of simplifying this process by providing integral solutions that allow households to automatically coordinate with the smart grid to optimize energy consumption. However, the cost of building a smart home is still much higher than the cost of building a traditional one[1]. Moreover, this do not consider the majority of existing households which do not possess any built-in smart capabilities. As a result and with the surge of Internet of Things devices, affordable modular solutions that can be incorporated to existing residential buildings are becoming more popular. These systems bring closer the benefits of smart homes to all those people without such kind of homes.

Several systems are now available for providing feedback on energy usage. However, such systems still lack the ability to provide appliance level consumption feedback and personalized recommendations in real-time to the occupants. One of the most important benefits of appliance level usage information is providing automated personalized recommendations by identifying which appliances could most effectively reduce energy usage in a household. Furthermore, fine-grained information can also be used to identify faulty or malfunctioning appliances that consume more energy than they should. Consequently, occupants know where the energy is being wasted. Several utility companies, or utilities, are now interested in providing appliance level consumption feedback as a service to their customers.

[1] How much does a smart home system cost?: `http://www.smarthome.eu/a/how-much-does-a-smart-home-system-cost.html`

The most common way of obtaining appliance level information is by deploying sensors on individual appliances in the household. Such a deployment is intrusive, cumbersome to maintain and has high installation cost. Alternatively, recent home energy monitoring techniques have utilized *non-intrusive load monitoring* (NILM) algorithms that aim to break down a household's aggregate energy consumption into individual appliances [7]. NILM techniques are gaining popularity due to low cost sensors for measuring energy usage, large-scale smart meter deployments to obtain household's aggregate energy consumption and inference algorithms proposed for energy disaggregation [6, 7, 9].

However, there still exist several challenges preventing NILM techniques to be widely adopted in households: (i) Most of the proposed mechanisms consider only a subset of appliances – a few high energy consuming appliances – for disaggregation. This is due to the exponential computational complexity associated with the number of appliances, hence tractable only for a small number of appliances [10]. (ii) Several appliances with similar energy consumption profiles may exist and, moreover, each appliance may have multiple states. Thus modeling and inferring the states of an appliance accurately is not trivial. (iii) NILM is often performed in a centralized manner with third-party services or utilities having privacy-sensitive information of consumers. Commercially available NILM systems are required to send smart meter data to a cloud service for energy disaggregation. This approach raises several issues related to scalability and privacy. (iv) Lastly, only a few NILM systems manage to provide near real-time energy disaggregation. The ones that do provide, require detailed household and occupants information, and utilize cloud services.

Hitherto, several NILM techniques have been proposed to address some of the above mentioned challenges. For example, Kolter et. al [7], propose variants of factorial hidden markov models (FHMMs) to derive appliance level information for all the appliances in the household. The complexity of the proposed algorithms increases with the number of appliances and exact inference of the appliance state is generally intractable. Hart [6] proposes a combinatorial optimization (CO) approach to find the optimal combination of appliances in different states. However, this approach cannot distinguish between appliances with similar states and has exponential complexity. Other energy disaggregation techniques use additional sensors like acoustic, magnetic and light placed near to the appliances for accurate disaggregation [11]. These mechanisms are intrusive, incur high deployment cost and cumbersome to maintain.

Finally, regardless of the method how the fine-grained appliance level information is obtained, one important issue to resolve is how this information is presented back to the users in a useful way. Ultimately, the idea of provid-

ing feedback to the user about their consumption statistics is to promote better energy behavior from their side. However, simply stating which appliances are the most energy consuming might not be enough to reach that goal. Users need help to understand the data and make sense of what it implies. Analizing this information and organizing it into actionable suggestions that actively promote improvements in energy behavior might be as important as obtaining the fine-grained information. Therefore, in this thesis we present a case study for the use of the fine-grained information obtained after energy disaggregation. The proposed recommendation system is able to inform the occupants on potential savings by deferring usage of an appliance to the time of a day when the electricity price is lesser. To minimize the resulting discomfort associated to the change in their daily routines, the system takes into account the behavioral patterns of the users to suggest cost-saving recommendations that do not abruptly disrupt those patterns. Thus, the proposed recommendations have a higher probability to be followed by the users and, consequently, the ultimate goal of improving energy behavior can be achieved.

Thesis Outline

The thesis is organized as follows. We first introduce the problem and discuss existing related work in Chapter 2. Next, in Chapter 3 we outline some theoretical and practical foundation for understanding the characteristics of the data streams required for energy disaggregation. Then, in Chapter 4 we explain our proposed NILM-Loc framework. This is followed by the description of the evaluation process and the discussion of the experimental results in Chapter 5. A case study for the use of the fine-grained appliance level energy data obtained by NILM-Loc is showcased in Chapter 6. Finally, we conclude and state proposals for future work in Chapter 7. Additional information containing the pseudocodes of the implemented algorithms can be found in the Appendix.

Chapter 2

Problem statement

Energy consumption by lighting, electronics and other consumer appliances has surpassed every other category in the residential sector [45, 46]. This proportion is forecasted to continue growing in the foreseeable future. To be able to meet the required energy needs, energy consumption patterns must be studied so the necessary response actions can be taken. However, this presents a real challenge given that personal energy behavior itself is poorly understood, unmonitored and uncontrolled. Users have no clear idea where their energy is spent; they do not know *when*, *which*, and in many cases even *why* their devices are currently consuming energy. Providing detailed energy consumption information enables consumers to understand their usage behavior and potential ways to optimize their electricity usage [12]. Nonetheless, users have very limited tools to manage their energy use, cost, and comfort [45].

Today, appliance monitoring for energy management solutions allow to obtain fine-grained appliance-specific energy consumption statistics. This can be achieved by deploying smart power outlets on every device that needs to be monitored. However, this imply extra hardware costs and installation complexity which have avoided its widespread adoption by the public [46].

Recently, large scale deployments of smart meters have rekindled the interest among academia and industries towards developing effective non-intrusive load monitoring (NILM) solutions. This approach requires a single sensor that measures the aggregated consumption from the whole household, from which the individual consumption of each appliance is inferred. Thus, NILM can significantly reduce the hardware costs and deployment issues of traditional intrusive monitoring. However, what NILM reduces in hardware requirements it also increases in software complexity. Only one single sensor is required but mathematical algorithms must be devised to separate the load into the different appliances that constitute it. These algorithms can become complex enough not to be able to run them real-time nor in a local

system within the house due to memory and processing constraints [10]. Some commercial products in the market that offer energy disaggregation offload the algorithmic computations to the cloud to overcome these issues, e.g. Bidgely, PlotWatt. In turn, this raises other issues that also need to be addressed. For example, a significant portion of users is wary about possible disadvantages of smart grid technologies, specially showing a lack of trust in the level of privacy [47].

Therefore, to summarize, the question we try to answer in this thesis is if we can *design a non-intrusive load monitoring system with reduced computational complexity that can run on a local embedded system without sacrificing disaggregation accuracy.*

2.1 Related work

Hitherto, several intrusive and non-intrusive methods for monitoring energy consumption in the households have been proposed. Several NILM algorithms have been proposed in literature to derive fine-grained appliance level information. These algorithms rely on various techniques (supervised, semi-supervised or unsupervised) and additional data [13]. We first provide details on the existing algorithms and then describe how our approach enhances the current state-of-the-art NILM algorithms.

NILM Techniques

Unsupervised NILM techniques use no prior knowledge of the appliances but often require appliances to be manually labeled and work on low frequency (i.e., 1 Hz) data. These techniques typically rely on accurate detection and modeling of the state change in the aggregate consumption data [7, 15, 16]. Several variants of factorial hidden markov models (FHMMs) to model the states of the appliances are proposed in [7, 15]. Furthermore, other machine learning approaches such as artificial neural networks (ANNs) and genetic algorithms are also used [16]. These approaches are computationally intensive and exact inference from models with large number of HMMs are intractable.

Supervised NILM techniques assume that ground truth appliance level data is available to train and develop appliance models prior to performing disaggregation. Hart's algorithm identifies step changes in the aggregate electricity consumption and matches them with appliance signature database to learn the states of the appliance [6]. Other approaches employ both real and reactive power measurements for energy disaggregation [18]. These algorithms require extensive training on appliance level data to model the states accurately.

Semi-supervised NILM techniques avoid the need to intrusively install sensors for deriving appliance signatures [9, 20]. Nambi et al. [9] propose a semi-intrusive approach to determine the most optimal number of appliances to be monitored for accurate energy disaggregation. Parson et al. [20] utilize prior models of general appliance types, which are tuned to specific appliance instances using signatures extracted from the aggregate load. In general, due to the complexity involved in training and inference, these algorithms require a high processing power system for energy disaggregation and hence are not suitable for low power embedded systems.

Additional data considered in NILM

NILM algorithms also use different additional information (either energy related or contextual data) to simplify energy disaggregation and enhance its accuracy. Some algorithms rely only on real power consumption of the household [7, 6]. However, other algorithms require both real and reactive power for energy disaggregation [18]. Recent algorithms use information on how loads are distributed across different phases in a household [18, 22] or use transient and harmonic information with very high frequency sampling [23]. However, sampling at high frequency requires expensive hardware and determining appliance distribution across different phases is not trivial.

Algorithms described in [24, 11] employ information provided by other sensors as additional input for energy disaggregation. Rowe et al. [24] propose an event detector to determine the state change by sensing the electromagnetic field (EMF) in the surrounding. Kim et al. [11] utilize ambient signals from inexpensive sensors placed near appliances to estimate power consumption. While the aforementioned approaches improve NILM accuracy, they also require additional deployment and maintenance of these sensors. Moreover, algorithms developed by using these additional data are generally constrained to a particular dataset or a household; consequently, making it nearly impossible to employ the algorithm with other publicly available datasets.

2.2 Our solution

This work presents NILM-Loc framework that utilizes user occupancy information along with aggregated energy data to derive appliance level energy information. We propose a modified combinatorial optimization (CO) algorithm to accurately infer the states of the appliances. The motivation for using location information is threefold. *First*, by utilizing location information of occupants, NILM algorithms can reduce the number of potential appliances considered for energy disaggregation. *Second*, by reducing the

state explosion, the processing power and storage capacity required for disaggregation are also reduced, making NILM algorithms tractable. *Third*, with the large-scale proliferation of smartphones and wearables, it is now possible to monitor location of the occupants (indoor room-level localization) in a non-intrusive and cost-effective manner.

NILM-Loc is able to perform energy disaggregation at the household on a low-cost embedded system such as Raspberry Pi, due to which consumer's privacy-sensitive data is stored and processed locally. This approach further is highly scalable and avoids sharing of privacy-sensitive information to the utilities. Furthermore, the proposed framework can be employed on any dataset containing room-level location information of the occupants. For example, SMART* [29] and iAWE [30] datasets collect occupants room-level location information using PIR sensors. In a similar fashion, our own collected DRED [35] dataset derives room-level location information. However, it obtains it from received signal strength (RSS) data from occupants' smartphones or wearables. We show the efficacy of the proposed NILM-loc framework by evaluating it across several publicly available datasets and our own dataset. To the best of our knowledge (apart from NILMTK [10]), we are one of the firsts to validate and compare NILM algorithms across multiple datasets.

In summary, the main contributions of this thesis are:

1. We propose a novel real-time location aware energy disaggregation framework (NILM-Loc) to derive appliance level information with lesser computational complexity (Chapter 4).

2. We propose several accuracy metrics to determine the efficacy of NILM-Loc both at house level and at appliance level. NILM-Loc was empirically evaluated across several publicly available datasets (Chapter 5).

3. We provide details of a case study for load scheduling in households to minimize cost and discomfort based on hourly day-ahead pricing. Our algorithm employs the disaggregated data to derive appliance usage behavior and provide cost effective schedule for a household (Chapter 6).

Chapter 3

Data modelling

In this chapter we explain the characteristics of the data streams required for energy disaggregation. We first discuss how these data streams are obtained and later describe the process employed to derive the required information from them.

3.1 Occupancy modeling

In general, occupancy information is used to develop efficient energy management systems for smart homes [33]. For example, occupancy information can be used to control the HVAC system efficiently or turn off appliances (lights) when user has left the room. Occupancy data can be obtained using either a *direct* or an *indirect* sensing approach. Direct sensing refers to the method where occupancy data is obtained directly from the sensors without any further postprocessing of such data. For example, low cost sensors such as passive infrared (PIR), reed switches, RFID tags are used to determine room-level occupancy information using this approach [33]. Other systems actively trigger a biometric sensor (e.g., thumbprint/retina scanner) or require cameras in the house. However, these systems are often cumbersome to maintain or perceived as invasive to personal privacy [44].

On the other hand, indirect sensing refers to the method where occupancy data is derived from sensing different parameters other than occupancy. Hence, a postprocessing step is necessary to obtain the desired occupancy information. Indirect sensing methods can therefore make use of existing infrastructure to acquire the data. One of the most common techniques of indirect sensing for determining user location in indoor environments is done by scanning radio frequency signals from WiFi and/or Bluetooth (BT) radios. The intensity of these signals varies depending on the position of the user within the household. Smartphones and wearables enable the collection of received signal strength (RSS) from radio enabled devices. Some advantages of determining indoor location of occupants using these include:

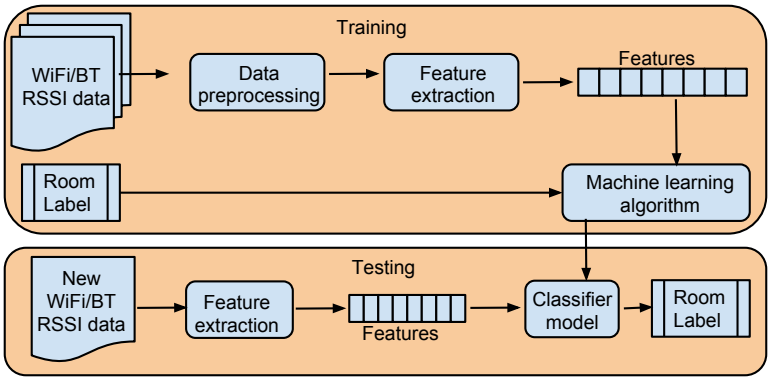

Figure 3.1: Localization.

1. Smartphones, and even more wearables, are personally associated and carried by a user. This enables the possibility of a user-level energy allocation system.

2. Change in sensor information such as accelerometer can be used to detect user movements. This enables the option for activity monitoring which, in turn, could enhance the accuracy of energy disaggregation.

3. Localization techniques can use WiFi and/or Bluetooth radios to identify user location.

Classification techniques such as Bayesian, Support Vector Machines, K-nearest neighbor, decision trees, etc., have been proposed in the literature to derive room-level occupancy using RSS information. Most commonly, these techniques have two main phases: training and testing. Fig. 3.1 shows the process for obtaining user location using a machine learning algorithm. During the training phase, WiFi scans are performed periodically at each location. This phase is also called the fingerprinting stage, where data obtained from the scans is used to learn the available APs and their RSS at different locations. The feature vectors for different time periods at each location are used as training set for classification. The classifier model built on the feature vectors is used for indoor localization. In the evaluation phase, a new feature vector is evaluated with the classifier model to determine the room level location information of an occupant.

The data stream from a new scan includes the list of all visible access points (APs) and their RSS values along with the timestamp information. In case of a WiFi scan, the list of APs indicate the access points from the neighboring houses, whereas the BT scan indicates the Bluetooth beacons available in the house. Currently there exist several Bluetooth enabled devices in a household such as laptops, mobile phones, speakers, etc. Furthermore,

in the near-future most of the household appliances will be Bluetooth enabled [1]. Bluetooth enabled devices can now determine accurately indoor location information of the occupants. This data stream is summarized to obtain a feature vector **l**, which contains mean, maximum, minimum, and standard deviation of the signal strength for each visible access point (AP). It can be represented as,

$$\begin{aligned} l_t =< & rss_t^{max}(1), rss_t^{min}(1), rss_t^{mean}(1), rss_t^{std}(1), ..., \\ & rss_t^{max}(k), rss_t^{min}(k), rss_t^{mean}(k), rss_t^{std}(k) > \end{aligned} \tag{3.1}$$

where k is the number of visible APs at time period t.

Occupancy models often use zoning systems to subdivide a single floor into multiple rooms [44]. In our work, we make room-level divisions in terms of logical function of the appliances; for example, kitchen, living room, basement and bedroom. For the purpose of improving the accuracy of appliance disaggregation, the ideal case would be to have a one-to-one relationship between zones and appliances. In such a scenario, we could have a higher certainty of which appliances are being used at the moment based on the occupancy data. However, in practical terms this is not feasible as it requires determining the user location with an accuracy of under one meter. This is not easily achievable using indirect sensing, and with direct sensing it would imply over-instrumenting the household with many more sensors which, in turn, elevates the installation and maintenance costs. Hence, it is not trivial for energy disaggregation systems to directly use this information. In this work, the occupancy model refers to room-level division of the household.

NILM-Loc can work with systems designed with occupancy models using either direct or indirect sensing approaches. As long as the output of these systems include timestamped room-level location of the users in the household, no other occupancy data is required by NILM-Loc to operate.

3.2 Energy modeling

Load monitoring can be done *intrusively*, which involves placing a sensor on each different appliance in the household. In this work we focus on the alternative *non-intrusive* monitoring, which measures the data from a single point at the power meter of the household. Nowadays, there already exist several smart meters in the market which provide almost real time feedback to the user of the aggregated energy consumption of the household, e.g., Plugwise Smile P1[2]. Non-intrusive load monitoring (NILM) provides a more convenient method for collecting aggregated data as it requires less number of sensors and, consequently, fewer points of failure and lower cost

[1] http://www.bluetooth.com/Pages/Smart-Home-Market.aspx

[2] Plugwise Smile P1: `http://www.plugwise.com/smile-p1/`

for deployment and maintenance. However, an energy disaggregation algorithm is required to obtain the energy information of each appliance from this aggregated data. The goal of an energy disaggregation algorithm is to provide estimates of actual energy consumed by each appliance from the aggregate energy consumption data.

Some algorithms rely only on real power consumption of the household [7, 6]. However, other algorithms require both real and reactive power for energy disaggregation [18]. Recent algorithms use information on how loads are distributed across different phases in a household [18, 22] or use transient and harmonic information with very high frequency sampling [23]. However, sampling at high frequency requires expensive hardware and determining appliance distribution across different phases is not trivial. In this work we aim to give a solution for energy disaggregation that is not only non-intrusive, but one that also does not imply expensive deployment costs for the user. Therefore, we focus on algorithms that can be implemented using lower frequency data (1 sec to 1 min) coming from off-the-shelf sensors. In this section we provide a description of the most widely used energy disaggregation algorithms and explain how each of them uses the data to infer appliance usage information.

3.2.1 Hart's algorithm

The beginnings of energy disaggregation go back to G.W. Hart, who proposed a NILM algorithm in 1992 that is based on changes of active and reactive power of electrical appliances.

The system monitors the total load, checking for certain *signatures* which provide information about the activity of the appliances which constitute the load [6]. For example, if the household contains a refrigerator which consumes 250 W, then a step increase of that characteristic size indicates that the refrigerator is turned ON, and a decrease of that size indicates that it is turned OFF. Other appliances have other characteristic signatures. Fig. 3.2 shows an example of the appliance detection method using Hart's algorithm. Each event detected, i.e., each step in the signal, is matched to the appliance with the most resembling signature.

As it can be deduced, this technique requires an appliance model that describe the signature of each appliance found in the household. To obtain these models, a training phase is carried out in the beginning. During this period, signatures are observed and named as appliances are manually turned ON and OFF.

Hart's algorithm has some drawbacks that hinder its efficacy. First, it relies on accurate event detection to accurately disaggregate the energy. However, aggregate power is usually noisy and distinguishing a step is there-

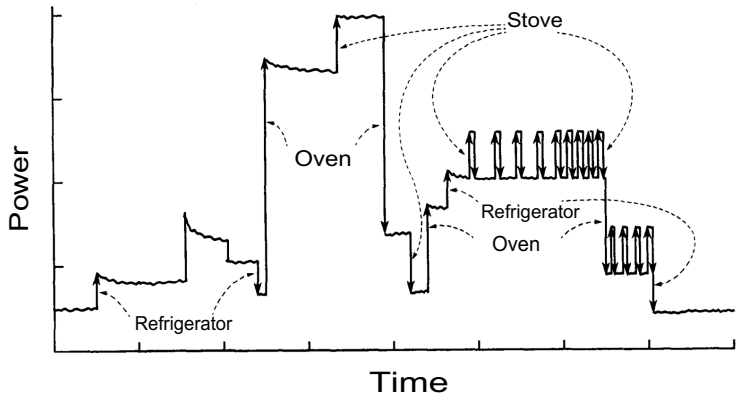

Figure 3.2: Step changes detection in Hart's algorithm [6].

fore troublesome. This results in overlooking transitions coming from low powered appliances whose signature's value is comparable to the magnitude of the noise. Moreover, Hart's algorithm does not consider the scenario where multiple appliances change states during the same time period. Given the low frequency of the data, this situation may arise quite frequently.

3.2.2 Combinatorial Optimization (CO)

We provide a brief description of the CO algorithm for energy disaggregation [6]. Let $\hat{y}_t^{(n)}$ be the estimated energy consumed by each appliance, $\overline{y}_t$ be the aggregate energy reading and the predicted state of the appliance such as ON, OFF, Idle is represented by $\hat{x}_t^{(n)}$. CO finds the optimal combination of appliance states, which minimizes the difference between the sum of predicted appliance power and the observed aggregate power. It is given by,

$$\hat{x}_t^{(n)} = \arg\min_{\hat{x}_t^{(n)}} \left| \overline{y}_t - \sum_{n=1}^{N} \hat{y}_t^{(n)} \right| \tag{3.2}$$

where N is the set of all appliances in the household and t is the current time period. Similar to Hart's algorithm, this approach also requires an appliance model, which includes power consumption details for each state of the appliance. This is used during inference to predict the current state of the appliance. The appliance models are defined during a training phase where data from each monitored appliance is obtained. This data is then analyzed to determine the different power states in which each appliance can operate. This is usually obtained by applying clustering techniques. One common method that is used for this is K-means clustering.

The computational complexity of disaggregation for T time periods is $O(TS^{|N|})$, where S is the number of appliance states and $|N|$ is the car-

dinality of the set of all appliances, i.e. the number of appliances in the household.

CO has several drawbacks. Firstly, this optimization problem resembles to subset sum problem and is NP-complete. Furthermore, the computational complexity in CO increases exponentially with the number of appliances. Secondly, this algorithm does not differentiate between appliances with similar power consumption and appliances with similar states. Third, this algorithm assumes all the appliances in the household are being monitored and assigns some portion of energy to appliances even if they are not currently used, resulting in low disaggregation accuracy.

3.2.3 Factorial Hidden Markov Models (FHMM)

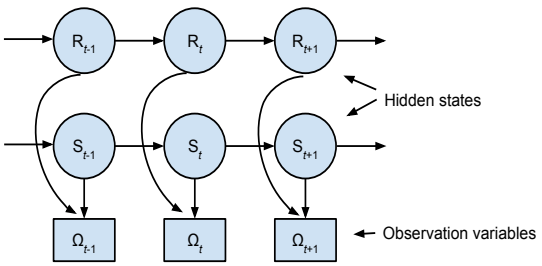

Figure 3.3: FHMM with two underlying Markov chains.

Factorial Hidden Markov Models (FHMM) are an extension of the basic hidden Markov model where several HMMs evolve independently in parallel, and the observed output is some joint function of all the hidden states [7]. Fig. 3.3 shows an example of a FHMM. In a regular HMM, information about the past is conveyed through a single discrete variable. This discrete variable is the hidden state. FHMM allows the state to be factored into mutiple state variable, therefore, FHMMs can be used to represent a combination of multiple independent signals where the characteristics of each one are described by a different Markov chain [8].

As energy disaggregation involves jointly decoding the power draw of n appliances, hence a factorial HMM is well suited for this task [10]. The power demand of each appliance can be modelled as the observed value of HMM. The hidden component of these HMMs are the states of the appliances. A FHMM can be represented by an equivalent HMM in which each state corresponds to a different combination of states of each appliance. Such a FHMM model has three parameters: (i) prior probability (Π) containing S^N entries, (ii) transition matrix containing S^{2N} entries, and (iii) emission matrix containing $2S^N$ entries. The complexity of exact disaggregation for such a model is $O(TS^{2N})$.

As a result of its complexity, the FHMM approach scales even worse than CO. For example, in a scenario with 14 appliances, each one with only two states, computing the transition matrix requires 8 GB of RAM [10]. Consequently, this makes the use of FHMMs for energy disaggregation unsuitable for implementation in resource-constrained embedded systems. Moreover, besides exact inference being not computationally tractable, existing implementations are highly susceptible to local optima [7].

Chapter 4

NILM-Loc

In this chapter we describe our proposed solution for improving energy disaggregation. We present NILM-Loc, a framework to derive accurate appliance information using occupancy information. Fig. 4.1 shows the high level block diagram of location-aware energy disaggregation. As previously detailed in Chapter 3, aggregate energy consumption is obtained from the smart meters deployed at the household. Occupancy information of the residents can be obtained using PIR sensors, RFID tags, smartphone/smartwatch based indoor localization, etc. In our framework, user occupancy information refers to the room or rooms where the occupants are currently located within the household.

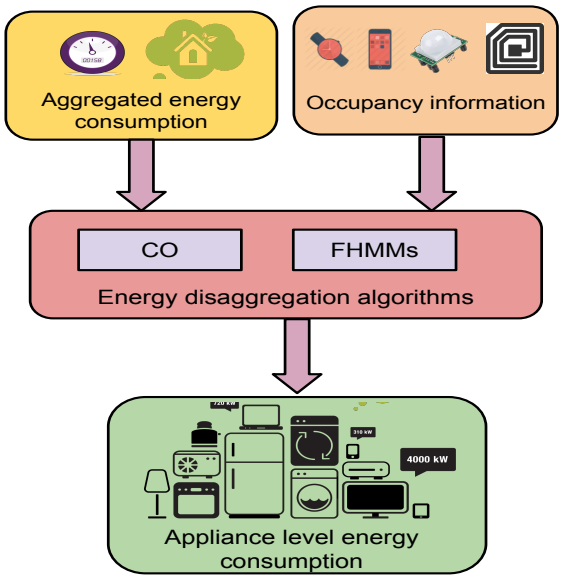

Figure 4.1: Location-aware energy disaggregation.

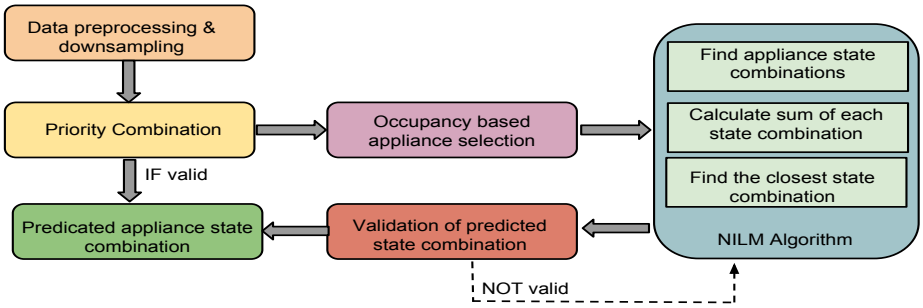

Figure 4.2: An overview of NILM-Loc Framework.

NILM-Loc can be used with any NILM algorithm proposed in the literature. The framework supports use of several NILM algorithms like Hart, FHMMs, ANNs, etc. In this work, we show the benefits of NILM-Loc framework with a simple NILM algorithm like Combinatorial Optimization (CO). We introduced some changes to the original algorithm to further improve its accuracy.

We propose a modified CO algorithm to overcome some of the drawbacks of original CO. Our modified CO algorithm, constrains the number of appliances considered for disaggregation based on the current location of the occupants. This results in exponential reduction in state space for disaggregation. Furthermore, we employ a crowd-sourced generic appliance model from the power consumption database. For example, the power consumption database provides crowd-sourced information on maximum and idle power for a wide range of loads indexed by type, manufacturer, and model number[1]. This information can be obtained *a priori* based on the appliances in the household from the manufacturers datasheet or crowd-sourced data, thus eliminating appliance level energy modeling. However, our modified CO requires to know the number of appliances and their location in the household. This metadata information is collected only once and, except from a few appliances like vacuum cleaner or hair dryer, the location of the appliances is generally static. Fig. 4.2 shows an overview of our proposed NILM-Loc framework and is described in detail next.

4.1 Data preprocessing

NILM-Loc framework includes preprocessing techniques that can simplify the NILM computation and improve energy disaggregation accuracy. Our framework can handle various data sampling rates and is designed to work with several datasets. In general, during data collection there might be gaps

[1]The Power consumption database. [Online] http://www.tpcdb.com/

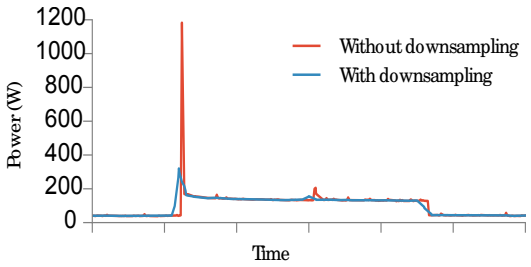

Figure 4.3: Downsampling.

in the data due to sensor malfunction, network connectivity, etc. Hence, it is important to preprocess these gaps either by removing them or by using statistical models such as smoothing, interpolation, forward filling, etc. Furthermore, different datasets include different sampling intervals typically from 1 second to 15 minutes. NILM-Loc applies a downsampling mechanism, to filter transients that occur due to high starting current of an appliance.

In this work, we use filters such as mean and median to downsample the aggregated energy data to 1 minute frequency. Fig. 4.3 shows the comparison of refrigerator energy consumption with and without downsampling. As it can be observed, the downsampled signal significantly reduces the transients at the beginning of the refrigerator's duty cycle. This step allows us to work with cleaner data since extreme points, far from the typical operation values, are greatly filtered out.

Sampling of 1 minute frequency was chosen because it allows having reduced storage requirements without sacrificing too much granularity resolution, maintining the shape of energy loads really close to the original ones, albeit eliminating the mentioned transients (see Fig. 4.4). For example, in the case of REDD, the original space requirement for storing the whole dataset is reduced over 20 times. Moreover, in terms of occupancy data, a 1 minute resolution gives enough information to the algorithm as, most of the times, users spent at least that amount of time in a room when using an appliance. As users do not move from one location to another that fast, a 1 second resolution of occupancy data does not provide more information as compared to 1 minute, however it brings unnecessary computational burden to the system.

4.2 Priority combination

In original CO, at each time period the algorithm tries to find the set of appliances whose combined energy consumption is closest to the current aggregated energy consumption. This may result in a different set of appliances being selected for consecutive time periods. The main reasons are

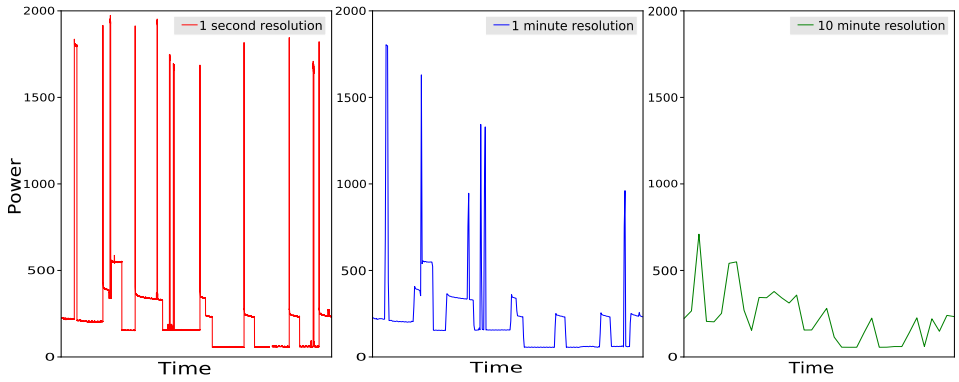

Figure 4.4: Comparison of aggregated energy shape at different sampling frequencies.

(i) the fluctuations in the aggregated value and (ii) multiple combinations of appliances with similar combined energy consumption. Given the sheer number of possible combinations that exists as the number of appliances increases, even with the expected small fluctuations in the aggregated energy consumption, the closest combination to the aggregated value changes at each time period. For example, at time period 't', CO may determine appliance TV and microwave are being currently used, at time period '$t+1$' it may select fan and microwave, and at time period '$t+2$' it may select TV and microwave again. This is due to the fact that TV and fan may have similar energy consumption profiles. In practice, this result would mean TV is switched ON in one minute and switched OFF the next minute and so on. Hence, it is necessary to preserve consistency in selection of appliances during consecutive state estimations.

NILM-Loc introduces the concept of *priority combination*, which is the set of appliances that are assumed to be currently running. This information can be retrieved from the last iteration of NILM-Loc algorithm. At each time period, NILM-Loc first evaluates the priority combination to check whether the sum of all appliances in the priority combination matches the current aggregated value. If the difference between the sum of priority combination and aggregated energy is within a threshold δ, then the current priority combination is retained as the prediction. NILM-Loc evaluates the following equation to determine whether the current priority combination of appliances is still valid or not.

$$|\overline{y}_t - \sum_{n=1}^{K} \hat{y}_t^{(n)}| \leq \delta \tag{4.1}$$

In this equation, K is the set of appliances present in the priority combin-

ation and δ is the variation threshold. The variation threshold parameter ensures small fluctuations in aggregate power ($\overline{y}_t$) has minimal effect. However, when the difference between current priority combination and aggregate consumption is greater than δ, NILM-Loc finds the new set of appliances that are used. The value of δ can be configured to adapt the algorithm to these fluctuations in the aggregate power. This value can be either fixed or dynamic. A dynamic value of δ allows the algorithm to vary the threshold as the magnitude of the aggregated value changes. This is due to the fact that the noise in the aggregate power increases as its value grows.

4.3 Occupancy-based appliance selection

If the current priority combination does not match the aggregate energy consumption, NILM-Loc then estimates the set of appliances that could be currently in operation. This stage identifies the set of appliances which are considered valid. In general, the appliances considered for evaluation at a particular time period include,

1. **Appliances present in the current location of the occupants.** For example, if the current location information indicates that there are occupants in both Kitchen and Living room, only appliances present in these locations are contemplated during that time period for energy disaggregation.
2. **Appliances that are already "ON".** This is the *priority combination*, i.e. the appliances selected in the last iteration of the algorithm.
3. **Appliances that are autonomous.** These are appliances which can change operation state at any given time without user intervention, e.g. refrigerator.
4. **Appliances that can be remotely controlled.** For example, lights and other smart appliances.

We refer to these appliances as our *constrained set of appliances*. NILM-Loc uses this constrained set for energy disaggregation rather than the complete set of appliances present in the household. If for a time period, there is no occupancy information available, all appliances present in the household are considered for evaluation. This may occur due to malfunction of PIR/RFID sensors, power outages, or communication network issues.

4.4 NILM algorithm

This stage finds the optimal combination of appliance states given the *constrained set*. As mentioned earlier, NILM-Loc can be used with other NILM

algorithms proposed in the literature such as FHMM. In this work, we employ a modified CO algorithm as the NILM algorithm for disaggregation.

In constrast to the original CO algorithm, we calculate the sum of all possible state combinations only from the appliances in the *constrained set* rather than from the whole set of appliances in the household. After that, we select the closest combination of appliances that match the aggregated energy consumption. The computational complexity of disaggregation for T time periods in NILM-Loc is then $O(TS^{|N_c|})$, where S is the number of appliance states, $|N_c|$ is the cardinality of the constrained set of appliances. It is worth noticing that $N_c \subseteq N$, meaning that $|N_c| \leq |N|$. This ensures that for any given iteration, in the worst case, the complexity of the modified CO algorithm using NILM-Loc is at most the same as the one from the original CO algorithm. This case would occur only when $N_c = N$, however, in practice $|N_c| \ll |N|$. This reduced computational complexity enables NILM-Loc to determine the state of appliances in real-time.

4.5 Validation

The final stage of the NILM-Loc framework validates the set of appliances predicted in the previous stage. Despite having already constrained the set of valid appliances in the occupancy-based selection stage, an extra validation step is required. Occupancy-based appliance selection helps NILM-Loc ensure not to turn "ON" an appliance when user is not present in that location. On the other hand, validation stage ensures not to turn "OFF" an already "ON" appliance when the appliance location is not included in the current user location. However, this depends on the type of the appliance. In this work, we broadly classify the set of appliances into:

1. **User dependent appliances.** Appliances that require user interaction to turn "OFF", e.g., TV, fan, etc.

2. **User independent appliances** Appliances that can turn "OFF" themselves and require no user interaction, e.g., microwave, washing machine, dishwasher, etc.

If the set of appliances selected in the previous stage implies one or more user dependent appliances is being turned "OFF", even though that appliance's location is not included in the user locations at that moment, then validation stage eliminates this combination of appliances. NILM-Loc then selects the second closest combination from the previous stage and re-validates.

Chapter 5

Evaluation and results

5.1 Evaluation

In this section, we first discuss the datasets considered to evaluate the proposed NILM-Loc framework. We review the characteristics of each dataset, examine the differences they present and explain how these differences were dealt with. Finally, we describe the set of accuracy metrics used to study the efficacy of NILM-Loc across several datasets.

5.1.1 Datasets

It is important to compare and evaluate energy management algorithms across datasets from different countries due to change in usage behavior of appliances. The Reference Energy Disaggregation Dataset (REDD) was the first publicly available dataset to test NILM algorithms [32]. This was followed by other datasets such as SMART* [29], iAWE [30], ECO [31], BLUED [36], AMPds [37], UK-Dale [38] and Pecan Street [39].

To validate our work, we provide performance evaluation results of the proposed framework across multiple datasets. NILM-Loc imports data from popular publicly available datasets such as REDD (House no.1), Smart* and iAWE. In addition to these, we also make use of our own collected DRED (Dutch Residential Energy Dataset) [1]. Hence, we show performance results for four different datasets collected in different countries. To the best of our knowledge, apart from NILMTK [10], we are one of the firsts to validate and compare NILM algorithms across multiple datasets.

Datasets characteristics

Each dataset includes data from different set of appliances and for varying time duration. In order to evaluate the performance of NILM-Loc across

[1] DRED Dataset. [Online] http://www.st.ewi.tudelft.nl/∼akshay/dred/

Datasets	No. days	No. appliances	No. rooms	Occupancy data
iAWE	61	10	7	yes
SMART	93	25	6	yes
DRED	65	12	5	no
REDD	37	15	N/A	no

Table 5.1: Dataset statistics considered.

multiple datasets, it is necessary to understand the characteristics of each dataset. Table 5.1 shows the different datasets considered and the data used for our evaluation.

Energy submetered: Fig. 5.1 shows the percentage of total energy measured at the appliance level for all days in the dataset. Most of the datasets do not monitor all the appliances in the household, leading to large amounts of unaccounted energy in the aggregated consumption data. In most datasets, these amounts account up to more than 50% of the total energy. Furthermore, the variation of this unaccounted energy data significantly reduces the accuracy of disaggregation algorithms. This decrease in accuracy comes from the fact that all energy from the aggregated value is allocated only to the known appliances even though, in reality, this energy consumption might be coming from unmonitored ones.

DRED has around 75% of energy submetered and all other datasets have around 45% of energy measured at the appliance level. It is clear from the figure that DRED captures significant proportion of the energy consumed in the household, whereas, iAWE dataset has the lowest percentage of energy captured at the appliance level. The importance of this becomes more evident in Section 5.2 where DRED, the dataset with highest energy submetered, is the one that presents better results.

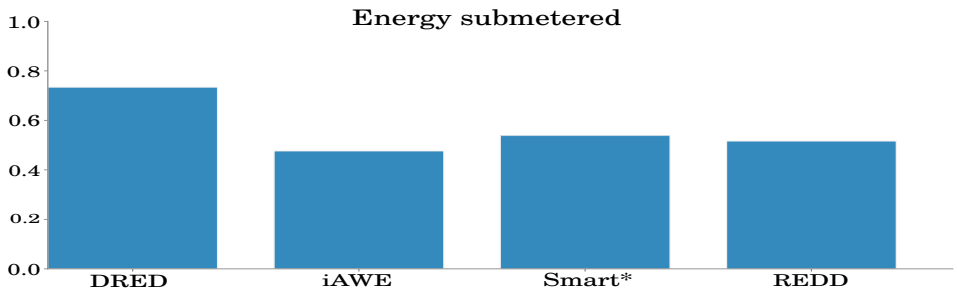

Figure 5.1: Energy submetered in each dataset.

Aggregated available data: Another important statistic to be considered is the percentage of aggregated data available in the dataset. This is the ratio of the number of data points recorded over the total number of data points that can be collected in a day. Fig. 5.2 shows the histogram of average aggregated data available throughout each dataset. The y-axis indicates the percentage of days and x-axis indicates the rate of data availability. In DRED, all of the days have at least 90% of aggregated data available (Fig. 5.2 (1)). However, this availability rate reduces for the rest of the datasets due to communication issues or malfunctioning of sensors deployed. For example, for iAWE, Smart* and REDD it drops to 78%, 59% and 32% respectively.

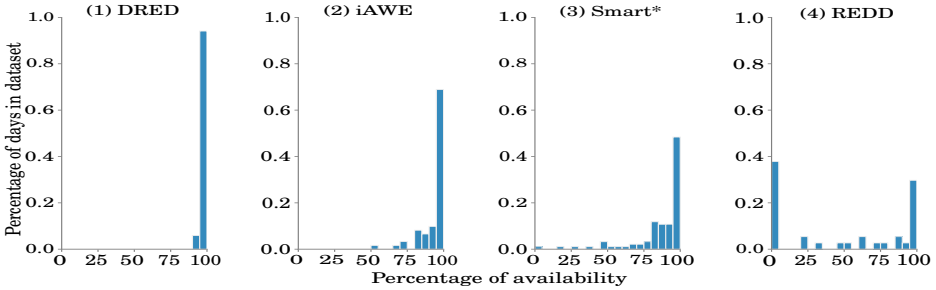

Figure 5.2: Percentage of daily aggregated data available in each dataset.

Top-k appliances: In general, only a few appliances constitute the majority of power consumed in a household. Therefore, it is important to get more accurate information for these top consuming appliances during energy disaggregation. Fig. 5.3 shows the proportion of energy consumed by top-5 appliances and the rest of appliances present in the household in each dataset. It is interesting to see the variation of top-5 appliances across datasets, indicating the varying preference of appliance usage in different countries. The top-5 appliances in DRED cover around 60% of total energy consumed while for the other datasets they represent around the 50%.

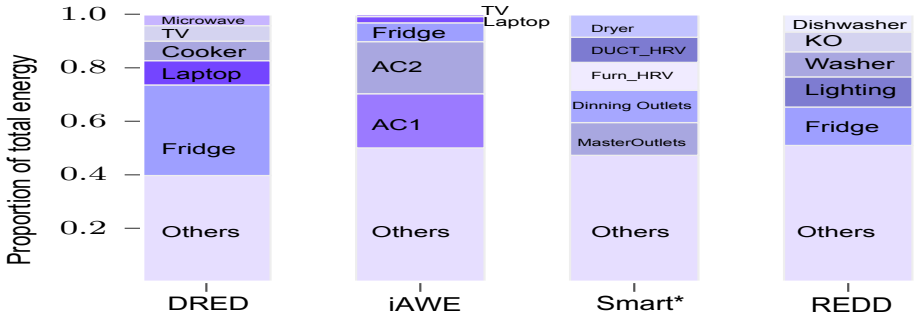

Figure 5.3: Proportion of energy consumed by top-k appliances.

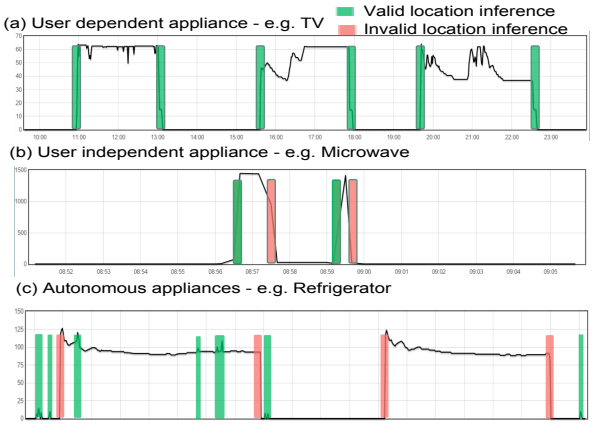

Figure 5.4: Location inference from REDD dataset.

Occupancy data: Finally, since NILM-Loc relies on the occupancy information collected, it is important to find the occupancy data availability rate. The occupancy availability rate represents the ratio of total number of occupancy data recorded over the total number of expected occupancy data. iAWE and Smart* has occupancy rate of 76% and 36% respectively. We further determine the relevant occupancy information that corresponds to the amount of occupancy data available whenever there is an actual energy consumption event. For example, if an appliance currently being used is in living room and the occupancy data includes living room as one of the occupant's location, then, this occupancy data is considered as relevant. iAWE has 53% of valid occupancy information, whereas, Smart* has only about 10%.

Location inference: As observed in Table 5.1, REDD does not originally include any occupancy data. DRED does include occupancy data, however, the data was not yet available at the time the experiments were executed. Therefore, to enable fair comparison across popular datasets, we infer user location with the help of appliance level data. As defined in Section 4.5, NILM-Loc differentiates between user dependent and user independent appliances to accurately infer occupancy information. Fig. 5.4 shows the locations inferred based on the appliance energy consumption information. For a user dependent appliance, a user is present in that location when this appliance is being turned ON or OFF (see Fig. 5.4 (a)). Similarly, for a user independent appliance, a user is present during the ON event but may or may not be present during the OFF event. Hence, as we cannot determine with full certainty the user location at that moment, we label this as an invalid location as shown in Fig. 5.4 (b).

Furthermore, special consideration needs to be given for appliances such as the refrigerator, where occupancy information is valid only when a user

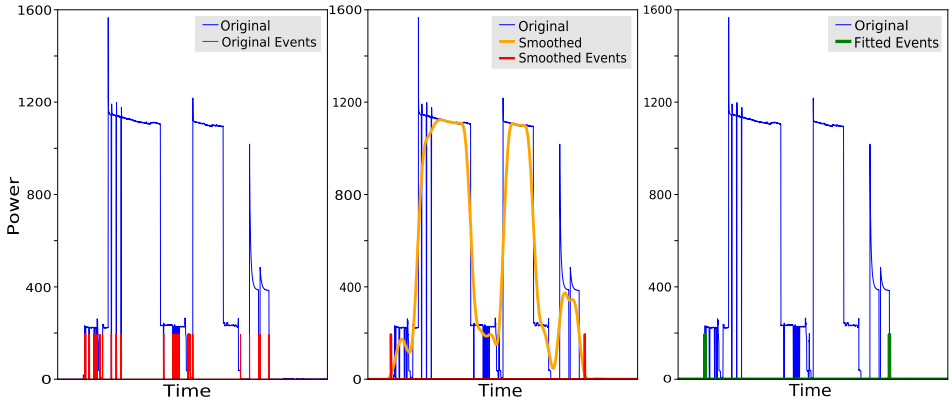

Figure 5.5: Location inference on dishwasher usage with and without smoothing.

either opens or closes the door and not necessarily when it is consuming the most energy which is when the compressor is working. Consequently, we eliminate the compressor energy consumption and infer locations only when the refrigerator door is opened or closed. These events can be observed in Fig.5.4 (c) as small spikes in the energy load which are related to the light bulb inside the refrigerator turning ON and OFF. However, we need to distinguish between these kind of spikes and the initial spikes at the beginning of every compressor cycle. These energy surges are still present, albeit smaller, even after downsampling the signal. Therefore, these spikes due to the compressor are filtered out before determining any user events. In some cases, it may coincide that a spike due to a user event occurs at the same moment as the spike due to the compressor's duty cycle. Given that the magnitude of the latter is much larger, some user events may be filtered out in those cases.

Another challenge in inferring user location from appliance level data is the issue with multi-state appliances. These appliances have more than two states such as, ON, Idle, intermediate state and OFF state, e.g. dishwasher. These appliances can vary its consumption significantly as shown in Fig. 5.5. In a single usage cycle, the energy profile of the appliance may go several times to intermediate states with 0W power consumption. This circumstance, if not handled adequately, would incorrectly result in multiple events being inferred when, in reality, they all correspond to one single event (see leftmost plot of Fig. 5.5). In order to take care of this, our location inference module, upon determining the multi-state appliance, tries to smoothen the energy data of the appliance to remove intermediate states with 0 power consumption. These smoothing techniques converts the highly varying energy signal across multiple states to only one ON and one OFF event as

observed in the middle plot of Fig. 5.5. However, these two events from the smoothed signal are not alligned in time to the corresponding events from the original signal (they may be shifted several minutes). Hence, an allignment step is required to match the point in time in which these events really occurred (see rightmost plot of Fig. 5.5). Finally, a distinction between user dependent and user independent appliances is necessary to distinguish if the OFF event is considered or not. In the case of the dishwasher, as it is a user independent appliance, only the ON event is taken into account for inferring user location.

Please note that location information only when an appliance is being used will be available with the above mentioned inference procedure.

5.1.2 Metrics

Several accuracy metrics both at house level and at appliance level are considered for evaluation of NILM-Loc.

House level metrics

House-level metrics characterize the energy disaggregation accuracy of all the appliances in the household, or a subset of them, as a group. Different metrics at house level are described below:

Fraction of total energy assigned correctly (FTE): It measures the fraction of energy correctly assigned to an appliance and is one of the common accuracy metrics for NILM algorithms [32, 10]. FTE is the overlap between the actual fraction of energy consumed by each appliance and the fraction of energy assigned to each appliance.

$$FTE = \sum_{n} min\left(\frac{\sum_{n} y_t^{(n)}}{\sum_{n,t} y_t^{(n)}}, \frac{\sum_{n} \hat{y}_t^{(n)}}{\sum_{n,t} \hat{y}_t^{(n)}}\right) \tag{5.1}$$

where $n \in \{1, .., N\}$ and N is the total number of appliances. Also $t \in \{1, .., T\}$ and T is the total time period considered.

Total disaggregation error (T_e): Total disaggregation error is the difference between the total assigned energy for all appliances and the actual energy consumed by the appliances, normalized by the total energy consumed. It directly measures how well the disaggregation algorithm recovered the total outputs of all appliances.

$$T_{error} = \frac{\sum_{n,t} |y_t^{(n)} - \hat{y}_t^{(n)}|}{\sum_{n,t} y_t^{(n)}} \tag{5.2}$$

Number of appliances identified correctly (J_a): Jaccard similarity coefficient is used to measure the similarity between the predicted set of appliances (J_a^p) and the actual set of appliances (J_a^a) used over a time period. J_a measures the percentage of appliances correctly identified by the disaggregation algorithm.

$$J_{apps} = \frac{|J_{apps}^p \cap J_{apps}^a|}{|J_{apps}^p \cup J_{apps}^a|} \tag{5.3}$$

Number of appliance states identified correctly (J_s): It measures the similarity between the predicted set of appliance states (J_s^p) and the actual set of appliance states (J_s^a). It indicates the percentage of appliances' states correctly identified by the disaggregation algorithm.

$$J_{states} = \frac{|J_{states}^p \cap J_{states}^a|}{|J_{states}^p \cup J_{states}^a|} \tag{5.4}$$

Appliance level metrics

Appliance level metrics indicate the disaggregation accuracy per each appliance in the dataset. The appliance level metrics considered are:

Proportion error per appliance (P_e): It measures the difference between the proportion of the energy assigned to an appliance and the actual energy consumed by the same appliance.

$$P_{error} = \left| \sum_t y_t^{(n)} - \sum_t \hat{y}_t^{(n)} \right| \tag{5.5}$$

Normalized error per appliance (N_e): It measures the sum of the differences between the assigned energy and the actual energy consumed by the appliance, normalized by the total energy consumed by the appliance. It measures how well the algorithm disaggregated the individual appliance.

$$N_{error} = \frac{\sum_t |y_t^{(n)} - \hat{y}_t^{(n)}|}{\sum_t y_t^{(n)}} \tag{5.6}$$

F_1 score per appliance (F_1): F_1 score is a measure of test's accuracy and is obtained by calculating the harmonic mean of precision and recall. F_1 score measures the percentage of energy correctly assigned to each appliance in the dataset. It can be interpreted as the weighted average of the precision and recall. A higher F_1 score value indicates a better result.

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.7}$$

5.2 Results

We compare NILM-Loc disaggregation results across the four datasets described in Section 5.1.1. We show the performance of NILM-Loc and the original CO algorithm. Furthermore, to ensure fair comparison, both NILM-Loc and CO utilize the same appliance models required for implementing combinatorial optimization as described in Section 3.2.2, i.e., they both have the same input.

We considered one week of data (the week with highest data availability rate) across all the four datasets. In general, FTE, J_a and J_s can vary between 0 and 1, and T_e can take any non-negative value. As we will further explain below, it can be seen that NILM-Loc performs significantly better across all the datasets for all the metrics. NILM-Loc performs better than CO mainly due to two reasons,

1. NILM-Loc ensures that the predicted set of appliances does not vary significantly for consecutive time periods, thanks to *priority combination* as discussed in Section 4.2. Original CO is sensitive to small changes in aggregated power.

2. NILM-Loc constrains the number of appliances considered to disaggregate based on occupancy information, ensuring similar appliances from a different location are not selected.

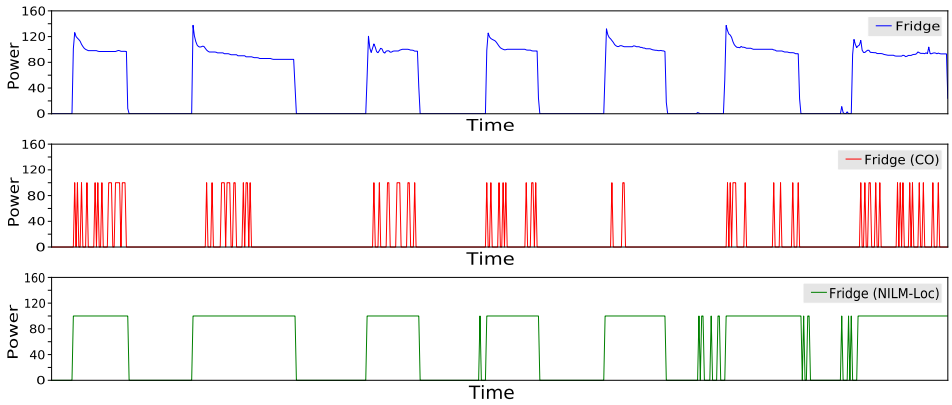

Figure 5.6: Effect of using priority combination during disaggregation.

Fig. 5.6 shows the original load produced by the refrigerator (top) and the resulting disaggregation output using the original CO (middle) and NILM-Loc (bottom). It can be clearly observed the positive effect of using priority combination during energy disaggregation. The resulting output from CO shows an interrupted load, while the output from NILM-Loc is more consistent to the original load from the appliance.

5.2.1 House level results

Fraction of total energy assigned correctly (FTE)

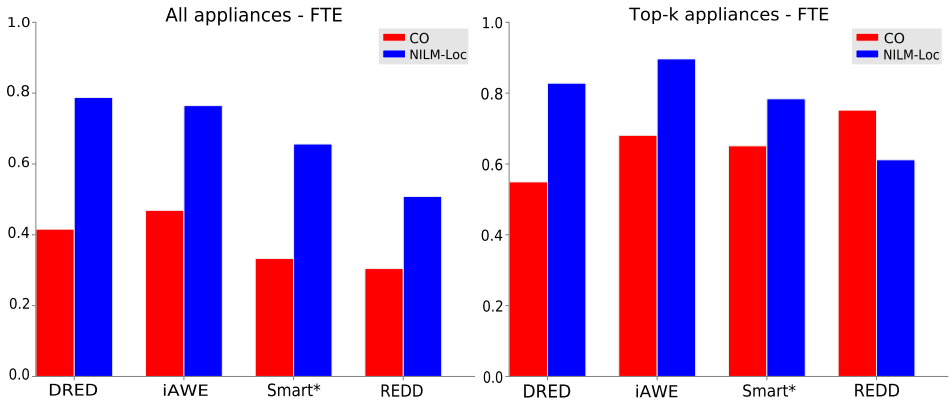

Figure 5.7: Disaggregation performance of FTE in CO and NILM-Loc across datasets (1 week).

Fig. 5.7 shows the fraction of total energy correctly assigned across all datasets. It displays the performance result for all appliances in the household (left) and for only the top 5 most consuming appliances (right). Considering all appliances in the household, NILM-Loc obtains better results than CO. In DRED it improves from 41% to 78%. Similarly, in iAWE it goes from 46% to 76%. In Smart* it is enhanced from the original 33% in CO to 65% in NILM-Loc. Finally, in REDD it improves from 30% to 50%. In the case of the top 5 appliances NILM-Loc also outperforms CO in three out of the 4 datasets. In DRED, iAWE and Smart* this metric goes from 54%, 68%, 65% to 61%, 89%, 78% respectively. In REDD, FTE considering the top 5 appliances performs better in CO than in NILM-Loc, 75% to 61%. This is likely due to the wrong inference of locations from appliance ground truth data in a top 5 appliance.

Number of appliances identified correctly (J_a)

Fig. 5.8 shows the result of the Jaccard similarity index between the appliances consuming energy in the ground truth and the predicted output of CO and NILM-Loc. The higher the similarity index the better the result is as it means the output predicted by the algorithm has higher resemblance to the actual load. Considering all appliances in the household (left), NILM-Loc obtains better results than CO. In DRED it improves from 14% to 49%. Similarly, in iAWE it goes from 21% to 32%. In Smart* it is enhanced from

the original 16% in CO to 54% in NILM-Loc. Finally, in REDD it improves from 12% to 25%. In the case of the top 5 appliances NILM-Loc also outperforms CO. In DRED, iAWE, Smart* and REDD J_a, goes from 30%, 31%, 20% and 39% to 61%, 39%, 57% and 46% respectively.

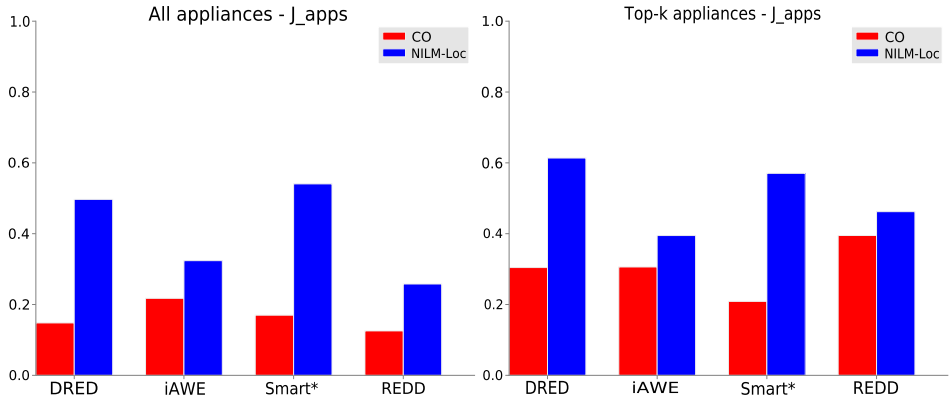

Figure 5.8: Disaggregation performance of Jaccard Apps in CO and NILM-Loc across datasets (1 week).

Number of appliance states identified correctly (J_s)

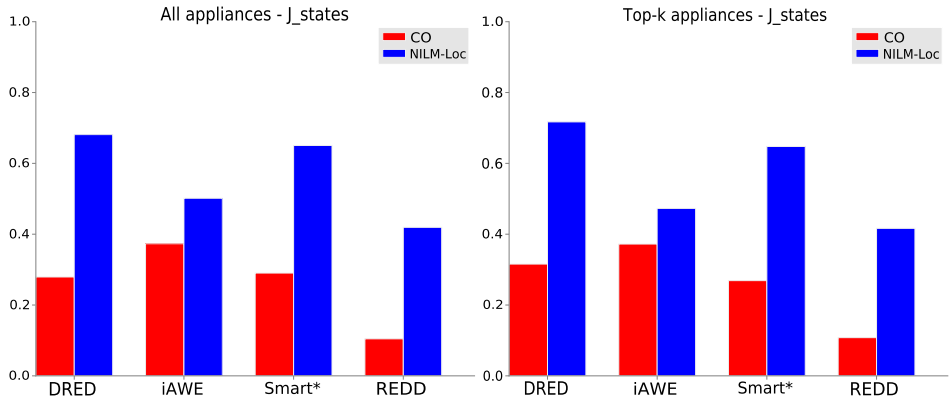

Figure 5.9: Disaggregation performance of Jaccard States in CO and NILM-Loc across datasets (1 week).

Similarly to with J_a, Fig. 5.9 shows the result of the Jaccard similarity index between the predicted states of the appliances and the actual states of the appliances in the ground truth. Considering all appliances in the household (left), NILM-Loc obtains better results than CO. In DRED it improves from 27% to 68%. Likewise, in iAWE it goes from 37% to 50%. In Smart* it is

enhanced from the original 29% in CO to 65% in NILM-Loc. Finally, in REDD it improves from 10% to 41%. In the case of the top 5 appliances NILM-Loc also outperforms CO. In DRED, iAWE, Smart* and REDD J_s, goes from 31%, 37%, 26% and 10% to 71%, 47%, 64% and 41% respectively.

Total disaggregation error (T_e)

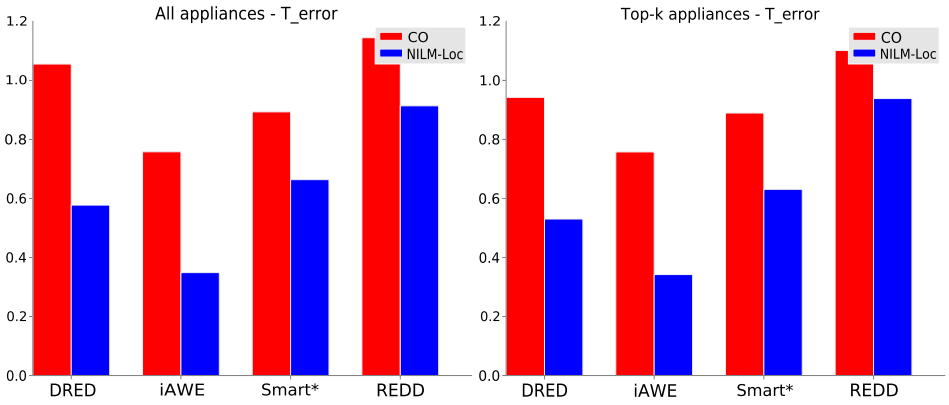

Figure 5.10: Disaggregation performance of Total Error in CO and NILM-Loc across datasets (1 week).

Fig. 5.10 shows the total error of the results of CO and NILM-Loc. The lower T_e, the better results it is. Considering all appliances in the household (left), NILM-Loc's total error is less than the one from CO. In DRED it improves from 1.05 to 0.57. Similarly, in iAWE it goes from 0.75 to 0.34. In Smart* it is enhanced from the original 0.89 in CO to 0.66 in NILM-Loc. Finally, in REDD it decreases from 1.14 to 0.91. In the case of the top 5 appliances NILM-Loc also outperforms CO. In DRED, iAWE, Smart* and REDD T_e, goes from 0.94, 0.75, 0.88 and 1.1 down to 0.53, 0.34, 0.63 and 0.93 respectively.

House level metrics over all days

Table. 5.2 shows the percentage increase in disaggregation accuracy of NILM-Loc compared to CO for all the days across the datasets with all appliances and top 5 appliances. For this table we consider the results from all days of data available regardless of the availability rate they have. As it can be seen, NILM-Loc outperforms CO with similar percentages as when only one week of data (the week with highest data availability rate) is considered.

Dataset	All Appliances				Top-k Appliances			
	FTE	J_a	J_s	T_e	FTE	J_a	J_s	T_e
DRED	30.5	28.7	36.6	-37.9	22.4	23.3	36.5	-41.2
iAWE	8.5	3.2	2.2	-7.9	14.8	3.3	4.5	-9.6
Smart*	29.3	28.3	28.6	-14.4	1.9	28.1	30.4	-18.6
REDD	11.4	12.7	27.6	-13.6	-22.1	5.3	27.3	-8.8

Table 5.2: Percentage increase in performance of NILM-Loc over CO (all days).

5.2.2 Appliance level results

Dataset	CO		NILM-Loc	
	P_e	N_e	P_e	N_e
DRED	5.588	0.075	2.288	0.038
iAWE	0.088	0.522	0.067	0.451
Smart*	0.127	16.45	0.108	2.261
REDD	0.070	50.98	0.053	35.24

Table 5.3: Appliance level accuracy metrics (1 week).

Table. 5.3 shows the aggregated appliance level accuracy metrics across all datasets for one week duration. In general, P_e and N_e can take any non-negative values. It can be seen that across all the datasets, P_e and N_e values for NILM-Loc are lower compared to CO; indicating better energy assignment for all appliances.

Appliance	CO			NILM-Loc		
	$F1$	P_e	N_e	$F1$	P_e	N_e
*TV	0.246	1.929	0.152	0.156	1.394	0.051
*Fridge	0.314	0.910	0.351	0.876	0.532	0.177
*Laptop	0.432	1.044	0.048	0.275	0.975	0.071
*Mircowave	0.090	1.006	0.020	0.424	0.615	0.016
*Cooker	0.341	0.895	0.008	0.434	0.495	0.005
Fan	0.092	3.023	0.072	0.153	1.636	0.049
Oven	0.115	0.968	0.004	0.261	0.555	0.003
Mixer	0.001	13.710	0.087	0.101	5.701	0.021
WashingMachine	0.042	33.966	0.016	0.007	10.566	0.008
Toaster	0.000	1.430	0.002	0.042	0.871	0.000
Outlets	0.099	2.583	0.066	0.026	1.831	0.013
Average	0.161	5.588	0.075	0.250	2.288	0.038
*Average Top-k	0.284	1.156	0.1158	0.433	0.802	0.064

Table 5.4: Appliance level accuracy metrics in DRED for all appliances.

Table. 5.4 shows the accuracy metrics for all appliances in DRED. Moreover, it shows the average of each metric considering all appliances and only the top-k appliances. It can be seen that for the top-k appliances, the average F_1 score per appliance is much higher than CO. Furthermore, NILM-Loc has better F_1 score for top-k appliances than for all appliances. Errors associated with disaggregation is also much lower for top-k appliances than compared to all appliances. This is mainly due to removal of noisy, less frequent and highly varying appliances present in the household.

5.2.3 Complexity index

Dataset	CO		NILM-Loc	
	Average	*Maximum*	*Average*	*Maximum*
DRED	104976	104976	7.1	972
iAWE	59049	59049	162.3	19683
Smart*	8192	8192	59.6	8192
REDD	165888	165888	72.8	41472

Table 5.5: Combinations / iteration across datasets.

Finally, we also computed the number of state combinations evaluated in each iteration of the algorithm in both CO and NILM-Loc. We refer to this number as the *complexity index*, which is an indicator of the processing and memory requirements for the execution of the algorithms in an embedded system. As explained in Section 3.2.2, the computational complexity of disaggregation is directly related to the cardinality of the set of appliances that are evaluated and, thus, to the number of possible state combinations that can be generated from them. Original CO has a fixed number of state combinations depending upon the number of appliances and its states. However, for NILM-Loc the number of appliances considered varies and is determined based on the constrained set of appliances.

In iAWE and Smart* the average state combinations to be evaluated for disaggregating a value is 59049 and 8192 for CO and it is 162 and 59 for NILM-Loc. Similarly in DRED, CO evaluated 104976 combinations and NILM-Loc evaluated only 7 on an average. In REDD 16588 combinations were evaluated in CO while only 72 were evaluated in NILM-Loc. It can be seen that across all datasets the average number of state combinations evaluated by NILM-Loc is drastically reduced, consequently, decreasing the computational complexity for real-time disaggregation. If we focus on the maximum number of combinations evaluated in each dataset, Table. 5.5 shows that NILM-Loc always evaluates less combinations than CO. The only exception for this is in Smart*, where the maximum number of combinations is the same for both algorithms. This means that, in at least one instance, all appliances were considered.

Chapter 6

Scheduler

In this work we present a case study for the use of fine-grained appliance level energy data obtained by NILM-Loc. We propose a Demand Response (DR) system that creates a house-level energy schedule of the appliances. The aim of our scheduler is to reduce energy costs while minimizing user discomfort. Discomfort refers to the inconvenience experienced by the consumers due to the changes in appliances usage applied by the DR system. We further explain how the scheduler works in the following section.

6.1 Scheduler framework

The main objective of DR programs is to balance the energy supply and demand. Smart meters now enable two-way communication between household and utilities to provide immediate feedback on power usage and pricing details. Utilities can now communicate the pricing details to consumers and further induce them to shift their consumption from peak to off-peak hours, by charging more during peak time periods [40]. In this way, utilities reduce stress in the power grid and the costs coming from large peak loads. From the users' perspective, they also reduce costs, satisfy environmental concerns and, depending on the program, they receive incentives from the utilities. However, as DR programs involve actions to change residential electricity demand, they imply applying changes that impact users' daily life routines since they are not able to make use of their appliances as they would without the progam. In consequence, this means that the proposed changes are either infeasible or that user comfort should be severely hampered.

Demand planning can be achieved at the consumer level *via* several techniques including peak clipping, valley filling, strategic conservation, strategic load growth, flexible load shape and load shifting [43]. All of them try to reshape the load demand of the consumer. Load shifting, in particular, involves shifting loads from peak to off-peak hours, without significantly influencing the average load over time. To perform effective load shifting,

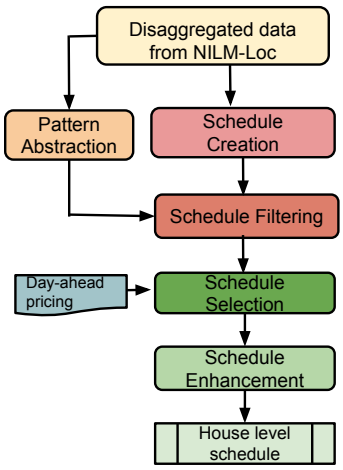

Figure 6.1: Overview of schedule generation algorithm.

consumers need to know detailed appliance level energy consumption and also the day-ahead pricing. For this work, we propose the use of load shifting as the load management technique. Given that our goal is to reduce costs with only a minimal effect to user comfort, by using load shifting, appliance usage duration remains the same albeit shifted in time.

We formulate a cost minimization problem at the consumer end by effectively scheduling loads based on hourly pricing. Appliance usage patterns of consumers are derived from the disaggregated energy data using NILM-Loc. The objective of this case study is to show how disaggregated energy data can be used to schedule loads effectively, to minimize the cost and the discomfort associated. Hitherto, load scheduling algorithms do not consider the heterogeneity of appliance usage at a household (i.e., different appliances have different time sensitiveness for shifting) or require detailed user and appliance level information for scheduling [40]. Our proposed scheduling algorithm consider patterns in historic appliance usage by determining the *flexibility* and *sensitivity* in shifting for every appliance. Furthermore, rule mining techniques are used to determine the *association rules* associated with the appliances. This ensures user preferences and appliance dependencies are considered in deriving cost effective load schedule.

Fig. 6.1 shows an overview of our schedule generation algorithm. Our scheduling algorithm has five stages *viz.,* schedule creation, pattern abstraction, schedule filtering, schedule selection and schedule enhancement.

6.1.1 Schedule creation

The first step in our scheduling algorithm is to find all possible schedules from the historic data of a household. To do so, given that the final output

of the scheduler is a one day energy schedule, we start by splitting the historical data into daily blocks. As a result, we obtain a set of schedules that represent past user energy behavior in a day. Thus, we create schedules that are feasible and with minimal impact in their comfort since users have already execute them at some point in the past.

Further to minimze discomfort, we group these schedules at different granularities (i.e., either weekdays or weekends, day of the week, etc.) and choose the subset of them that coincides to the type of day that corresponds to the schedule we are creating. For example, if we are designing a schedule for a Saturday we might choose only schedules coming from past Saturdays or from weekends. In such a way, the final proposed schedule retains a greater resemblance to what the user typically does on that day. Despite this, it may be possible that some of these schedules do not match user preferences either due to huge variation in demand profile on that day or due to arrival of guests in the household, etc. Hence it is necessary to determine the representative schedules that depict user preferences.

6.1.2 Pattern abstraction

To derive the representative schedules, we apply data mining techniques on the historic appliance usage data. We determine three indicators that show us different aspects of the usage pattern of each appliance:

1. **Flexibility coefficient.** Represents the average usage duration of an appliance in each hour of the day. This indicates the time periods when an appliance was used previously and how much time it was used. Fig. 6.2 shows a heat map of the flexibility coefficients of appliances in REDD.

2. **Sensitivity coefficient.** Indicates the preferred time delay in usage of an appliance. Some appliances can tolerate longer delays compared to others. For example, the coffee machine might allow for shorter delays than the washing machine as the user always prepares coffee within a specific (and shorter) time period.

3. **Association rules.** These associations provide information on appliance usage sequence. In general, the occupants have a daily routine making it possible to use an appliance in a sequence. For example, TV is always associated with a home theater.

This information will be used in the next step to determine the representative schedules.

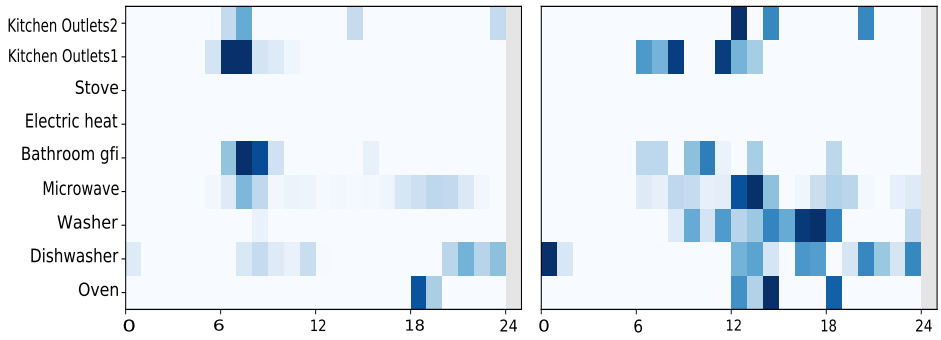

Figure 6.2: Usage patterns of appliances in weekdays and weekends respectively in REDD. Heat map depicts the average usage duration of each appliance in each hour of the day normalized to the hour where each appliance is most used, i.e., the hours with darkest blocks are the hours where the appliance is used the most.

6.1.3 Schedule filtering

This involves selection of schedules, which most accurately represent the user preference. From the set of schedules obtained after the creation stage, we select the subset of schedules that respect the usage patterns abstracted previously. Thus, we discard odd schedules that happened only a few times or that are not representative enough of what a typical day is, i.e., schedules where the usage of appliances do not adhere to their usual patterns or their average durations. To do so, we select schedules that adhere to the average flexibility coefficient of the appliances. Furthermore, we discard schedules which do not include the associations derived earlier. Fig. 6.3 shows the difference between a representative schedule and a non-representative one. In the top figure it is evident how a larger number of appliances in the schedule adhere to their average usage duration, while in the bottom figure only a few appliances do.

At this stage, we make a difference between schedulable and non-schedulable loads. The former ones are loads which have a direct relation with user behavior and thus are possible to shift, e.g. microwave, oven, dishwasher, etc. On the other hand, non-schedulable loads are independent of user behavior. For example, the compressor in the fridge is not directly manageable by the user and, consequently, does not make sense trying to schedule its usage. From this stage on, we focus on schedulable loads.

As the filtering is based on appliance level criteria, it may occur that on a particular schedule not all the appliances comply with their average flexibility coefficient or that not all association rules are satified. However,

in order to consider a schedule representative, a minimum percentage of the appliances must be compliant. This setting is adjustable by the user and determines the harshness of the filter. These filtered schedules are the representative schedules for that household, since they include the appliance usage pattern preferred historically by the users.

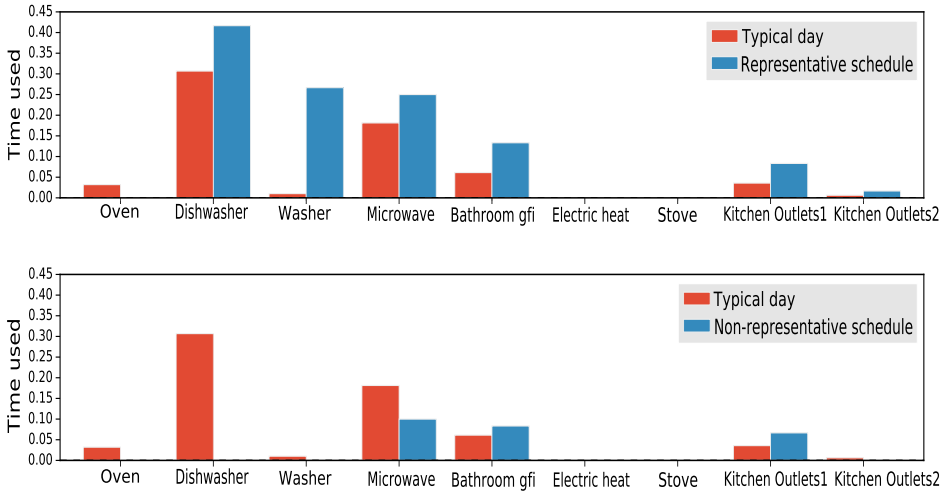

Figure 6.3: Appliance time usage duration in REDD during a typical day in comparison to one representative schedule (top) and in comparison to one non-representative schedule (bottom).

6.1.4 Schedule selection

From the representative schedules derived, we find the cost-optimal schedule based on the day-ahead hourly pricing provided by the utilities[1]. The outcome of this step is the schedule that minimizes the cost from the filtered schedules.

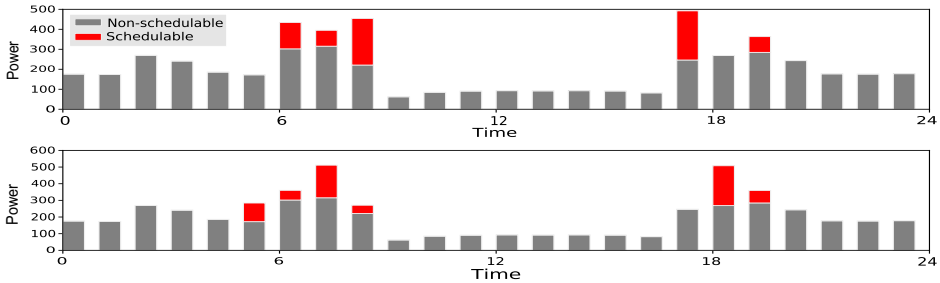

Figure 6.4: Cost-optimal schedule (top) and enhanced schedule (bottom) for a day in REDD.

[1]Day-ahead prices: `http://www.powersmartpricing.org/pricing-table/`

6.1.5 Schedule enhancement

Finally, in this last stage we try to enhance the cost-optimal schedule derived previously. Enhancements are performed by shifitng each appliance load based on their flexibility and sensitivity coefficients to further reduce the cost without increasing the discomfort associated. This means finding the schedule that satisfies the following equation,

$$min \sum_{i=1}^{24} C_i \quad s.t. \quad l^a_{f_c} \leqslant {f_c}^a \leqslant u^a_{f_c} \quad and \quad {s_c}^a \in (l^a_{s_c}, u^a_{s_c}) \tag{6.1}$$

where C_i is the cost of energy used during each hour of the day, ${f_c}^a$ and ${s_c}^a$ are the flexibility and sensitivity coefficients of each appliance, and $l^a_{f_c}$, $u^a_{f_c}$, $l^a_{s_c}$, $u^a_{s_c}$ their corresponding lower and upper bounds.

This is an iterative approach where each appliance event is either retained at the same time period (if the cost is lower) or shifted within the flexibility and sensitivity range determined previously. As mentioned in 6.1.2, the former indicates the average usage time period an appliance has in an hour and the latter indicates the time delay the appliance can tolerate. For the shifting process of each appliance event, we need to take care of several issues. For example, we must ensure that we retain the event as a single block and not subdivide it in smaller blocks to avoid more expensive hours that may exist in between. Moreover, besides respecting the average duration an appliance has during a particular hour, the appliance event should not span more time than it originally did. Avoiding to do so would mean extending the duration of the actual event while reducing its energy consumption. Fig. 6.4 shows how the schedulable loads from the cost-optimal schedule were shifted to obtain the enhanced schedule. Moreover, Fig. 6.5 shows a side by side comparison of the hourly power utilized in the cost-optimal schedule and the enhanced schedule. It also displays the hourly price of the energy. It can be observed how the enhanced schedule has shifted some power from more expensive hours to others less expensive.

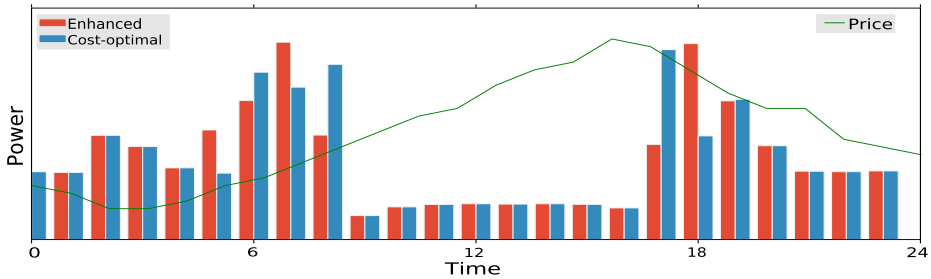

Figure 6.5: Comparison of cost-optimal schedule and enhanced schedule for a day in REDD.

6.2 Results

We evaluated our scheduling algorithm across several datasets. Fig. 6.6 shows the results from each step of the scheduler in DRED dataset. From all possible schedules, Fig. 6.6(i) shows the filtered schedules with schedulable and non-schedulable loads. Fig. 6.6(ii), (iii) shows the usage pattern of appliances in weekdays and weekends respectively. Fig. 6.6(iv) shows the cost effective schedule obtained based on the day-ahead pricing. Furthermore, Fig. 6.6(v) shows the enhanced schedule using flexibility and sensitivity coefficients. Finally, Fig. 6.6(vi) shows the load shifted and the day-ahead price. On this day, around 70% of schedulable load was shifted to achieve minimal cost and discomfort.

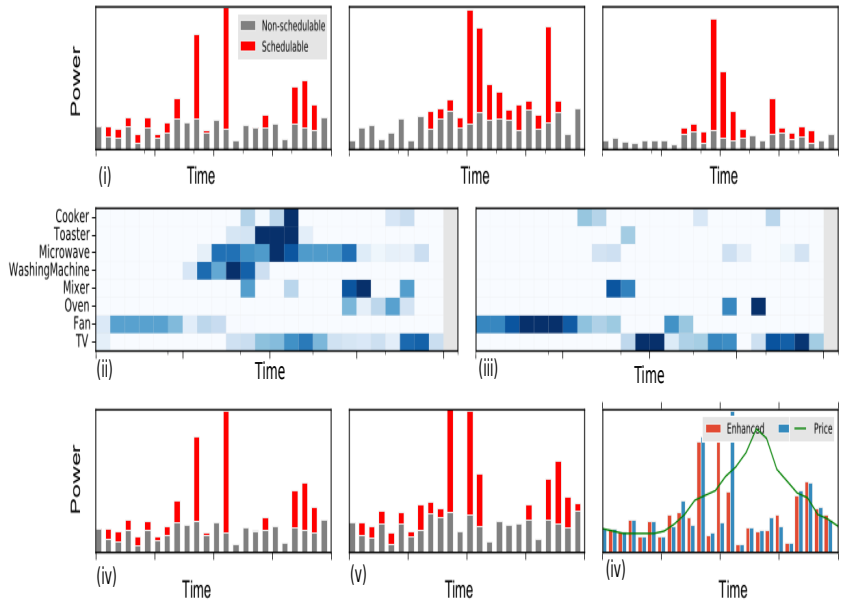

Figure 6.6: Optimal schedule for a day in our DRED dataset.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The core research goal of this thesis was to design a non-intrusive load monitoring system with reduced computational complexity that can run in a local embedded system without sacrificing disaggregation accuracy. We proposed a *novel* location-aware energy disaggregation framework (NILM-Loc) to derive accurate appliance level data. NILM-Loc can be used with any NILM algorithm proposed in the literature. We employed a modified CO algorithm to infer the state of the appliances. We evaluated NILM-Loc across multiple publicly available datasets such as DRED, iAWE, Smart* and REDD. Our evaluation shows that around 80% disaggregation accuracy can be achieved for all appliances on DRED and iAWE datasets. Furthermore, in a given day, up to 90% accuracy is achieved when only top-k appliances are considered for disaggregation in DRED and iAWE. The number of correctly identified appliances and states are 61% and 68% in DRED using NILM-Loc.

In general, all evaluation metrics registered a substantial performance boost when using NILM-Loc. This improvement can be observed across all datasets in both evaluation groups: all appliances and top-k appliances. NILM-Loc also substantially decreases the computational complexity of the algorithm by reducing the state explosion, thus, making NILM algorithms tractable and allowing them to run on low-cost embedded systems with constrained resources. This enables disaggregation to be implemented locally within the user's household, avoiding the need of third-party systems in the cloud that may expose sensisble private data.

Even with additional location information there are errors associated with disaggregation due to several factors. In most of the datasets, due to lack of knowledge on number of appliances and lack of monitoring of all appliances

in the household there exist a significant amount of unaccounted energy in the aggregate consumption. Only DRED dataset monitors almost all appliances and has a very low variation in baseline consumption. Also, the percentage of occupancy information available plays an important role in improving the accuracy. Therefore, the datasets with lesser unaccounted energy and datasets from which we have more accurate occupancy information have the best results overall. Furthermore, NILM-Loc uses a generic approximate model to find states of appliances. Accurate modeling of appliance states will further improve the disaggregation performance.

Finallly, we provide details of a case study for the use of fine-grained appliance-level energy data. We present a load scheduler that minimizes cost and discomfort based on hourly day-ahead pricing. Our algorithm employs the disaggregated data to derive appliance usage behavior and provide cost effective schedule for a household.

7.2 Future Work

The problem we investigated is complex and can be approached in many different ways. We have identified numerous points that need improvement or that can be further enhanced.

1. Implementation of other NILM algorithms in our NILM-Loc framework apart from combinatorial optimization.

2. In the case of using smartphone / wearables for retrieving occupancy information, data from the motion sensors of the device can be utilized to implement gesture recognition and infer the current user activity, e.g. cooking, watching TV, typing, etc. In turn, this information can be used to further improve accuracy of disaggregation by constraining even more the set of possible appliances responsible of energy consumption depending on the activity being detected.

3. Development of a personalized user-level scheduler for individual occupants to manage their load. The current implementation of the scheduler proposes a house-level energy programme that reduces costs without penalizing comfort. Nevertheless, the resulting cost-saving schedules can take a step further to minimize discomfort by taking into account energy behavioral patterns from each individual in the house. Given that NILM-Loc framework already requires user occupancy data, user-specific location information can be used to allocate energy consumption in a user-level basis. In this way, besides inferring *what* is being used, we could also deduce *who* is using it. Thus, allowing for more personalized and efficient schedules.

Bibliography

[1] Barker S., Kalra S., Irwin D., and Shenoy P., "PowerPlay: creating virtual power meters through online load tracking," In Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys '14), 2014.

[2] Energy outlook 2010. Energy Information Administration. [Online] http://www.eia.doe.gov/oiaf/ieo/index.html, 2010.

[3] Y Guo, M Jones, B Cowan, and R Beale. Take it personally: personal accountability and energy consumption in domestic households. *In CHI '13 on Human Factors in Computing Systems, 2013.*

[4] S. Darby. The effectiveness of feedback on energy consumption. A review for DEFRA of the literature on metering, billing and direct displays. 2006.

[5] C. Fischer. Feedback on household electricity consumption: A tool for saving energy? *Energy Efficiency, 1(1):79104, 2008.*

[6] G. W. Hart. Nonintrusive appliance load monitoring. *Proc. of the IEEE, 80(12):18701891, 1992.*

[7] J.Z.Kolter and T.Jaakkola. Approximate inference in additive factorial HMMs with application to energy disaggregation. *In Proc. AIS-TATS, 2012.*

[8] Ibe, Oliver. Markov Processes for Stochastic Modeling. *Newnes, 2013.*

[9] S. N. Akshay Uttama Nambi, T G. Papaioannou, D Chakraborty and K Aberer. Sustainable energy consumption monitoring in residential settings. *In Proc. IEEE INFOCOM 2013.*

[10] 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 Proc. e-Energy. ACM, 2014.*

[11] Y. Kim, T. Schmid, Z. Charbiwala, and M. Srivastava. ViridiScope: Design and implementation of a fine grained power monitoring system for homes. *In Proc. UbiComp. ACM, 2009.*

[12] M. Hazas, A. Friday, J.Scott. Look Back before Leaping Forward: Four Decades of Domestic Energy Inquiry. *Pervasive Computing, 2011.*

[13] M. Zeifman and K. Roth. Nonintrusive appliance load monitoring:Review and outlook. *IEEE Trans. on Consumer Electronics, 57(1):7684, 2011.*

[14] A. Zoha, A. Gluhak, M. A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors, 12(12):1683816866, 2012.*

[15] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han. Unsupervised disaggregation of low frequency power measurements. *In Proc. SDM. SIAM, 2010.*

[16] M. Baranski and J. Voss. Genetic algorithm for pattern detection in NIALM systems. *In Proc. SMCS. IEEE, 2004.*

[17] M. Berges, E. Goldman, S.H. Matthews, L. Soibelman. Learning systems for electric consumption of buildings. *In Proc. of the ASCE International Workshop on Computing in Civil Engineering, 2009.*

[18] M. Weiss, A. Helfenstein, F. Mattern, and T. Staake. Leveraging smart meter data to recognize home appliances. *In Proc. PerCom. IEEE, 2012.*

[19] A. Marchiori, D. Hakkarinen, Q. Han, and L. Earle. Circuit-level load monitoring for household energy management. *IEEE Pervasive Computing, 10(1):4048, 2011.*

[20] O. Parson, S. Ghosh, M. Weal, and A. Rogers. Non-intrusive load monitoring using prior models of general appliance types. *In Proc. AAAI, 2012.*

[21] L.Farinaccio and R.Zmeureanu. Using a pattern recognition approach to disaggregate the total electricity consumption in a house into the major end-uses. *Energy and Buildings, 30(3):245259, 1999.*

[22] N. Batra, H. Dutta, A. Singh. INDiC: Improved Non-Intrusive Load Monitoring using load Division and Calibration. *In Proc. of ICMLA, 2013.*

[23] S. Gupta, M. S. Reynolds, and S. N. Patel. ElectriSense: Single-point sensing using EMI for electrical event detection and classification in the home. *In Proc. UbiComp. ACM, 2010.*

[24] A. Rowe, M. Berges, and R. Rajkumar. Contactless sensing of appliance state transitions through variations in electromagnetic fields. *In Proc. BuildSys. ACM, 2010.*

[25] D. Jung and A. Savvides. Estimating building consumption breakdowns using ON/OFF state sensing and incremental sub-meter deployment. *In Proc. SenSys. ACM, 2010.*

[26] A. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. P. OHare. Real-time recognition and profiling of appliances through a single electricity sensor. *In Proc. of SECON 2010.*

[27]] M. Wytock and J. Zico Kolter. Contextually Supervised Source Separation with Application to Energy Disaggregation. *In Proc. of AAAI, 2014.*

[28] A. Kavousian, R. Rajagopal, and M. Fischer. Determinants of residential electricity consumption: Using smart meter data to examine the effect of climate, building characteristics, appliance stock, and occupants behavior. *Energy, 55(0):184 194, 2013.*

[29] 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 Proc. SustKDD. ACM, 2012.*

[30] N. Batra, M. Gulati, A. Singh, and M. B. Srivastava. Its Different: Insights into home energy consumption in India. *In Proc. Buildsys, 2013.*

[31] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, & S. Santini. The ECO Data Set and the Performance of Non-Intrusive Load Monitoring Algorithms. *In Proc. Buildsys 2014.*

[32] J. Z. Kolter and M. J. Johnson. REDD: A public data set for energy disaggregation research. *In Proc. SustKDD, 2011.*

[33] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The smart thermostat: Using occupancy sensors to save energy in homes. *In Proc. SenSys10. ACM, Nov. 2010.*

[34] T. A. Nguyen and M. Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings, 56:244257, 2013.*

[35] Dutch Residential Energy Dataset. [Online] http://www.st.ewi.tudelft.nl/akshay/dred/

[36] 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 Proc. SustKDD, 2012.*

[37] S. Makonin, F. Popowich, L. Bartram, B. Gill, and I. V. Bajic. AMPds: A Public Dataset for Load Disaggregation and Eco-Feedback Research. *In Proc. EPEC, 2013.*

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

[39] C. Holcomb. Pecan Street Inc.: A Test-bed for NILM. *In Proc. of NILM workshop, 2012.*

[40] C. Joe-Wong, S. Sen, Sangtae Ha, Mung Chiang. Optimized Day-Ahead Pricing for Smart Grids with Device-Specific Scheduling Flexibility. *In Journal of JSAC , 2012.*

[41] S. Rollins, N. Banerjee. Using rule mining to understand appliance energy consumption patterns. *In Proc. PerCom, 2014.*

[42] Zhuang Zhao; Won Cheol Lee; Yoan Shin; Kyung-Bin Song. An Optimal Power Scheduling Method for Demand Response in Home Energy Management System. *Smart Grid, IEEE Transactions on, 2013.*

[43] Gellings, C.W. The concept of demand-side management for electric utilities. *In Proc. IEEE, 1985.*

[44] Sookoor, T.; Whitehouse, K. Roomzoner: Occupancy-based room-level zoning of a centralized HVAC system. *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on , vol., no., pp.209,218, 8-11 April 2013*

[45] Christensen, D., L Earle, and B Spam. Nilm Applications for the Energy-efficient Home. . *National Renewable Energy Laboratory Technical Report, November 2012*

[46] Zoha A, Gluhak A, Imran MA, Rajasegarar S. Non-Intrusive Load Monitoring Approaches for Disaggregated Energy Sensing: A Survey. *Sensors (Basel, Switzerland). 2012;12(12):16838-16866.*

[47] Engel, D., Wavelet-based load profile representation for smart meter privacy. *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES, vol., no., pp.1,6, 24-27 Feb. 2013*

Chapter 8

Appendix

8.1 Algorithms pseudocodes

8.1.1 NILM-Loc

```
results = []
appliances_already_ON = None
while (True)
  #================================================
  #Get the aggr. energy and occupancy data for the last min
  occupancy_data = []
  energy_data = []
  priority_combo = appliances_already_ON

  timestamp = get_time()
  while ((current_timestamp - timestamp) < ONE_MINUTE)
    current_timestamp  = get_time()
    energy_data.append(get_current_energy_value())
    occupancy_data.append(get_current_user_locations())

  current_energy_value = downsample_data(energy_data)
  current_user_locations = aggregate_last_minute_user_locs()

  #================================================
  #Compare current aggregate value to priority combo
  delta = get_tolerance_treshold(current_energy_value)
  priority_combo_sum = get_sum_of_appliances_in_combination(
      priority_combo)

  if abs(priority_combo_sum - current_energy_value) <= delta:
    #============================================
    #Predicted appliances and their states remain unchanged
    results.append((timestamp, priority_combo))
  else:
    #============================================
    #Get constrained set of appliances
    constrained_appliances = []
```

```
    constrained_appliances.append(get_appliances_in_locations(
        current_user_locations))
    constrained_appliances.append(appliances_already_ON)
    constrained_appliances.append(metadata.
        appliances_autonomous)
    constrained_appliances.append(metadata.
        appliances_remotely_controlled)

    #=========================================
    #NILM algorithm (CO)
    state_combinations = find_all_possible_combinations(
        constrained_appliances)
    summed_power_of_each_combination = []
    for combo in state_combinations:
      summed_power_of_each_combination.append(
          get_sum_of_appliances_in_combination(combo))

    valid_prediction = False
    while valid_prediction == False:
      predicted_combo = find_nearest_combination(
          current_energy_value,
          summed_power_of_each_combination)
      #=====================================
      #Validation
      appliances_turned_OFF = get_appliances_turned_OFF(
          appliances_already_ON, predicted_combo)
      locations_of_appliances_turned_OFF =
          get_locations_of_appliances(appliances_turned_OFF,
          metadata)
      invalid_location_exists = compare_locations(
          current_user_locations,
          locations_of_appliances_turned_OFF)

      if (invalid_location_exists):
        #Invalidate last prediction
        summed_power_of_each_combination[predicted_combo] =
            None
      else:
        valid_prediction = True

    results.append((timestamp, predicted_combo))
    appliances_already_ON = predicted_combo
    priority_combo = predicted_combo
```

Listing 8.1: NILM-Loc pseudocode

8.1.2 Scheduler

```
#=====================================================
#Import appliance-level energy data (1 min freq.)
app_data = import_disagg_energy_data(start_date, end_date)

#=====================================================
#Resample data to hourly frequency
app_data_hourly = resample_data(app_data, freq='1Hr')

#Group data according to the type of day requested
app_data_grouped = group_data_according_to_type_of_day(
    app_data_hourly, type='weekdays/weekends')

#=====================================================
#Pattern abstraction
flexibility_coefficients =
    get_average_usage_duration_in_each_hour_of_day(
    app_data_grouped)
sensitivity_coefficients = get_range_of_hours_used(
    app_data_grouped)
association_rules = get_daily_sequence_of_usage(
    app_data_grouped)

#=====================================================
#Schedule creation
all_schedules_possible = get_daily_schedules(app_data_grouped,
    target='weekends')

#=====================================================
#Schedule filtering
schedulable_loads = get_user_dependant_loads(metadata)
filtered_schedules = []
for day_schedule in all_schedules_possible:
  if (min_required_patterns_satisfied(day_schedule) == True)
    filtered_schedules.append(day_schedule)

#=====================================================
#Schedule selection
day_ahead_hourly_pricing = get_pricing()
cost_optimal_schedule = None
for day_schedule in filtered_schedules:
  price_day_schedule = get_schedule_cost(day_schedule,
      day_ahead_hourly_pricing)
  price_cost_optimal = get_schedule_cost(cost_optimal_schedule,
       day_ahead_hourly_pricing)
  if price_day_schedule < price_cost_optimal:
    cost_optimal_schedule = day_schedule

cost_optimal_schedule_minute_resolution =
    get_minute_level_cost_optimal_schedule(
    cost_optimal_schedule, appliance_data)

#=====================================================
```

```
#Schedule enhancement
enhanced_schedule = None
apps_events_in_schedule = get_appliances_events(
    cost_optimal_schedule_minute_resolution)
for event in apps_events_in_schedule:
  appliance = event.get_appliance()
  app_sensitivity = sensitivity_coefficients[appliance]
  app_flexibility = flexibility_coefficients[appliance]

  #Add original time position of event
  possible_positions_of_event_during_day = []
  possible_positions_of_event_during_day.append(event)

  #Find all possible permitted positions for that event
  shift_possible = True
  while shift_possible:
    #Shift event considering the appliance characteristics
    shifted_event = shift_event(event, app_flexibility,
        app_sensitivity)
    possible_positions_of_event_during_day.append(shifted_event
        )

    #Check if there is still room for continuing shifting this
        event
    room_for_shifting = determine_if_shifting_is_still_possible
        (shifted_event, app_flexibility, app_sensitivity)
    if (room_for_shifting == False):
      shift_possible = False

  #Find cost-optimal placement of event
  optimized_event = None
  for possible_event in possible_positions_of_event_during_day:
    price_possible_event = get_event_price(possible_event,
        day_ahead_hourly_pricing)
    price_optimized_event = get_event_price(optimized_event,
        day_ahead_hourly_pricing)
    if price_possible_event < price_optimized_event:
      optimized_event = possible_event

  #Add event to enhanced schedule
  enhanced_schedule.add_event(optmized_event)
```

Listing 8.2: Scheduler pseudocode