Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Recently, the field of Smart Grid technology has been widely developed and solutions for appliance detection show great promise, offering valuable tools for prediction and energy management schemes. Future compliance with environmental mandates and energy conservation have given rise to a need for energy usage analysis methods, including the detection of user behavior patterns. This may be achieved by combining smart metering technologies with home energy management systems to monitor, evaluate and control the operation of domestic appliances.

This project describes a typically termed NIALM (Non-Intrusive Appliance Load monitoring) technique, which is based on a prototype genetic algorithm proposed by M. Baranski and J. Voss [4]. The aim of this work is to approach the detection problem of unknown devices, by applying clustering to the available power traces, a genetic algorithm for modelling all possible appliances, and a verification process of the aggregated power signal. In addition, a recently proposed solution for energy capping and concealment, provided by Toshiba Laboratory [9] is exploited. Finally, cross-correlation tests on protected and unprotected power traces are performed, extracting information about the overall detection accuracy.

Bristol City Council and Toshiba Telecommunications Research Laboratory are major partners of the '3eHouses' European Union FP7 project, dealing with energy efficiency technology. Power Consumption Data was collected using ZigBee powered sensors, from semi-detached family homes based in Bristol (UK). In general, our implementation consists of the following key steps:

1. Data Sanitisation
2. Event Detection
3. Learning Phase
4. Appliance Identification - Evaluation

Initially, a data-mining process for extracting series of discrete edge events is implemented in Java, using random power traces. In terms of sanitisation, this involves filtering and decomposing techniques, for uncorrupted records extraction. Also, differences within active power edges of the aggregated signal are identified and groupings of events are created, mapping important power signal step changes.

Next, using Matlab, K-means clustering is applied to classify similar edge events. All cendroid points are analysed and progressively build a genetic classifier, modelling all operating devices. Through this, a set of series of events is extracted, setting the ground for the rest of the NIALM learning phase. With no previous knowledge of the appliances which are being monitored, this step produces a Finite State Machines static table representing all possible detected appliances.

Finally, the detection phase is based on a modified knapsack algorithm which runs in parallel and processes trials of random days. The performance is tested by measuring correlation within real and hypothetical devices. Accuracy is also tested on protected and unprotected power loads.

1 Introduction

Recently, the field of Smart Grid technology has been widely developed, and a variety of different technologies focus on energy saving strategies. National governments and relevant stakeholders make significant efforts in the development of future electricity network, as smart grids are an emerging engineering challenge to reform the worlds electrical grids [13]. Their aim is to reduce cost and meet environmental goals such as the EU's 20-20-20 goals (20% increase in energy efficiency, 20% reduction of CO_2 emissions, and 20% renewables by 2020) [10].

There is no doubt that the need for environmental compliance and energy conservation is vital, thus extensive research in load forecasting, energy score-keeping and control of power-consumption equipment has been emerging. In the near future, millions residencies will be equipped with smart meters. Such intelligent monitoring devices, transmit power-usage information to the utilities, in order to allow load monitoring and power consumption analysis. Aside from the cost for special measuring devices, the intrusion into the local installation is the main problem for reaching a high market distribution. In the UK, there is a plan for mass roll out of about 53 million smart meter for gas and electricity between 2014-2019. Also, the Dutch government announced that all 7 million homes in the Netherlands would have been equipped with smart meters by 2013.

Making use of smart meters, appliance profiles can be generated by sampling similar households. Running below hundred megahertz, low-cost algorithms can be applied to run on top of existing smart meter devices for real time frequency monitoring and observation of high-low peak usage. Such devices, when installed, are expected to provide accurate readings automatically at requested time intervals to the utility company, and extract daily energy usage patterns. It is important to mention that behavior patterns detection can be extremely beneficial for electricity companies to statistically analyse their consumers, and distribute energy more efficiently according to high-low peak timings.

Since the inception of electricity deregulation and market driven pricing throughout the world, utilities have been looking for a means to match consumption with generation. Traditional electrical and gas meters only measure total consumption, and so provide no information of when the energy was consumed at each metered site (market use rates are readily available to utilities however). Smart meters provide a way of measuring this site-specific information, allowing price setting agencies to introduce different prices for consumption based on the time of day and the season.

There is a rich and ongoing line of research in the construction and upkeep

of NIALM techniques, providing means to identify appliance usage even when multiple household power signatures are aggregated [18]. Recent studies propose the use of load signatures, to detect basic operating characteristics [8], [24], which exploit any consequential power difference between successive operation states of the loads.

According to current proposals, load reduction will be confirmed using communication between the smart meters in the home and the individual appliances. This scheme has a number of apparent security problems as outlined in the NISTIR security report [23]. In order to confirm the correct operation of each appliance, numerous proposals suggest methods for demand management, involving remote attestation in order to confirm the correct operation of each appliance's reporting.

Despite the significant advantages of such technology, communication from the appliance to the meter has to be both reliable and secure from communication tampering in order to prevent malicious users changing data to falsely lower their bills, or disallow third parties to access critical household information. Service providers have to make use of secure schemes to ensure anonymity of the home users whose data are being mined, and control power load signatures from being distributed among unauthorised parties.

The objective of this work is to investigate the performance of a novel technique for Pattern Detection which has been originally proposed by M. Baranski and J. Voss [3]. The authors propose a NIALM technique able to separate the load into its major appliances, by accurate identification of load signatures of any possible appliances that may be operating in private households. Through a genetic algorithm, it is possible to detect appliances having no previous knowledge of the power signals or any other information regarding the appliances usage. This genetic algorithm offers a variety of parameters to be tuned, in order to improve the detection or to assimilate data sources of different applications.

Also, we measure the effectiveness of the detection on protected and unprotected power signals, using a privacy technique which has been proposed by G. Kalogridis in [14], [15]. The authors of this recent work suggest a simple but effective solution for preserving anonymity to the high-frequency energy measurement data, offering an increased level of privacy to the user. This was possible by using using a rechargeable battery to balance the power load, and conceal peak events when possible.

1.1 Non Intrusive Appliance Load Monitoring Overview

A SMART GRID is a network that integrates the actions of all users connected to it, by using smart meter data communications aiming to improve system analysis and control [10]. This technology enables us to measure energy consumption in more detail compared to conventional power meters. Therefore, the contribution of a smart grid network to the current area of power technology is to manage energy distribution better, compared with existing power grids, and is underpinned by the analysis of energy usage data, collected from smart meters as seen in Fig. (1).

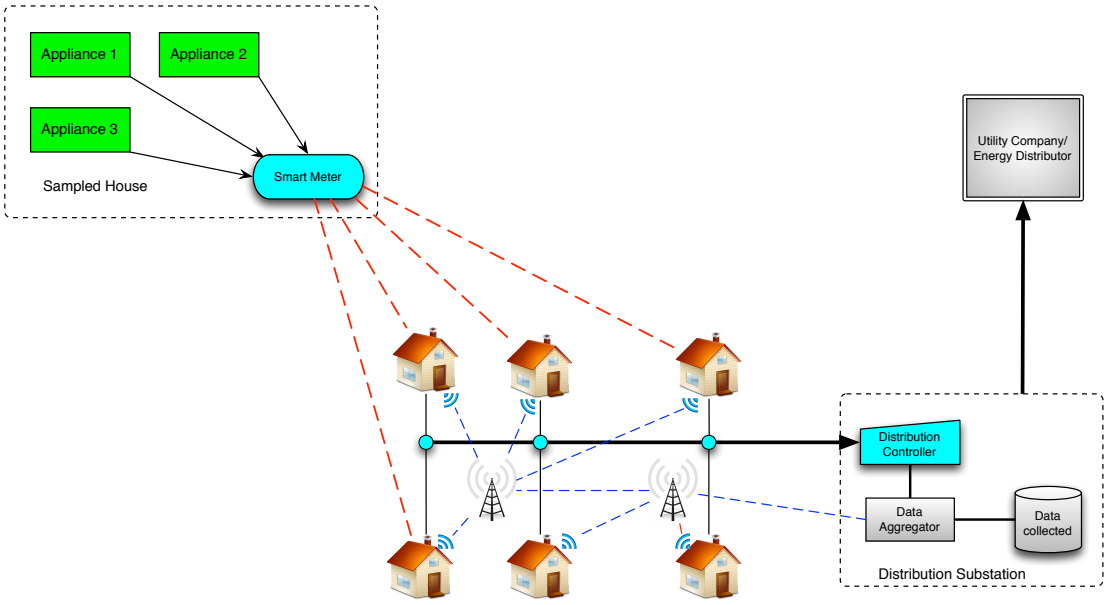


Figure 1: Distribution network and smart metering data structure

The problem of finding non-intrusive measurements methods for disaggregating total information collected by households, is being discussed since 1990s. This novel idea started when George W. Hart [12] was collecting and analysing load data as part of a residential photo-voltaic systems study. Hart suggested a method of analysis of the current and voltage of the load data, to simplify the collection of energy consumption data by utilities. Non-Intrusive Appliance Load Monitoring known as NIALM, is, since mid-nineties, a mature technique for disaggregating the entire electrical load into the uses of domestic appliances. Prevalently, an intermediate architecture between customers and power operators known as aggregators, has been deployed. Aggregators are mainly used for co-ordinating, transacting and implementing the demand services in parallel with the existing power infrastructure. Having installed smart meter aggregators in numerous residential households, sufficient amount of sampled active power data can be collected.

NIALM systems usually require sufficient and accurate power load information to separate the total amount of power consumption into the major appliances using all voltages and currents. Each individual household consists of numerous electric appliances, which may have different switch events based on their operating mechanisms. Detecting switch events in consideration of load peaks is commonly the only information source, which has to be exploited in order to find structures of unknown electrical appliances.

The most common monitoring system category is the intrusive model (see Fig. (2)), where multiple sensors are used to capture the power consumption of each device. This method can be expensive, as requires multiple sensors in order to get accurate results, in addition to data storage centres and other communications technology. Also, it requires physical access into the house, which creates privacy concerns, while raises the cost of installation.

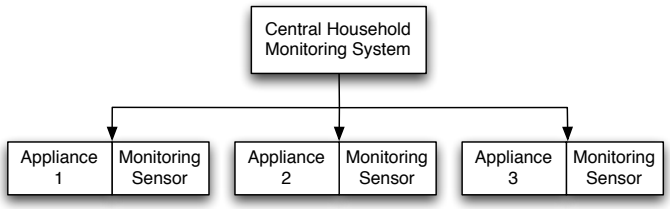


Figure 2: Intrusive monitoring system

In comparison, non-intrusive systems (Fig. (3)), do not require someone to *intrude* into a household and install separate devices over each appliance to measure different power consumptions. Such systems are less expensive and can be more flexible to install in existing households power meters. Non-intrusive methods, reduce the complexity of individual installations, requires less bandwidth in order to transmit information collected, and in general is preferable.The prototypical NALM for residential buildings utilise the difference between the two steady power levels (defined as event) and follow the following key steps:

1. Measurement of current and voltage using a 1-sec sampling interval, from which real and reactive power are calculated.
2. Detection of On and OFF events.
3. Clustering of similar events.
4. Matching of On and OFF events over time.
5. Device identification.

Some examples of NIALM systems that may be used in different fields of residential, commercial or industrial areas [12], are the following:

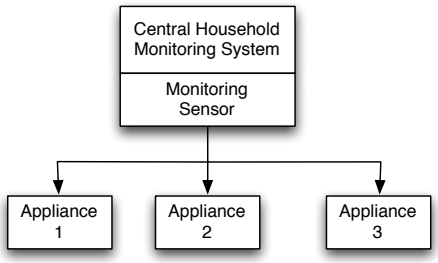


Figure 3: Non-Intrusive monitoring system

- Load Forecast: Continuous analysis of power consumption and optimisation of energy production, in order to reduce power cost of supplied electricity.
- Load Management Control: Reduce energy costs in households, as such systems can analyse characteristics of appliances, and suggest users how to reduce power consumption.
- System Failures: Reveal information about system failures in high risk electricity networks, and offer monitoring solutions.

1.2 Appliance modeling

NIALM systems are based on the fact that electrical appliances show special characteristics during operation time. In general, domestic appliances can be categorised into four major classes as described in Table 1, and the categories are the following:

1. Permanent consumer devices (constant load). These devices are 24/7 switched on, with constant and reactive power consumption.
2. Two-state appliances such as light bulbs or toasters, have two operations at any given moment (ON/OFF). Typical switching patterns which can be mainly detected in houses with constant switched on power consumption. Appliances included in this category have a single transition power value between the two power states.
3. Finite State Machines (FSM) appliances such as washing machines or dishwashers, having distinct types of ON states, e.g., fill, rinse, spin etc. These are described by having more than one distinct on power grade (a fan has distinct power settings).
4. Continuously-varying appliances are in general variable-speed hand tools like light dimmers, washing machine and drilling machine. These operate at a continuous range of ON states. Such appliances are difficult to monitor non-intrusively, as they do not generate step changes in power consumption,

and they do not offer much potential for optimisation.

Table 1: Classification of various household consumer appliances

	Active Control	Autonomous control
Permanent On appliances	Manually switched pumps, air conditioner, etc.	stand-by loads, heating control, aquarium pump
On-Off Appliances	TV, PC-monitor, kettle, hair-dryer boiler, stove, cooker, fan heater, electric iron	waterheater, fridge, waterbed, air conditioner, fish tank heating, freezer
Finite State Machines (FSM)	PC-printer, microwave, dish, washer, video, coffee machine	
Continuous Variable Appliances	lighting with dimmer, HiFi units	

All the above appliances, can be described by an electrical expression named as *Load Signature*. Load signatures are useful information to produce many useful services and products, such as determining the energy usage of individual appliances, monitoring power quality, and developing facility management tools. Some typical measurement such as power consumption, current waveform and harmonics, can be treated as load signatures. By the operating characteristics entailed in the load signature, automated systems are able to identify the load consumption and predict its demand and impact on the power system.

Virtually, all smart meters can send information to one or more devices across the home network. This allows utility companies to collect information regarding home usage, and transmit back to the consumers information regarding usage, price and cost data. Known as HAN, this innovative technology allows interconnection of digital devices from multiple computers and their peripheral devices to telephones, VCRs, televisions, home security systems, "smart" appliances and other various digital devices. Smart meters allow consumers to create HANs with an in-home display to help them monitor and manage their energy use as well as remotely monitor and control thermostats and other electric appliances. In the near future, a smart home power and communications network (see Fig. (4)), could possibly comprise the following components:

i. Smart metering two-way communication technologies
ii. Electrical devices or appliances with integrated control methods
iii. Private sources of energy
iv. Power router for Sensing and Measurement
v. Load Signature Moderator (LSM)
vi. Home Area Network (HAN)

In the near future, independent appliances, will be HAN-equipped and through Wi-Fi/HAN radio frequencies, a smart meter will be able to talk directly to the home computer or smartphone. This revolutionary data acquisition technology, aims to a more energy-friendly home, providing remote metering readings, automated outage notification and energy efficiency and savings. In contrast, HAN provided data, can raise privacy concerns, as such systems are insecure from the perspective of data transmission interception, encryption and other potential data leakage.

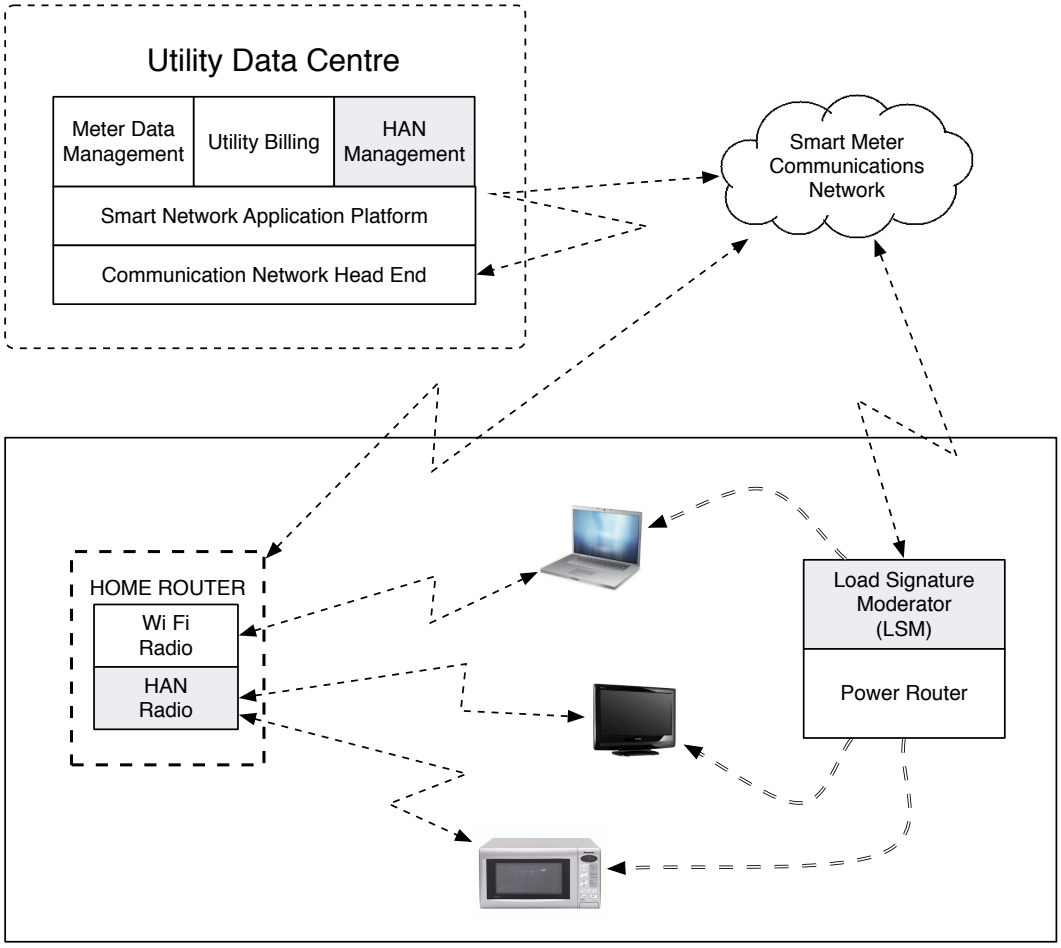


Figure 4: HAN distribution using Smart Meter Communications Network

1.3 NIALM Privacy Impacts

While NIALM techniques are based on total active power disaggregation of appliance specific data, privacy-preventing guarantees against adversary's knowledge are also important. The notion of differential privacy, as discussed in [2], is an important part of smart-grid data processing. NIST [11] refers to smart grid privacy

as being contradictory to the major benefit provided, i.e., the ability to get richer data to and from customer meters and other electric devices, is also its Achilles heel from a privacy viewpoint (see Table 2).

Numerous techniques for preserving privacy have been developed in order to secure encrypted messages between a meter and the supplier. They propose improved secured decryption methods for private Stream Aggregation algorithms, and other anonymity preserving data mining techniques relevant to suppliers and customers. Thus, it is important not only to construct efficient NIALM algorithms, but also preprocess, evaluate and classify accurately private and public power consumption information.

Table 2: Privacy Impacts that arise from collection and use of Smart Grid Data.

Who wants smart meter Data?	How could the data be used?
Utilities	Monitor electricity usage and load; determine bills
Electricity advisory companies	Promote energy conservation and awareness
Insurance companies	Determine health care premiums based on unusual behaviors that might indicate illness
Marketers	Profile customers for targeted advertisements
Law enforcers	Identify suspicious or illegal activity
Civil litigators	Identify property boundaries and activities on premises
Landlords	Verify lease compliance
Private investigators	Monitor specific events
The press	Get information about famous people
Creditors	Determine behavior that might indicate creditworthiness
Criminals	Identify the best times for a burglary or to identify high-priced appliances to steal

The National Institute of Standards and Technology (NIST) in a recent report [23] proposed the development of a security architecture for the Smart Grid Technology related to Transmission, Distribution, Operations, Bulk Generation, Markets, Customer, and Service Provider. In Europe, data privacy is mainly addressed by the European Union Data Protection Directive [7], which requires that "personal data should be collected for a specified purposes and not be further processed for other purposes".

Table 2 describes some major potential privacy impacts, summarising the importance of preserving the privacy of the individuals whose power data is being distributed. Currently, not many technological solutions exist to protect smart

grid privacy; mostly discussed solutions are based on aggregated and encrypted power metering data.

The current work focuses on two main NIALM problems: (a) the implementation of a novel algorithm for appliance and monitoring detection based on unprotected power signals, and (b) determining accuracy of the detection when applied on protected signals using a recently proposed LSM technique. In addition, this work is based on building appliance models without requiring a-priori knowledge of the operating devices, nor any information regarding usage behavior of the residents.

2 Systems Overview

The current work is based on collected smart meter data from real residencies. Toshiba TRL is a major partner of a European Union FP7 project entitled '3eHouses' [1]. Toshiba TRL's smart grid research consists of numerous research projects covering European smart grid stakeholders as well as academic institutions by developing algorithms feeding into smart grid and home networking applications. This €4M FP7 project focuses on utilising information and communication technology (ICT) in order to ultimately reduce domestic energy consumption. '3e-HOUSES' project deals mainly with the integration of the most established ICT technologies in social housing, to provide an innovative service for energy efficiency. This is highly related in developing efficient HAN domestic networks, which will provide real time monitoring of the energy consumption, integration of renewable energies, and creating the resources to lower energy consumption.

2.1 Trials and Measurements

Sampled data used in this paper is from the replicator project based in Bristol (UK) where, in collaboration with the Bristol City Council, the houses are typically semi-detached family homes. ZigBee powered sensors are used to measure energy consumption of six household appliances as well as aggregate (per house) meter data at 20 seconds intervals. The data used is taken from the benchmarking phase of the project so it can be assumed that the measurements represent typical behavior patterns without the effect of any ICT interventions.

ZigBee is a specification for a suite of high level communication protocols using small low-power radios based in the IEEE 802.15.4-2003 standard for wireless personal area networks. This technology is intended to be simpler and less expensive compared to other WPANs, such as Bluetooth. ZigBee sensors are applicable to RF applications that require a low data rate, long battery life and secure networking. In United States and European Union it has become the major driver for HAN, as several investors select ZigBee for Smart Grid applications. The primary attractiveness of ZigBee is its open standard platform that promises interoperability among multiple systems.

In addition, ZigBee offers a Smart Energy Application Profile that is specifically designed for utility applications within the HAN such as Demand Response, Smart Charging, Hybrid Electric Vehicle etc. (see Fig. (6)). It also uses frequency hopping spread spectrum radio technology, offering long range, reliable

performance, and immunity against interference and jamming. Such network is shown in Fig. (5).

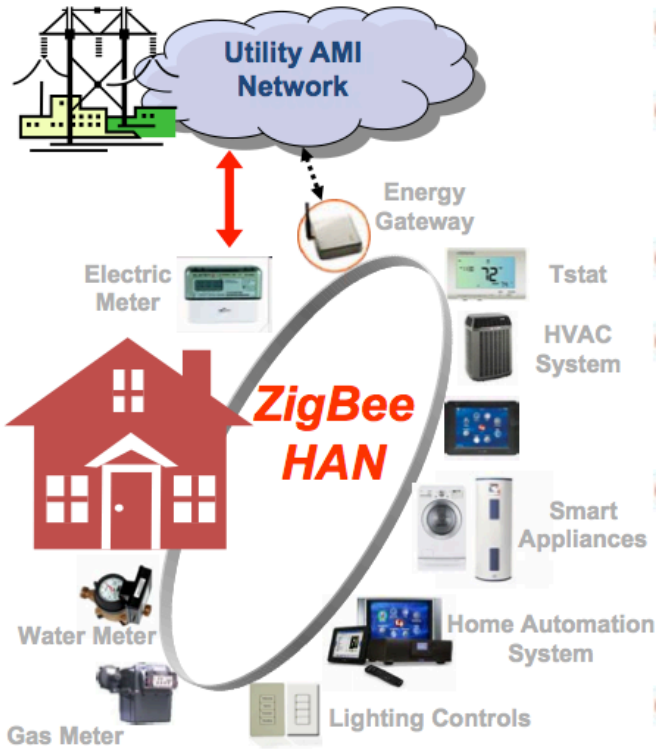


Figure 5: ZigBee HAN example

All samples indicate the active power consumption of the devices shown in Fig. (7i). Also, an aggregator has been installed to measure the overall consumption of each house, again sampling at 20 seconds interval time (Fig. (7iii)). Each deviceID represents a mapping of a specific sensor that is connected to a specific device, as shown in Fig. (7ii). All provided power trace sql records contain active power consumed (Watt) and cumulative energy (Watt-Hours), and are of the following form:

```
Record :  <deviceID, date, time, Instant power, Cumulative power>
```

2.2 Data Sanitisation

As part of both the learning phase and the monitoring phase, the fist goal of the meter is to detect abrupt changes in the power readings which correspond to loads changing state. An event detector analyses current power values, and whenever it detects sufficient change of the current power value, it generates a compressed form of meter readings.

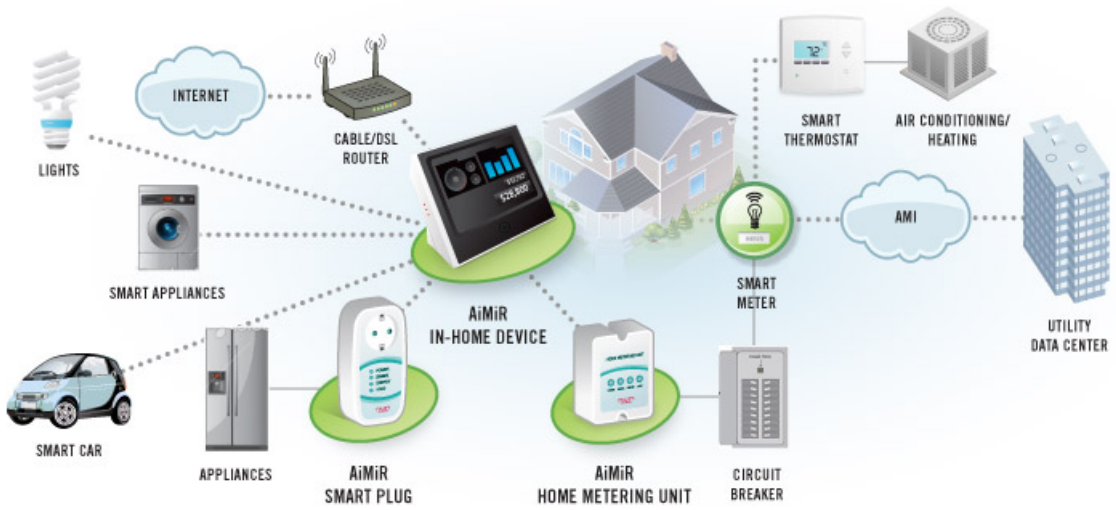
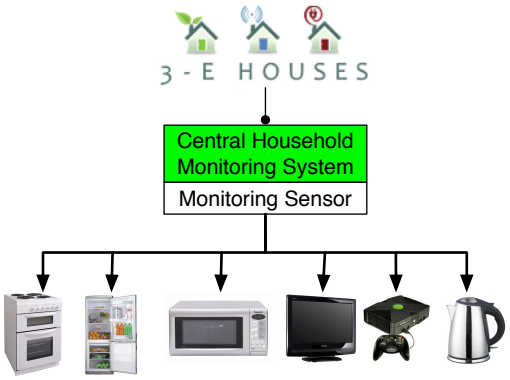


Figure 6: Home Energy Management example

The 3eHouses data measurement system is unfortunately inconsistent, as it is depended on a variety of factors ranging from installation errors, hardware or network failures, including interference in the 2.4 GHz carrier band and signal loss. Ideally, each ZigBee device samples at $\tau = 20$ sec intervals, which is not guaranteed. The provided dataset holds records of 5 months or more. Commonly, such files contain excessive records, and fast process is essential. In addition, during pre-processing, it is necessary to apply filtering normalisation in order to handle corrupted power traces. Thus, records were split according to timestamp and appliance ID. Faulty records, or inconsistent devices, were dropped off. Eventually, each device sampled, was analysed and individual files were generated for detection and verification purposes (§5).

Using Java we implemented the following steps:

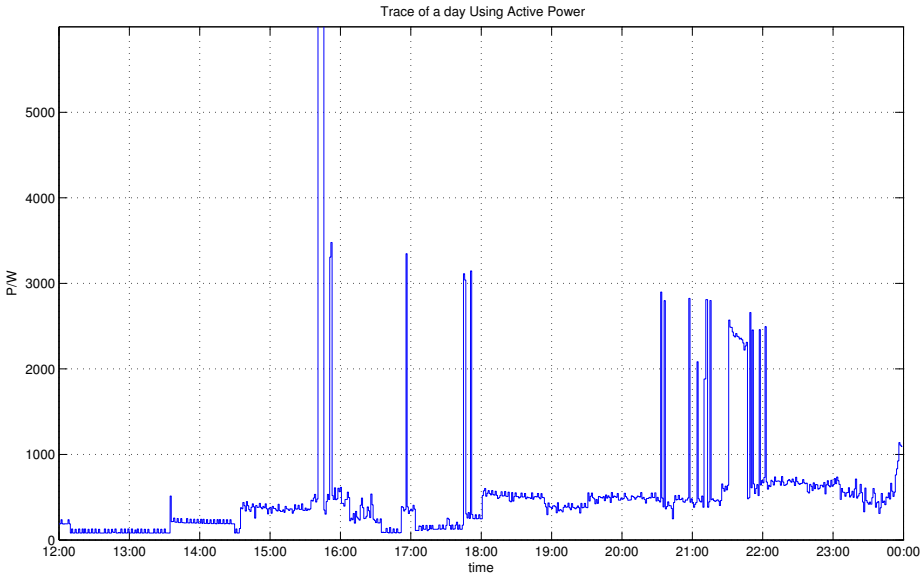
1. Fast sorting of dataset: Unix function `sort -k` is used to sort all records according to time and ID fields.
2. Partitioning of all appliance ID's : `Java.NIO` library provides support for fast buffered I/O. In that way, data is read and split into particular appliance ID files, sorted by time sequence. This allowed faster process of individual records.
3. Filtering of corrupted records: Smart meters can be broken, or stuck, or the wireless signal which ZigBee power sensors transmit may be corrupted for various reasons. Therefore, it is possible that some records will not be accurate. For this case, we drop any days that do not contain at least 1 record per minute, or their values of active power and cumulative energy are out of range, or invalid.



(i) Sampled Devices

deviceID	description
50	Fridge
51	Microwave
52	TV Audio
53	Xbox
54	Kettle
40	Total
41	Cooker

(ii) Smart Meter Device ID



(iii) Typical Power Graph

Figure 7: Sampled Devices from a typical 3eHouse in Bristol (UK).

4. Normalization of records: A sampling rate of 20s is not always guaranteed, so it was essential to apply normalisation (Algorithm 1) in order to produce consecutive records, 1 per second, as shown in Fig. (8). A typical power graph of a random day is shown in Fig. (7iii), derived from the total aggregate smart meter with `deviceID` = 40, after normalisation.

The above process can be easily comprehended from a simple example shown in Fig. (8). Given four values of active power ($P_1 = 90W$, $P_2 = 180W$, $P_3 = 140W$, $P_4 = 250W$), the method populates each power value evenly among all distinct seconds intervals (1 per second). This is achieved by finding all ΔT differencies within each pair of (P_t, P_{t-1}) and replicating P_t within the time space of

Algorithm 1 Data Normalisation

Input: $P(i)$: Active Power at time t_i
Input: N : Last t_i sample of Day
for all i $\in [2, N]$ **do**
  $\Delta P_{i+1} \leftarrow |P_{i+1} - P_i|$
end for
for all integer j $\in \left[\frac{\Delta P_{i-1}}{2}, \frac{\Delta P_{i+1}}{2}\right) \bigwedge \frac{\Delta P_{i-1}}{2} \leq P_i \leq \frac{\Delta P_{i+1}}{2}$ **do**
  populate $P(j) \leftarrow P(i)$
end for

$\left(\Delta T_{P_{t-1}} \leq T_{P_t} < \Delta T_{P_t}\right)$. Eventually, the extracted dataset is consistent, allowing learning process to handle data more efficiently.

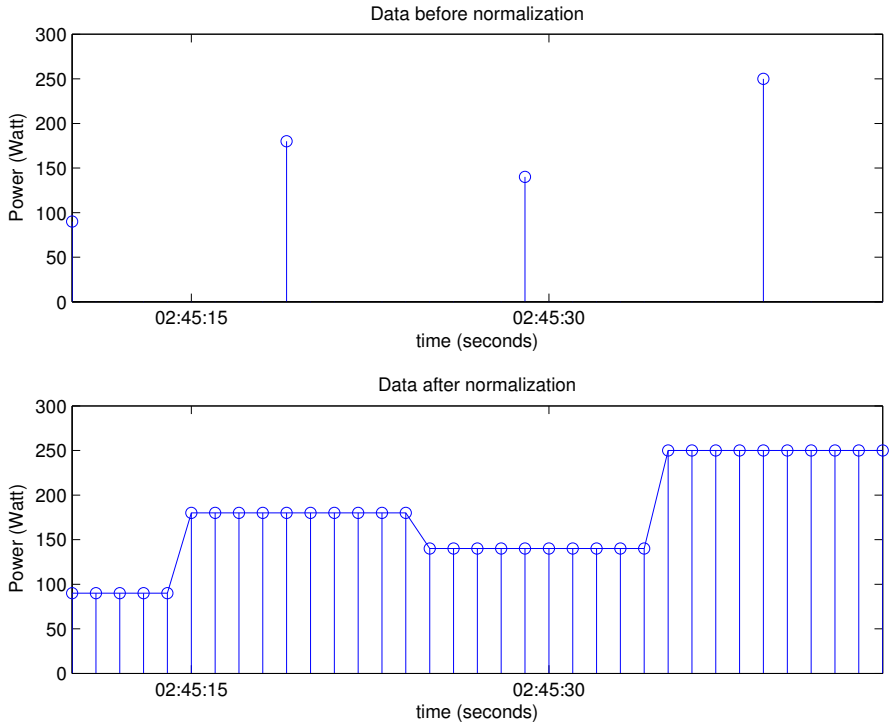


Figure 8: Normalization of Data

3 Edge Detection and Clustering

Eventually, having filtered out only interesting days (§2.2), a process of edge detection and clustering, is initialized (Fig. (9)). Let us consider a set $\{P_t\}$ of active load series. With respect to the behavior of typical appliances, the active power could be assumed to be the total load at any given time defined by series of discrete power during a time period of one day. Our algorithm implements a simple custering analysis of the events, and then develops a static state table using a genetic algorithm to optimize a set of possible finite state machines.

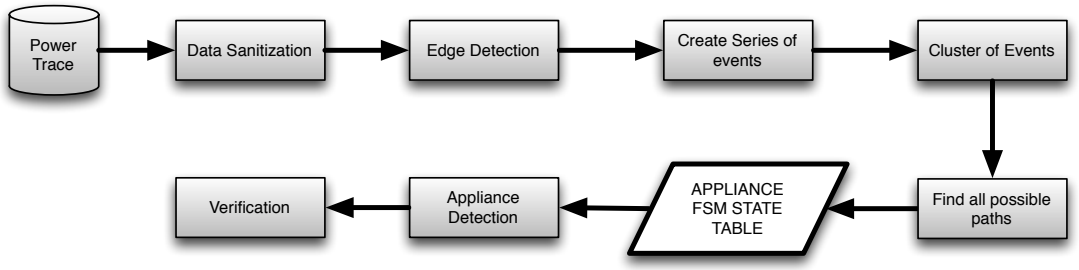


Figure 9: Structure of the overall NIALM process

Each independent electric appliance will be handled as a Finite State Machine. All possible power step values of each appliance, can be detected by discovering recurring sequences which consist of the same fixed order. The motivation for clustering is based on the observation that power traces have fluctuations around a set of values that correspond to significant power levels of the underlying system. Provided that the fluctuations are small, these power levels can be extracted from the trace. In order to remove the subjective nature of spotting these levels and permit automation of the technique, we decided to use one of the many cluster analysis techniques, to locate the presence of these levels, measure distance metric and group them into n clusters that minimise the distance between points.

Through this, we attempt to discover as many as possible accurate N_C clusters, and later, all power C_r values, will be assigned to each power centre $P(C_r)$, with respect to time sequence. The implementation was based on the method proposed by Baranski [4], and, using Java and Matlab programming, we implemented from scratch the basic algorithm, while adding some important modifications regarding accuracy and performance. Eventually, clustering produces a decomposition of any given power load into its major components. This technique is analysed in §3.1 and §3.3.

3.1 Power load disaggregation

In order to decompose the load into its components, we define the following:

$\{P_t\} :=$ Power load value at any given time defined by series of discrete power during a time period (24H), where:

$$P_t \geq 0, \quad \forall t \tag{1}$$

$\{P(S_t\}) :=$ Active Power value of the t^{th} switch event; the total load (measured in Watt) at any given time defined by series of discrete power during a time period (one day). It clearly depends on which appliances are switched on at any given moment, so a switch process $S(t)$ has to be described.

$\{\Delta P_t\} :=$ Difference series of the load (Watt). This metric will be used extensively for the project, and is defined as:

$$\Delta P_t = P_t - P_{t-1}, \quad \forall t \tag{2}$$

$\{S_t\} :=$ Series of detected switch events; a new S_i is created whenever a new ΔP_t is detected, exceeding a specific threshold δ, and has opposite sign to the previous ΔP_{t-1} which was assigned to that S_i. Algorithm 2 is referred to this method, extracting all possible series of events S_i, and relevant switching characteristics are detected, by computing subsequent differences in the power level. Suppose there are n appliances identified by i, $1 \leq i \leq n$.

$$S_{i,t} = \begin{cases} 1 & \text{if appliance } i \text{ is on at time t} \\ 0 & \text{if appliance } i \text{ is off at time t} \end{cases} \tag{3}$$

$\{\hat{P}\,(S_i)\} :=$ Denotes the maximum power step value, or boost value, according to a peak event.

$\{T(S_i)\} :=$ Represents the corresponding time distance between the first and the last ΔP_t that was stored in each S_i.

$\{N_S\} :=$ Number of switch events evaluated from the total load of one day.

Authors in [4], describe a mapping rule (eq. 4), which symbolise Difference Series as neighbouring values ΔP_t that are assigned to a switch event.

$$\Delta P_t = \begin{cases} S_i & \text{if } |\Delta P_t| > \delta \bigwedge sgn(\Delta P_t) = sgn(\Delta P_{t-1}) \\ S_{i+1} & \text{if } |\Delta P_t| > \delta \bigwedge sgn(\Delta P_t) \neq sgn(\Delta P_{t-1}) \end{cases} \tag{4}$$

The edge detector algorithm, takes an input of threshold δ and iterates through all detected ΔP_t, by reading sequentially each P_t and P_{t-1}. In order to successfully create different switch models, all events are handled as complex data types, which contain objects Q_i for each appliance. This process is eventually terminated by storing all power series, holding objects Q_i in sequential time ordering:

$$Q_i = \left[P\left(S_i\right), \hat{P}\left(S_i\right), \Delta T \right] \tag{5}$$

Each Q_i is a new object vector, and `createSeries()` iterates through all sorted time series of active power that have been generated after data Sanitisation `SanitizeData(Dataset File, ApplianceID)` function.

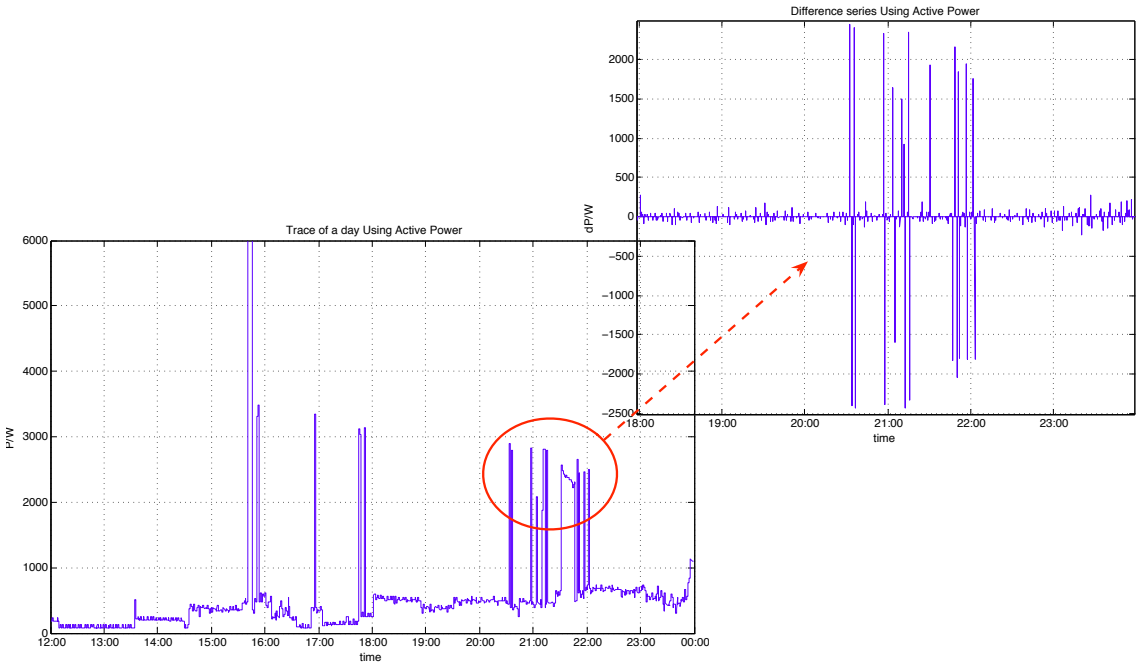


Figure 10: Difference Series of Events

3.2 Combining Sequences of Events

Subsequent to creating all S_i, it is necessary to reduce all detected elements, by examining time differences that correspond to each ΔP. It is possible that each S_i contains events based on more than one appliance, thus we have to assure that if each S_i occurs too often, is may be generated by more than one appliance. In general, combining sequences of events aims to eliminate records which have

Algorithm 2 CreateSeries of Events

Input: $P(t) \leftarrow$ Active Power at time t, N $\leftarrow$ last t_i record of dataset
Input: $\delta \leftarrow$ threshold power barrier (eliminate minor events)
Output: Series of Events $\{S_N\}$
for all t $\in [2, N]$ **do**
  $\Delta P_t = P_t - P_{t-1}$
  if sizeof $\{S_i\} =$ `null` **then**
    Create new $\{S_i\}$
  end if
  if $|\Delta P_t| > \delta$ **and** $sgn\{\Delta P_t\} = sgn\{\Delta P_{t-1}\}$ **then**
    $S_i \leftarrow \langle \Delta P_t, t \rangle$
  else if $|\Delta P_t| > \delta$ **and** $sgn\{\Delta P_t\} \neq sgn\{\Delta P_{t-1}\}$ **then**
    $S_{i+1} \leftarrow \langle \Delta P_t, t \rangle$
  end if
end for

occurred within same family of events , and do not comprise adequate time duration (ΔT threshold). Implemented in Java, `combineSeries_Pow()`, processes all Q_i objects, and exports sufficient number of events as follows:

$$S_i = S_i + S_{i+1} \forall S_i : \{P(S_i) \geq 0; P(S_{i+1}) \leq 0; t_{i+1} - t_i \leq \Delta t_{peak}\} \tag{6}$$

This procedure is described by algorithm 3 in more detail.

Algorithm 3 Combine Series of Events

Input: Series of events $\{S_n\}$, $\Delta t_{peak} \leftarrow$ time threshold
Output: Combined, time Sorted series of events $\{S_n\}$
for all $\{S_i\} \in \{S_n\}$ **do**
  find all $\Delta P \in \{S_i\}$
  $P(S_i) \leftarrow \sum_{k=1}^{|S_i|} \Delta P_k$, where n is the last element of $\{S_i\}$
  $\hat{P}(S_i) \leftarrow \max\{\Delta P_i\} \forall \Delta P_i \in \{S_i\}$
  $\{\tau_i\} \leftarrow t_{\Delta P_{first}}$ of $\{S_i\}$, $\{\tau_{i+1}\} \leftarrow t_{\Delta P_{first}}$ of $\{S_{i+1}\}$,
  if $P(S_i) > 0 \bigwedge P(S_{i+1}) < 0 \bigwedge \Delta t_{peak} > \tau_{i+1} - \tau_i$ **then**
    $S_i \leftarrow S_i \bigcup S_{i+1}$
    Drop S_{i+1}
  end if
end for

Finally, an example of combined switch events are represented in Fig. (12). The algorithm concentrates only on significantly strong or frequently occurring appliances, and it is comprehensible that combined switch events can minimise series of stored events, and make the clustering process more accurate. Detecting

peaks with very small distance Δt_{peak} reduces the number of events due to measurement noise. In Fig. (11), a power signal is transformed into sequence of Series of Events using ΔP/Watt signal, which was created by all detected ΔP events. Rare and small events are neglected due to their irrelevant impact. The process of clustering is then *initialized*.

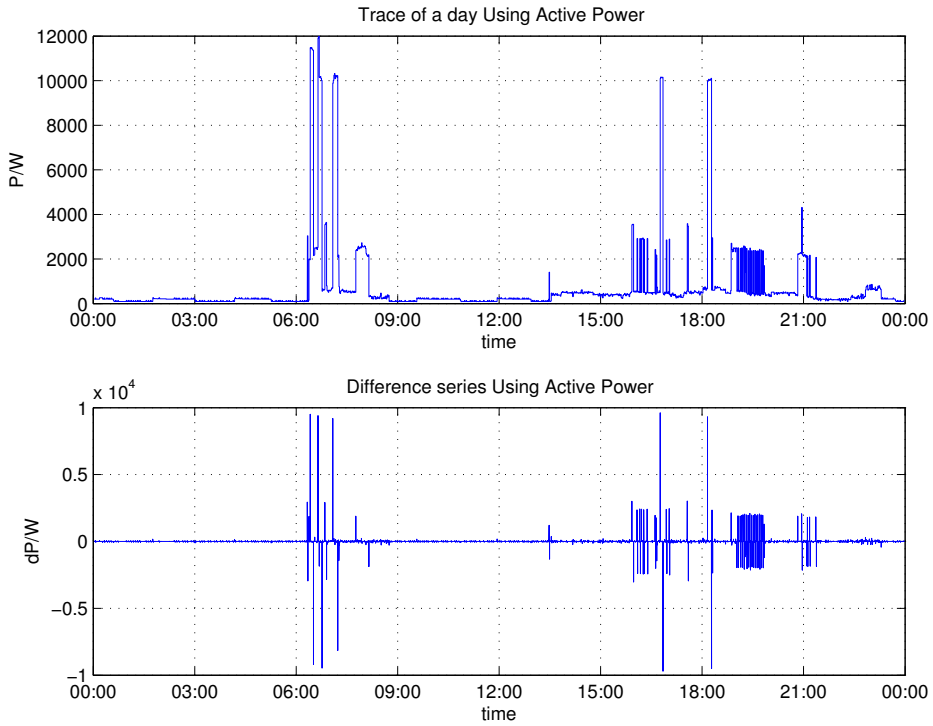


Figure 11: Difference Series of Events, using $\delta = 10W$, $\Delta T_{peak} = 60$ sec

3.3 Pattern Detection and K-means Clustering

Clustering is based on the events caused by electrical appliances. These events consist of changes in the power consumption caused when devices are turned on and off. By plotting those changes in a two-dimensional space, it is possible to investigate how to form clusters of appliances, using machine learning techniques. The existing methods for energy disaggregation can be split into into two main categories:

1. Unsupervised methods as described in [17, 16] that require labelling of detected appliances, and prior knowledge of the number and type of appliances.

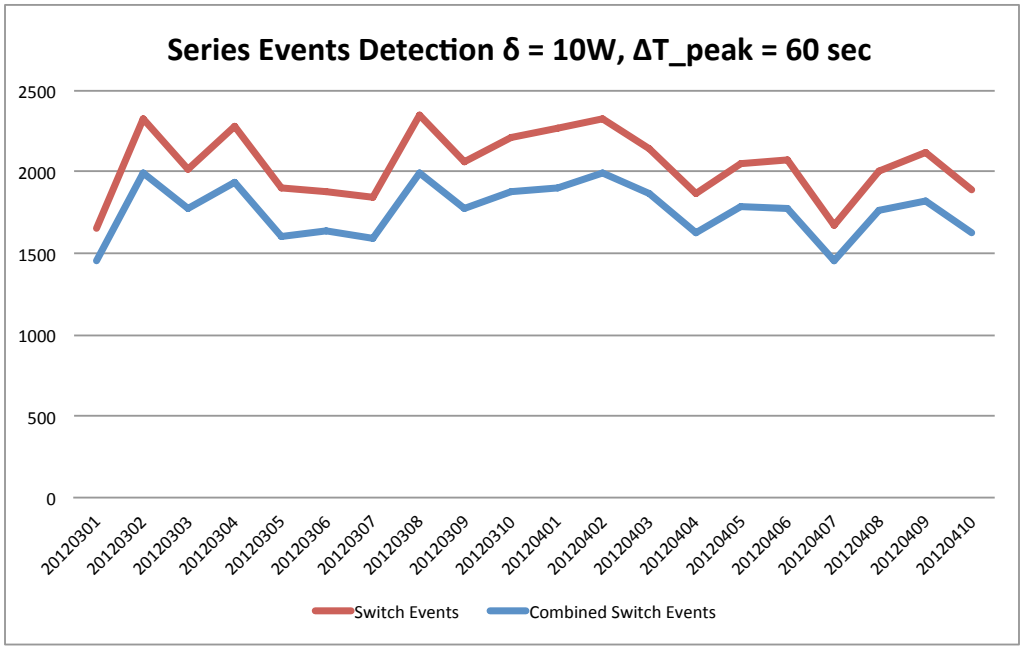


Figure 12: Series combined for various days, using $\delta = 10W$, $\Delta T_{peak} = 60$ sec

2. Supervised methods that mainly focus on training the designed models given sufficient datasets, prior to performing disaggregation.

For the current supervised method analysis, we apply 2-dimensional K-means clustering of the combined series of events. In data mining, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean value. The algorithm tends to find clusters of comparable spatial extend. Given a set of observations $(x_1, x_2, ..., x_n)$ where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k sets $(k \leq n)$: $S = \{S1, S_2, \ldots, S_n\}$. The process is repeated to minimise each sum of squares within each cluster S_i, which is denoted as $\arg\min \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$, with μ_i is the mean of points in S_i. Repetitively calculations are performed until all calculated new means are centroid of the observations in the cluster (see Fig. (13)).

Using Matlab, all combined series mapped to stored Q_i objects, are processed. K-means takes as input pairs of $\left[P\left(S_i\right), \hat{P}\left(S_i\right)\right]$ values, and, a specified number of maximum clusters $\#C_i \geq \#N_c$ with N_c is the number of all real appliances. The problem of finding patterns of unknown appliances can be formulated as an optimisation problem. Each appliance V_j is described by as sum of its events. For each appliance, only a subset of all N_S events can be linked to this appliance. We limit the solution space of possible events by reducing the num-

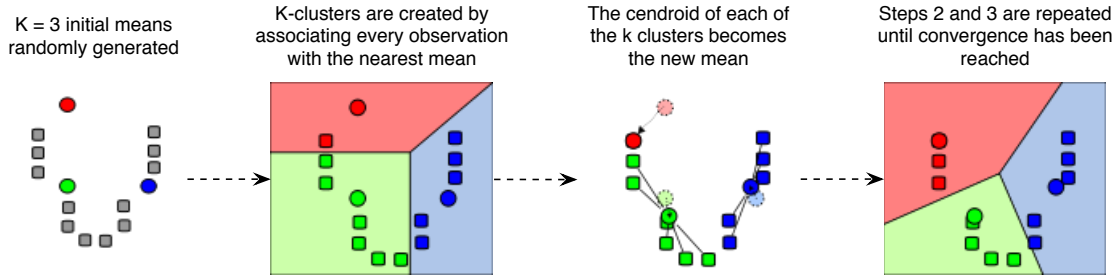
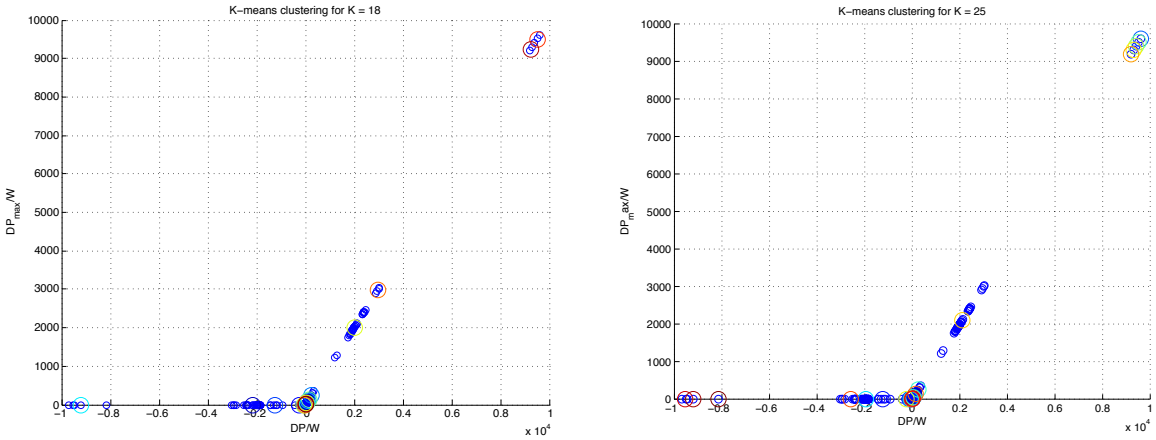


Figure 13: Demonstration of the standard K-means process

ber of detectable appliances. Only appliances with recurring patterns arouse our interest.

Using efficiently K-means with a predefined number of C_i clusters, conglomerations of similar switch events are generated, producing N_C clusters for all Q_i (Fig. (14)).

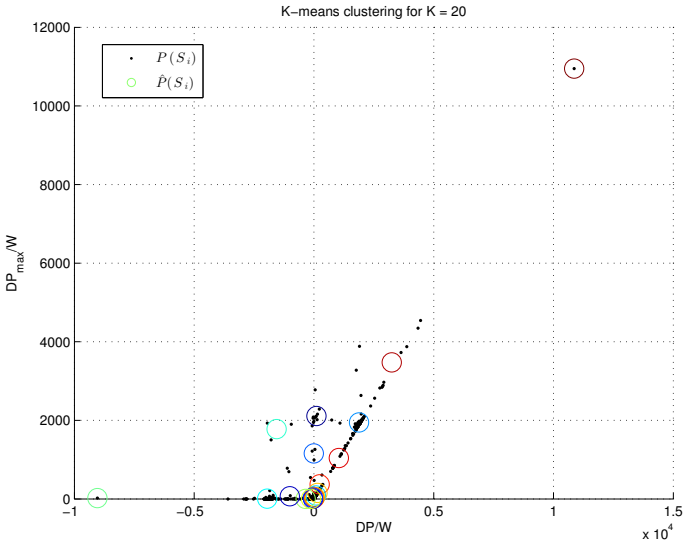


(i) K-means for K = 18 Clusters

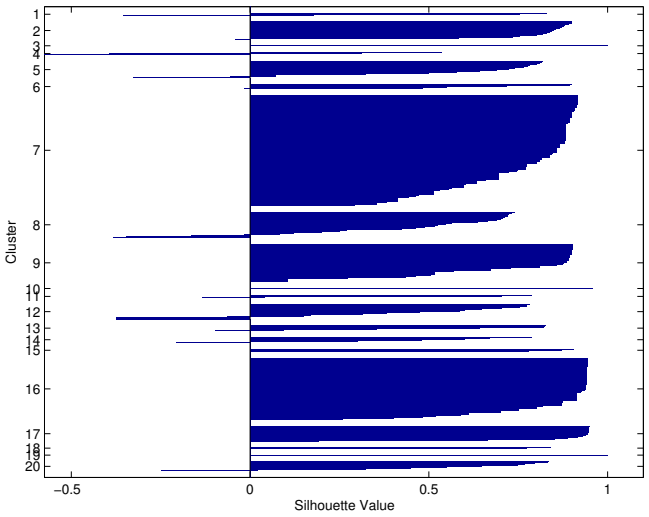
(ii) K-means for K = 25 Clusters

Figure 14: K-means Clustering

Another example of clustering and the relevant silhouette plot is denoted in Fig. (15). Silhouette refers to a method of interpretation and validation of clusters of data. It provides a succinct graphical representation (see Fig. (15ii)) of how well each object (centroid) lies within its cluster. The average $s(i)$ of a cluster is a measure of how tightly grouped all the data in the cluster appear. The average $s(i)$ of the entire dataset is a measure of how appropriately the data has been clustered. In K-means algorithm, some of the clusters will typically display much narrower silhouettes than the rest. A Silhouette plots and averages may be used to determine the natural number of clusters within a dataset.



(i) K-means for K = 10 Clusters



(ii) Silhouette (clustering K = 20)

Figure 15: K-means and Silhouette clustering for 2012-01-22

In Fig. (15), the initial value N_C was set to K=20, whereas in Fig. (15ii), the cendroid clusters that form most of events, are approx.regarding 6-8. This information gives a small idea regarding possible number of appliances that expected to have been operating in the household for the specific day. With red dots (Fig. (16)) are represented the clustered results of K-means algorithm, in time-sequence order. Having produced 20 clusters, each value of ΔP is matched to a specific cluster value.

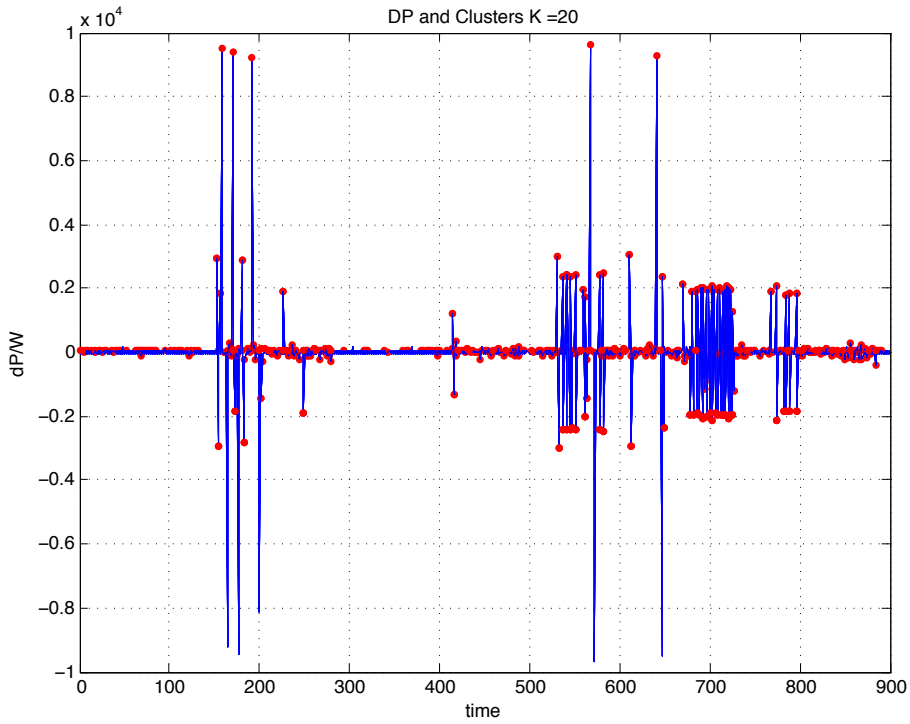


Figure 16: Clustering of ΔP detected edge events

As each object Q_i holds time-stamp information of time observed within the day, then all ΔP stored in that Q_i object, are mapped to each distinct power clusters. Combinations of different clusters represent potential candidates of appliances. If the distribution of appliance on/off events is uniformly distributed in the plane of real power, then it would be difficult for the clustering algorithm to make intelligent decisions based upon the data. Proceeding in §4, each possible binary combination of power clusters $P(C_r)$ will be used as a possible FInite State Machine.

4 Learning Phase - Building FSM appliance tables

The idea of recurring pattern detection is based on the following notion. Given a random appliance V_j, at any time t, then the power series of V_j can be represented as

$$P_{j,t} = \sum_{i=1}^{N_S} u_{j,i}\sigma(t - t_i)P(S_i) \tag{7}$$

with

$$\sigma(t - t_i) = \begin{cases} 0, & \text{if } t < t_i \\ 1, & \text{otherwise} \end{cases} \tag{8}$$

Using a binary vector $[u_{i,1}, \ldots, u_{i,N_S}]$ for each possible i appliance, only one position will be one, and all the others have to be zero, for a random t_i. Also the time series of each appliance have to satisfy an additional constraint

$$P_{j,t} \leq P_t \qquad \forall t \tag{9}$$

Each appliance V_i holds its own vector $\underline{u_i^T}$ containing binary combinations of u_{i,N_S}. In this way, special events can be selected while masking all the other by setting the corresponding values $u_{i,j}$ to zero. All valid events, are then clustered; this is accomplished by taking an interval of data and grouping similar events according their respective power changes, summarising events with similar structures. Valid combinations of different clusters represent potential candidates of appliances (Finite State Machines). We include also the following additional assumptions:

1. The time series $P_{j,t}$ of any detectable appliance should be considered as a recurring pattern with parameters range in a limited area around their expectation value
2. Every state from the set of states defining a FSM is activated exactly one time in each sequence
3. All state changes within the recurring sequences will have all the same fixed order.

4.1 Learning Phase

The most basic power scheme is the ON/OFF model; however, not all loads can be explained by just ON/OFF states. Authors in [4] propose a genetic algorithm

which takes as input the centroids of all switch events §3.3, and generates a static table by calculating the most likely combination of FSM to fit the input data. The use of a genetic algorithm reduces execution time when the number of different clusters exceeds about 25.

Using Matlab, our work follows a process of finding best step changes for hypothetical appliances. The first step (Fig. (17)) is to select clusters that represent all possible states of a load according to binary vector u. For N_c clusters, at most 2^{N_c} possible binomial combinations of clusters can be produced, compared to $N_c!$ permutations. Or, one could represent this notation as

$$Z_j = [c_{j,1}P(C_1);\ c_{j,2}P(C_2);\ c_{j,3}P(C_3);\dots\ ;c_{j,N_C}P(C_{N_C})] \tag{10}$$

where

$$c_{j,r} \in \{0,1\} \quad \forall j,r : r = 1,\dots,N_c \tag{11}$$

Altering all $c_{j,r}$ with zero or one, a matrix with all possible binomial combinations of power clusters $P(C_i)$ is produced. This matrix represents ALL possible power states of hypothetical appliances, and corresponds to all possible appliances. Because the number of combinations becomes huge when N_C is large, it is impossible to examine all possible combinations. Thus, a genetic algorithm is utilised to select only promising combinations of clusters by understanding which of all $P(C_i)$ variations may correspond to actual appliances.

As $|N_C|$ increments (typically $\{15,\dots,20\}$), the size of associated binary table containing all (10),(11), grows exponentially:

$$M = \begin{bmatrix} C_{1,1} & \cdots & C_{1,N_C} \\ \vdots & \ddots & \vdots \\ C_{j,1} & \cdots & C_{j,N_C} \end{bmatrix} \Bigg\downarrow 2^{N_C} \text{ combinations} \tag{12}$$

For this purpose, we used a predefined function from Matlab Central Exchange [22], which produces matrix similar to the binary-coded decimal (BCD) table, minimising size and process time to get all permutations.

Next, in order to decide on the best rows of M, (denoting the best variations of step changes), we calculate the following qualities for each row of $\underline{M}$:

$$Qv_j = \gamma_1 Q_{v_j}^{(1)} + \gamma_2 Q_{v_j}^{(2)} + \gamma_3 Q_{v_j}^{(3)} \tag{13}$$

$$Qv_j^{(1)} = \frac{|\sum_{r=1}^{N_C} c_{j,r} P(C_r)|}{\max_{c_{j,r}=1} (\{|c_{j,r} P(C_r)|\})} \tag{14}$$

$$Qv_j^{(2)} = \frac{|\sum_{r=1}^{N_C} h(C_r) c_{j,r} P(C_r)|}{\max_{c_{j,r}=1} (\{|c_{j,r} P(C_r)|\})} \tag{15}$$

$$Qv_j^{(3)} = \frac{1}{N_C} |\sum_{r=1}^{N_C} c_{j,r}| \tag{16}$$

$$h(C_r) = \frac{H(C_r)}{\sum_{r=1}^{N_C} H(C_r)} \tag{17}$$

All weightings γ_1, γ_2, γ_3 allow user to optimise/configure the quality function that will create a set of possible FSM possible appliances. Eq. (14) evaluates the congruence of all power values, relative to the greatest absolute value of the corresponding combination (with $c_{j,r} \neq 0$), resulting in zero for an optimal appliance. Eq. (15) also adds a term of assessing small numbers of clusters combined, to create an appliance better than greater ones, and is highly related to the frequency. Finally, $H(C_r)$ denotes the number of elements clustered in each C_r, addressing the total number of different states.

In order to analyse each quality, and extract best rows, the genetic algorithm repeatedly evaluates results based on how close the sum of power changes is to zero. This is done because the state transitions of any load should start and end with an off state (see example in §4.2).

For each row, the algorithm calculates the quality of individual rows, and eliminates 50% individuals with lower quality. Then, creates 50% new individuals from the remaining rows, and recalculates each Q_{v_j}. If the algorithm decides (based on new qualities) that this set of rows infers better solution, it saves this set of rows as best solution, else restores previous set of rows.

In Fig. (17) the block diagram of the described genetic algorithm is shown, which assigns cendroids of clusters to Finite State Machines. As the number of #rows increases exponentially, a predefined threshold ϱ sets a limit value of maximum #rows which will be stored and processed by the genetic algorithm.

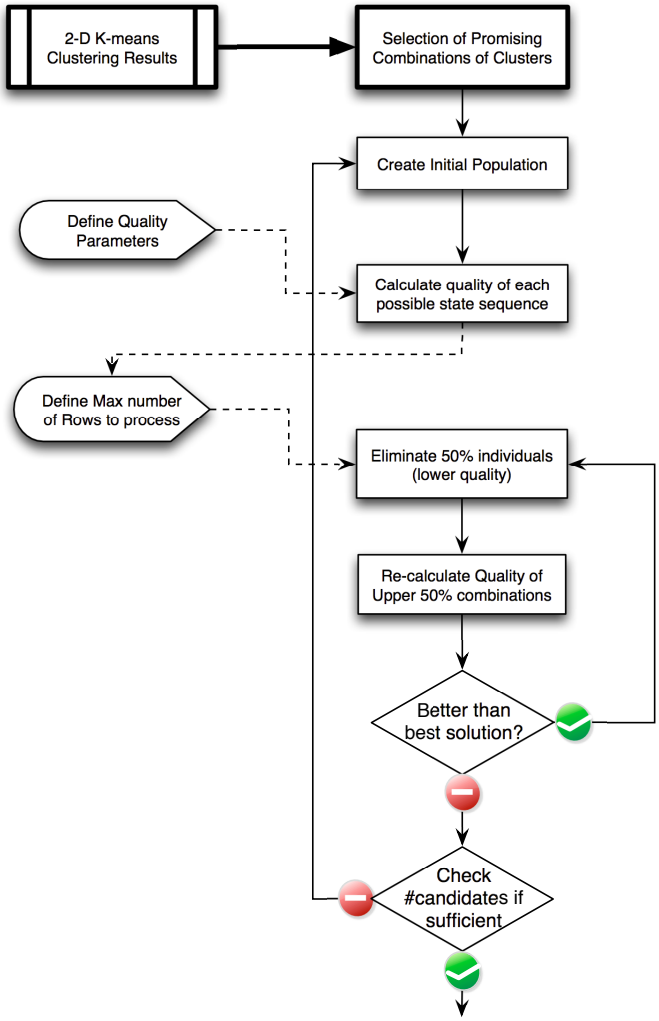


Figure 17: Block diagram using K-means centroids results to build the initial FSM population.

4.2 Initialising FSM tables

The overall target is to create valid sequences of power step events, which may represent any appliances that operated in a residence. Until now, we have described a process of training, where power clusters $P(C_i)$ are created, representing all ΔP_i of strong detectable trace events.

One critical step for the overall process, is to determine which of the FSM static table rows are valid, and discard from the total population any disqualified FSM sequences. For this purpose, a new optimisation process is initialised.

After initialisation of FSM static table, consider a set of cendroid clusters $\{C1; C2; C4\}$ that may represent a hypothetical appliance. In order to successfully find the best sequence path, we follow an example derived from [4]. For a given

set of $P(C_i)$ cendroids:

$$P(C_i) : \{P(C_1) = +50W; P(C_3) = +100W; P(C_4) = -150W\}$$

then all possible step permutations $k! = 6$ for a hypothetical device are the following:

Table 3: Variations of sequence patterns

	ΔP_i Series (Watt)			Power Step Transitions P0 $\rightarrow$ P1 $\rightarrow$ P2 $\rightarrow$ P3	Valid ?
	$P(C_1)$	$P(C_2)$	$P(C_4)$		
Z_1	+50	+100	-150	0 $\nearrow$ 50 $\nearrow$ 150 $\searrow$ 0	✔
Z_2	+50	-150	+100	0 $\nearrow$ 50 $\searrow$ -100 $\nearrow$ 0	✗
Z_3	+100	+50	-150	0 $\nearrow$ 100 $\nearrow$ 150 $\searrow$ 0	✔
Z_4	+100	-150	+50	0 $\nearrow$ 100 $\searrow$ -50 $\nearrow$ 0	✗
Z_5	-150	+100	+50	0 $\searrow$ -150 $\nearrow$ -50 $\nearrow$ 0	✗
Z_6	-150	+50	+100	0 $\searrow$ -150 $\nearrow$ -100 $\nearrow$ 0	✗

Note that each centroid corresponds to a value $\Delta P_i = P_i - P_{i-1}$. Thus, each value represents the power difference between two active power detected values. Also, given that $P_t \geq 0 \quad \forall t$, only $\langle Z_1 \rangle, \langle Z_3 \rangle$ can be valid, while $\langle Z_2, Z_4, Z_5, Z_6 \rangle$ denote a negative power value in the middle or the initial step of their state transitions. Then, the best variation will be handled as the finite state machine to model the appliance, the other one will be discarded.

To easily perceive the implications of the above, a basic rule is that the algorithm excludes variations that result to 0W in the middle of any state transition, or that do not come back to 0W in their last transition. Also, it is evident that state transitions that start with negative power, are also excluded. After creating a valid variation Z_1, all events S_i referenced by the corresponding clusters C1, C3, C4 are assigned to this finite state machine and sorted by time in ascending manner.

The above optimisation phase, filters out any invalid rows similar to the previous example. Then, the algorithm searches again for the best variations from the remaining rows, based on the frequency with which each sequence pattern occurs in the observed switch events, and respectively, the quality criterion (ref. eq. (13)).

4.3 Finding frequencies of occuring patterns

Again, all valid rows are examined, and for each one, the corresponding ΔP are retrieved. The scope of this process is to count relevant frequency that each combination (each distinct row), appears within the total power ΔP series (i.e. select

best variation from $\langle Z_1 \rangle, \langle Z_3 \rangle$). This is achieved using timestamp information stored in every object Q_i (see eq. (5)) and running algorithm (4). In particular, for the above example where $\langle Z_1 \rangle, \langle Z_3 \rangle$ are valid, all mapped events of C_1, C_2 and C_4 clusters, are sorted by time in ascending manner and create a subset $S_{j,t} = S_i \in \langle C1 \bigvee C2 \bigvee C4 \rangle$.

Algorithm 4 Finding frequencies of patterns

Input: Valid rows containing $\{P_{C_r}\}$ from FSM static table,
Input: Series of events sorted by Time
Output: $\epsilon \leftarrow$ Occurring Frequency for each row
for each row $r \in$ FSM Table **do**
  $Q_r \leftarrow$ find all ΔP that are mapped to each cendroid $P(C_r) \in r$
  sort Q_r by time reference
  $\epsilon \leftarrow$ #count distinct occurrences of r in all Q events (by ascending time order).
end for

Eventually, rows of state transitions with frequency $\epsilon \geq 1$ are saved, and sorted by frequency order.The rest of the population, i.e. transitions that do not occur in any of the power trace defined by all ΔP_t, are discarded. Eventually, a new FSM table is recreated, and a new process of finding overlappings of clusters is initialised (optimisation phase).

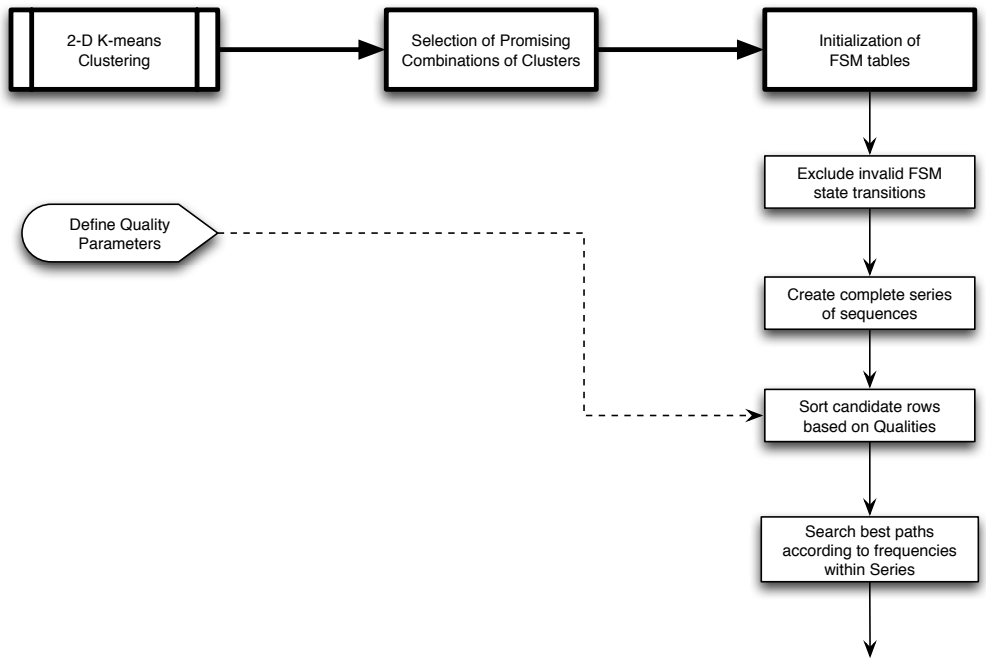


Figure 18: Block diagram of initialization process of FSM population

4.4 Optimizing FSM static tables

The above procedures result in building valid finite states machines, corresponding to each row of $\underline{M}$. However, some clusters can be distributed to different loads simultaneously, when the number of ones in any column of $\underline{M}$ exceeds one $\big($remember that each $c_{j,r} \in \{0, 1\}$ designates contribution of each $P(C_r)\big)$.

The optimisation phase, examines overlaps for each column, and if overlaps exist, it selects the best row based on the quality value of the best path for each overlapping appliance. In general, an overlap occurs when a power cluster P_{C_i} exists in more than one row. Each row, represents a possible appliance, and, based on the fact that power load signatures of each device cannot be identical, we assume that these overlappings are not valid; thus they must be rejected.

This procedure is repeated until all overlappings are solved and the algorithm has assembled the best-fit static table. Once the static table is returned, the learning phase is complete. An overall block diagram of the learning phase is shown in Fig. (19).

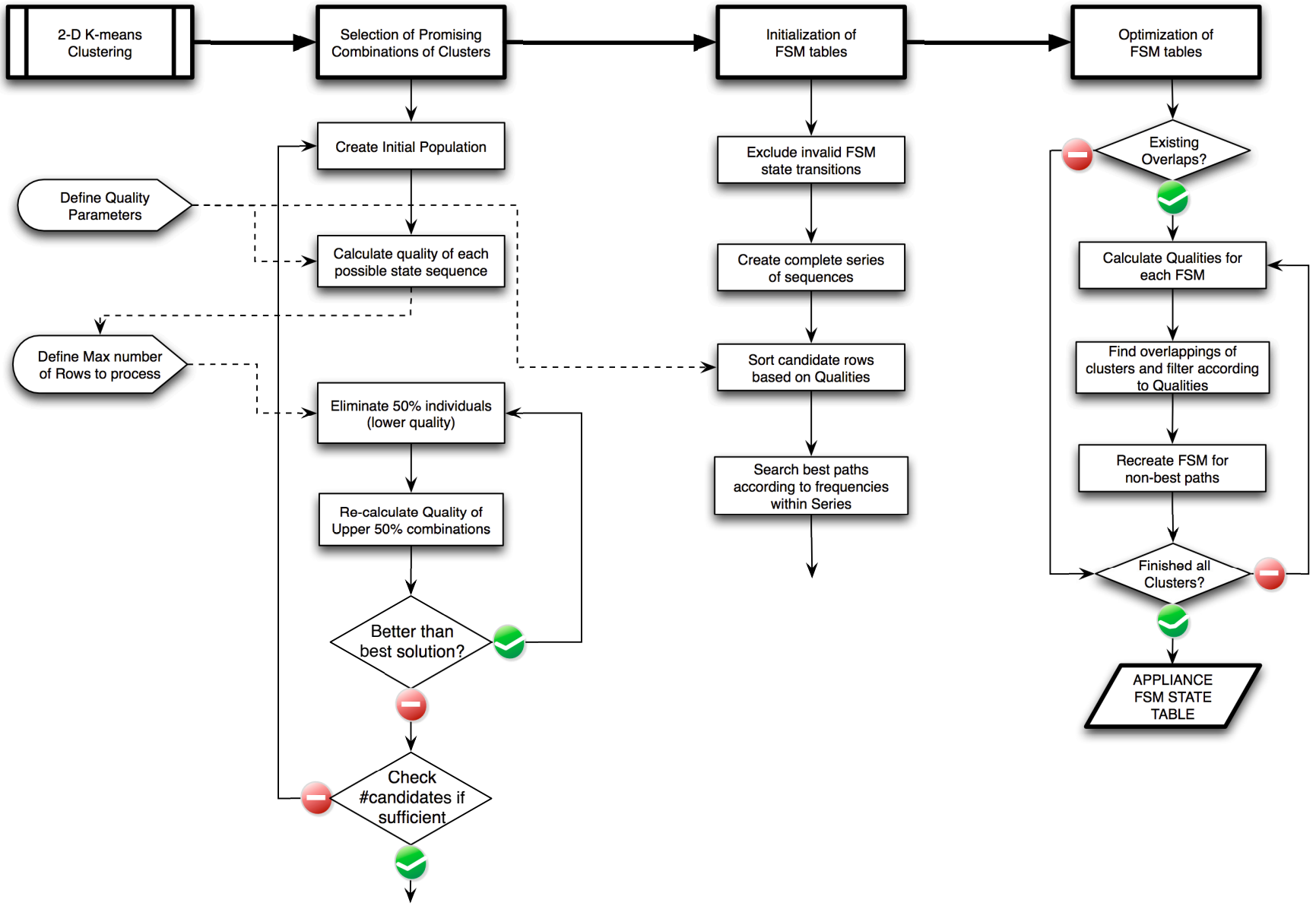


Figure 19: Block Diagram of Genetic Algorithm for Building Appliance Static Table

5 Detection Phase - Appliance Monitoring

Smart Grid technology relies on a process of improving energy consumption efficiency, by real time management of power flows and providing a bi-directional metering needed to compensate local producers of power. It consists of an integration of the traditional electrical power network to modern ICTs. The objectives of this innovation, is to yield a system capable of self-healing and self-organising. The latter is intended to provide proficiency and resilience to ageing critical infrastructure. It integrates the physical electrical network components with the logical cyber ICTs.

During peak hours, it is important for power suppliers to identify heavy consuming appliances, and distribute power load accurately, or even change tariffs according to appliance usage behavior. The smart grid technology allows for systematic communication between suppliers (their energy price) and consumers (their willingness-to-pay), and permits both the suppliers and the consumers to be more flexible and sophisticated in their operational strategies.

Demand response support enables generators and loads to interact in an automated fashion in real time, coordinating demand to flatten spikes. For this project, we implement a detection method, which runs a modified knapsack algorithm on each edge event. Having only one sensor installed on the main electric entrance of the house, we show that it is possible, to identify target appliances and find the electric characteristics.

The detection process uses the Appliance Static Table $\underline{M}$ that was created after optimisation phase (§4.4), and proceeds to an incremental analysis. Given a random power signal, we use a modified knapsack algorithm to search for a combination of loads, whose sum power is maximised under the constraint that the total power is less or equal to the current observed power values.

5.1 The Knapsack Problem

Knapsack is a well-known problem in combinatorial optimisation: Given a set of items, each with a weight and a value, it is possible to find the maximum number of each item to include in a collection, such that the total weight is less than or equal to a given limit threshold, and the total value is as large as possible. For this project, using Dynamic Programming, a modified knapsack problem solution is implemented, in order to compute any appliances that are responsible for the overall power signal collected.

Given n power states, the naive knapsack solution consists of $O(2^n)$ com-

plexity across all loads in the static state table. In contrast, authors in [5] propose '0-1 knapsack problem' solution, which restricts the total number x_i copies of each kind of item to zero or one. This can be formulated as:

$$Maximise : \sum_{i=1}^{n} w_i x_i \quad \text{subject to} \sum_{i=1}^{n} w_i x_i \leq W + T, \qquad x_i \in \{0, 1\} \tag{18}$$

where

1. W is the real power observed in a n edge event.
2. T is the tolerance value, the sum of the standard deviation of all possible loads given the current detected power.
3. $x_{i,j}$ On/Off of state j of appliance i, $x_{i,j} \in 0, 1$.
4. $w_{i,j}$ real power consumption of load i in state j.

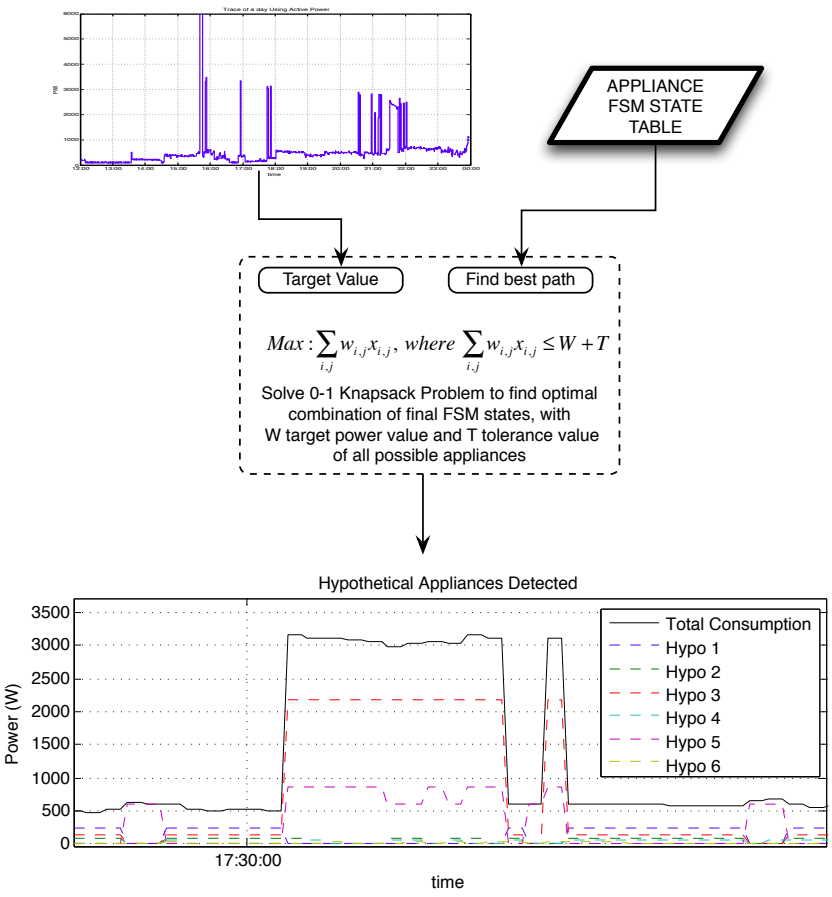


Figure 20: 0-1 Knapsack problem

For the detection process, a random power dataset holding values of active power $P_t \geq 0$ (Watts) is fed to the system. Thus, we decided to transform the appliance static table from ΔP_t values, to active power values, and store it in a new table named $\underline{\hat{M}}$.

Also, we define that all $w_1, w_2, \ldots, w_n$ weight target values are nonnegative and equal to 1 as all power clusters have the same probability to appear. Recursively, we define the value of an optimal solution in terms of solutions to smaller problems. This optimal solution consists of the *maximum* power value detected within a load trace.

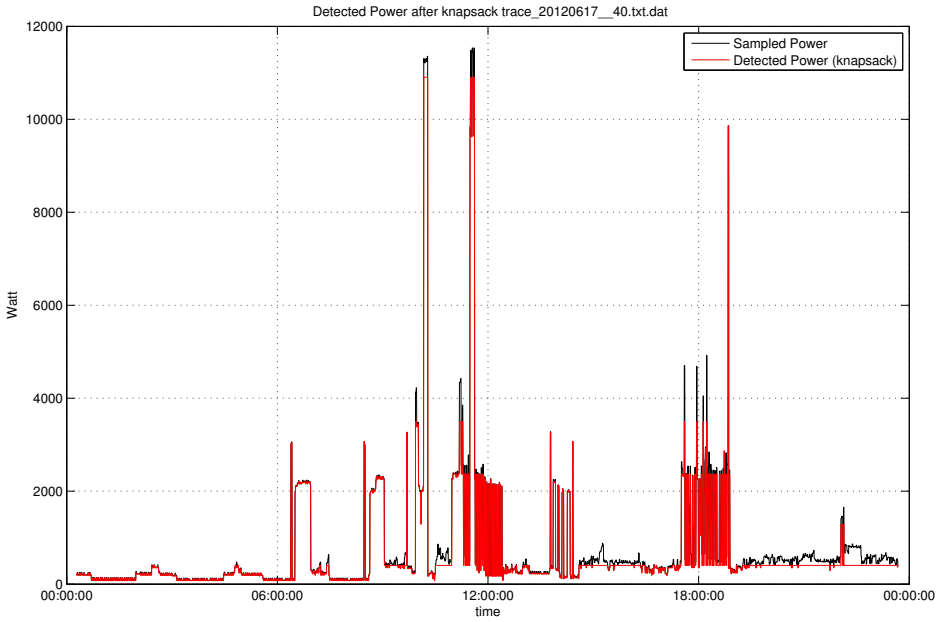


Figure 21: Knapsack results

5.2 Solving '0-1' Knapsack, using Parallel Programing

The total range of hypothetical appliances which are stored in $\underline{\hat{M}}$, is a metric highly linked to one of the initialisation values of K-means. In order to instantiate clustering process, a value of the total number $\#C_i$ possible clusters has to be predefined, which represents the maximum number of possible K-means cendroids that the algorithm will attempt to produce. As this initial value grows, the number of $P(C_r)$ clusters becomes larger, and therefore, more hypothetical appliances are possible to be created. A random number of 8 hypothetical devices, contain at least 16 states of (On/Off) transitions.

In addition, let us consider a regular power trace of a single day, provided by a smart meter. We wish to analyse the daily consumption, and detect appliances, using active power P_t records. It is possible that such dataset, can hold records for every $\sim$1 sec interval (as discussed in §2.2). In that case, our detection algorithm would require to iterate at most through all (60s x 60m x 24h = 86400) values, and for each one, process '0-1' knapsack solution. Given all possible combinations of $P(C_r)$, a maximum number of 2^{N_C} would have to be examined. According to the following:

1. No two loads have the same power signature.
2. Each load has discrete finite states (at most 4).
3. There is an accurate static table containing valid sequences.

then, the total size of $\underline{\hat{M}}$, is possible be reduced to binary combinations that do not hold values corresponding to the same hypothetical device. Based on the fact that all devices have at least two power states (On/Off), this limits solution space to 2^{N_C-1} minimum binomial combinations.

For the detection process, a naive method would be to examine each power value stored in the given dataset, and run knapsack to derive the optimum solution from $\underline{\hat{M}}$. Using algorithm 5, and a preprocess of transforming ΔP_t values to active power, our method *precomputes* all possible sum values for all possible binary combinations of appliances stored in $\underline{\hat{M}}$. This data file, holds every possible solution that the knapsack would produce. As a result, a new table is created, smaller than $\underline{\hat{M}}$, holding appropriate power states, and their sum values. Having precomputed everything, the speed process is increased and time required is subsided, by looping through a new table of all possible solutions, instead of running extensive repetitions of knapsack for each target value W.

Finally, to accelerate even more the process, we used built-in parallel computing for Matlab simulation. The algorithm takes as input $\underline{\hat{M}}$ solution space, and an initial value of the number of X jobs a user wishes to process. Then, $\underline{\hat{M}}$ is split into X parts, which are processed in parallel.

The jobs are finally submitted to the cluster, searching the maximum sum closest to $(W+T)$. When the optimum value for each target is found, all jobs are terminated, and the best binary combination of clusters is returned. This process exports a new signal containing information of which appliances operated during the day, sampling the processed power signal every $\tau = 20$ seconds (algorithm 6).

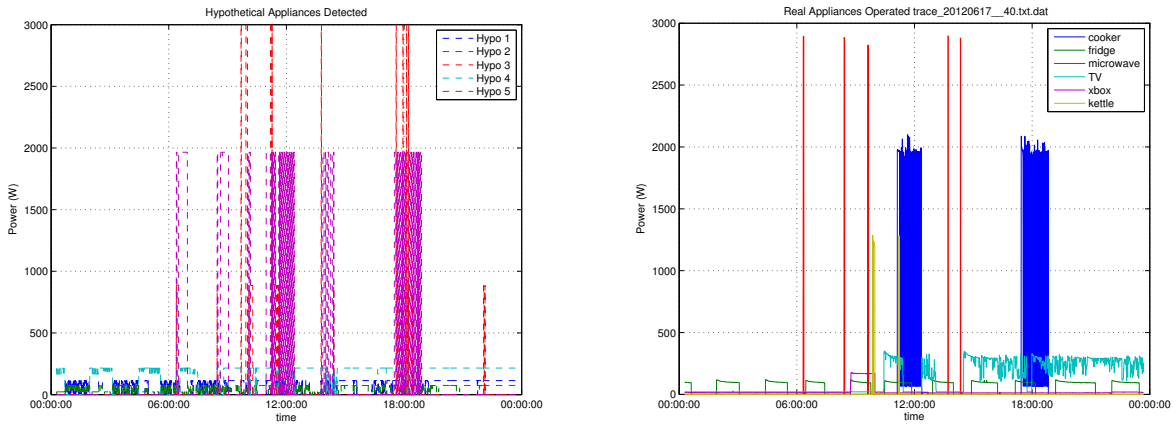
An example of knapsack results, and the relevant detected appliances is shown in Fig. (22).

Algorithm 5 Knapsack Sum Extraction Results

Input: $\underline{M} \leftarrow$ FSM Static Table
Output: $\underline{P_M} \leftarrow$ Table containing [k -perms; sum_{k-perm}]
for all unique elements $P(C_i) \in \underline{M}$ **do**
  Create set $S : S = \langle P(C_1), \ldots, P(C_{N_C}) \rangle$
  Find k-permutations $\forall |N_C| : \sum_{0 \leq n \leq N_C} \binom{N_C}{n} = 2^{N_C}$
  Remove k-ith permutation containing zero elements
  Remove k-th permutation containing $P(C_i)$ that belong to same appliance
  for each k-ith permutation **do**
    Compute $\sigma_i \leftarrow Sum$ all distinct elements in k
  end for
end for

Algorithm 6 Modified DP 0/1 Knapsack

Input: $W \leftarrow$ Power trace to be analysed
Input: $\tau \leftarrow$ Time sampling intervals, $\underline{P_M}$ table
Input: jobs $\leftarrow$ # processors
Output: $P(C_i)$ best clusters solution
for all $w_\tau \in W$ **do**
  $\mu_j \leftarrow$ Split $\underline{P_M}$ $/\#N_{jobs}$
  for each $job_i = \{1 : \#N_{jobs}\}$ **do**
    Process each portion of $\underline{P_M}$
    Find c_i position from $\underline{P_M} : \max \underline{P_{M_i}} \leq w_\tau + T$
  end for
  output all clusters in $[c_i]$
end for



(i) Detected Devices

(ii) Real Devices

Figure 22: Knapsack results

6 Detection Results and Analysis

In [20] a load signature is described as 'the unique consumption pattern intrinsic to each individual electrical appliance/piece of equipment'; a characteristic which all devices distinctly possess. Each electrical appliance holds unique features such as voltage, current, and power measurements. Thus, load signatures can reveal patterns of electric loads, which may then be used in a variety of innovative applications.

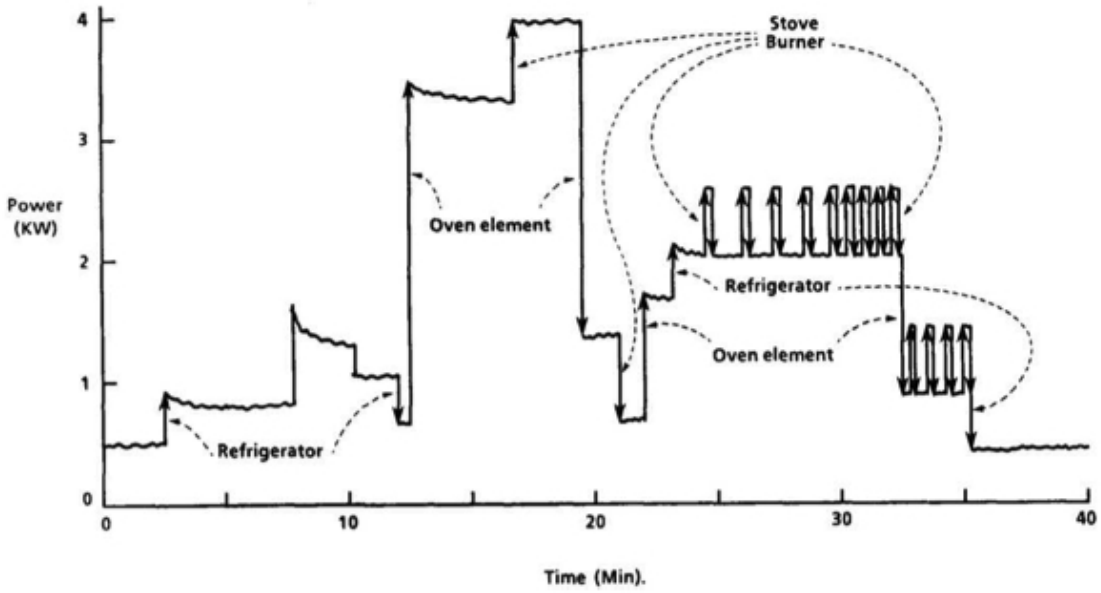


Figure 23: Example of various load signatures

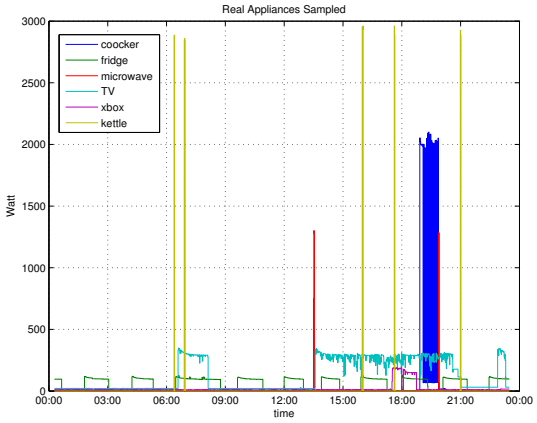
The scope of the detection process, is to analyse through edge detection, the signal of an aggregated power trace (provided by a smart meter installed on the main supply power-line), detect appliance load signatures, and extract information about operation time usage of each device. According to classification method in [25] most of the household devices are resistive appliances.

In general, these can be modelled using steady-state signatures, which have the following main advantages:

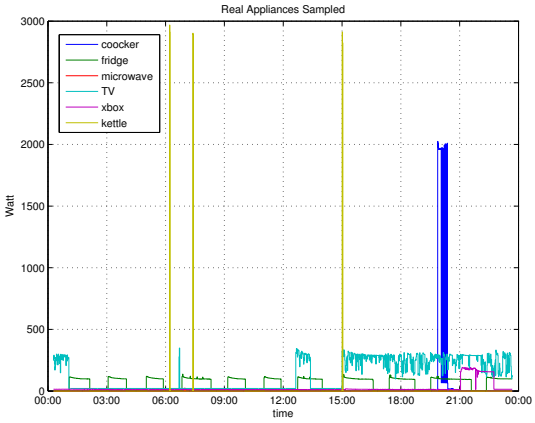
i. they provide a continuous indication of the appliance's operating state, making it much easier to detect the change;
ii. the sum of the power changes in any cycle of state transitions is zero;
iii. they are additive when two or more appliances are activated.

Within a day, most of the referred devices are probably randomly operated, according to household needs. In order to understand and discover recurring

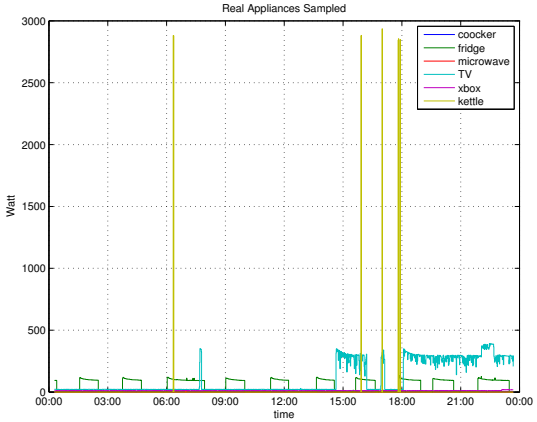
patterns of those devices, figures below (Fig. (24)) represent 4 random days of a typical '3eHouse'. Active power (Watt) is sampled to represent each individual appliance with corresponding edge events (ON state):



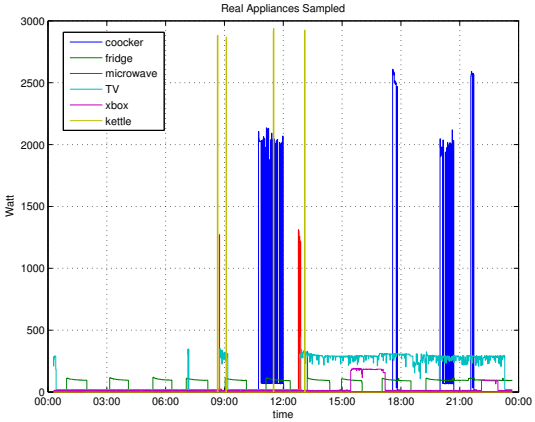
(i) Date: 2012-06-01



(ii) Date: 2012-06-03



(iii) Date: 2012-06-09



(iv) Date: 2012-06-07

Figure 24: Individual Appliances Operated

Through optical observation, the above figures revealed that corresponding ON states are the following: Later, having built FSM static tables for any possible device, an evaluation process is described, which will compare the results of the detection phase compared to the following 6 pre-specified devices:

1. TV ← 316 Watt
2. Fridge ← 112 Watt
3. X-Box ← 187 Watt
4. Kettle ← 2850 Watt
5. Cooker ← 2025 Watt
6. Microwave ← 1132 Watt

6.1 Edge Detection

In [6] authors propose a method of supplying a library of appliances, and match the results of the detection process with the real data. This process involves previous information about the appliance classes, duty cycles and other individual power consumption levels. In contrast, the learning phase which was implemented for this project, does not require previous knowledge of timestamp, consumption or power load signatures for any of the appliances. The edge detection algorithm is dependent exclusively upon the static appliance FSM table that was built during learning phase. Through knapsack detection process, decides for the best paths resulting to occurring appliances (see alg. 6).

In order to initiate the tests, randomly selected days are extracted to train the classifier and build corresponding FSM static tables. For each day, the classifier is trained, using the following parameters to process series of detection events (normalisation is also applied prior to learning phase):

1. $\delta = 10$ Watt (ref. alg. (2) and eq. (4))
2. $\Delta T = 60$ seconds (ref. alg. 3)

A simple example of detection process is shown in Figure (26), for which the corresponding classifier was build upon power trace of day '2012-05-21'. Running edge detector for days '2012-06-01', '2012-06-17', it is evident that major consumer appliances such as fridge, TV, kettle and microwave produced detectable edge events, corresponding to accurate load signatures.

Occasionally, residents of a random household use multiple devices. The power graph demonstrated in Fig. (25), describes (blue colour) a typical signal of $\sum_{i=1}^{N_k} P_t$, where N_k is the number of known appliances as described in Fig. (7i). In contrast (red colour), the power trace represents all possible (probably unmetered) appliances. For training and testing purposes, the overall detection process, is applied only on top of the aggregate signal (blue line).

Also, similar results using different day trace for training ('2012-5-12') and testing ('2012-06-03') purposes are shown in Fig. (26). Through optical comparison between real and detected results, it is evident that the detection process successfully detected most of the devices. In both examples, 5 out of 6 devices could be identified, with the most powerful edge events to be more distinguishable.

In addition, example of edge detection is shown in Figures (27, 28) where knapsack runs over aggregated data of active power P_t derived from a test house during day '2012-05-25'. The closer the knapsack results match with the actual power signal, then the more accurate results are expected to be produced. Based

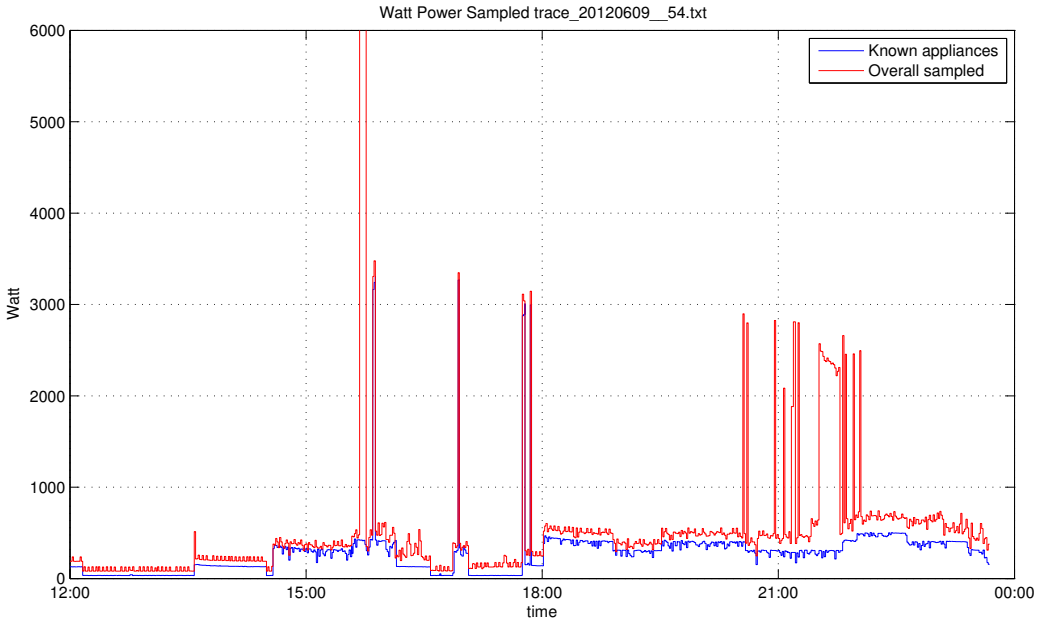


Figure 25: An example of un-metered power signal, compared to overall active power measured during a random day.

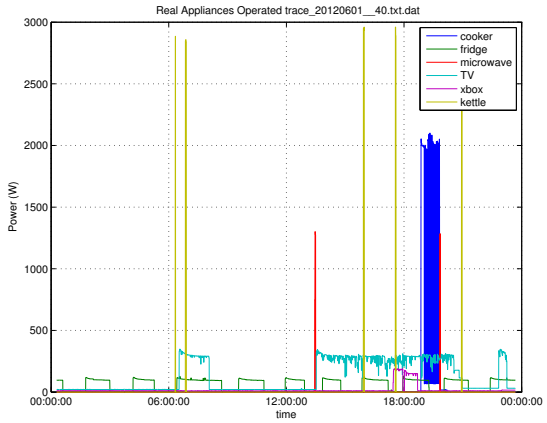
on the results of knapsack, the algorithm has information regarding which FSM static paths contributed to the overall sum result of knapsack detection, and user is able to derive information regarding individual devices that have been operating at any given time, even if they could possible be simultaneously in use.

6.2 Efficiency and Accuracy Results

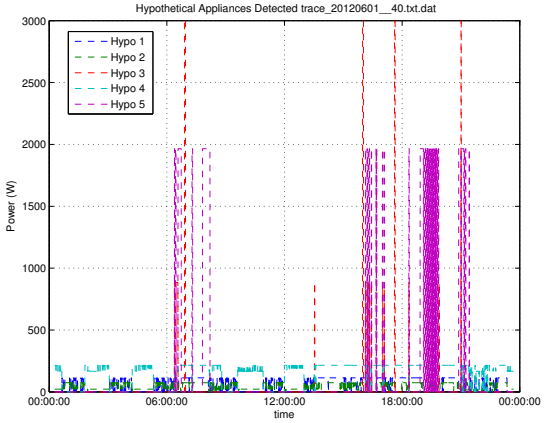
In order to measure accuracy detection performance, it was indispensable to compare the detected appliance signals, with those collected from real devices. To achieve this, a cross validation method was applied; this method, also used in machine learning applications, is exploited to determine model parameters, following a simple process; the dataset is randomly partitioned into N sets of equal size, and the detection process is run N times. Each time, a different one of the N sets is deemed as a test set, and the classifier is trained on the remaining $N-1$ sets.

For the experiments, we used data derived from a '3eHouse' for 5 days. In order to have an estimate of the accuracy of the classifier, we decided to do a 5-fold cross validation as follows:

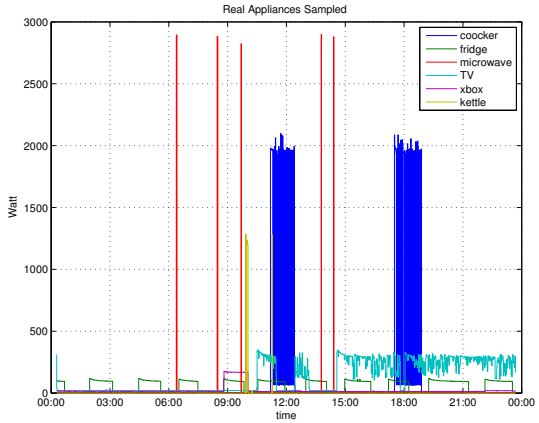
1. Processed data derived from 5 random days.
2. Normalised power trace records.
3. Trained the classifier using 1 power load (24H) set each time.



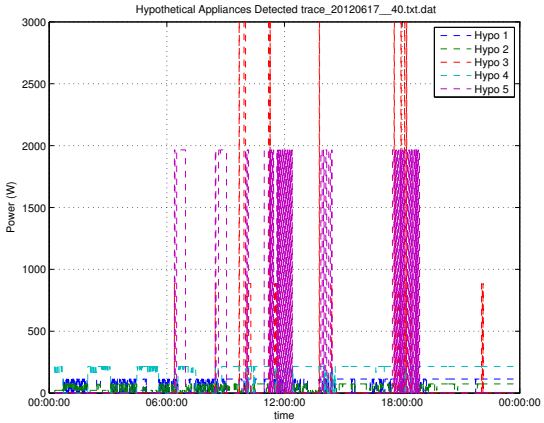
(i) Real appliances {2012-06-01}



(ii) Detected appliances {2012-06-01}



(iii) Real appliances {2012-06-17}



(iv) Detected appliances {2012-06-17}

Figure 26: Results of detection using FSM static table derived from '2012-05-21'

4. Run Detection algorithm over the rest of 4 power load sets.
5. Compared correlation within hypothetical and real appliances.
6. Repeat for all $n-1$ sets.

For the experiments, ZigBee smart meters were configured to sample 5-6 major appliances, installed by the electricity engineers among various '3eHouses'. Sampling at 20 secs, we created $\hat{M}_{hypo}$ and $\hat{M}_{real}$ matrices, containing power series of hypothetical and real devices. In order to compare and validate the matrices, two comparable correlation metrics are applied, trying to match any detected appliance.

Pearson product-moment correlation coefficient PPC is a product-moment coefficient that measures the dependence between two quantities. It is obtained by dividing the covariance of the two variables, by the product of their standard deviations. In our case, X, Y, will be the values which have been stored

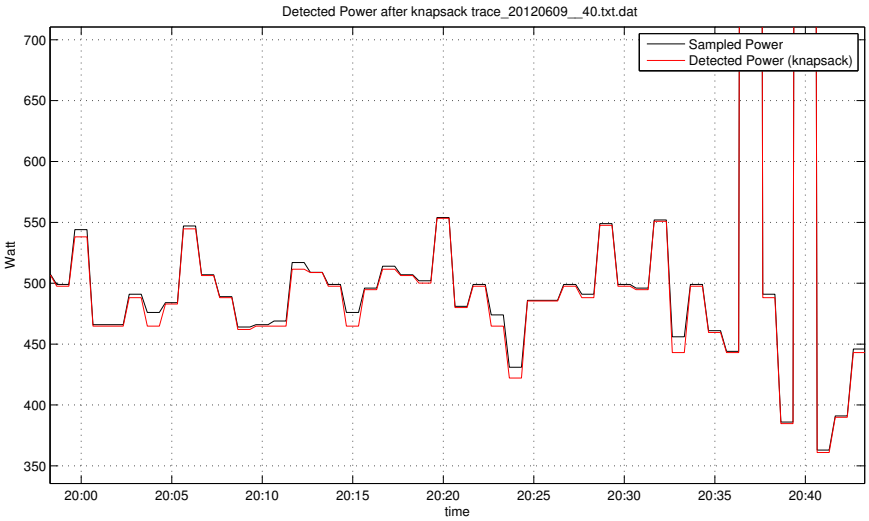


Figure 27: Edge detection using modified knapsack, day '2012-06-09'

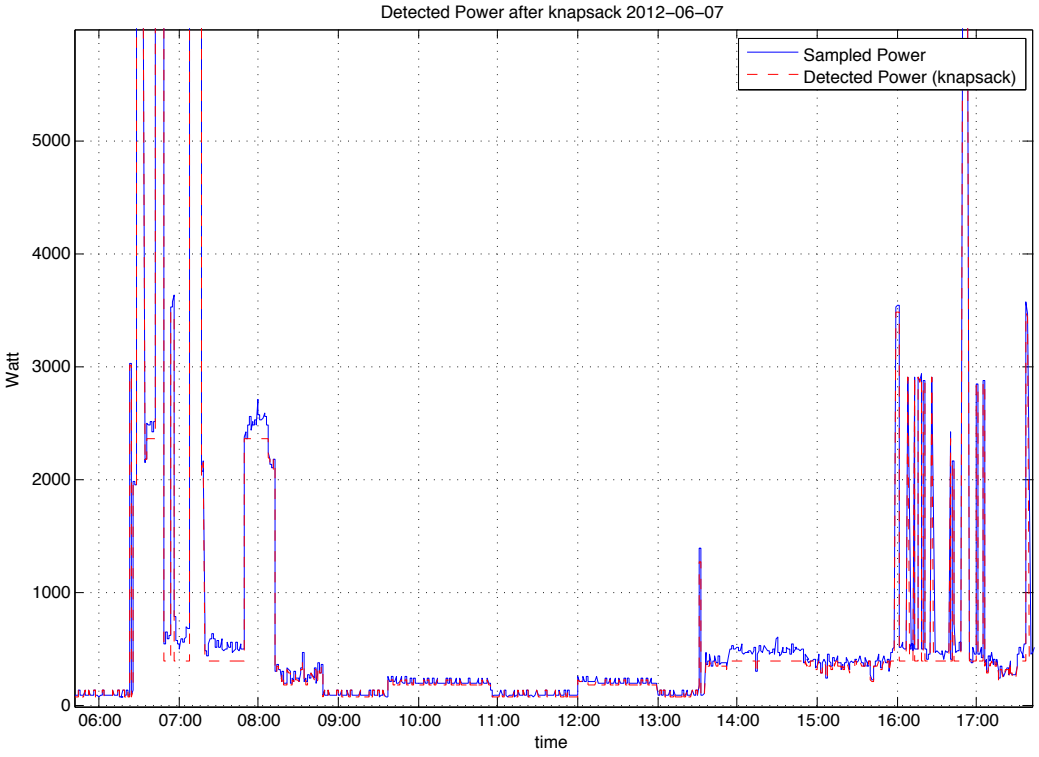


Figure 28: Edge detection, day '2012-06-07'

in each vector of $\hat{M}_{hypo}$ and $\hat{M}_{real}$ matrices.

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y} \tag{19}$$

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{20}$$

For each column, pairwise comparison is processed, and equivalent ρ values are examined. The positive sign reveals that the variables increase and decrease together. The large magnitude (close to 1) expose existence of strong positive relationship between the two variables.

Spearman's rank correlation coefficient The Spearman correlation is defined similar to the PPC method, producing ranked variables. It is a metric to compute ranks x_i, y_i that correspond to the average of the positions of X_i, Y_i vectors, in the ascending order of the values. The sign of Spearman's correlation indicates the direction of association between X and Y variables. As in PPC, a Spearman correlation result close to zero indicates that there is no tendency, while a correlation coefficient close to 1 reveals perfectly monotonically relationship. For a sample of size n, the n raw scores X_i, Y_i are converted to ranks x_i, y_i and r is computed as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{21}$$

Using traces from 5 different days, and examining results with $r \geq 0.2$ minimum rank coefficient, we attempt to investigate strong correlations of appliances, filtering only significantly different from zero coefficients. For this purpose, *corr* Matlab function is used, and results are shown in Figure 29.

The results show the following regarding the classification performance:

i. Detected more than 2 devices with ∼90% accuracy.
ii. Detected more than 3 devices with ∼80% accuracy.
iii. Overall, a success rate of ∼55% is achieved, corresponding to successful match of major consuming devices such as fridge, cooker, microwave and kettle (see Fig. (30)).

Dates	20120602	20120609	20120122	20120204	20120220	Success Rate
20120602		3/5 (0.2056-0.7160)	5/6 (0.2130-0.5972)	4/5 (0.2879-0.6146)	3/5 (0.2568-0.6498)	70.83%
		3/5 (0.2034-0.6513)	6/6 (0.2513-0.6126)	3/5 (0.2109-0.5465)	3/5 (0.2844-0.7062)	70%
20120609	4/6 (0.2372-0.7689)		3/5 (0.3402-0.6632)	2/5 (0.3740 - 0.6664)	3/4 (0.2758-0.6609)	60.42%
	4/6 (0.2303-0.6688)		2/5 (0.3068-0.5563)	2/5 (0.2556-0.5065)	3/4 (0.3566-0.6483)	55.42%
20120122	4/6 (0.2053-0.7893)	3/5 (0.2115-0.7920)		2/5 (0.2013-0.6457)	3/5 (0.2958-0.8434)	56.67%
	4/6 (0.2141-0.6505)	2/5 (0.3935-0.6384)		2/5 (0.2028-0.4661)	3/5(0.2033-0.8198)	51.67%
20120204	4/6 (0.2102-0.6338)	3/5 (0.4396-0.8141)	3/5 (0.2107-0.5332)		2/4 (0.2979-0.6130)	59.17%
	4/6 (0.2147-0.6442)	2/5 (0.2664-0.7035)	3/5(0.2755-0.4406)		2/4 (0.2898-0.6630)	54.17%
20120220	4/6 (0.2459-0.4534)	2/5 (0.2040-0.5848)	3/5 (0.2108-0.5261)	2/5 (0.2514-0.2640)		51.67%
	4/6 (0.2742-0.5287)	2/5 (0.2026-0.5485)	4/5 (0.2137-0.4304)	2/5 (0.2404-0.4513)		56.67%
					Total Pearson Rank	59.75%
		Pearson (r_min - r_max) coefficient			Total Spearman Rank	57.58%
		Spearman (r_min - r_max) coefficient				

Table 4: Cross Validation on real devices and hypothetical power series derived from detection process.

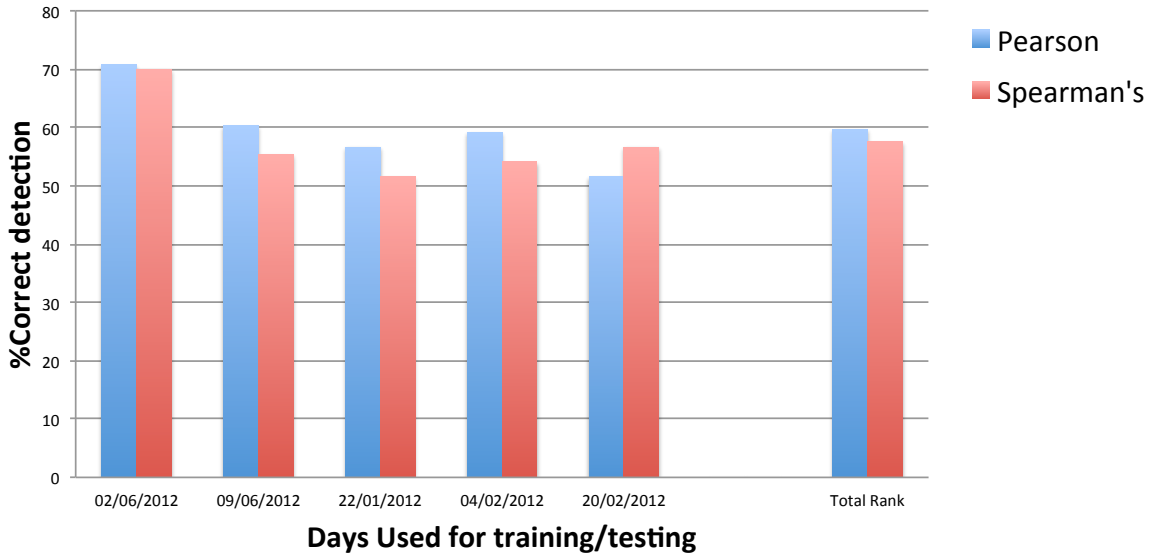


Figure 29: Cross Validation on real devices and hypothetical power series derived from detection process.

iv. Devices that consume over 400W of active power, were easily matched, using $\rho \geq 0.2$ threshold, whereas other devices resulted in smaller ρ results.

In addition, we can observe that for various tests, $\rho_{\max} \geq 0.5$, which implies strong correlations. Also, the square of the correlation coefficient (known as 'coefficient of determination') is considered significant between the observed and model appliances. Values of ρ^2 close to 1 indicate that regression line perfectly fits the data and cases when $\rho^2 \geq 0.5$ denote that the variations of one of the correlated parameters explains at least 50% of the variation of the other.

Regarding timing performance, the tests were run on a 2.8GHz laptop with i7 processor, and MacOS operating system. Approximately, a detection process for a 24H power trace, lasted less approximately than 3 minutes, sampling at sin 20

seconds intervals, using the pre-defined knapsack solution table. Comparable results of model appliances are shown in Fig. (30). In some cases, the detection produced even more hypothetical devices than already known sampled devices. Generally, for a more reliable accuracy measurement, it is eminent to compare only strong devices that operate, and discard weak devices that operate below 400 Watt. In a recent distributed NIALM implementation of [4], authors imply that their detection process discarded appliances that consumed less than 1000W ([5] §C. dNILM Analysis). More analysis and further improvements are discussed in §8.

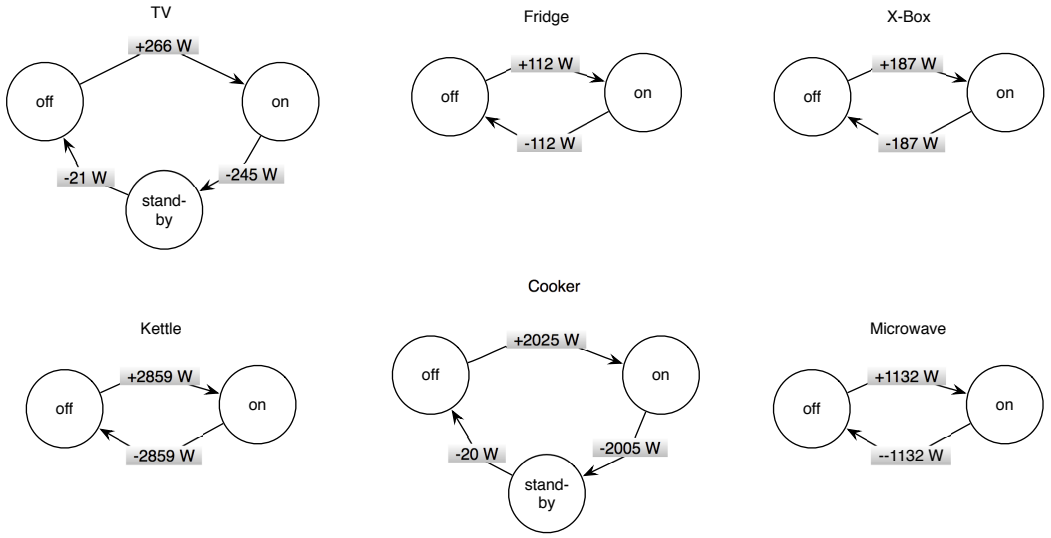


Figure 30: Various appliances and power load states

7 Privacy Enabled Home Energy Management System

As mentioned in §1, one of the aims of this project, comprises a comparison of detection algorithm performance, on protected and unprotected power sets. For this purpose, we utilise an algorithm provided by TRL, named as SimplyCAP. This novel energy preserving idea, offers a patented solution regarding a Home Energy Management System (HEMS) and has been provided by authors of [15]. The basic idea of PeHEMS is based on using a rechargeable battery, which can be charged and discharged [14] according to load volume, a method that can be integrated in a home power and communications network.

Traditional load monitoring techniques can be described as intrusive technologies due to the physical placement of sensors on individual appliances to gather end-use load data. Thus, this poses a long term-intrusion onto the private life and property. The proposed solution, is attempting to hide the load signature of individual devices, and disguise the overall power consumption using a simple battery mixing method.

7.1 HEMS model

Home Energy Management Systems are mainstream security systems, that monitor electricity use. With a smart meter, one can use home monitoring device to see electricity use, usually reported every τ time intervals. HEMS have evolved into home energy management systems by providing visibility in overall energy consumption, through identification and isolation of large load centres such as air conditioners, water heaters etc. Such systems are operating on top of Home Area Networks (HAN), using smart meters and ZigBee or other standardised RF mesh technology (based on IEEE 802.15.4) for HAN deployment.

An example of advanced home power system with smart appliances, plug-in vehicles, power storage systems and automated management, is represented in Fig. (31), where the electric car holds a rechargeable battery, and plays the role of storing power using charge/discharge automated management methods.

A recent research conducted by IMS Research [21], forecasts than "More than 100 Million Smart Meters with Integrated HAN Gateways will be deployed in the next five years, driving HEMS Market past US $ 9 Billion". Also, EU regulations require the deployment of smart meters to most consumers within the next eight years (EU regulation 2007/72/EC).

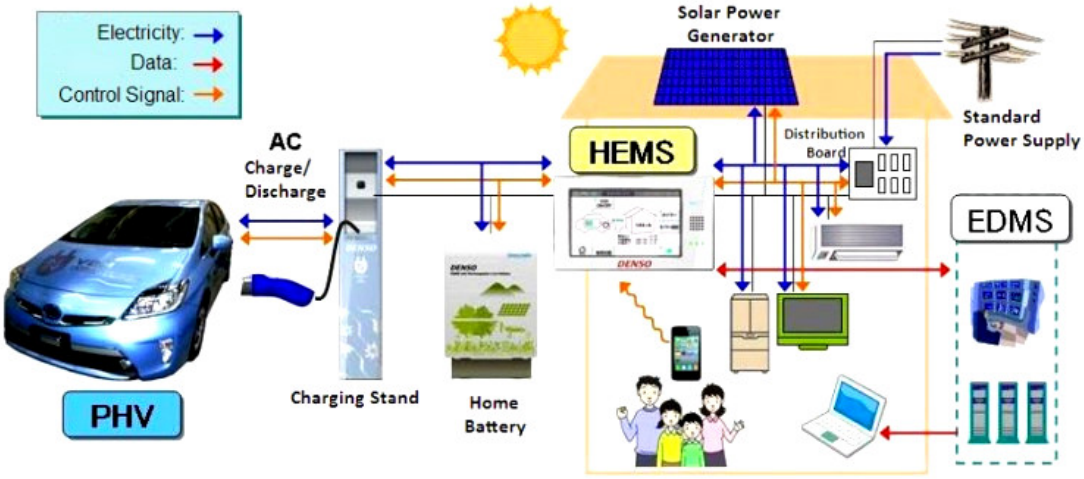


Figure 31: Advanced home power system with smart appliances,

In general, those systems are designed to educate consumers about energy conservation, while improving distribution of the power load provided by utilities. On top of those systems, one can install a LSM (Load Signature Moderator) to detect a privacy thread and respond by configuring power routing, via a power line-ZigBee gateway, to give home energy management systems access to consumption data and other information available.

PeHEMS (Privacy Enabled HEMS and Load Balancing Prototype) offers a privacy level that allows the user to configure privacy parameters. This energy capping system is enforced by triggering appropriate UPS switch on and off commands. The smaller the cap the bigger the privacy level and vice versa. For example, a kettle requires 2KW of power when switched on; while the power router can be configured so that 1kW is supplied from a solar panel, 0.5 kW from a battery and the rest 0.5 kW from the mains electricity supply.

Authors denote a simple battery power mixing moderation model as shown in Fig. (32), to improve efficiency of load shaping signatures, and raise individual security of Home Area Networks.

7.2 The Rechargeable Battery Model

The battery model presented in Fig. (32) uses active power $p_B(t)$, and can be re-charged or discharged within a metering interval Δt. Utilising a battery of capacity C_B, the system is configurable to hide consumption load $p_A(t)$. With the use of the battery, the metered home load becomes $p(t) = p_a(t) - p_b(t)$, and

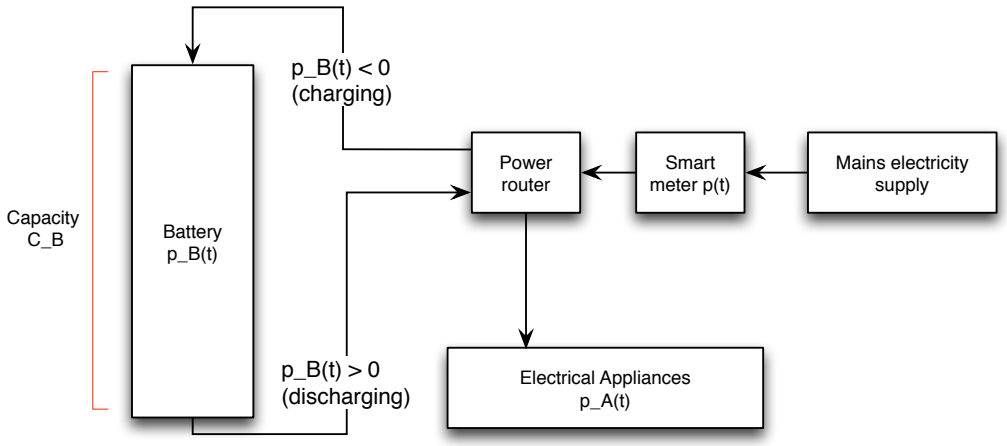


Figure 32: Battery power mixing moderation model

the ultimate target is to disguise $p(t)$ and make it undetectable, i.e. $p_a(t)$ cannot be determined given $p(t)$. Through this idea, energy information leakage can be prevented using appropriate parameters and a simple rechargeable battery.

In particular, the battery must be specified by the following characteristics:

1. It compromises finite capacity B_E (hence, the battery has to maintain its energy by recharging), i.e. $0 \leq \int_0^T p_B(t)\,\mathrm{d}t \leq B_E, \quad \forall T \geq 0$ (assuming that for $t = 0$ the battery is fully charged.
2. The battery has a maximum discharge and recharge power of B_P, i.e. $|p_B(t)| \leq B_p$.

Given the above, ideally, the level of privacy protection should approach the maximum when $p(t) = 0$ for $0 \leq t \leq T - \Delta T$, and $p(T) = \int_0^T p_B(t)\,\mathrm{d}t$, for $T - \Delta t \leq t \leq T$.

The SimplyCap solution, enforces the battery to either discharge or recharge whenever the required load $p_a(t)$ is either larger or smaller (respectively) than the previously metered load $p(t - \Delta t)$. The power and duration of battery charging/discharging cycles are configured to equal the power differences, unless battery limits (B_E, B_P) are reached.

Training and Detection using PeHEMS: The following process, describes the methodology that was followed, in order to train the classifier, and detect any possible devices. Using the following configurations, and a preprocessing of the dataset (see §3, §4), we derived results of protected power loads as described in ([15] §V):

1. Energy data were mapped to 30 minute aggregates, by extrapolating boundary samples.

Algorithm 7 SimplyCap Energy Capping Algorithm

Input: Energy CAP for metering cycle duration, $T = 30$ minutes
Input: Internal measurements every $\tau = 30$ seconds
Input: Allocated battery capacity C and charge/discharge rate P_B
Input: Estimated battery energy loss coefficient $\alpha > 1$
while Metering cycle $i = \lfloor t/T \rfloor$, where t is current time **do**
  for $(t = iT; t < (i+1)T; t = t + \tau)$ **do**
    Measure consumption q for current metering cycle
    Remaining time to end of cycle: $\delta = (i+1)T - t$
    Predict expected consumption Q for current cycle
    Monitor the available battery charge level E_B
    if $Q > CAP$ battery discharging case **then**
      Use battery energy D
      $D = min\big(E_B, P_B\delta, a\,(Q - CAP)\,\big)$
      Factor in energy losses: Provided energy $= D/a$
    else
      Recharge the battery with energy R
      $R = min\big(\,(C - E_B)\,, P_B\delta, (CAP - Q)\,/a\big)$
      Factor in energy losses: Additional energy $= \alpha R$
    end if
  end for
end while

2. A constraint of the daily total energy consumption dropped days with corresponding consumption of less than 5kWh or more than 25kWh.
3. Data sampled every 30 seconds on average, with 20 seconds being the interval sampling rate of ZigBee sampling rate. In particular, days that were not providing intervals within 5 minutes, where also discarded.

In addition, for the rest of the testing purposes, we run SimplyCAP with parameters:

1. CAP = 313.5 Wh.
2. $\alpha = 1.1$ (i.e. a 10% energy loss during either charging or discharging).

A sample of the results of a random day that SimplyCAP was used is shown in Fig. (33), compared with Fig. (34) revealing some major appliances operated during the same day. It is obvious that capping method, achieves sufficient privacy hiding of the derived plane of real power trace. This becomes more evident by comparing Fig. (36iii) and Fig. (36ii), where clearly, the algorithm achieved to reduce information leakage, even when the cooker was On for a sufficient time.

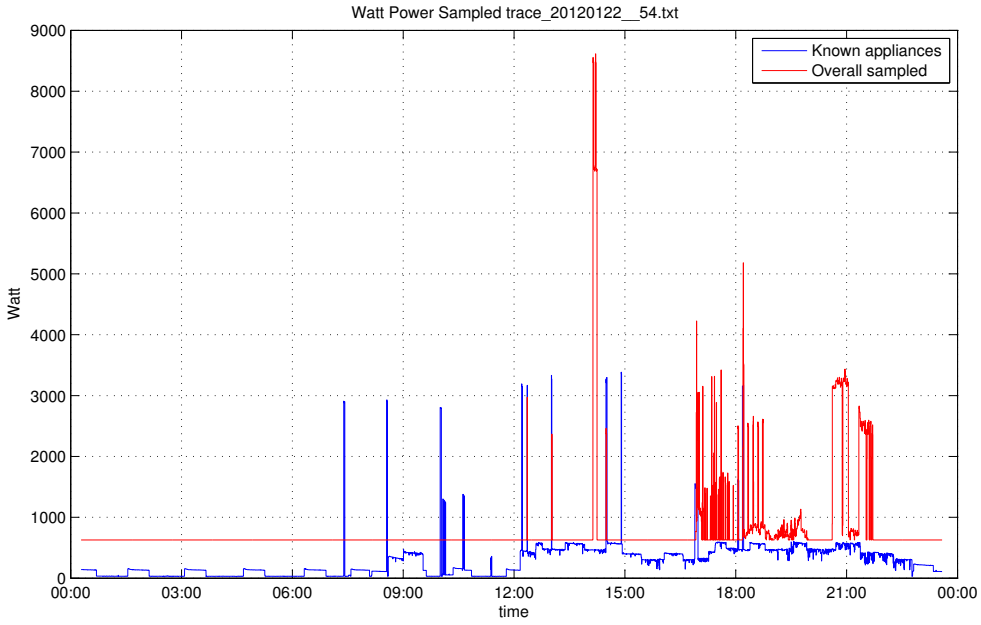


Figure 33: Capping results on '2012-01-22' aggregate

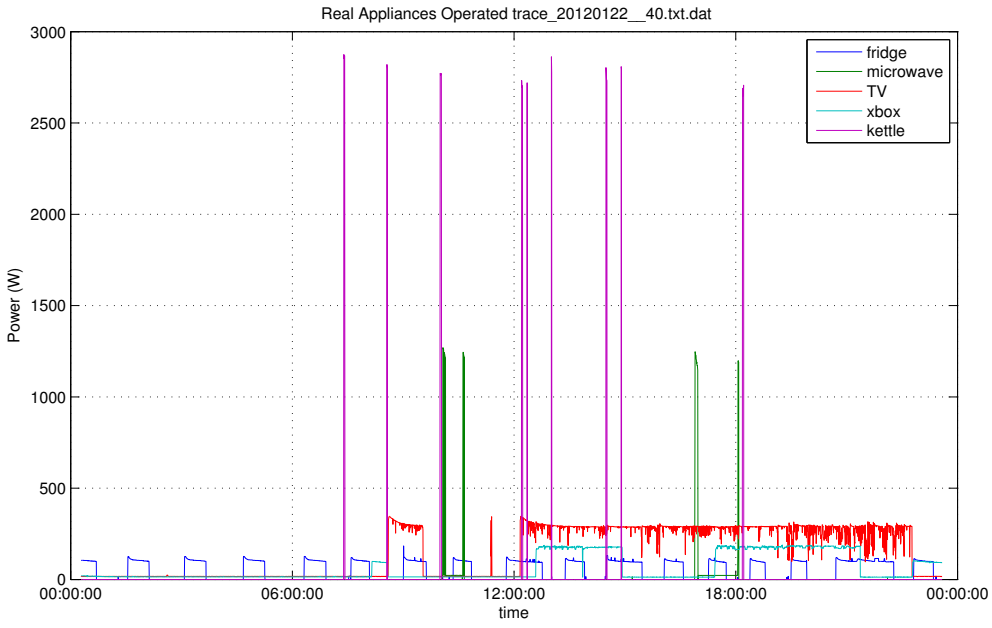


Figure 34: Real appliances operated on '2012-01-22'

7.3 PeHEMS detection results

The scope of the following experiment, is to investigate the effectiveness of SimpleCAP by applying the battery-mixing technique on the identical 5 days used in §6.2 for the previous 5-days cross-validation experiment. In particular, the

static appliance FSM tables extracted from unprotected power trace aggregates (see Fig. (4)), are also used for the Cross Validation test.

Afterwards, having built the FSM appliance models for these days, we wish to detect any appliances derived from protected power signal. For this process, Spearman's and Pearson's correlation coefficients are also computed as shown in Figure 35. Accuracy, as expected, was dropped to $\sim$ 21%, which was anticipated. Though, it achieved to correlate at least 1 device with $\sim$ 53% and at least 2 devices with $\sim$ 33% success rate respectively.

Dates	20120602	20120609	20120122	20120204	20120220	Success Rate
20120602	3/6 (0.2174-0.5661)	1/5 (1.652)	2/5 (0.2548-0.3846)	0	0	22.00%
	2/6 (0.2594-0.4943)	0/5	3/5(0.2105-0.3072)	0	0	19%
20120609	3/6 (0.2196-0.5587)	1/5 (0.1933)	2/5 (0.2189)	0	0	22.00%
	2/6(0.2082-0.4310)	1 (0.1652)	2/5 (0.2570-2955)	0	0	18.67%
20120122	2/6 (0.1109-0.6042)	1/5 0.2489	2/5 (0.1611-0.3975)	0	0(0.0924)	18.67%
	3/6(0.1011-0.4929)	0	2/5 (0.2538-2987)	0	0 (0.1048)	18.00%
20120204	1/5 (0.1663)	2/5 (0.1598-0.3287)	1/5 (0.3704)	0 (0.0538)	0 0.0881	16.00%
	1/5 (0.1578)	1/5 0.2481 0.3680	1/5(0.3075)	0 (0.0417)	0 0.1186	12.00%
20120220	2/5 (0.1829-0.2698)	1/5 (0.1902)	2/5 (0.3142)	2/5 (0.1585-0.3142)	0 (0.0854)	28.00%
	3/5 (0.1501-0.2450)	1/5 (0.3579)	1/5 (0.2891)	3/5 (0.1511-0.2891)	0 (0.1256)	32.00%
					Total Pearson Rank	**21.33%**
		Pearson (r_min - r_max) coefficient			**Total Spearman Rank**	**19.93%**
		Spearman (r_min - r_max) coefficient				

Table 5: Cross Validation of real devices and hypothetical power series derived from PeHEMS capping

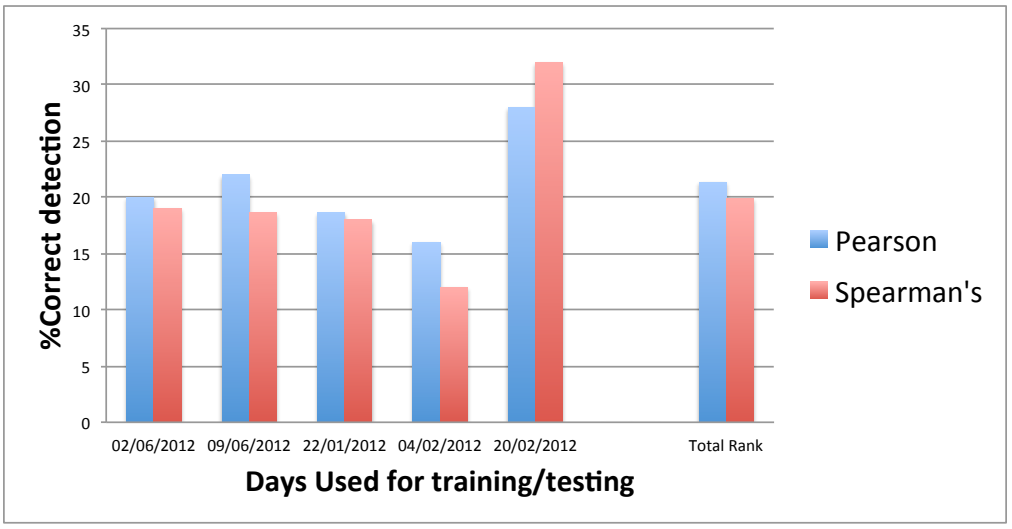
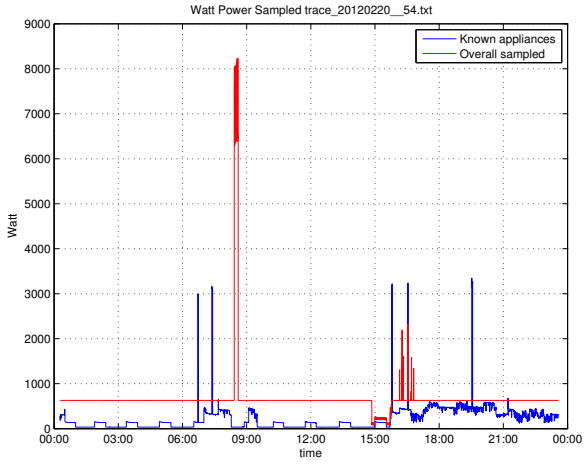
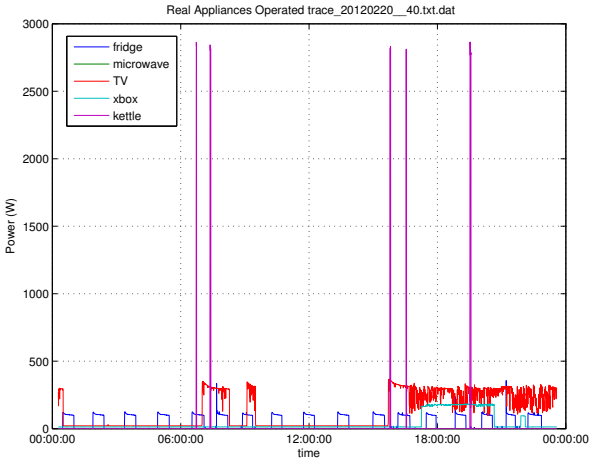


Figure 35: Cross Validation validation on PeHEMS capping datasets.

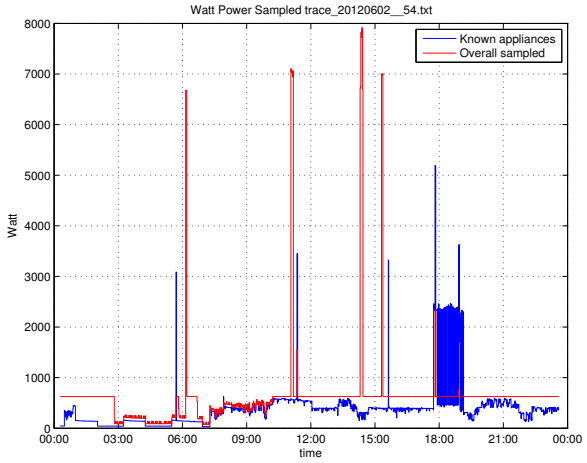
It is obvious that SimplyCAP reduced accuracy of the process, by $\sim$ 30% compared to Table 4 in §6.2. Even with this capping solution, the genetic algorithm achieved in identifying some appliances, as shown in Table 5.



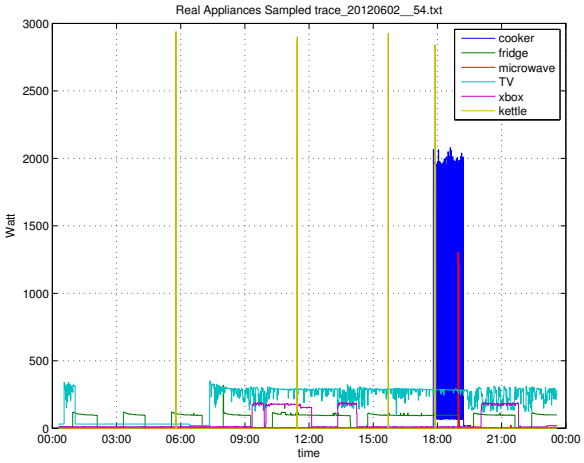
(i) Capping on '2012-02-20' aggregate



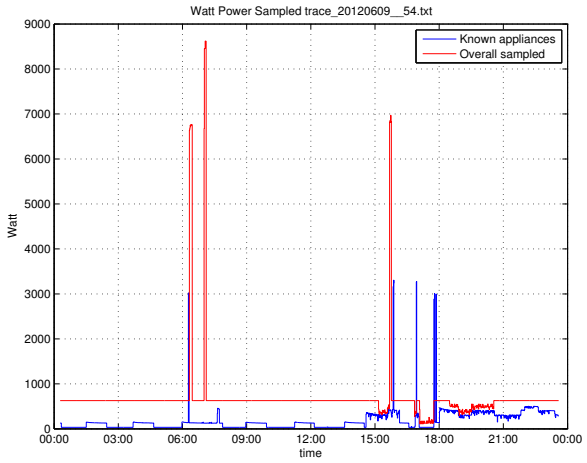
(ii) Real appliances on '2012-02-20'



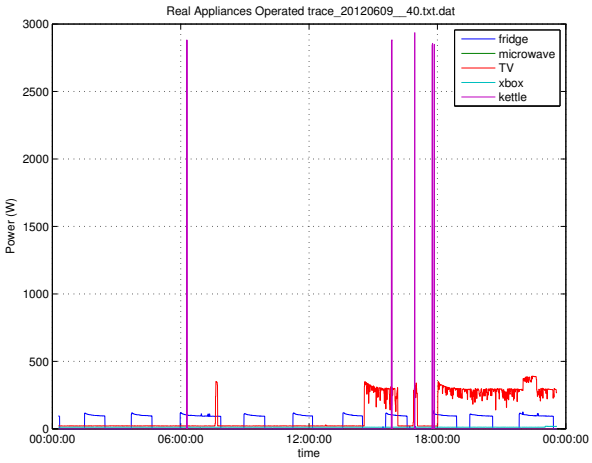
(iii) Capping on '2012-06-02' aggregate



(iv) Real appliances on '2012-06-02'



(v) Capping on '2012-06-09' aggregate



(vi) Real appliances on '2012-06-09'

Figure 36: SimpleCAP results vs Real appliances.

8 Conclusions and Future Work

This project studies a method for analysis and detection of home appliances, and PeHEMS, a home energy management load balancing algorithm, that helps protection of electrical metering privacy. Electricity loads studied in this research were generated from typical household appliances, located in Bristol (UK). Since there has been conducted an extensive research regarding reoccurring patterns of appliances, it was important to create an autonomous method for disaggregating the total load into its major devices.

The method implemented is based on Harts contribution [12], due to the use of only step changes in real power, and shows a promising potential for application in residential buildings. Results prove that the whole house electricity consumption can be disaggregated into its major end-uses, by using a pattern recognition approach where only one single sensor installed on the main electric entrance of the house. Also, all training and testing operations where conducted in terms of daily load profiles. The proposed methods were tested with monitored data from random days within the period of January - June 2012, obtained from local residencies. Datasets were provided by Toshiba TRL, from a trial in Bristol (UK), that is part funded by '3eHouses', and EU FP7 project of which Toshiba and Bristol City Council are major partners.

Due to the complex nature of the genetic algorithm, its running time is supra linear, as the combinatorics involved in generating the possible sequence patterns consume significant processing power. Testing has proven that overall appliance tables for various days, are both distinct and consistently observable over time. In addition, the genetic algorithm offers a variety of parameters be tuned and improve the detection of appliances or to assimilate data sources of different applications. In particular, the contribution of this project is the study of the impact of a number of algorithmic parameters to help develop smarter, i.e. self-evolving and truly autonomous, algorithms. The most significant parameters considered in this study as the following:

- Weightings γ_1, γ_2, γ_3, which allow to configure the quality criterion filtering for generating promising FSM static tables.
- A predefined threshold ϱ which sets a limit value of maximum number of rows which will be stored and processed by the genetic algorithm.
- A number of maximum clusters $\#C_i \geq \#N_c$ with N_c is the number of all real appliances, that a user wishes to create using 2-D K-means clustering. As this number increases, the complexity of the algorithm is increasing.
- A threshold Δt_{peak} which effectively reduces the number of stored events,

and processes events that come into view for larger time period.

- A barrier δ defines a limit power barrier to eliminate minor detected events.
- A parameter of `#jobs` for splitting Knapsack processing into X parallel processes for faster detection.
- A threshold value of max_{states} of possible power states that FSM table is possible to store. This value was obligatory, to disallow building FSM step events for hypothetical devices that could probably exceed 4 step changes.

In particular, the implemented algorithm does not require any sub-metering of the target appliances during the training period, as it builds Finite State Machines for every possible appliance with no a-priori knowledge of the users behavior, nor the appliances installed. Also, the clustering method finds typical switch states of frequently occurring appliances. These patterns are mainly two-state ON/OFF devices, or FSM appliance model consisting of normally 3-4 states. Limitations encountered is that certain classes of appliances cannot be easily monitored such as low power and continuously varying power devices. But, surprisingly, this method achieves to detect FSM and also appliances which are constantly on or in stand-by mode.

The results revealed that chief consumer load devices such as cooker, fridge, TV, kettle could be easily detected. Additional tests are affordable to elaborate a suitable and robust parameter set to detect appliances for a wide spectrum of electric devices. The total execution time ranges from 3-7 minutes computed by a 2.8 GHz-personal computer. Using parallel computing, for pre-computation of sum results, '0-1' knapsack detection algorithm processes detection for a single day in less than 2 minutes.

In general, the overall performance shows a promising potential in disaggregating whole-house electricity consumption into its major end-uses, using pattern recognition approach and a genetic algorithm. Even when the clustering method results in 30 different state transitions to create FSMs, the genetic algorithm reduces the number of binary combinations to find finite state machines from $2^{30} \approx 10^9$ to $1.5 * 10^4$ with sufficient quality.

8.1 Future work I

Within the three-months limited time period available for conducting this project, most of the time was devoted to the research and implementation phase. Extensive evaluation and investigation for further improvements/extensions is compelling. Thus, analysis regarding all possible future work that can be made is described

below, in order to produce a more stable system, able to work independently, and provide permissible accuracy.

The most important problem that faced the clustering algorithm was what kind of granularity would exist between separate edge events. On average, k-means clustering produced 8-15 clusters $P(C_r)$. In addition, due to the fact that dataset was generated from uncontrolled home environment, ON/OFF active state power values where not always identical for the same device, and also, FSM devices such as fridge of microwave, were not consuming identical power at each distinct multi-state power event.

The first extension would be to experiment with various machine learning clustering methods such as hierarchical clustering, fuzzy clustering or distribution-based clustering. Applying a variety of clustering techniques, could probably improve the accuracy of extracted $P(C_r)$, and build more accurate static FSM appliance tables. Also, 2-dimensional clustering can be easily extended to 3-dimensional, as Q_i objects stored after edge detection are of the form $Q_i = \left[P\left(S_i\right), \hat{P}\left(S_i\right), \Delta T\right]$ hence, adding time duration to the clustering process can prove useful and extract more accurate cendoid $P(C_r)$ values.

The second extension suggested would be to experiment with different values of configurable parameters, and train the classifier with different settings. Detection accuracy experiments could reveal optimal parameters for best fitting results. In particular, maximising the barrier δ from 10Watt to 40-50Watt or more, would probably reduce the series of detected series of events, by keeping only strong edge events. Also relative $\gamma_1, \gamma_2, \gamma_3$ parameters used in §4.1 may be changed, according to the necessitated level of state transition accuracy for each appliance model.

A third modification would be to pre-define, if possible, all possible number of distinct power states of hypothetical appliances. During learning phase, the genetic algorithm creates all possible binary combinations

$$Z_j = [c_{j,1}P(C_1);\ c_{j,2}P(C_2);\ c_{j,3}P(C_3); \dots\ ;\ c_{j,N_C}P(C_{N_C})] \tag{22}$$

for each possible i appliance, where only one position will be one, and all the others have to be zero, for a random t_i. If the algorithm was able to perform iterations for decreased non-zero $P(C_r)$ elements, the volume of total binary combinations would be lessened, and pre-process time/efficiency could be more adequate. This could also reduce the overlappings that were discussed in §4.4. Definitely, as the implementation was based upon [4] and [5], such modifications have to be widely researched.

8.2 Future work II

The breakdown of whole-house electricity consumption among the major end-uses is considered beneficial to increase the house owners' awareness regarding the actual energy performance of houses. Due to the nature of computations, and the complexity of Learning and Detection Phase, it is necessary that the algorithm is extended and split into two parts, one that could possibly run on a capable backend system, and another that can be run on a typical residential advanced meter.

The accuracy of detection for unprotected data was proved to be fairly good. Thus, we believe that modifying the existing method to operate in a fully automated distributed manner, could be practical and contribute to the Smart Metering technology. Unfortunately, the architecture of the smart grid severely limits the usefulness of the recent NILM solutions. This is due to the fact that smart meters lack the necessary computational power to conduct the spectral analysis locally at the home.

Furthermore, a distributed solution implies a completely autonomous procedure of disaggregation of power load via a distributed system. The goal of such NILM scheme, is to utilise computationally limited AMI meters to analyse as much data as possible, while still allowing NILM learning phase to be employed at the backend. Existing techniques such as frequency analysis require meter readings sampled at the millisecond scale [19], while the meter is configured to take a maximum of one reading per 20 seconds. Thus, the low processing power and other computational constraints of the meters, could be bypassed by transmitting all readings back to a dedicated server for automated processing. In addition, such solution, would not require extra communication overhead between appliances and meters, and use only the available bandwidth of the channel while still allowing the meter to accurately detect appliances, and learn about the environment without initial setup.

In general, on-meter edge detection is a viable way to reduce the amount of data that needs to be transmitted across the entire AMI network, as a server could prioritise the relearning phase and provide reliable appliance models. In this scenario, the smart meter would monitor the actual state of the appliances in real time, and transmit the results back to a server (both active and re-active power). There, extensive computation processes such as a modified learning process and dynamic FSM model computations would be aggressively executed, based on the received results and real-time error-rate estimation. Reinforcing learning process, is also an important aspect that could also contribute to the current method.

We believe that future versions of the current work will focus on:

i. An algorithm for re-active and autonomous finite states machine static table construction, based on error detection that will provide high accuracy to the overall results.
ii. Detection of FSM state table errors that may arise due to unknown major appliances being added to the house. This would trigger the meter to enter the relearning phase and eventually receive an up-to-date state table from the AMI head end.
iii. Support of advanced pattern recognition methods for measuring accuracy of detection results compared to real power spectrums.
iv. Implementation of an online NIALM system that could be a suitable application for consumers to remote control special appliances of their home in absence.

Appendix A

According to Toshiba Research and Development Laboratory Regulations, I am not allowed to include the whole part of the code, thus only the main Matlab and Java functions, for data sanitisation, learning and detection phase are included.

Process.java – data sanitisation and detection of important events

```
// This function reads the provided 24H day power trace and creates series of events
private static void createEvents() {
  float Sum_of_all_assigned_DP_T = 0;
  float Boost_value = 0;
  int time_distance = 0;
  // active power
  for(Entry<Integer, ArrayList<Record> > e : sorted_series_pwr.entrySet() ){
    ArrayList<Record> tt = new ArrayList<Record>();
    tt = e.getValue();
    int timeVal = tt.get(0).getTimeval();
    for(Record d : tt ){
    /** equation (7)
    * find total power step P(S_i)
    * and maximum power setp value P'hat
    * ~~ boost value is called
    */
    Sum_of_all_assigned_DP_T = Sum_of_all_assigned_DP_T + d.getDP_power();
    // find boost value
    if( d.getDP_power() > Boost_value)
      Boost_value = d.getDP_power();
  }
  time_distance = tt.get( tt.size() -1 ).getTimeval() - tt.get(0).getTimeval() ;
  if(time_distance < 0 && tt.get( tt.size() - 1).getDate() != tt.get(0).getDate() )
    time_distance = Math.abs(time_distance);
    // add to vector
    ObjectT obj = new ObjectT(e.getKey() , Sum_of_all_assigned_DP_T, Boost_value,
        time_distance );
    ObjectVector_pwr.put(e.getKey(), obj);
    Sum_of_all_assigned_DP_T = 0;
    Boost_value = 0;
  }
  sorted_ObjectVector_pwr = new TreeMap<Integer, ObjectT>(ObjectVector_pwr);
}
```

Process.java – Combining Series DT threshold for detected events

```
// This function extracts combined Series of Power according to Delta T peak threshold
private static void combineSeries_Pow(){
  ArrayList<Integer> rememberSi_combined_deleted = new ArrayList<Integer>();
  // sorted_series_pwr hold < Si, set of Records >
```

```
  for(int i = 1 ; i < sorted_series_pwr.size(); i++){
    for( Record d : sorted_series_pwr.get(i) )
      check_P_Si_ = check_P_Si_ + d.getDP_power();
    // get first time val from S1
    check_T_Si = sorted_series_pwr.get(i).get(0).getTimeval();
    for( Record dd : sorted_series_pwr.get(i+1) ){
      check_P_Si_plus_1_ = check_P_Si_plus_1_ + dd.getDP_power() ;
    }
    // get first time val from S2
    check_T_Si_plus_1_ = sorted_series_pwr.get(i+1).get(0).getTimeval();
    /** Equation 6
    * if P(S1) > 0 and P(S2) < 0;
    * P(Si) denotes TOTAL power STEP ( sum of all assigned DP_t )
    **/
    if( check_P_Si_ > 0 && check_P_Si_plus_1_ < 0 )
    // if time < dt_peak
    if( ( check_T_Si_plus_1_ - check_T_Si ) < DT_peak){
    // copy S2 -> temp2
    ArrayList<Record> temp2 = sorted_series_pwr.get(i+1);
    // copy S1 -> temp1
    ArrayList<Record> temp1 = sorted_series_pwr.get(i);
    // temp1 = temp1 U temp2
    int timeStampOfSeries = temp2.get(0).getTimeval() ;
    for( Record ss : temp2 ){
      if( ss.getTimeval() <= timeStampOfSeries){
        temp1.add(ss);
        timeStampOfSeries = ss.getTimeval() ;
      }
      sorted_series_pwr.put(i, temp1);
      rememberSi_combined_deleted.add(i+1);
      }
    }
    System.out.println( "#Power Series Si " + sorted_series_pwr.size() );
    for(Integer d : rememberSi_combined_deleted)
      sorted_series_pwr.remove(d);
    System.out.println("#Combined Power Series " +sorted_series_pwr.size() );
  }
```

SanitizeData.java – Read File, Filtering, Normalisation of power trace

```
public class SanitizeData {
  public SanitizeData(String FileToSanitize, Integer idTotal) throws IOException,
      InterruptedException, ParseException{
    FileInputStream fin = new FileInputStream( FileToSanitize );
    FileOutputStream fout = new FileOutputStream( FileToSanitize + "_tmp.txt");
    FileChannel fcin = fin.getChannel();
    FileChannel fcout = fout.getChannel();
    ByteBuffer buffer = ByteBuffer.allocate(1024);
    while(true){
```

```
9           buffer.clear();
10          int r = fcin.read(buffer);
11          if(r == -1 )
12            break;
13          buffer.flip();
14          // tokenize the data file
15          CharBuffer cbuf = Charset.forName("ISO-8859-1").decode(buffer);
16          String ss = cbuf.toString().replace("-", "").replace(":", "").replace("\"",
                "").replace(" ", ",");
17          ss = ss.replace(",", " ");
18          fcout.write( buffer.wrap( ss.getBytes() ) );
19        }
20        Runtime sorting1 = Runtime.getRuntime();
21        try{
22        sorting1.exec("sort -k 2,2 -k 3,3 -k 1,1 " + FileToSanitize + "_tmp.txt" + " -o
              " + FileToSanitize + "_tmp.txt" ).waitFor();
23        }catch( IOException ioe){
24        ioe.printStackTrace();
25        }
26        System.out.println("DataSet Sorted !! " + FileToSanitize + "_tmp.txt");
27        /** Sanitize dataset
28         * based on some rules :)
29         */
30        DataInputStream in = new DataInputStream( new FileInputStream( FileToSanitize
              + "_tmp.txt" ) );
31        BufferedReader br = new BufferedReader(new InputStreamReader(in) );
32        FileWriter fstream = new FileWriter("writeTotals" + "__"+ idTotal + ".txt");
33        BufferedWriter out = new BufferedWriter(fstream);
34        while( ( line = br.readLine() )!= null ){
35          i++;
36          if( i > 1 && line.split(" ")[0].equals(String.valueOf(idTotal)) )
37            out.write( line + "\n" );
38        }
39        out.close();
40        HashMap<String, String[]> t = init_temp_DayCollection();
41        FilterPowerDataSplitIntoSecs(t, idTotal);
42        System.out.println("Deleted " + DaysDeleted.size() + " days due to not having a
              record every minute");
43        System.out.print(DaysDeleted);
44      }
```

training.m – 3-D K-means clustering and silhouette plotting

```
1  %% K-means clustering
2  opts = statset('MaxIter', 200, 'Display', 'iter');
3  [clustIDX, clusters, interClustSum, Dist] = kmeans(data, K, 'options',opts, ...
4      'distance','sqEuclidean', 'EmptyAction','drop', 'replicates', 200);
5  %% plot data+clusters
6  figure, hold on
```

```
scatter3(data(:,1),data(:,2), data(:,3), 50, clustIDX, 'k.')
scatter3(clusters(:,1),clusters(:,2), clusters(:,3), 200, (1:K)' )
hold off, xlabel('x'), ylabel('y'), zlabel('z')
title(['K-means clustering for K = ', num2str(length(clusters) )] );
lgnd = legend('$ P(S_i) $ ','$ \hat{P}(S_i) $')
set(lgnd, 'Interpreter','latex');
xlabel('DP/W')
ylabel('DP_{max}/W')
grid on
% plot clusters quality
figure
[silh,h] = silhouette(data, clustIDX);
avrgScore = mean(silh);
```

training.m – building Finite State Machines Table

```
% the algorithm creates a matrix table_A_i and holds 0 or 1, then, finds the maximum
    number of combinations using the quality of each row. This is based on how close
    the sum of power changes is to zero. This is done because the state transitions of
    any load should start and end with an off state
%Fast population of all binomial combinations using VChooseKRO function
R = VChooseKRO(int8(0:1), K);
table_A_i = zeros(2^K-1,K);
saveQ = zeros(2^K-1 , 2);
for m = 1 : 2^K-1
    A_i = zeros(K,1) ;
    for k = 1 : K
        % ci * p(cj) ~> [0 or 1] * p[cj]
        if( R(m,k) ~= int8(0) )
            A_i(k) = powerClust(1,k); %* R(m,k);
        else
            A_i(k) = 0;
        end
        table_A_i (m,k) = A_i(k);
    end
    max_Ai = max( abs(A_i (A_i ~= 0 ) ) ) ;
    Q1 = abs (sum( A_i ) ) / max_Ai ;
    Q2 = abs (sum( A_i' * num_in_each_cluster) ) / max_Ai ;
    Q3 = abs (sum( R(int8(m),:) ) ) / length(clusters) ;
    total_Q = g_1*Q1 + g_2*Q2 + g_3*Q3;
    saveQ(m, [1:2] ) = [total_Q, m];
end
```

training.m – 3-D K-means clustering and silhouette plotting

```
% for each row in SelectedSequences search for permutations that do not have zero
    values inside and do not start with negative and then positive
FINAL_FSM_TABLE = zeros(1,K+1);
for d = 1 : length(SelectedSequences)
    % Sum of Power steps must be positive or near zero
```

```
5       % and maximum power steps is 2 <= x <= 4
6       non_zero_times = nnz( SelectedSequences(d,[2:end]) );
7       % get permutations with Positive values of eacy row in
8       tempPositives = SelectedSequences(d, 1+find(SelectedSequences(d, [2:K+1]) > 0));
9       % get permutations with Negative values of eacy row in
10      tempNegatives = SelectedSequences(d, 1+find(SelectedSequences(d, [2:K+1]) < 0));
11      if( sum ( SelectedSequences(d,[2:end]) ) >= 0 && ...
12          length(tempPositives) >= 1 && length(tempNegatives) >= 1 && ...
13          non_zero_times >= 2 && non_zero_times <= 3 )
14          %%% new filter method for negative positive and DP > 0
15          candidate_raw = SelectedSequences(d , [2:end] ) ;
16          toStore_new = zeros( 0, length(tempPositives) + length(tempNegatives) );
17          % at least 2 elements
18          if( nnz(candidate_raw) >= 2 )
19              perm_candidate_raw = perms( candidate_raw(find(candidate_raw ~= 0 )));
20              % store possible combinations which have at front positive
21              % and at the end negative value and Pt = Pt-1 + DPt always positive
22              perm_to_delete = 0;
23              for i = 1 : length(perm_candidate_raw)
24                  % take first permutation
25                  toCheckRaw = perm_candidate_raw(i,:);
26                  if( toCheckRaw(1) > 0 && toCheckRaw(end) < 0 )
27                          for j = 1 : length ( toCheckRaw )
28                           % sum in row at every step must always be positive
29                           if( sum( toCheckRaw(1 : j) ) < 0 )
30                               if(perm_to_delete == 0)
31                                   perm_to_delete = toCheckRaw;
32                               else
33                                   perm_to_delete = vertcat(perm_to_delete,
                                         toCheckRaw);
34                               end
35                               i = length(perm_candidate_raw);
36                           end
37                      end
38                  else
39                      f( toCheckRaw(1) < 0 || toCheckRaw(end) > 0)
40                          if(perm_to_delete == 0)
41                              perm_to_delete = toCheckRaw;
42                          else
43                              perm_to_delete = vertcat(perm_to_delete, toCheckRaw);
44                          end
45                          i = length(perm_candidate_raw);
46                      end
47                  end
48              end
49              toStore_new = setdiff(perm_candidate_raw, perm_to_delete ,'rows' );
50              % for each row in toStore_new find frequency that
51              % these steps occur in total PCi <-> Si mapping
```

```
52                [toStore_new_size rrr] = size(toStore_new);
53                frequency_for_each_row = zeros(toStore_new_size, rrr+1);
54                for u = 1 : length( toStore_new(:,1) )
55                    frequency = length(strfind( Sequence_of_PCi, toStore_new(u,: ) ) );
56                    if( frequency > 0)
57                        frequency_for_each_row(u,:) = [frequency, toStore_new(u,:) ];
58                    else
59                        frequency_for_each_row(u, :) = NaN;
60                    end
61                end
62                frequency_for_each_row(any(isnan(frequency_for_each_row),2),:)=[];
63                frequency_for_each_row = sortrows(frequency_for_each_row, -1);
64            end
65            if( length( frequency_for_each_row) > 0 )
66                zz = zeros(length(frequency_for_each_row(:,1)), K -
                      length(frequency_for_each_row(1,:) ) + 1 );
67                FINAL_FSM_TABLE = vertcat(FINAL_FSM_TABLE,
                      horzcat(frequency_for_each_row, zz) );
68            end
69        end
70    end
```

```
1   %% create all possible sums from the knapsack algorithm
2   fprintf('..... Initializing knapsack results .....\n');
3   R = VChooseKRO(int8(0:1), length(uniques) );
4   R = R(2:end, :);
5   job = createJob( 'configuration', 'local', 'FileDependencies', {'job_try.m'} );
6   combos = round ( 2^length(uniques) );
7   endValue = -1+1 ;
8   % Parallel process of Knapsack Solutions
9   for task = 1 : 4
10      startValue = endValue + 1;
11      endValue = ( task / 4 ) * combos -1
12      task_id = createTask ( job, @job_try, 1, ...
13      { uniques, [startValue, endValue] , DP_to_Pwr, R} ) ;
14  end
15  submit( job );
16  waitForState ( job, 'finished' );
17  results = getAllOutputArguments ( job );
18  for task = 1 : 4
19      if ~isempty ( results{task} )
20          disp(' results '); disp( results{task} );
21          if result == 0
22              result = results{task};
23          else
24              result = vertcat( result, results{task} );
25          end
26      end
27  end
```

```
destroy( job );
clear sched job task task_id tasknum startValue endValclear aue combos
%%
sum_table = result;
sum_table_row = sum_table(:, 1);
fprintf('..... Finished knapsack results .....\n');
```

detection.m – Disaggregating knapsack edge detection results to hypo devices

```
%% Uses knapsack sum results to do the detection
disp('Loading knapsackresults...')
load knapsackresults.mat
sum_table_row = sum_table(:, 1);

for ii = 1 : length(powerss)
    tj = 0;
    target = powerss(ii,:) ;
    %%%%%% new knapsack %%%%%%
    for j = 1 : length(sum_table_row)
        max_value_found = 0;
        if sum_table_row(j) == target
            max_value_found = sum_table_row(j)
            tj = j;
            break

        elseif sum_table_row(j) < target & sum_table_row(j) > max_value_found
            max_value_found = sum_table_row(j);
            tj = j;
        end
    end
    result = sum_table(tj, [2:end] ) ;
    sol(ii) = sum( result );
    vector_appl_detected = ismember(DP_to_Pwr, result) .* DP_to_Pwr;
    indipendent_appl = 0;
    for i = 1 : length(vector_appl_detected(:,1) )
        indipendent_appl(i,1) = max( vector_appl_detected(i,:)) ;
    end
    if (indipendent_appll == 0)
        indipendent_appll = indipendent_appl;
    else
        indipendent_appll = horzcat(indipendent_appll , indipendent_appl);
    end
end
```

detection.m – Finding correlations between real and detected appliances

```
%% accuracy measurement using pearson and spearman correlations
real__matrix = appliances_storage_powers( [startTime:time_distance:endTime] , :);
hypo__matrix = indipendent_appll.';
compare_pears = corr(real__matrix, hypo__matrix);
```

```
[r,s] = find( compare_pears > 0.150) ;
ss = [r,s];
rr = unique(r);
fprintf('Pears found %d devices with corr > 0.15 = \n', length(rr))
min(compare_pears( find( compare_pears >=0.15 ) ) )
max(compare_pears( find( compare_pears ) ) )
compare_spear = corr(real__matrix, hypo__matrix, 'type', 'Spearman');
[r,s] = find( compare_spear > 0.150) ;
ss = [r,s];
rr = unique(r);
fprintf('Spearman found %d devices with corr > 0.15 = \n', length(rr) )
min(compare_spear( find( compare_spear>=0.15 ) ) )
max(compare_spear( find( compare_spear ) ) )
```

```
% loads sampled devices to verity all appliances that are included in folder of training or
    testing folder
disp('..... Starting verification .....')
fprintf('\n\n..... Choose a folder with test data to analyse .....')
folder = uigetdir(pwd, 'Folder with real appliances :' ) ;
fileList = getAllFiles(folder)
% merge toClusterPower datasets
% this loop finds all subfiles in a folder and returns those which
% include 'toClusterPower'
data1 = 0;
total_power = [];
startT = 3600* input('Start time (01-24) HH : ');
endT = 3600* input('End time (01-24) HH : ');
T1 = datenum('00:05:00', 'HH:MM:SS');
T2 = datenum('23:55:00', 'HH:MM:SS');
appliances_storage_powers = 0;
figure()
for i = 1 : length(fileList)
    [pathstr, name, ext] = fileparts( fileList{i} );
    getFile = regexp( name, '(trace_).*(?i)(41|50|51|52|53|54)', 'match');
    if( length(getFile) ~= 0 )
        ffile = strcat(pathstr,'/',name,ext)
        s = load(ffile);
        fid = fopen(strcat(pathstr,'/',name,ext), 'rt');
        C = textscan(fid, '%f %f %f %f %f %s', 'Delimiter', ' ',...
        'HeaderLines', 0, 'MultipleDelimsAsOne',true, 'CollectOutput', false);
        fclose(fid);
        appliance_power = C{3};
        size(C{6})
        if( appliances_storage_powers == 0)
            appliances_storage_powers = appliance_power;
        else
            appliances_storage_powers = horzcat(appliances_storage_powers,
                appliance_power);
        end
```

```
34            dt = datenum(strcat( C{6} ), 'HH:MM:SS' );
35            t1 = find(dt == T1);
36            t2 = find(dt == T2);
37            dt = dt( t1 : t2 ) ;
38            appliance_power_ = appliance_power(t1:t2);
39            %%%% set axis of time %%%%
40            dt_set = dt([startT : endT], 1);
41            appliance_power_dt_set = appliance_power([startT:endT], 1 ) ;
42            plot(dt_set , appliance_power_dt_set, '-')
43            total_power = horzcat(total_power, appliance_power_) ;
44            datetick('x')
45            hold all
46        end
47    end
48    legend('cooker', 'fridge', 'microwave', 'TV', 'xbox', 'kettle')
49    xlabel('time')
50    ylabel(' Watt ')
51    title(['Real Appliances Sampled ', num2str(name) ])
52    grid on
53    hold off
54    for i = 1 : length(total_power(:,1) )
55        total_(i) = sum(total_power(i,:));
56    end
57    save totalpowers.mat appliances_storage_powers;
58    %%%% set axis of time %%%%
59    total__ = total_.';
60    total_dt_set = total__([startT:endT], 1 ) ;
61    for i = 1 : length(fileList)
62        [pathstr, name, ext] = fileparts( fileList{i} );
63        getFile = regexp( name, '(trace_).*(?i)(40)', 'match');
64        if( length(getFile) ~= 0 )
65            ffile = strcat(pathstr,'/',name,ext)
66            s = load(ffile);
67            fid = fopen(strcat(pathstr,'/',name,ext), 'rt');
68            D = textscan(fid, '%f %f %f %f %f %s', 'Delimiter', ' ',...
69            'HeaderLines', 0, 'MultipleDelimsAsOne',true, 'CollectOutput', false);
70            fclose(fid);
71            appliance_power = D{3};
72            dt = datenum(strcat( D{6} ), 'HH:MM:SS' );
73            t1 = find(dt == T1);
74            t2 = find(dt == T2);
75            dt = dt( t1 : t2 ) ;
76            appliance_power_ = appliance_power(t1:t2);
77            %%%% set axis of time %%%%
78            dt_set = dt([startT : endT], 1);
79            appliance_power_dt_set = appliance_power_([startT:endT], 1 ) ;
80             plot(dt_set, appliance_power_dt_set, '-r')
81            axis auto
```

```
82          hold on
83      end
84  end
85  hold all
86  datetick('x', 'keepticks')
87  legend('Known appliances', 'Overall sampled')
88  xlabel('time')
89  ylabel(' Watt ')
90  title(['Watt Power Sampled ', num2str(name)] )
91  grid on
```

jobtry.m – Knapsack Job Process submitted to cluster for parallel computing

```
1   function [table_T_ii ] = job_try( uniques, range, DP_to_Pwr, R )
2   % replaced knapsack problem. Find all possible sums at once and use table_T_ii to get
        the results
3   table_T_i = [];
4   uniquess = uniques';
5   K = length( uniquess );
6   table_T_ii = zeros(1, K);
7   for index = range(1) : range(2)
8       c_combo = bitget( index, 1:length( R(index,:) ) );
9       t_combo = find(c_combo >=0 );
10      t = c_combo(t_combo);
11          T_i = zeros(1 , K);
12          for k = 1 : K
13              if( t(k) ~= int8(0) )
14                  T_i(k) = uniquess(k);
15              else
16                  T_i(k) = NaN;
17              end
18          end
19          track_ro = 0;
20          for i = 1 : length( T_i )
21              [r w] = find( bsxfun (@eq, DP_to_Pwr, T_i(i) ) );
22              track_ro = [track_ro unique(r')];
23          end
24          track_ro = track_ro( find( track_ro > 0 ) ) ;
25          if length( unique( track_ro ) ) == length( track_ro ) & max( T_i) > 0
26              T_i(isnan(T_i)) = 0 ;
27              z = length( table_T_ii( : , 1) );
28              table_T_ii( z + 1, 1:K+1 ) = [sum(T_i(:) ) T_i];
29          end
30      if( index == range(2) )
31          table_T_ii = table_T_ii( [2:end], :)
32          return
33      end
34  end
35  end
```

Glossary

AMI	Advanced Metering Infrastructure, allows 2-way communciations with the meter.
Appliance Signatures	Electrical measurable parameters that give information about the nature and operating status of an appliance.
Cluster Analysis	Studying the different groups or clusters formed by the appliances events in a graphical interpretation.
Differential Privacy	Guarantees that a user is not at increased risk of privacy whan participating in a certain statistical database.
EDMS	Energy Data Management System.
Event	A step change in power due to the change of an appliance's operating state to another.
HAN	Home Area Network - home communications network for energy management and other purposes.
ICT	Information and communication technology.
Knapsack Algorithm	A dynamic programming algorithm for finding the most likely sequence of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.
Load signature	Series of time-stamped aberate power loads $p(t)$ derived from energy values $e(t)$ metered at intervals Δt, $p(t) = e(t) - \frac{e(t-\Delta t)}{\Delta t}$.
NIALM	Analysis of electrical changes to deduce what appliances are being used as well as their individual energy consumption.
Non-intrusive Systems	They do not require to intrude into an applaince to collect data about it.
PeHEMS	Privacy Enabled Home Energy Management Systems and Load Balancing Prototype.

Power Load	Anything that uses electricity. They may be divided into resistive, inductive or capacitive.
Smart Power Meter	New generation of meters with additional features for easy and efficient monitoring of power consumption.
WPAN	Wireless Personal Area Network.
ZigBee	ZigBee is a low-cost, low-power, wireless mesh network standard, based on an IEEE 802 standard for personal area networks.

References

[1] 3-EHouses European Union FP7 project. `http://www.3ehouses.eu/`, 2012. [Online; accessed 24-September-2012].

[2] Gergely Ács and Claude Castelluccia. I have a dream!: differentially private smart metering. In *Proceedings of the 13th international conference on Information hiding*, IH'11, pages 118–132, Berlin, Heidelberg, 2011. Springer-Verlag.

[3] M. Baranski and Voss J. Nonintrusive appliance load monitoring based on an optical sensor. In *Power Tech Conference Proceedings, 2003 IEEE Bologna*, volume 4, page 8 pp. Vol.4, june 2003.

[4] M. Baranski and J. Voss. Genetic algorithm for pattern detection in nialm systems. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pages 3462 – 3468 vol.4, oct. 2004.

[5] D.C. Bergman, Dong Jin, J.P. Juen, N. Tanaka, C.A. Gunter, and A.K. Wright. Distributed non-intrusive load monitoring. In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, pages 1 –8, jan. 2011.

[6] A.J. Bijker, Xiaohua Xia, and Jiangfeng Zhang. Active power residential non-intrusive appliance load monitoring system. In *AFRICON, 2009. AFRICON '09.*, pages 1 –6, sept. 2009.

[7] P.M. Scwartz D. J. Solove, M. Rotenberg. Data protection conference on the implementation of directive 95/46/ec. `http://ec.europa.eu/justice/newsroom/data-protection/events/020930_en.htm/`, 2012.

[8] S. Drenker and A. Kader. Nonintrusive monitoring of electric loads. *Computer Applications in Power, IEEE*, 12(4):47 –51, oct 1999.

[9] C. Efthymiou and G. Kalogridis. Smart grid privacy via anonymization of smart metering data. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 238 –243, oct. 2010.

[10] Zhong Fan, Georgios Kalogridis, Costas Efthymiou, Mahesh Sooriyabandara, Mutsumu Serizawa, and Joe McGeehan. The new frontier of communications research: smart grid and smart metering. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, e-Energy '10, pages 115–118, New York, NY, USA, 2010. ACM.

[11] U.S. Government. *Nist Framework and Roadmap for Smart Grid Interoperability Standards, Release 1 0.* General Books LLC, 2011.

[12] G.W. Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870 –1891, dec 1992.

[13] A. Ipakchi and F. Albuyeh. Grid of the future. *Power and Energy Magazine, IEEE*, 7(2):52 –62, march-april 2009.

[14] G. Kalogridis, C. Efthymiou, S.Z. Denic, T.A. Lewis, and R. Cepeda. Privacy for smart meters: Towards undetectable appliance load signatures. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 232 –237, oct. 2010.

[15] G. Kalogridis and D. Saraansh. Pehems: Privacy enabled hems and load balancing prototype. In *Smart Grid Communications (SmartGridComm TAIWAN) , 2012 IEEE International Conference on*, pages 232 –237, oct. 2012 (authors preprint).

[16] Hyungsul Kim, Manish Marwah, Martin F. Arlitt, Geoff Lyon, and Jiawei Han. Unsupervised disaggregation of low frequency power measurements. In *SDM*, pages 747–758, 2011.

[17] John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors. *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*. Curran Associates, Inc., 2010.

[18] H.Y. Lam, G.S.K. Fung, and W.K. Lee. A novel method to construct taxonomy electrical appliances based on load signaturesof. *Consumer Electronics, IEEE Transactions on*, 53(2):653 –660, may 2007.

[19] K.D. Lee, S.B. Leeb, L.K. Norford, P.R. Armstrong, J. Holloway, and S.R. Shaw. Estimation of variable-speed-drive power consumption from harmonic content. *Energy Conversion, IEEE Transactions on*, 20(3):566 – 574, sept. 2005.

[20] Jian Liang, S. Ng, G. Kendall, and J. Cheng. Load signature study ;part i: Basic concept, structure, and methodology. *Power Delivery, IEEE Transactions on*, 25(2):551 – 560, april 2010.

[21] IMS research. Smart home energy management systems world. `http://imsresearch.com/products-services/pr-detail.php?pr_id=2045/`, 2012.

[22] Jan Simon. VChoseKRO(V,K), Matlab function. `http://www.mathworks.com/matlabcentral/fileexchange/26242/`, 2012. [Online; accessed 24-September-2012].

[23] The Smart, Grid Interoperability, Cyber Security, and Working Group. Introduction to nistir 7628 guidelines for smart grid cyber security. *Energy*, (September):20, 2010.

[24] F. Sultanem. Using appliance signatures for monitoring residential loads at meter panel level. *Power Delivery, IEEE Transactions on*, 6(4):1380 –1385, oct 1991.

[25] F. Sultanem. Using appliance signatures for monitoring residential loads at meter panel level. *Power Delivery, IEEE Transactions on*, 6(4):1380 –1385, oct 1991.