Exploiting Appliance State Constraints to Improve Appliance State Detection

Tarun Khandelwal
tarunrkhandelwal@gmail.com
RBCCPS, IISc Bangalore

Karan Rajwanshi
k.rajwanshi93@gmail.com
RBCCPS, IISc Bangalore

Priti Bharadwaj
bharadwaj.priti@gmail.com
RBCCPS, IISc Bangalore

Shayan Srinivasa Garani
shayan.gs@dese.iisc.ernet.in
Electronic Systems Engineering
IISc Bangalore

Rajesh Sundaresan
rajeshs@ece.iisc.ernet.in
RBCCPS, and Electrical
Communication Engineering
IISc Bangalore

ABSTRACT

This work deals with non-intrusive load monitoring using a single inexpensive device at the mains. We argue that very low sampling rates (of 1 Hz) may suffice. This enables significant compression and cheaper end-devices. There are challenges when operating at such low sampling rates, of course. To achieve good appliance inference performance we propose improved event detection, feature extraction, and inference algorithms. The inference algorithm exploits state transition constraints and proposes the use of a maximum likelihood sequence detection for improved performance.

CCS CONCEPTS

• **Mathematics of computing** → **Cluster analysis**; *Maximum likelihood estimation*; • **Computer systems organization** → *Embedded and cyber-physical systems*;

KEYWORDS

NILM; energy disaggregation; HMM; k-means clustering

ACM Reference format:
Tarun Khandelwal, Karan Rajwanshi, Priti Bharadwaj, Shayan Srinivasa Garani, and Rajesh Sundaresan. 2017. Exploiting Appliance State Constraints to Improve Appliance State Detection. In *Proceedings of e-Energy '17, Shatin, Hong Kong, May 16-19, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3077839.3077859

1 INTRODUCTION

Non-intrusive load monitoring (*NILM*) deals with the estimation of individual appliance usages from observations of aggregate power consumption. Prior algorithms assume availability of data sampled at high sampling rates from, as of now, expensive smart meters. This paper explores the feasibility of NILM using data sampled at lower sampling rates. There are three motivations for doing this: we want a single smart meter device connected into the mains, a cheap smart meter and data compression right at the source point.

Problem statement: Given the time-series data of aggregate power consumption obtained from a single smart meter sampling at a low rate of $\sim 1Hz$ and installed in series with the *mains* supply, infer the sequence of appliances' ON/OFF states.

This inference problem is not straightforward because the low sampling rate leads to some complications. Device signatures can be tracked only at a much lower resolution. Moreover, multiple events may have occurred during the relatively large 1s sampling period.

Our methodology involves a series of steps – data processing, event detection and feature extraction; learning of essential feature parameters; and inference of appliance states.

1) We first preprocess a component of the signal, which is active power, to suppress spikes and irregularities. This step results in a waveform which is essentially a sequence of step changes in the active power level and hence facilitates event detection.

2) We then use the first and second differences to detect changes in the power level. Epochs with significant changes are marked as events, and this constitutes the event detection step. Our event detection algorithm offers robust performance in terms of both *precision* and *recall* [4] values when compared with competing algorithms operating at a much higher sampling rate. The results are presented in Subsection 3.1. The highlight of our proposed procedure is that we are able to detect more events for an acknowledged challenging context as compared to the existing approaches of [2, 3].

3) For extracting features corresponding to each event, where change in active and reactive power levels (ΔP and ΔQ respectively) are our features of interest, we use a clustering based approach inspired by the method in [3]. This approach utilizes the shifts in clusters on the (P, Q)–plane associated with the occurrence of events.

4) For learning the feature parameters, we assume that the feature vectors corresponding to every appliance will form a cluster over the $(\Delta P, \Delta Q)$–plane. A hierarchical approach, though currently executed with some manual supervision, gives better performance than a vanilla k-means clustering. We then proceed to compute the mean and variance for each of the clusters to learn the noise parameters associated with an appliance. For the sake of simplicity

e-Energy '17, Shatin, Hong Kong

DOI: http://dx.doi.org/10.1145/3077839.3077859

we assume points in each cluster to be Gaussian distributed with mean as the cluster centers and independent noise components.

5) The mean and variance values of each appliance's load are then used to create a hidden Markov model for net instantaneous consumption. A Viterbi algorithm then performs the maximum-likelihood detection under the constraint that there is an alternation of states, from ON to OFF to ON and so on, for simple devices. We also force other reasonable constraints over state transitions that limit switching of appliances to at the most two per event epoch. This setup is capable of handling false alarms and missed detection to an extent. We use this setup to predict appliance states for the BLUED dataset.

This paper covers a part of work being carried out as a part of a larger program which aims to provide households in a small town called Aluva, Kerala, India, with some insight into their own electricity consumption. Towards this, information related to the ownership of appliances was obtained by a survey administered on a subset of households and is used in a bottom up model to disaggregate the total consumption into different components. For an even a smaller subset, smart meters were installed in series with the mains for capturing electric power consumption data at a sampling rate of 1Hz. This work was motivated by our need to infer appliance usage from the smart meter data. The knowledge gained from such an inference, the survey data, and historical electricity consumption data for each household are being used to disaggregate total consumption into components for cooling, heating, lighting etc. The findings of the larger program will be reported elsewhere. Here we restrict attention to inferring appliance usage. See Section 3 for inference of appliance usage in BLUED dataset [2].

The remainder of this paper is organized as follows. In Section 2 we provide a detailed explanation of our working methodology and the algorithms used for disaggregation. In Section 3 we provide a performance comparison of our algorithm with those in the literature.

2 METHODOLOGY

Our methodology involves five steps as indicated in the introduction. These are preprocessing, event detection, feature extraction, parameter learning and inference. We shall discuss each of these steps in detail in this section. Let us first briefly discuss a few terminologies that will be extensively used in the sections ahead.

2.1 Terminology

(1) The *appliance state* of appliance i at time t, denoted $a_i(t)$, is either ON or OFF. The state (ON/OFF) of this appliance is encoded as:

$$a_i(t) \in \{0, 1\} \tag{1}$$

with 0 being OFF and 1 being ON. Multiple appliance states, as in the case of a ceiling fan, can be handled with more states at the expense of complexity.

(2) The *state of the system* containing k appliances, at time t and denoted by $\boldsymbol{a}(t)$, is the vector

$$\boldsymbol{a}(t) = \Big[a_1(t), \quad a_2(t), \quad \cdots \quad , a_k(t)\Big] \in \{0, 1\}^k \tag{2}$$

(3) An *event* is said to have taken place when the state of the system changes. This happens when at least one appliance state changes. This leads to change in observed power as can be seen in Figure 1.

(4) Each event is associated with a set of *features* that characterizes it. In this work, the amplitudes of changes in active and reactive powers (ΔP and ΔQ respectively) are the features of interest.

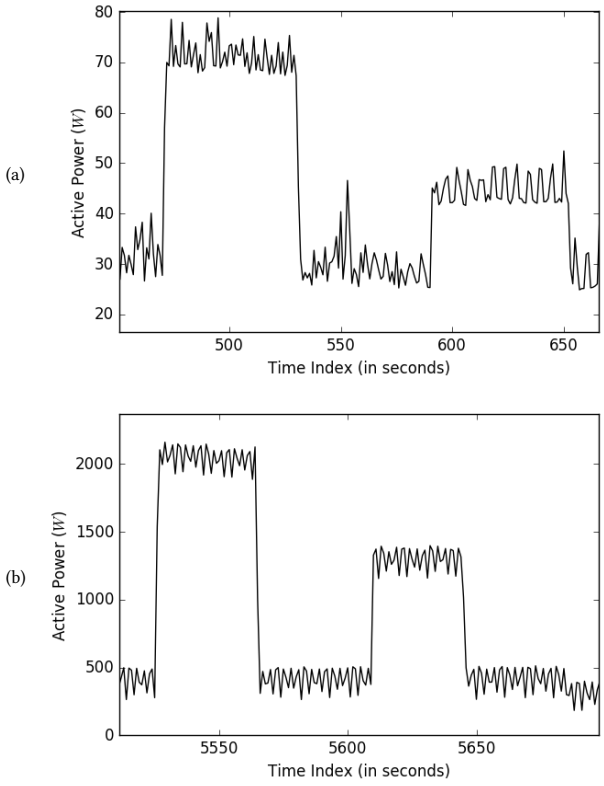

Figure 1: Example of events in presence of (a) minor fluctuations, and (b) high amplitude fluctuations arising from the operation of a washing machine.

2.2 Event Detection and preprocessing

An event leads to a change in aggregate power. This will also happen when an appliance switches to an intermediate state in case of multistate appliances like a fan, electric iron, washing machine, etc. The power consumption data is recorded by a logger at the rate of 1 sample/second. At this sampling rate all events are captured essentially as a step rise or fall in active power waveform. Fluctuations, if any, will appear to ride on top of a step change as can be observed from Figure 1. An event detection algorithm captures the epochs where an appliance(s) switches states.

Event detection through clustering, as described by Barsim et al. [3], uses the idea that if no new appliance is turned ON/OFF the active and reactive power levels would not change. If a device turns ON/OFF there will be change in corresponding power levels. Of course, the observed power levels are noisy. What clustering does is to obtain a scatter plot of active power and reactive power indexed by time. The samples will be clustered around a location

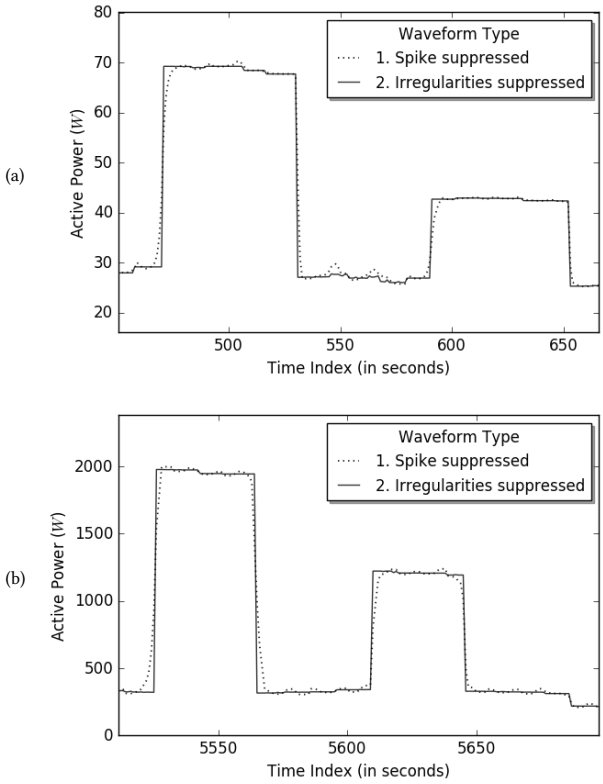

Figure 2: Results of preprocessing of the signal shown in (a) Figure 1a; and (b) Figure 1b.

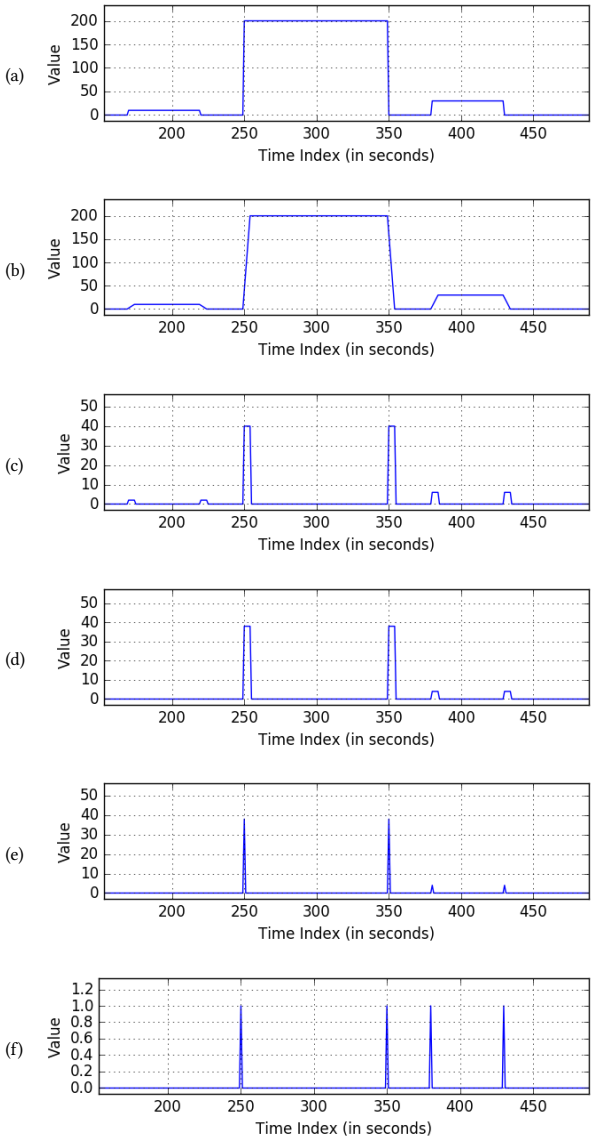

Figure 3: Illustration of event detection process: (a) an example waveform with 3 events; (b) output of moving average with $W_{\text{ma}} = 5$, (c) first difference, (d) first difference values above a threshold of 2, (e) positive values from first difference of signal obtained in step 3d, (f) sign of values obtained in step 3e; the impulses give the event locations.

if no appliance changes state. A shift in location of clusters will be observed in case of an event. Because the number of points are more in clusters corresponding to steady states, it becomes possible to ignore occasional spikes/surges and transients that last for a small duration so that they are not captured as events. A couple of samples corresponding to a spike or a transient will not form a sufficiently large cluster to be considered as an event. The advantage of this approach is that we are able to obtain average initial and final values of active and reactive power corresponding to each event. Also, we can obtain a reasonable estimate of time duration for the power level to transit from one stable state to another stable state upon occurrence of that event. On the other hand, there are some limitations as well. If there are significant fluctuations in the signal or if two events are located less than $\sim$ 10 seconds apart from each other in time the algorithm tends to miss them. Also, if there is an appliance like washing machine running in the background, whose operations produce high amplitude transitions in the background (see Fig 1b), then events for appliances of less than 25W power rating tend to be missed. On the other hand, as mentioned by Barsim et.al in [3], this approach can handle sinusoidal steady states for a 6Hz sampling rate signal.

Our event detection approach detects a level change using first and second differences. The central idea is that whenever there is a step change, we should be able to observe a high magnitude of the corresponding derivative (first difference) and change in the sign of second derivative (second difference) due to the sigmoidal nature of the active power samples. However, the implementation requires preprocessing of the signal to suppress spikes and other irregularities in the waveform. These are discussed in the self-explanatory Algorithms 1 and 2 respectively. This step is helpful because the original waveform is observed to contain fluctuations and occasional surges of small duration. These could result from the supplied power itself or due to operation of different appliances.

Algorithm 1 Spike Suppression

procedure SpikeSuppression(x)
  // x is the input waveform
  SignalLen ← #Samples in x
  for $i < 10$ **do**
    // 10 iterations of positive spike suppression
    for $j <$ SignalLen **do**
      if $x[j] > x[j-1]$ AND $x[j] > x[j+1]$ **then**
        $x_1[j] \leftarrow \frac{x[j]+x[j-1]}{2}$
      else
        $x_1[j] \leftarrow x[j]$
    $x \leftarrow x_1$
  for $i < 10$ **do**
    // 10 iterations of negative spike suppression
    for $j <$ SignalLen **do**
      if $x[j] < x[j-1]$ AND $x[j] < x[j+1]$ **then**
        $x_1[j] \leftarrow \frac{x[j]+x[j-1]}{2}$
      else
        $x_1[j] \leftarrow x[j]$
    $x \leftarrow x_1$
return x

Algorithm 2 Suppression of Irregularities

procedure SuppressIrregularities(x, W)
  // x is the input waveform.
  // W is the window size, i.e. given a sample, the number of neighboring samples to be observed
  SignalLen ← #Samples in x
  for $j <$ SignalLen **do**
    $M_1 \leftarrow \frac{1}{W}\Sigma_{i=1}^{W} x[j+i]$
    // Mean value of W samples on the right of j^{th} sample
    $M_2 \leftarrow \frac{1}{W}\Sigma_{i=1}^{W} x[j-i]$
    // Mean value of W samples on the left of j^{th} sample
    if $|x[j] - M_1| < |x[j] - M_2|$ **then**
      // Choose whichever out of M_1 and M_2 is closer to the j^{th} sample
      $x_1[j] \leftarrow M_1$
    else
      $x_1[j] \leftarrow M_2$
  $x \leftarrow x_1$
return x

Detecting events on the original waveform increases the number of wrong detections or false discoveries. In Algorithm 1, ten iterations each for suppression of positive and negative spikes are used. In Algorithm 2, we allow $W \in \{2, 4\}$ as choices of window size W. Initially, a higher value of W is used which is then gradually reduced. The number of iterations reduce along with the value of W. The preprocessing step reshapes the waveform to appear mostly as a series of step rise(s) or step fall(s) in power level. The effect of preprocessing steps on the signal is illustrated in Figure 2 using the example waveforms shown in Figure 1.

After preprocessing, the waveform is fed to the event detection procedure described in Algorithm 3. Using this approach, it is possible to detect events separated in time by more than six seconds. If the separation is any lesser, those events may be detected as a single event. The performance of the event detection algorithm depends upon the threshold value used for declaring a change in active power ΔP and upon the moving average window size W used for smoothing. As per our observations, it is difficult to detect level changes less than 12.5 Watts. Also, if there is an appliance like washing machine running in the background, there is a possibility of false alarms (false positives). The results of our event detection algorithm are presented in Section 3.

Algorithm 3 Event Detection

procedure EventDetection($x, W_{\text{ma}}, \Delta P_{\text{thresh}}$)
  // x is the input waveform
  // W_{ma} is moving average window size
  // ΔP_{thresh} is the threshold to detect change in active power level
  // n represents time index
  EventIdx ← $\emptyset$
  // Create an empty set of event time indices
  SignalLen ← #Samples in x
  $T \leftarrow \frac{\Delta P_{\text{thresh}}}{W_{\text{ma}}}$
  // Normalized threshold power level change
  for $n <$ SignalLen **do**
    $g[n] \leftarrow \frac{1}{W_{\text{ma}}}\Sigma_{i=0}^{W_{\text{ma}}-1} x[n+i]$
  // Compute moving average
  for $n <$ SignalLen **do**
    $f[n] \leftarrow |g[n] - g[n-1]|$
  // Compute magnitude of first difference
  for $n <$ SignalLen **do**
    $f_T[n] \leftarrow [f[n] - T]_+$
  // Apply threshold to values obtained from first difference
  for $n <$ SignalLen **do**
    $s[n] \leftarrow [f_T[n] - f_T[n-1]]_+$
  // Get positive values of the first difference of f_T
  for $n <$ SignalLen **do**
    $y[n] \leftarrow \text{sign}(s[n] - s[n-1])$
  // First difference of $s[n]$
  for $n <$ SignalLen **do**
    if $y[n] == 1$ AND $y[n+1] == 0$ **then**
      EventIdx ← EventIdx ∪ $\{n\}$
  // Mark the time index where the value of y changes from 1 to 0
return EventIdx

Our event detection module does not give features for further operation unlike the approach by Barsim et al. [3]. This is the responsibility of a separate feature extraction module which is described in the next subsection.

2.3 Feature Extraction: Clustering and a Gaussian model for each appliance

The event detector identifies the epochs of significance where a state change has been detected. In order to extract the features to be used for the identification of appliances associated with the event, we use an approach inspired by [3]. To capture the features, the changes in active and reactive powers, ΔP and ΔQ, for an event located at any time index n, we process twenty samples each of active power and reactive power about index n using the approach

of [3]. The features of interest for an event are taken to be the changes in active and reactive powers ΔP and ΔQ respectively, that result either from a switching ON/OFF of an appliance(s), or from a change in state in case of a multistate appliance. Features for different classes of appliances should form different clusters on the $(\Delta P, \Delta Q)$ – plane because each class of appliance offers a different load in terms of impedance and hence consumes a certain complex power. Even in case of a multistate appliance like refrigerator, a cluster would appear for each multistate if data has been collected for sufficiently long duration. In principle, one could treat each of these clusters as a separate appliance. For each appliance we model the $(\Delta P, \Delta Q)$ as a random variable with the Gaussian distribution. We must then learn the mean and variance for the $(\Delta P, \Delta Q)$ of this appliance. There are two ways in which these model parameters are learnt.

In case we know the appliance label associated with each event, we can collect all such samples and estimate the mean and variance from the samples. This is the case with the BLUED dataset because we know the appliance that caused event.

However, in most practical situations, we do not know the appliance associated with $(\Delta P, \Delta Q)$ for an event. This is the case in the data collected by us in uncontrolled environment in some Kerala households as part of our field trials. On these, we use a clustering algorithm to group events into different categories (clusters), with each one to be treated as a different appliance entity. We use a variation of k-means clustering on the $(\Delta P, \Delta Q)$ feature vectors associated with each event. Before feeding the feature vectors to the k-means algorithm, every feature x_i was normalized, and re-scaled in the log domain using following transformation:

$$x^{\star} = \text{sign}(x_{\text{normalized}}) \cdot \log(1 + |x_{\text{normalized}}|). \tag{3}$$

This transformation, reminiscent of speech compression algorithms, proved useful when high power appliances, that consume more than 1000W of power, are used along with low power appliances and there is a need to suppress the spread of features at high power ranges. The bias term of 1 ensures that signs are preserved and that the transformation is approximately linear for low values of $x_{\text{normalized}}$.

We roughly cluster events by using a small value of k and then look deeper into each cluster to see if further clustering could be done within it. This hierarchical approach was found to yield better clusters as compared to using a larger value of k right at the beginning. While working with our own internal data set, which contained lot of events with less than 100W of change in power levels, our approach yielded better clustering. If a cluster appeared to have been wrongly split into multiple parts, those parts were manually grouped together as one. Once clustering is completed, mean and variance for each cluster are estimated and are taken to be the learnt model parameters. Figure 5 in Section 3 shows the results of clustering using this approach on a real dataset.

2.4 Appliance Inference using a hidden Markov model

To improve upon the clustering results which so far did not take into account the *ON/OFF* state-matching, we model the states of the system and the total consumption observations using a *hidden Markov model* and deploy the *Viterbi algorithm* for a maximum-likelihood sequence detection of appliance states. There are approaches that use HMMs [1, 6–8, 10] for disaggregation. Instead of modeling individual appliances using HMMs [1, 7, 8], we model the entire system and infer states of this system when events occur. Our final goal is to use this inference for disaggregation. In a low sampling rate setting we also have to deal with compound events. We extend the HMM approach by allowing more than one appliance to change state in a one second window.

The *observed state* for the Viterbi algorithm is the aggregate power consumption consisting of both active and reactive powers at an *event* e_i. The *hidden states* are all the possible 2^k combinations of *states of the k appliances* $\boldsymbol{a}(t)$. Given observations, we must find out that sequence of hidden states which best explains the sequence of actual observations of aggregate power $\bar{Y}(t)$. We shall discuss this approach in detail in this section.

Section 2.2 and 2.3 provide us with information about *cluster centers* which includes cluster mean $\boldsymbol{\mu}^i = \begin{bmatrix}\mu_P^i & \mu_Q^i\end{bmatrix}^T$ and cluster standard deviation $\boldsymbol{\sigma}_i = \begin{bmatrix}\sigma_P^i & \sigma_Q^i\end{bmatrix}^T$ for an appliance/ cluster i. Assuming each cluster represents an appliance, suppose we have k appliances denoted as $A_1, A_2, \cdots, A_{k-1}, A_k$, the state of appliance i at time t is $a_i(t)$ as defined in Section 2.1. The *state of the system* is $\boldsymbol{a}(t) = \begin{bmatrix}a_1(t), a_2(t), \cdots, a_k(t)\end{bmatrix}$, as defined in Section 2.1. Let $\boldsymbol{S}$ be the set of all possible states; $|S| = 2^k$.

The mean of the total consumed complex power in system state S_j is the algebraic sum of the means of the appliances that are ON ($a_i(t) = 1$) in state S_j. This is given by:

$$\Delta^{(S_j)} = \sum_{i=0}^{k} a_i(t|S_j)\boldsymbol{\mu}_i \,, \tag{4}$$

where $\Delta^{(S_j)} = \begin{bmatrix}\Delta_P^{S_j} & \Delta_Q^{S_j}\end{bmatrix}^T$ is the mean active and reactive power for state S_j, and $a_i(t|S_j) = 1$ if $a_i(t) = 1$ for state S_j.

We assume that each event results in a noisy observation whose variance is the estimated variance of the corresponding appliance's state change. Suppose there are N detected events. The objective is to find that best possible sequence of system states $(\boldsymbol{a}(0), \boldsymbol{a}(1), \cdots, \boldsymbol{a}(N))$ which best explains the sequence of observations $\bar{Y} = (Y(0), Y(1), \cdots, Y(N))$ as described below. The state transitions allow toggling of multiple appliances in an event epoch (up to 2) but imposes the constraint of two consecutive ON's or OFF's are not possible. Let t be index of an event under consideration.

We model the aggregate power consumption recorded in the logger at event-index t to be $Y(t) = \begin{bmatrix}P(t) & Q(t)\end{bmatrix}^T$, where

$$Y(t) = Y(0) + \sum_{1 \le t' \le t} \sum_{i=1}^{k} (\boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i(t'))(a_i(t') - a_i(t'-1)), \tag{5}$$

with $P(t)$ = observed aggregate active power at index t,

$Q(t)$ = observed aggregate reactive power at index t,

$\boldsymbol{\mu}_i$ = mean active and reactive power consumption of appliance i,

$\boldsymbol{\epsilon}_i(t)$ = residual mean noise vector corrupting the power profile of appliance i represented as a Gaussian random variable,

$a_i(t)$ = state of appliance i at event index t.

The aggregate power consumption (5) can also be written as

$$\boldsymbol{Y}(t) = \boldsymbol{Y}(t-1) + \sum_{i=1}^{k} (\boldsymbol{\mu_i} + \boldsymbol{\epsilon_i}(t))(a_i(t) - a_i(t-1)). \quad (6)$$

2.4.1 Assumptions. The following assumptions have been made while modeling the system using a *Hidden Markov Model* (HMM):

(1) At most *two* appliances are turned *ON/OFF* at a time, i.e., each event shall correspond to a change of appliance state $a_i(t)$ for at most two appliances. In the BLUED dataset there are 39 compound events (two events occurring together at the same time-stamp at the 1 Hz resolution) out of 865 events, and no triple events on Phase A. There were 2 compound triple events out of 1513 events on Phase B. So at most two state changes per event seems to be a good assumption.
(2) No appliance(s) can have multiple states. They can have exactly two states (ON/OFF) corresponding to 1/0 respectively. This is clearly a simplifying assumption. The approach can be extended to appliances with multiple states under certain constraints. We have discussed the multi-state modelling of refrigerator in Section 3.
(3) Each appliance has a Gaussian distributed power demand given by $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$ where $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are respectively the mean and the standard deviation of appliance i.
(4) The distribution (modelled as a Gaussian) of the changes in the active and reactive powers for switching an appliance from $0 \rightarrow 1$ and from $1 \rightarrow 0$ are identical except for a sign change.

2.4.2 Data Requirements. We use the traditional Viterbi algorithm [5] that takes into account the *transition probabilities of a time-homogeneous Markov chain, emission probabilities and initial probabilities*. We can also exploit time-of-use information by using a time-varying transition matrix. For example, change in aggregate power consumption at night (sleeping hours) is more likely due to a refrigerator than other appliances with similar characteristics. A morning sequence of resistive loads is more likely to be a toaster or a water heater and less likely to be an electric iron. The following data is required to run the algorithm:

(1) *Transition Probabilities*: The transition probability is defined as t_{ij} = probability of transition from state $i \rightarrow j$. In order to exploit the constraint that consecutive ON or consecutive OFF's of the same appliance(s) are not possible, we modify the transition matrix to accommodate this constraint. Taking an assumption that at most $l = 2$ appliances can toggle states corresponding to a single event (as discussed in Section 2.4.1), we will essentially have p number of possible states to transit into from any given state. We calculate p as follows,

$$p = \sum_{m=0}^{l} \binom{k}{m}. \quad (7)$$

The transition probabilities to these p states are based on estimates from the BLUED dataset, and are 0 for the remaining $2^k - p$ states. But this could be updated to more general t_{ij} in future implementations. Note that we can consider higher values of l and make the model more general. The value $l = 2$ was chosen based on the observations on the BLUED dataset. The dimensions of the transition matrix $\boldsymbol{T}(t)$ are $|\boldsymbol{T}(t)| = 2^k \times 2^k$. The transition matrix will also take into account any misdetection of events that might occur due to the inefficiencies of either the event detection or the feature extraction algorithms. This is because the probability of being in the same state is non-zero, and for small changes in active and reactive power, the emission probabilities corresponding to the appliance state changes will be small suggesting that the change may be noise. The transition matrix can incorporate the time of the day. We can also enhance the matrix by making use of time of use information.
(2) *Emission Probabilities*: The values of a Gaussian *PDF (probability density function)* for the observation $Y(t)$, centered at the mean of the system state given by (4), are referred to as *emission probabilities*. The observed state (o_t) is a continuous variable. We discretize it into steps of size 1 Watt each ranging from 0 to 2000 Watts making it $|\boldsymbol{O}| = 2000$ possible observed states. The matrix is given by matrix $E = e_{S_j}(o_t)$ = state observation likelihood of observation o_t given current system state S_j. The dimensions for E are $|E| = 2^k \times |\boldsymbol{O}|$. The emission probability for a system state S_j corresponding to an observed aggregate power consumption of $Y(t)$ would be given as the value of the PDF $\mathcal{N}(\Delta^{(S_j)}, \sigma^{(S_j)})$. When we consider both *active* and *reactive* power as our observations, we modify the emission probabilities as the product of emission probabilities obtained form both active and reactive power observations. This is under the assumption that active and reactive powers changes are independent random variables.
(3) *Initial Probabilities*: We assume the trellis to begin at a particular time-stamp say 00:00 hrs and calculate the frequency distribution of all appliance states at that particular time-stamp, over several days, thus giving us the initial probability distribution

$$I = P(S_j) = \frac{\#S_j}{\#D}, \quad (8)$$

where
$\#S_j$ = number of times the state of system is S_j at 00:00hrs;
$\#D$ = number of days over which I is calculated.
The time of the day for calculating I is chosen to be a time of minimum household activity in order to reduce initial inference errors. The size of I is $|I| = 2^k$.

2.4.3 Maximum likelihood sequence detection. The traditional Viterbi algorithm [5] is applied on the set of hidden-states to extract the most likely sequence $\boldsymbol{a}(0), \boldsymbol{a}(1), \cdots, \boldsymbol{a}(N)$ corresponding to a sequence of observed total consumption $\boldsymbol{Y}(0), \boldsymbol{Y}(1), ..., \boldsymbol{Y}(N)$, where N is the number of observations in the sequence.

Recall the constraint of no two consecutive ONs of OFFs for a single appliance and the constraint that the maximum number of appliances changing states at an event is *two*. We have used $(\cdot)$ to represent the dot product and $(\odot)$ to represent element wise multiplication. See Algorithm 4.

Most NILM algorithms take into account only the active power while running a *hidden Markov model*. Appliances having a very similar distribution in the active power co-ordinate but distinct distributions across the reactive power co-ordinate can be disaggregated more effectively by using both active and reactive power change observations.

Algorithm 4 Inference Engine

procedure INFER(ObsStat of len N, TransProb T, InitProb I, EmissProb E) **return** best path

Intialization :
  for each state s **from 1 to** 2^k
    $trellis[s,1] \leftarrow I[s] * E[s,1]$
    $backpointer[s,1] \leftarrow 1$

Recursion:
  for each observation t **from 1 to** N
    for each state s **from 1 to** 2^k
      $\max_{\forall s \in S} trellis[s,t] \leftarrow trellis[s,t-1] \cdot E[t] \odot T(t)$
      $\operatorname{argmax}_{\forall s \in S} backpointer[s,t] \leftarrow trellis[s,t-1] * T(t)$

Termination:
  return backtrace path by following the states back from $backpointer[s_F, N]$

In order to accommodate multistate appliances, such as the refrigerator, we take each possible state to be a separate virtual appliance. We must then allow two or more simultaneous transitions to handle transitions.Then OFF → mustistate 1 → multistate 2 → OFF, will be interpreted as, OFF → virtual appliance 1 ON → virtual appliance 2 ON → both virtual appliances 1 and 2 turned OFF.

3 RESULTS

To benchmark our framework's performance, we applied it to the BLUED dataset [2]. This contains current and voltage values sampled at 12000Hz. This dataset was used because it is fully labeled in terms of event epochs and the corresponding appliances. Since our low sampling rate framework involves the sampling rate of 1Hz, we used current and voltage samples from the dataset to compute the 1 second-averaged active and reactive powers. The resulting data was used for event detection, feature extraction and as input to the Viterbi Algorithm for appliance state detection. In the following subsections we present and discuss the results. Let us mention in passing that the BLUED dataset has missing current and voltage samples for two durations totaling to 71 seconds thereby resulting in a loss of 5 events on phase B. There are also instances where more than one event take place within a one second duration. We treat each these as a compound event. The summary of single, multiple and lost events is presented in Table 1. Our results are based on the events corresponding to observed signal (power) sampled at 1 Hz.

3.1 Event Detection

Due to our preprocessing step, it is not be possible to mark precisely an event's epoch. We must allow some error margin. Let the actual and detected event locations be denoted by T_{actual} and T_{detected} respectively. Let the error margin be k samples. Then the time difference ($\Delta T = T_{\text{detected}} - T_{\text{actual}}$) in seconds between an actual and detected event satisfies:

$$-k \leq \Delta T \leq k$$

Let us now recall a few standard definitions related to performance of the event detection algorithm:

(1) When an event is detected within $\pm k$ of the location where it is actually present in the signal, the result is a True Positive (TP).
(2) When no event is detected within $\pm k$ of a location where there is *no* event, the result is a True Negative (TN).
(3) When an event is detected within $\pm k$ of a location where none is present in the signal, the result is a False Positive (FP).
(4) When an event is *not* detected within $\pm k$ of the location of an event, the result is a False Negative (FN).
(5) Recall describes the completeness of detection. It is the ratio of number of events detected correctly to the number of actual events present. It is defined as

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{9}$$

(6) Precision describes the correctness of detection. It is a ratio of number of events detected correctly to the number of detected events. It is defined as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{10}$$

(7) F-measure score is defined as

$$\text{F-measure} = 2\left(\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}\right). \tag{11}$$

We use Recall and Precision to evaluate the performance of event detection. Higher values of Recall and Precision indicate better performance. We have especially aimed at achieving comparable and high values for both Precision and Recall for phase A as well as for the challenging phase B of the BLUED dataset. Also, the error between the detected and actual event time indices should be as small as possible. The values of precision and recall correspond to the scenario where an event is said to be detected if it falls within ±3 samples (seconds) of the actual event index.

Our Event detection depends on two parameters – Moving Average Window size (W_{ma}), and the Threshold (ΔP_{thresh}) for detecting change in active power level. Figure 4 shows plots for Precision and Recall as one parameter is varied while the other is kept fixed. We studied the variation of Precision and Recall with W_{ma} and ΔP_{thresh}. We considered different error margins as well for identification of an event. As seen in Table 2, for $\Delta P_{\text{thresh}} = 16W$ and $W_{\text{ma}} = 4$ we get high values of Precision and Recall for *both* the phases. For $\Delta P_{\text{thresh}} = 10W$ and $W_{\text{ma}} = 4$, the performance on Phase A data is optimized.

We compare our results with those of approaches presented in references [2] and [3]; see Table 2. We reiterate that our sampling rate is 1Hz against the 60Hz sampling rate in both [2] and [3]. As mentioned before, we use an error margin of 3 samples between detected and actual event indices. For Phase A, even with the best parameters, the performance of our algorithm at 1Hz sampling rate is inferior in terms of Recall and slightly inferior in terms of Precision. However, our approach performs better for both Recall and Precision values on the more challenging Phase B data. The robustified parameters affect the performance on Phase A, but only marginally while providing a huge improvement on Phase B. If one were to maximise the worst precision and recall values across the two phases, our framework fares better than those in [2] and [3].

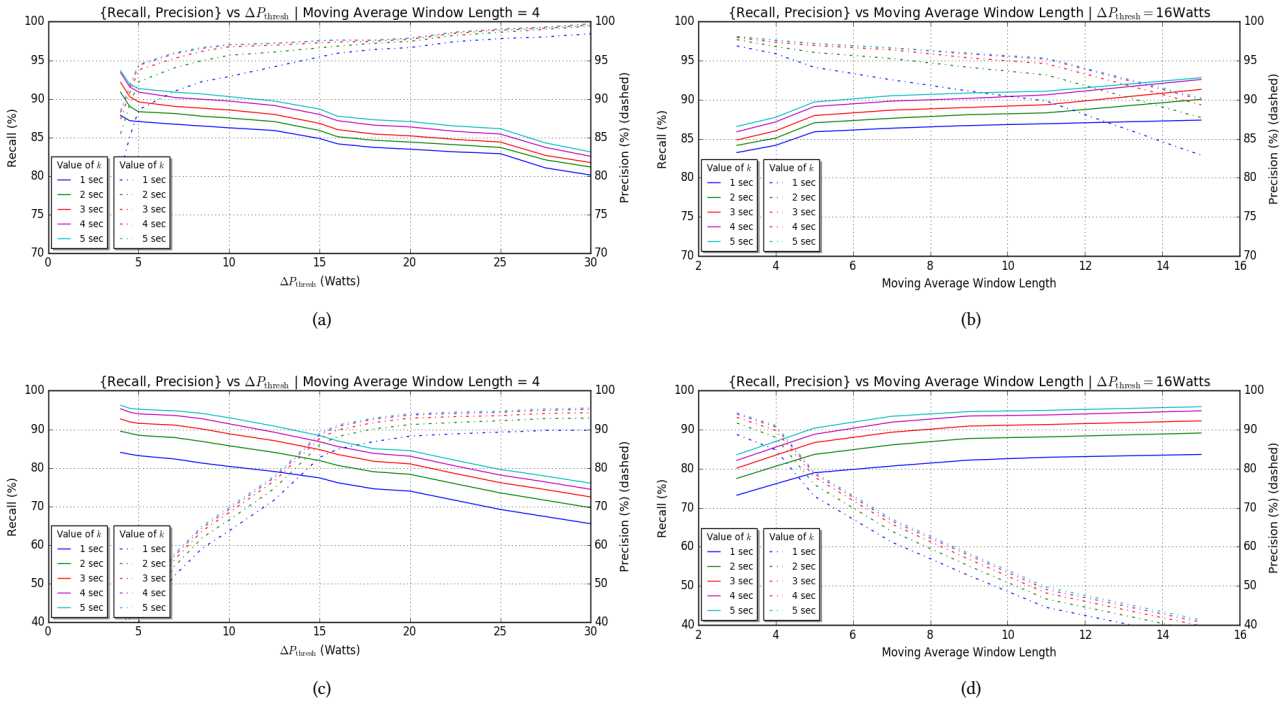

Figure 4: Performance analysis plots for different error margins (k). For Phases A and B respectively–(a) and (c) show variation versus ΔP_{thresh} keeping $W_{\text{ma}} = 4$ samples; (b) and (d) show variation versus W_{ma} keeping $\Delta P_{\text{thresh}} = 16$ Watts.

Table 1: Total events present in 1Hz sampled power data from BLUED dataset

Phase	#Single events (1)	#Compound events (2)	#Events (3) = (1) + (2)	#Events within lost samples (4)	Total Events (3) − (4)	Total Events in Dataset
A	826	39	865	0	865	904
B	1460	58	1518	5	1513	1578

3.2 Appliance Labelling using HMM

For the results, we have evaluated two approaches (see Table 3) on the BLUED dataset. First, keeping the transition probabilities to be constant for all the p possible states (A). Second, giving time independent transition probabilities according to the probability of an event corresponding to an appliance, based on their frequency distribution (B). We obtained better results in (B) as compared to (A). A time-varying transition matrix will be taken up in future work. We will discuss in detail the performance of (B) in this section. Performance evaluation with time-varying transition matrix is left as future work since this requires analysis of some additional survey data that we have on activities of individuals during the course of a typical day.

The BLUED dataset contains some appliance labels which correspond to unknown appliances. Also the computational complexity of the algorithm for even 14 clusters/ appliances is quite high. For simplicity we attached all the unknown appliances to a single label and fed it into the Viterbi algorithm. The performance of the inference engine was evaluated under the following three settings.

The first set of results for disaggregation on Phase A were obtained after relabelling the unknown appliances to a single label. This keeps the number of appliances down to $k = 10$. Of the 904 events on Phase A, 616 are corresponding to the refrigerator, which is a major contributor to the accuracy on the overall dataset. Refrigerator is taken as a single appliance even though it has multiple states (see later for a refinement). The accuracy (49.2%) is affected to a large extent because of large variances of the unknown appliances and the fact that only active power was used in calculations. Most of the events in this case were being attributed to the label of unknown appliances because of the huge variance. The results have been summarized in Table 3.

We next removed all the events corresponding to the unknown appliances in order to keep the clusters confined. We were now left with $k = 9$ appliance clusters. The refrigerator is again treated as

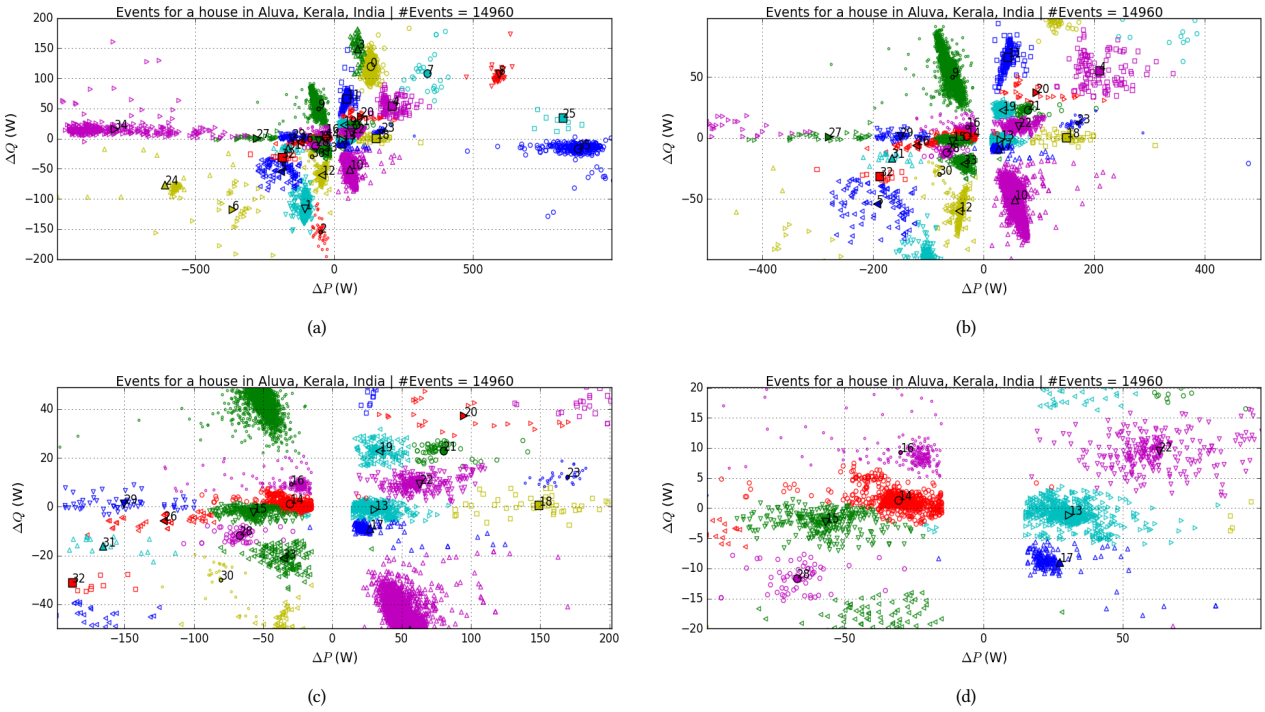

Figure 5: The outcome of hierarchical clustering mentioned in Section 2.3. (a), (b), (c) and (d) show the same scatter plot at different zoom levels to illustrate how hierarchical k-means clustering enabled us to separate out points into different clusters. This can be best viewed in color.

Table 2: Comparison of event detection performance. Algorithm 3 results have been obtained using parameters $(\Delta P_{\mathbf{thresh}}, W_{\mathbf{ma}}) = {}^{\dagger}(16W, 4), {}^{\ddagger}(10W, 4)$

Approach			Authors (Algorithm 3)		[2]	[3, 9]
Sampling Rate			1Hz		60Hz	60Hz
			Robust Values†	Phase A Optimized‡		
Phase	A	Recall	86.01%	88.55%	98.16%	98.41%
		Precision	97.36%	96.76%	97.94%	99.43%
		F-measure	0.913	0.925	0.980	0.989
	B	Recall	83.48%	88.83%	70.40%	70.48%
		Precision	89.83%	68.39%	87.29%	88.97%
		F-measure	0.865	0.773	0.779	0.787

a single appliance with large variance. The accuracy in this case improved to 62.8%.

We now discuss the multiple states of the refrigerator. We found three peaks from plotting the histogram for the changes in active power of the refrigerator. These were located around 42, 85 and 126W. Hence, we split the original cluster for refrigerator into 3 subclusters corresponding to the 3 peaks on the histogram. Similar, was the case with *bathroom upstairs lights* which had 2 subclusters centered around 63 and 126W. Since, the clusters centered at 126W for both refrigerator and bathroom upstairs lights were too close to be differentiated, we incorporated the effect of reactive power component into the model, by introducing a factor for reactive power in the calculation of *emission probabilities*. The accuracy obtained after this splitting into $k = 9 + 2 + 1 = 12$ clusters yielded a low accuracy of 52.1%. This was because of the fact that due to splitting of clusters, the variance for these clusters was dramatically reduced

Table 3: Performance of appliance inference using HMM on Phase A

Data	#Clusters	#Events	Accuracy(%)	
			A	B
Unknown appliances relabelled	10	865	25.2	49.24
Unknown appliances removed	9	764	40.31	62.82
Refrigerator and bathroom lights split	12	764	32.6	52.09
Refrigerator and bathroom lights split; $\sigma = 0.02\mu$	12	764	88.2	89.39
Refrigerator and bathroom lights split; (Constraint: No compound event)	12	764	7.4	22.25

and many refrigerator events were being incorrectly inferred as other appliances that had high variances.

The variance estimation for the appliances in the BLUED datasets was too noisy for most appliances except for the refrigerator and bathroom lights because of the lower number of points in clusters of other appliances leading to statistically insignificant second-order statistics. So we next experimented with σ_i being set as $\sigma_i = 0.02\mu_i$, by observing similar behaviour for clusters of fridge and bathroom light clusters. The standard deviation is observed to be $\sim$ 2% of the mean. This model cuts out the noise in the estimation of the standard deviation. The resulting accuracy was 89.4% which is quite promising.

Also, as can be seen in Table 3, the accuracy is very low if the model operates under the constraint of absence of compound events.

4 SUMMARY AND FUTURE WORK

Our event detection algorithm is robust to noise as it has performed significantly better on the challenging Phase B of the BLUED dataset which is a lot more noisy compared to Phase A. This better performance is when compared to the existing methods of [2, 3]. The Viterbi algorithm takes into account toggling of multiple appliances. Its performance is significantly affected in case of huge variance in observed changes in active and reactive power levels. This therefore requires good estimates of variances or appropriate models (such as $\sigma_i = 0.02\mu_i$) for better performance. But in case of clean clusters the algorithms performs exceptionally well.

Our study suggests that 1Hz sampling rate is good enough. This enables us to make the end device very cheap and efficient in terms of storage and communication.

Methods to improve the performance of event detection procedure to detect more low power appliances would be useful, particularly when fluctuations are large. Two approaches are being explored by our group. First, identification and incorporation of more features would help in disambiguating between appliances of similar power ratings. Second, time-of-use is an important information which can help in identifying an appliance.

5 ACKNOWLEDGMENT

The project was supported in part by the Shakti Sustainable Energy Foundation (Grant number G15 SSEF-154) and in part by the Robert Bosch Centre for Cyber-Physical Systems. The authors would like to thank Clytics for providing valuable support.

REFERENCES

[1] José Alcalá, Oliver Parson, and Alex Rogers. 2015. Detecting anomalies in activities of daily living of elderly residents via energy disaggregation and cox processes. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM, 225–234.

[2] Kyle Anderson, Adrian Ocneanu, Diego Benitez, Derrick Carlson, Anthony Rowe, and Mario Berges. 2012. BLUED: A Fully Labeled Public Dataset for Event-Based Non-Intrusive Load Monitoring Research. In *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability (SustKDD)*. Beijing, China.

[3] Karim Said Barsim, Roman Streubel, and Bin Yang. 2014. An approach for unsupervised non-intrusive load monitoring of residential appliances. In *Proceedings of the 2nd International Workshop on Non-Intrusive Load Monitoring*.

[4] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[5] Daniel Jurafsky and James H. Martin. 2016. *Hidden Markov Models* (3 ed.). 1–21. https://web.stanford.edu/~jurafsky/slp3/9.pdf

[6] Jacob A Mueller, Anusha Sankara, Jonathan W Kimball, and Bruce McMillin. 2014. Hidden Markov models for nonintrusive appliance load monitoring. In *North American Power Symposium (NAPS), 2014*. IEEE, 1–6.

[7] Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. 2012. Non-intrusive load monitoring using prior models of general appliance types. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Toronto, Ontario, Canada.

[8] Oliver Parson, Siddhartha Ghosh, Mark Weal, and Alex Rogers. 2014. An unsupervised training method for non-intrusive appliance load monitoring. *Artificial Intelligence* 217 (2014), 1–19.

[9] B. Wild, K. S. Barsim, and B. Yang. 2015. A new unsupervised event detector for non-intrusive load monitoring. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. DOI: https://doi.org/10.1109/GlobalSIP.2015.7418159

[10] Michael Zeifman and Kurt Roth. 2011. Viterbi algorithm with sparse transitions (VAST) for nonintrusive load monitoring. In *2011 IEEE Symposium on Computational Intelligence Applications In Smart Grid (CIASG)*. IEEE, 1–8.