DAISE - Reimplementing a State-Of-The-Art Algorithm for Non-Intrusive Load Monitoring

Daniel Illner
Technical University of Munich
Munich, Germany
ge98riy@mytum.de

Mikkel Groenning
The Technical University of Denmark
Copenhagen, Denmark
Mikkel@Groenning.net

ABSTRACT

The disaggregation of power consumption data of selected appliances, called Non-Intrusive Load Monitoring (NILM), is an important part of energy metering that promises energy savings and higher comfort levels for occupants. By providing publicly available datasets, that are used to test, verify, and benchmark possible solutions to tackle the tasks of NILM, it is much more easier to get a deeper understanding of the power consumption process in order to reduce the waste of energy.

This paper presents a reimplementation of the state-of-the-art event detection algorithm *Sequential Clustering-based Event Detection For Non-Intrusive Load Monitoring*, originally proposed by Karim Said Barsim and Bin Yang [14]. The algorithm will be applied to a new Non-Intrusive Load Monitoring (NILM) dataset, called building-level office environment dataset of typical electrical appliances (BLOND). The algorithm is a Sequential Clustering-Based Event Detection. It clusters sequential detection of abrupt changes in an aggregate electricity consumption profile. The input signal is decomposed into stationary and non-stationary segments and determines events happening based on a certain event model. This paper presents the results of the algorithm for the BLOND in comparison to the BLUED dataset.

KEYWORDS

Non-Intrusive Load Monitoring, NILM, Energy Disaggregation, BLOND Dataset, Event Detection, Event Model, DBSCAN, In-Depth Review

1 INTRODUCTION

A convenient means of determining the energy consumption and the state of operation of individual appliances based on analysis of the aggregate load measured by the main power meter in a building is known as non-intrusive load monitoring, moreover known as NILM or energy disaggregation. It states a process of analyzing changes in the voltage and current going into a building and therefore creates an individual consumption profile of each appliance. This can be used to inform the customers how much energy each appliance consumes in order to develop strategies to reduce their energy consumption in an appropriate way.

In times of energy crisis and climate change, both directly caused by the growth in energy consumption, applications like NILM have a actual global relevance. A significant reduction in the energy wastage can be achieved through fine-grained monitoring of energy consumption and giving this information back to the consumer. Leading to a reduction in energy consumption of every single consumer the overall effect will have certain impact on worldwide resource management [2]. Therefore NILM techniques have earned a true relevance in research and development [15].

The concept of NILM introduced by Hart [5] is fundamental and states the basis of several of today's load-disaggregation techniques. The general concept can be dived in certain steps [7].

(1) The power and voltage is measured and recorded at a certain sampling rate.
(2) The estimated power signals are normalized depending on the power line voltage
(3) Afterwards an edge-detection algorithm is applied to the normalized power signals. Therefore you get power consumption steps and within their timestamp labels.
(4) Cluster the output of the edge detection algorithm.
(5) From the clusters obtained ffinite state machine (FSM) models are created. An example for a specific model is an simple on/off appliance.
(6) These estimated appliance models are tracked. Whenever a model performs a state transition, the algorithm recognizes.
(7) Compute statistics and characteristics of the obtained models.
(8) In the final step, the algorithm attempts to assign each observed model to an actual appliance in the system.

Many previous works on NILM have been proposed with different kind of approaches that can be categorized into event-based and non event-based depending on whether they rely on detecting and classifying each appliance-state transition or they do not.

In this paper we will reimplement the algorithm *Sequential clustering-based Event detection for non intrusive load monitoring* created by Karim Said Barsim and Bin Yang [14]. The originally algorithm was applied on the BLUED dataset with big success. In this paper we apply the reimplemented algorithm to BLOND dataset. The paper is organized as it follows. In section 2, event detection will be introduced in general. Section 3 will provide a description about the BLOND dataset and what data we used as input to the algorithm. Section 4 will describe how the algorithm was implemented and what assumption were necessary in order to make it work. The evaluation and the results will be presented in section 5. Finally, section 6 summarizes and concludes this paper.

2 EVENT DETECTION

Without a reliable way of detecting events, NILM algorithms cannot proceed with the tasks of feature extraction and classification. As stated before, the focus lies on reimplementing an event-based detection algorithm. Therefore the paper gives a closer look at the topic.

Event detection is the process of identifying the relevant changes in the aggregate consumption data, which are caused by appliances. According to an overall background work that summarized existing literature you can classify event detection approaches into three different categories [1].

First group is called *Expert Heuristic*. The State-of-the-Art concept of Hart [5] introduced earlier is an exemplary model for this category. These algorithms are probably the least complex, and follow the basic principle of scanning the time based data looking for changes that hold up to a certain threshold. Moreover, in [11] for example the method senses when an appliance switches ON or OFF and uses a two-step classification algorithm to identify which appliance has caused the event. In order to detect the event, two criteria have to be fulfilled. The absolute magnitude of the signal's measurement at the time must be greater than a predefined threshold compared to a measurement certain time before. Also in order to avoid the presence of noise and reduce the chance of false positives, the previous event detected must not have occurred in a certain time period.

Probabilistic Models build the second class of event detection algorithms. According to literature there are three state-of-the-art algorithms that are commonly used in this area. These are the Generalized Likelihood Ratio (GLR) [10] test, the Goodness-of-Fit (GOF) [6] and the CUmulative SUM (CUSUM) filtering [12]. In order to state whether a event is happening in a certain timeframe, GLR calculates a likelihood ratio. The ratio is compared to a critical value or threshold which the decision is based on. The CUSUM is an adaptive threshold feature and detects both the beginning and the end of the transient-state feature. The detection of the beginning and the end of the transient is based on a simple statistic rule. The beginning of a transient happens when the stop rule occurs in steady state and the end of a transient occurs when the stop rule occurs in the transient state. By assuming that two consecutive timeframes have a certain distribution in common, the GOF test will detect an event occurring in these timeframes. Summarizing, all three methods are statistical tests, that act in the time domain and divide the measured signal into windows. Moreover, a second step is necessary to actually detect events, because in itself these methods are not capable to detect slow changes in the signal. In order to achieve that an extra analysis is added [12] or in a simpler way this is done via threshold [6, 10].

Last category, that literature defines, called *Matched Filters*. These algorithms or filters correlate a known template signal (mask) with an unknown signal in order to detect whether the mask also occurs in the unknown signal or not. The goal is to find known appliance transients in the aggregated consumption signal, which represents the unknown signal. In [9] these filtering techniques were applied to NILM. The event detector attempts to match segments of startUp transients to the aggregated signal using two transversal filters sequentially. In a first step the filter is set to find the transient shapes in the consumption signal. The second filter afterwards has to make sure that the found matches are not simply occurring noise but correspond to actual transients.

Furthermore, a typical event-based NILM approach is based on four processes: power measurement, event detection, event classification, and energy estimation [1]. In a first step electrical activity in specific environment has to be measured and monitored.

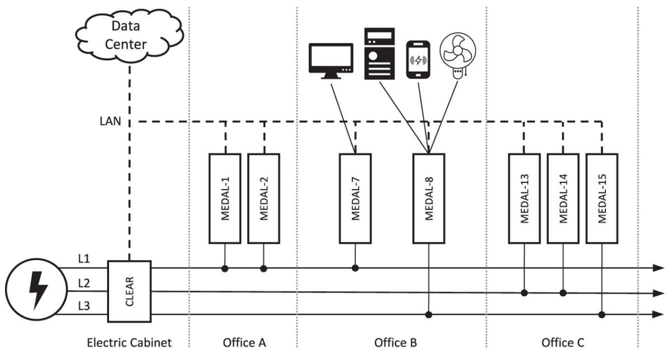

Figure 1: The measurement architecture of BLOND with physical placement of DAQ systems and connected appliances [8].

The environment can vary from private households up to a building-level office environment, like BLOND dataset represents. Event detection is made by searching for changes in the state of the appliance. At the stage of classification the system attempts to determine what caused this change in the signal in order to track its electrical power consumption (energy disaggregation).

3 BLOND DATASET

The work in this paper is based on data from a public NILM dataset named the building-level office environment dataset of typical electrical appliances (BLOND) [8].

As the BLOND dataset represents power usage of office environment, it states a collection of data at a typical office building in Germany, with the main occupants being academic institutes and their researchers. The department area of the measurement is a single floor with 9 dedicated offices with 160 m^2 of office space and central (non-electric) heating. The average power density of a weekday over the entire measurement period was 11.7 W/m^2. Compared to similar office buildings, that typically lie between 9.5 and 13.5 W/m^2, the office buildings for BLOND fits into this category. The population working in the monitored offices were between 15 and 20 people throughout the collection of the dataset.

The data shows differences between the days and in the days itself, because it indicates a typical German working schedule, which reaches from Monday to Friday with working hours from 9:00 to 18:00. Moreover on weekends and public holidays like 3rd of October there is no usage of the office spaces and therefore also no electricity consumption. The performances vary from light-duty office work, utilizing personal computers, monitors, and other electrical appliances. The majority of the working time is spent at the desk with different breaks like meetings or other activities outside their assigned offices.

In Figure 1 the structure of power system is seen. It is built by 50Hz mains with 3 circuits with a nominal phase shift of 120° (typical 3-phase supply), named L1, L2, and L3. The connection between each office room is one or two circuits and neighboring offices being on alternating circuits. In order to measure voltage and current waveforms with high sampling rates and bit-rate for a 3-phase power grid for the BLOND dataset a special CLEAR unit

[4] was designed. Furthermore, for the collection of ground truth appliance energy consumption data, a MEDAL system was used. The multiple units are installed in office rooms and connected to different circuits. The overall 15 medals are distributed to the three offices as seen in 1.

	Medals
Office A	1, 2, 3, 7, 12
Office B	6, 10, 11, 13, 14
Office C	4, 5, 8, 9, 15

Table 1: The distributions of the 15 medals towards the offices are stated.

BLOND consists of 945,919 files, amounting to 39 TB in total. In order to provide a proper usability to handle the data and apply an algorithm to it, Kriechbaum et al. [8] created a 1-second data summary. The data summary is derived by computing the root-mean-square of voltage and current, real power, apparent power, power factor, and mains frequency. This states a summary of 50.000 measurement per second into 1-second values. Therefore the use of that 1-second data summary allows us to work with the data without developing code that first preprocesses the big amount of data in a manageable way. Additionally, we stick to one month (November 2016) of dataset.

The BLOND data files are provided in HDF5 format. HDF5 is a unique technology suite that makes possible the management of extremely large and complex data collections. It provides the usability in most scientific computing languages and in particular Python (h5py/numpy/scipy). HDF5 offers high-performance computing, including partial read and write and therefore fast access to the data structure and a proper basis to focus on further tasks like implementing an algorithm.

4 DETECTION ALGORITHM

In the algorithm presented by Karim Said Barsim and Bin Yang [14], the *novel clustering-based* event detection algorithm is used in order to find events in Non-Intrusive Load Monitoring (NILM) systems. The approach of the algorithm is combining any clustering-based event with any event model.

This section is structured as follows. In section 4.1 we introduce the Event Model in which we will utilize the algorithm. In section 4.2 we describe how the algorithm works, assuming clustering has already taken place. For actually clustering the data, we have utilized the algorithm Density-Based-Based Spatial Clustering for Application with Noise (DBSCAN) [3]. We used the implemented algorithm from *scikit-learn* library. In section 4.3, we describe how we found the two parameters ε and *minPts* necessary to utilize the clustering algorithm DBSCAN. And lastly, we will go through an example in section 4.4 on how the algorithm works on interesting time series sample.

4.1 Event Model

The original paper suggests tree different event models, however anyone can be used and combined with any pre-clustering method. The event model we have implemented is heavily inspired by the proposed **Event Model 3** from the original paper. We will not present the **Event Model 1** and **Event Model 2** from the original paper as they are more or less, just special cases of **Event Model 3**, which we used for inspiration to build our own event model. As the first two event models will not be presented, we encourage the reader taking a look at the Paper of Karim Said Barsim and Bin Yang [14]. The first two event models provide a good impression of the idea behind the event model we implemented. Before we can introduce the event model, a few definition are needed.

Let the matrix

$$X = [\boldsymbol{x_1}, \boldsymbol{x_1}, ..., \boldsymbol{x_n}], \boldsymbol{x_n} \in \mathbb{R}^w \tag{1}$$

denote the input signal from which event is wished to be detected. As it can be seen, the matrix contains n, equal size and consecutive $w-$dimensional data samples. In other words, a large input signal has been split up into smaller part, which we will denote as time series samples. In this context, $\boldsymbol{x_n}$ contains the measured current from one of the three rooms in the office building. It is assumed that all n time series samples have been pre-clustered into m cluster sets $c_0, c_1, c_2, ..., c_m$. Note that the amount of cluster set varies from each time series sample x_i. It is assumed that the noise-aware clustering algorithm assigns noisy observations and outliers to cluster c_{-1}. Minus 1 is used as it is the default location of the DBSCAN algorithm to store noise and outliers, at least when using *scikit learn* library's implementation). The sum of the cardinality of all clusters sums up to the cardinality of the time series sample i.e.

$$\sum_{i=-1}^{m} |c_i| = n = |\boldsymbol{x_n}| \tag{2}$$

where $|c_i|$ and $|x_n|$ describe the cardinality of the respective cluster and time series sample. That is the amount of measurement in the cluster c_{-1} (noise) plus the amount of measurement in cluster c_0 plus the amount of measurement in cluster c_1. This continues up to the number of observation in the sample time series $\boldsymbol{x_n}$. We now introduce the following two definitions as metrics of a cluster. The definitions are needed to understand how the event model works.

Definition: *The Temporal Length* $\text{Len}(c_i)$ of cluster c_i is defined as the length of the cluster. Let u be the index of the first observation of a cluster and let v be the index of the last observation registered in the cluster. In other words u and v are described as the first and last observation belonging to cluster c_i, respectively. Thus the *The Temporal Length* is described as:

$$\text{Len}(c_i) = v - u + 1 \tag{3}$$

Definition: *The Temporal Locality ratio* $\text{Loc}(c_i)$ of cluster c_i is defined as:

$$\text{Loc}(c_i) = \frac{|c_i|}{\text{Len}(c_i)} \tag{4}$$

This is a measurement of how "pure" a clusters is. A high temporal locality indicates that a cluster contains low noise and no (or few) overlapping from other clusters. Thus $\text{Loc}(c_i)$=1 indicates maximum temporal locality that is a cluster which contains no noise and no overlapping from other clusters. I.e. the window of cluster is as big as the cardinality. The two measures is utilized for the implementation of the event model.

Event Model: An event is detected in a pre-clustered time series sample $\boldsymbol{x_n}$ if

(1) it contains *at least* two clusters $c_0, c_1, ..., c_n$ and the cluster containing the noise c_{-1} is not necessarily empty.
(2) the clusters have high temporal locality ratio i.e.

$$\text{loc}(c_i) \geq 1 - \varepsilon \tag{5}$$

Note that is possible here to adjust the parameter ε to allow more or less purity of the clusters.
(3) And *at least* two clusters $c_1, c_2, ..., c_n$ do not overlap in in the time domain i.e

$$\exists u, v, \boldsymbol{x_n} \notin c_j \forall (n > u) \text{ and } \boldsymbol{x_u} \in c_i \text{ and} \tag{6}$$

$$\boldsymbol{x_v} \notin c_j \forall (n < v) \text{ and } \boldsymbol{x_v} \in c_j \text{ where} \tag{7}$$

$$c_j, c_j \in \boldsymbol{X}. \tag{8}$$

This event model created here varies a little from the original paper **Model 3** as a few things were unclear. In the original paper only two clusters are allowed at a time however it is not clear how it was implemented. Thus the proposed event model from the paper has only been an inspiration to our event model.

4.2 Utilizing the Event Models and the Pre-clustering of the Input Signal

The algorithm's purpose is to search the time series samples for segments which match the event model. This is done by using a clustering algorithm on the time series samples and afterwards cheeking how the clusters of the time series samples match the event model. In the event model the clustering cardinality m is unknown thus the utilized clustering algorithm should be non-parametric. The Density-Based-Based Spatial Clustering for Application with Noise (DBSCAN) algorithm as this cluster algorithm require no prior knowledge of the number of clusters, furthermore the algorithm is noise aware, it therefore is a good choice for pre-clustering.

The complete event detection algorithm can be described as follows,

(1) Load the data set X and split up in bits of length w i.e.

$$\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}] \in \mathbb{R}^w$$

note here another parameter w which describes how big the time series samples should be.
(2) Receive new time series sample $x_{n+1} \in \mathbb{R}^w$
(3) Cluster the time series sample x_n utilizing the DBSCAN algorithm
(4) Check if the event model is satisfied if not satisfied according to the purity measure ε, remove the time series sample go to step 2
(5) Declare the a section of time series sample an even. I.e. register the the ending of cluster c_i i.e. v and the beginning of cluster c_j i.e u, and declare the event to occur within the interval.
(6) After declaration of the event remove the time series sample and go back to step 2. (repeat until no more time series samples are left).

The algorithm thus features a sequential clustering, as only subparts $\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}$ are clustered at a time. This makes the algorithm very efficient for real time NILM systems.

4.3 Parameter Tuning DBSCAN

Estimating parameter is a general problem for almost every data mining task. The clustering algorithm DBSCAN requires the two user specified parameters ε and minPts. Optimally, the parameter ε is given by the problem (e.g. a physical distance), and the parameter minPts is then the desired minimum cluster size. However, in the case of BLOND dataset no information were given and we did not posses any domain knowledge to quantify the parameters, neither was is possible to find any information in the dataset description. To introduce as little bias as possible we guessed that the parameters minPts were quantified as follows,

$$minPts = \lfloor \log(n) \rfloor \, . \tag{9}$$

where $\log(n)$ represents the natural logarithm. To determine the optimal ε, we found inspiration from Nadia Rahmah and Imas Sukaesih Sitanggang paper [13] regarding determination of optimal Epsilon (Eps) Value on DBSCAN algorithm. They state that Eps value is obtained by calculating the slope between points of the *K-Nearest Neighbour* where k in this context represent minPts value in a sorted manner. This is carried out with equation $(y2-y1)/(x2-x1)$ using a threshold value of 1 % so that the slope has a slope of 1 % difference is an optimal Eps value. When implemented, we have used a threshold value of 1 %.

In other words, we looked for the relative increase of the slope of the tangent of a point, where the relative increase of 1 % in the slope was relative to the point. The slope, where the relative increase was closest to 1 %, was used as the ε parameter. [13]. In section 4.4 follows an example on the parameter estimation as well.

4.4 Example

Consider the time series sample of root mean square current from office 1 measured 2nd November 2016 from 09:24:30-09:26:30, i.e figure 2. In order to find the parameters for the DBSCAN algorithm we consider the number of measurements. Here 2 min of data is represented i.e $2 \cdot 60 = 120$ measurement thus $minPts = \lfloor \log(120) \rfloor = 4$. To find ε we consider the 4 nearest neighbours as this was the number used for $minPts$. We consider the neighbour relative to all 30 observation before and after the point as the nearest neighbour will be close in time to the respective observation, as we are working with time series. All the closest neighbours were found and sorted in ascending order. This can be seen in figure 3. The relative slope was calculated as follows $(y_i - y_{i-1})/(x_i - x_{i-1})$. Thus the slope of point 2 was calculated relative to point 1 and point 3 calculated relative to point 2 and so and so forth. The slope closest to a increase of 1 % relative to the slope of the point before was used. In the context to the example the relative slope closest to an increase of 1 % relative to the previous slope was 0.0477. I.e. $eps = 0.0477$ and was used as a parameter for ε.

Thus in the example the clustering algorithm DBSCAN was utilized with the parameters $minPts = 4$ and $eps = 0.0477$. In figure 4 it can be seen how the DBSCAN clusters the time series sample. Note that here -1 is equivalent to cluster c_{-1} i.e. the cluster containing the noise. The algorithm produces a total of three

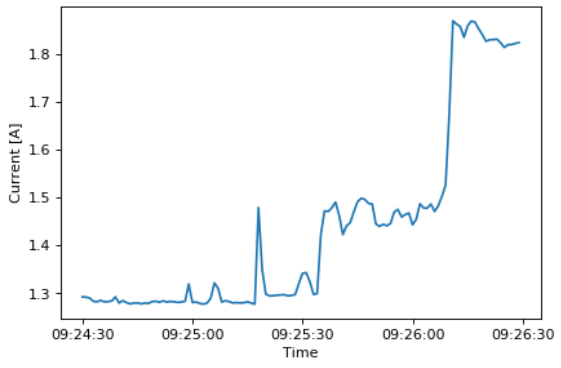

Figure 2: 2nd November 2016 - Current_rms1

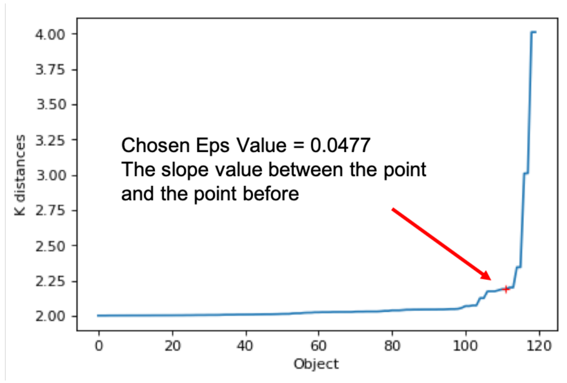

Figure 3: Observation from the time series sample sorted according to their fourth nearest neighbour ($minPts = \lfloor\log(120)\rfloor = 4$, were the best Eps Value is marked. I.e. the point were the slope increases with percent relative to the slope of the point before.

clusters $m = 3$, (excluding the noise). The output of DBSCAN cluster algorithm can also be seen in figure 5 where the the measurements are equipped with their appropriate cluster label.

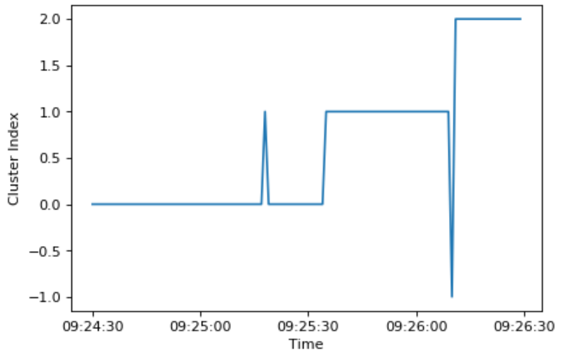

Figure 4: The clusters generated by the DBSCAN algorithm using the estimated parameters $minPts = 4$ and $eps = 0.0477$.

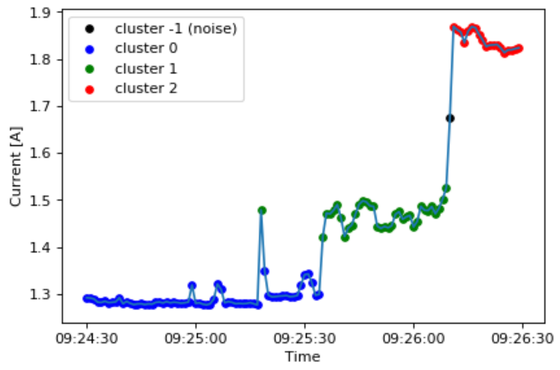

Figure 5: 2nd November 2016 - Current_rms1 with the appropriate cluster label.

When evaluating the clusters in term of the event model, we need to find the cardinality $|c_i|$, *temporal length* $\text{len}(c_i)$ and the *temporal locality ratio* $\text{loc}(c_i)$ of all three clusters at first (again excluding the noise). The start of the window of each cluster u and the end of each cluster window v in terms of their index can be seen in table 2. The cardinality $|c_i|$ as well as two length measurement can be seen as well. The *temporal locality ratio* is high for both Cluster 0 and Cluster 2 with values of 0.977 and 1 respectively. Whereas the *temporal locality ratio* of cluster 1 is significant lower with a value of 0.692.

	Cluster 0	Cluster 1	Cluster 2
Start (u)	33920	33948	34001
End (v)	33964	33999	34039
$\|c_i\|$	44	36	39
$\text{Len}(c_i)$	45	52	39
$\text{Loc}(c_i)$	0.977778	0.692308	1

Table 2: In the table can the start u and the end v of each cluster be read as well as the cardinality of the cluster, the *temporal length* as well as the *temporal locality ratio* from the time series series sample which can be seen in figure 2.

In order to be specified as an event, according to our event model, the two clusters were not allowed to overlap in the time domain. To check this one, you can look at the table 2 and see the starting value u and the ending value v of each cluster. As it can be seen looking at the table and the figure 5 cluster 0, and cluster 1 overlap whereas cluster 0 and cluster 2 do not. Furthermore, cluster 1 and cluster 2 do not overlap in the time domain. This allows us to construct a symmetric matrix $m \times m$ (recall m is number of clusters generated by DBSCAN excluding noise) with either zeros or ones. Each row and each column represent a cluster. If the value is "0" it indicates that the clusters overlap in the time domain, if "1" is placed it indicates that clusters do not over lap in the time domain. The diagonal of the matrix is always "0" as the clusters overlap with them selves. Thus in the context of the example, a 3×3 matrix is created. And the number "1" a placed where the clusters do not overlap. Furthermore, if the temporal locality of a cluster is too low according to a level ε, it ca not be detected as an event, thus the rows an col how the respective cluster becomes zero. So using a level $\varepsilon = 0.25$, in context of the example, one can find the following matrix representation of cluster overlap and temporal locality,

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \xrightarrow{\text{Low loc}(c_1)\text{ thus}} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{10}$$

To check if the event module is fulfilled one can check the sum of all the numbers within. If the sum of the matrix is above 0 we declare that as an event occurring in the time series sample. Looking at the matrix from the example, we can conclude the event occurs between cluster 0 and cluster 2, thus we declare the interval [33964, 34001] (the end value of cluster 0 and start value of cluster 2) as a place where at least one event occurs.

5 EVALUATION AND RESULTS

To evaluate the result of the algorithm, a comparison of the result will be considered against actual labeled events from the dataset. Let *true events* be the labelled recorded event and let *detected events* be the event the algorithm returns. The algorithm has two inputs and the results are heavily affected on the choice of the two parameters. The result covered in this paper are based on the input *Window Size* = 100 and $\varepsilon = 0.4$. These parameters provide a good event detection for the dataset. An in-depth description how the parameters were found, can be read in the next section.

5.1 Parameter Tuning for The Detection Algorithm

The input parameters heavily affect the event detection, thus finding good input parameters are essential for proper event detection. Recall the input parameters,

(1) ***Window Size*** : This parameter describes how large the time series sample is, that the algorithm processes. In general the smaller the window size the more sensitive the algorithm is to noise.
(2) $\boldsymbol{\varepsilon}$: This number between [0-1[describes how much *Temporal Locality* will be allowed to each cluster. When zero, only pure clusters are allowed, thus the smallest noise will be sorted out. When the parameter is close to 1, then a lot of noise is allowed.

In order to find the optimal parameters, the event found by the algorithm has to compared to true labeled event. So we are interested in events matching *true events* and *detected events*, but we are also interested in the algorithm not classifying noise as events so we want to penalize detecting too many events as well. In order to bring this down to one number, we created the measurement score. This measurement is constructed using the following,

- **Number of *true event***:
  This parameter describes the number of *true events* on a given day in a given office.
- **Number of *detected events***:
  This parameter describes the number of *detected events* on a given day in a given office.
- **Event Ratio**:
  This parameter is defined as: Event Ratio $= \frac{\text{true event}}{\text{detected events}}$.
- **True Positives (*TP*)**:
  The number of detected events that match *true events*.
- **True Positive Rate (TPR)**:
  This parameter is defined as: TRP $= \frac{TP}{\textit{detected events}}$.
- **Score**:
  This measurement is used to define the optimal parameters. It is defined as follows,

$$\text{Score} = \begin{cases} TPR \cdot \text{for Event Ratio} & \text{Event Ratio} > 1 \\ \frac{TPR}{\text{for Event Ratio}} & \text{for Event Ratio} < 1 \end{cases}$$

  it thus penalties to few event detected and too many events detected while it rewards a high TRP.

So when the score is the highest, the input parameters are the most optimal. However running the algorithm and finding the score on just one day and one office is pretty computationally expensive so running it on all three office and 30 different days in November experimenting with different values of *Window Size* and ε would be computation infeasible. Running the algorithm on one day and one office takes on average 6.5 seconds. The code were ran on MacBook Pro (late 2013) with the processing power:

- **Processor:** 2 GHz Intel Core i7
- **Memory :** 8 GB 1600 MHz DDR3

Instead we picked a random working day as well as an office and found the optimal values function inputs for that day and that office. For this purpose, we picked the time series of Office A on the 14th November. So in order to keep the computations to a low level, we picked 9 different values for ε and 9 different values for window size so the algorithm only had to run $9 \times 9 = 81$ times. In figure 6 the 9 different values of the ε can be read on y-axis vice versa can the 9 values of window size be read on the x-axis. The figure describes how the score changes when the input parameters are changing. The highest score is achieved when *Window Size* = 160 s (2 minutes and 40 seconds) and $\varepsilon = 0.5$.

If we had ran the algorithm on the whole dataset i.e. 30 days 3 offices, and 81 parameters it would have taken roughly

$$\begin{aligned} 30 \cdot 3 \cdot 81 \cdot 6.5 &\approx 47385 \text{ seconds} \\ &\approx 790 \text{ minutes} \\ &\approx 13.15 \text{ hours.} \end{aligned}$$

Thus it is our hope that parameter turning using day 14 office A generalize well.

5.2 Results

In this section, we show the results of applying the event detection algorithm on the BLOND dataset, based on the parameter tuning, that was described in the previous section. All result displayed in this section are based on the input parameters *Window Size* = 160 and $\varepsilon = 0.5$.

The determination of the True Positives, is found by comparing the *detected events* to true event. However, because the true events are registered as time stamp, and our algorithm returns a time interval, we had to cope that. We declare a true positive, if a true event lays in the interval ±15 measurements. I.e. 15 seconds before the interval and 15 second after the interval. The 15 seconds are added to cope with noise in the signal and measurement errors of the true event. In table 3 the results of the algorithm can be seen

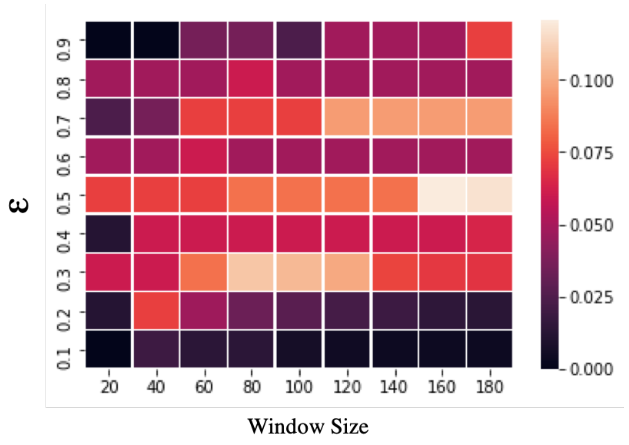

Figure 6: The heatmap describes the measurement score has been evaluated of the time series of Office A on the 14'th November. The two axis indicate different values of ε and window size. As can be seen the highest score is achieved when *Window Size* = 160 s and $\varepsilon = 0.5$.

evaluated on November 2nd, November the 7th, November the 21st and November 30th. All measurements are from office A.

	Day 2	Day 7	Day 15	Day 21	Day 30
Number of Events	90	118	73	95	82
Number of Detections	48	86	63	78	67
True Positives (TP)	1	7	2	1	4
False Positives (FP)	47	79	61	77	63
Precision	**2.1%**	**8.1%**	**3.2%**	**1.3%**	**6.0%**

Table 3: This result of the algorithm using the optimal parameters *Window Size* = 160 and $\varepsilon = 0.5$. The data is registered from office a and the days are from November 2016.

The results for the True Positives Rate (Precision) are ranging between 1.3% and 8.1% and are mostly caused by just 2 or 3 matched detection per day. Compared to the results that are presented in [14] for the BLUED dataset, which determined a True Positives Rate of 95.8% in total, it differs quite a lot.

5.3 Evaluation of The Results

As it can be seen from the result table 3 the algorithm does a very poor job in terms of hitting true positives (TP) values. In fact the job is done so poorly that it is save to say the implemented algorithm is not suited for registering the *true event* from the BLOND dataset. The registered true events are user triggered events e.g. when an employee turns on a computer. However, many true events are close to impossible register when just considering the time series data of the current RMS (the root mean square current). In these kind of measurement series the user triggered true events somehow appear to be random and not really different from noisy measurements in many cases. For example, consider figure 7 and 8, the first and second true event can be seen marked with red. As it can be seen looking at the time series, from the two figures, the implemented

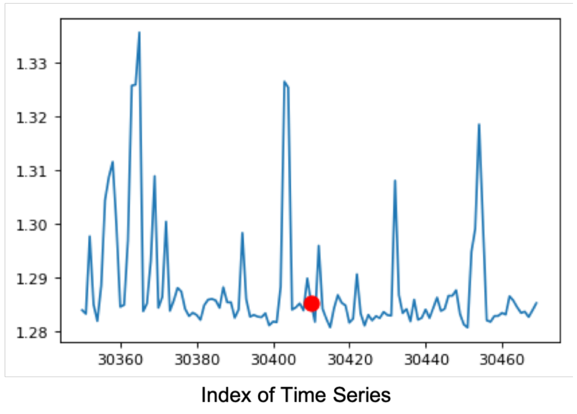

Figure 7: The first true event is marked with red on the time series sample describing office A on the 2nd of November.

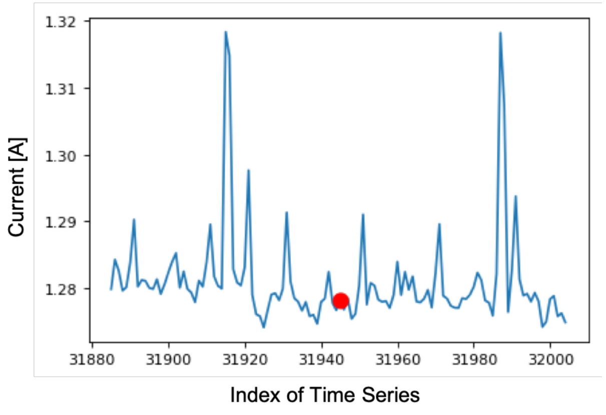

Figure 8: The second true event is marked with red on the time series sample describing office A on the 2nd of November.

algorithm would not be able to detect the true event. This is the case for many true event, that their time series sample do not indicate an event. In general, it can be seen that it would be very difficult to detect the two displayed true events. However, the issue might be a product of using the root mean square data rather than 50.000 measurements per seconds. The root mean square current is easier to work with, but it implies loss of information. Perhaps, the full dataset is necessary to make proper labeling using a high frequency algorithm.

Looking a part from the some-what disappointing result regarding matching true event to detected events, the detection of the event as stand alone tool seem quite good. Consider figure 9, which describes the time series of root means square current data of 2nd November, office A. Here the true events, has been marked with red and detected events has been marked with blue. As it can be seen, many of the detected events fall on places in the times series where big changes occur on the RMS current. In fact, we will argue that the detected event follows the change of the current better

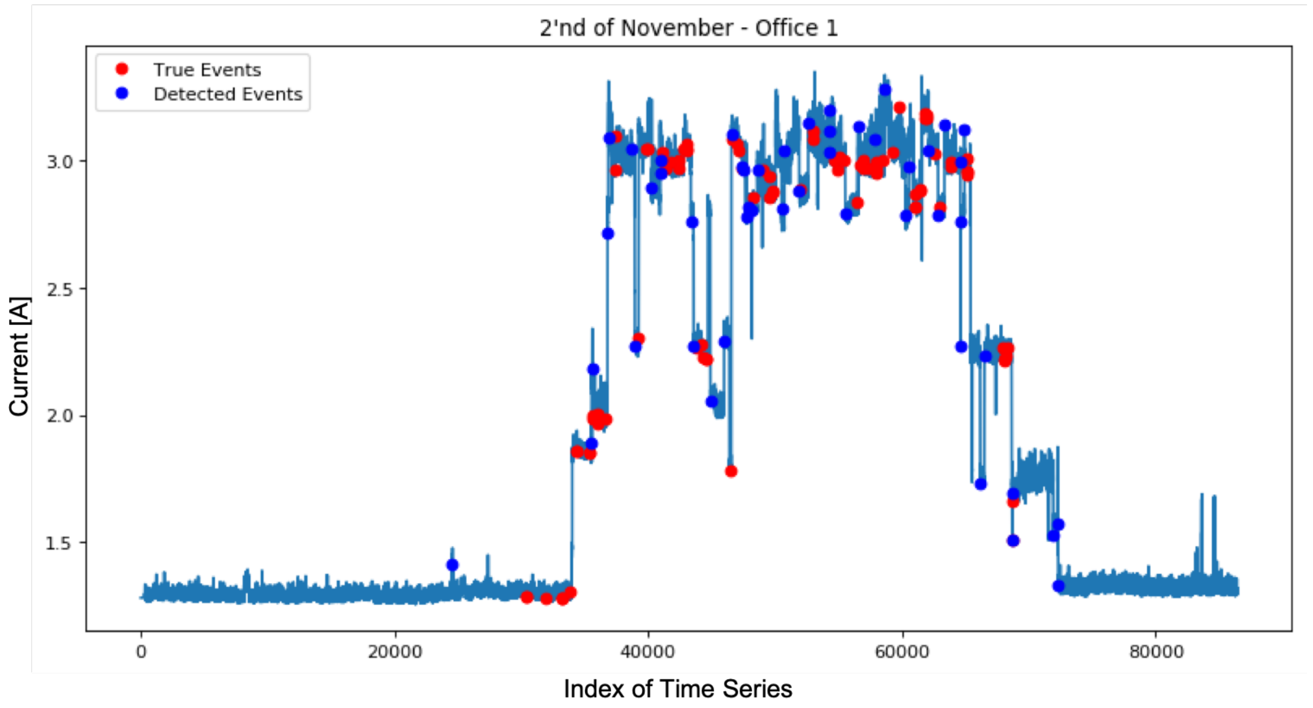

Figure 9: The full time series of Office A - the 2'nd of November with red can the true event be seen and with blue can detected events. Using the optimal parameters found in section 5.1.

than the true events. In general, the algorithm returns quite prosperous results, judging from a graphical point of view, on all the November data from all three offices when looking at figure like the one displayed in figure 9.

6 CONCLUSION

We successfully implemented a detection algorithm like the one implemented by Karim Said Barsim and Bin Yang [14] using our own event model, heavily inspired by *event model 3* from Karim Said Barsim and Bin Yan's paper. The algorithm is however not suited for matching *true events* to detected events. The event detected act much different from the *true events*. However we suspect that the *true event* labelled in the BLOND dataset are not very likely to be registered as an event using a different detection algorithm as there for many of the *true events* are no indication in the time series that an event occur. However this could potentially also be because we work with a root mean square dataset rather than the original dataset. In the original dataset there are 50.000 measurements made each second. And event could potentially be smoothed out over these 50.000 measurements. Thus using the compressed dataset we loose some information, but it a lot easier to work with. Looking a part from the match of *detected events* and *true events*, our interpretation is that the algorithm is still suited for finding event in a time series signal. Looking at the locations of *detected events* in the time series the occur when larger changes in current appear unlike many of the *true events*. Maybe the implemented low frequency algorithm still has a potential as a general NILM detection algorithm.

ACKNOWLEDGMENTS

The authors would also like to thank Daniel Jorde for his valuable comments, helpful suggestions and support. The work was done in the course of the module 'Data Analytics and Intelligent Systems in Energy Informatics' and is supported by TUM.

REFERENCES

[1] Kyle D. Anderson, Mario E. Berges, Adrian Ocneanu, Diego Benitez, and Jose M.F. Moura. 25.10.2012 - 28.10.2012. Event detection for Non Intrusive load monitoring. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 3312–3317. DOI: http://dx.doi.org/10.1109/IECON.2012.6389367

[2] K. Carrie Armel, Abhay Gupta, Gireesh Shrimali, and Adrian Albert. 2013. Is disaggregation the holy grail of energy efficiency? The case of electricity. *Energy Policy* 52 (2013), 213–234. DOI: http://dx.doi.org/10.1016/j.enpol.2012.08.062

[3] Ester M., Kriegel H.P., Sander J., and Xu X. A density based algorithm for discovering clusters in large spatial databases with noise. Vol. in proceedings of Knowledge Discovery and Data mining (KDD), 1996.

[4] A. U. Haq, T. Kriechbaumer, M. Kahl, and H.-A. Jacobsen. CLEAR – A Circuit Level Electric Appliance Radar for the Electric Cabinet. *2017 IEEE International Conference on Industrial Technology* 2017 (????), 1130–1135.

[5] G. W. Hart. 1992. Nonintrusive appliance load monitoring. *Proc. IEEE* 80, 12 (1992), 1870–1891. DOI: http://dx.doi.org/10.1109/5.192069

[6] Yuanwei Jin, Eniye Tebekaemi, Mario Berges, and Lucio Soibelman. 22.05.2011 - 27.05.2011. Robust adaptive event detection in non-intrusive load monitoring for energy aware smart facilities. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4340–4343. DOI: http://dx.doi.org/10.1109/ICASSP.2011.5947314

[7] Christoph Klemenjak and Peter Goldsborough. 2016. Non-Intrusive Load Monitoring: A Review and Outlook. (2016).

[8] Thomas Kriechbaumer and Hans-Arno Jacobsen. 2018. BLOND, a building-level office environment dataset of typical electrical appliances. *Scientific data* 5 (2018), 180048. DOI: http://dx.doi.org/10.1038/sdata.2018.48

[9] S. B. Leeb, S. R. Shaw, and J. L. Kirtley. 1995. Transient event detection in spectral envelope estimates for nonintrusive load monitoring. *IEEE Transactions on Power Delivery* 10, 3 (1995), 1200–1210. DOI: http://dx.doi.org/10.1109/61.400897

[10] Luo D., Norford L., Shaw S., and Leeb S. Monitoring HVAC Equipment Electrical Loads from a Centralized Location - Methods and Field Test Results, ASHRAE Transactions. Vol. 108, no. 1. 108, no. 1, pp. 841–857, 2002.

[11] Paula Meehan, Conor McArdle, and Stephen Daniels. 2014. An Efficient, Scalable Time-Frequency Method for Tracking Energy Usage of Domestic Appliances Using a Two-Step Classification Algorithm. *Energies* 7, 11 (2014), 7041–7066. DOI:http://dx.doi.org/10.3390/en7117041

[12] Kien Nguyen, Eric Dekneuvel, Benjamin Nicoll, Olivier Zammit, Cuong Nguyen Van, and Gilles Jacquemod. Event Detection and Disaggregation Algorithms for NIALM System. In *Jun. 2012.*

[13] Nadia Rahmah and Imas Sukaesih Sitanggang. 2016. Determination of Optimal Epsilon (Eps) Value on DBSCAN Algorithm to Clustering Data on Peatland Hotspots in Sumatra. *IOP Conference Series: Earth and Environmental Science* 31 (2016), 012012. DOI:http://dx.doi.org/10.1088/1755-1315/31/1/012012

[14] Karim Said Barsim and Bin Yang. 01022016. Sequential Clustering-Based Event Detection for Non-Intrusive Load Monitoring. In *Computer Science & Information Technology (CS & IT).* Academy & Industry Research Collaboration Center (AIRCC), 77–85. DOI:http://dx.doi.org/10.5121/csit.2016.60108

[15] Ahmed Zoha, Alexander Gluhak, Muhammad Ali Imran, and Sutharshan Rajasegarar. 2012. Non-intrusive load monitoring approaches for disaggregated energy sensing: a survey. *Sensors (Basel, Switzerland)* 12, 12 (2012), 16838–16866. DOI:http://dx.doi.org/10.3390/s121216838