Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04623-w

ATCI 2019

A new graph learning-based signal processing approach for non-intrusive load disaggregation with active power measurements

Ming-Yue Zhai[1]

Received: 2 August 2019 / Accepted: 22 November 2019

Abstract
Recently, there is a potential technology called graph-based signal processing (GSP) that is being used in many applications. GSP has been used successfully in the domains such as signal and image filtering and processing. In the paper, GSP is used as an applicable method to non-intrusive appliance load monitoring (NILM). In NILM, all of power consumption is disaggregated down to every appliance's consumption without hardware. Although there is over 30 years after NILM was proposed, there are still some problems faced by applications of NILM in real scenario if there is no training data. By combination of NILM with GSP concept, such a challenge is tackled with better performance over existing methods. As the first step, we propose a new graph learning algorithm to get a graph suitable for appliance load representation and for the disaggregation algorithm. In the following steps, graph-based signal processing method is used three times, from representation of the data sets of power measurements. Public datasets are used to demonstrate the proposed method's performance and feasibility.

Keywords Graph-based signal processing · Non-intrusive appliance load monitoring · Energy disaggregation · Graph learning

1 Introduction

In NILM or NIALM, all power consumption in one house can be disaggregated down to individual alliance level without additional sensors. Therefore, some important information can be delivered to consumers and supplying companies [1]. Although NILM was first proposed 30 years ago, some challenges are appearing when big data are obtained with large-scale smart meters deployment [2, 3]. Some reports from the business case show that energy savings far exceed costs in NILM technologies [4].

With the properties of low cost and easiness usage, NILM can be applied in household energy disaggregation. Smart meters have been widely deployed in the average household, and thus it is very important for NILM applications that the low sampling data by smart meters should be fully used to extract useful information [5]. In general, data from smart meters are with seconds or minutes sampling rates. Such low rates data cannot be used in most of the existing methods, which high rate is needed. Considering practicality and simplicity, the feasible NILM methods have focused on unsupervised schemes without training dataset [6].

Event-based NILM approaches are the first type of disaggregation schemes [7]. When power signals appear certain statistically significant changes, events are identified by edge detection. Such events may indicate that appliances have changed their working stages, for example, switching on or off, or changing operation modes [8]. After events are detected, these events will be analyzed and some characteristics can be extracted. Such features will be used to indicate certain appliances or operation modes. Many classification tools can be used in NILM, such as SVM (support vector machines), ANN (artificial neural networks), K-means, decision tree and naïve Bayes [9]. However, there are lots of factors which can affect the performances of these classification tools, such as load fluctuations, variances of active power readings, multi-stages of certain appliances.

✉ Ming-Yue Zhai
caizhai@gdupt.edu.cn

[1] School of Electronic Information Engineering, Guangdong University of Petrochemical Technology, Maoming, China

States-based NILM methods are the second class of disaggregation algorithms. In present, most of NILM methods have low resolutions, where different state transitions are used by state machine representing each appliance modes of operation. Such states can be constructed based on usage patterns. Since 2011, methods that use hidden Markov model (HMM) have become a focal point for most researchers [10, 11]. Since individual appliance internal states are not directly observed in the total consumption rate readings, hidden Markov models have been a natural choice for modeling the disaggregation process. HMMs are also a simple and efficient machine learning algorithm for modeling states over a length of time [12]. However, high computation load and local minimum restrict applications of HMM-based methods [13]. At the same times, expert knowledges are required to set prior values of each appliance state, and thus there is no feasibility to apply HMM-based methods in real-time situations.

The paper presents a different way to carry out disaggregation with GSP, where information behind energy readings is uncovered by connection between vertexes and edges in a graph [14]. By constructing a graph with energy data, signal processing methods can be performed to different applications [15]. After constructing a graph suitable for the measurement data, GSP is used to obtain adaptive threshold and perform feature matching, where adaptive thresholds are used to event detection. It is required that training data and the measurement should be with low resolutions.

Although there are some other papers presenting GSP-based disaggregation algorithms, such as [16, 17], Gaussian weights are directly used in the GSP in such methods, which is very different with the proposed method in the paper. We use the graph smoothness to calculate the weights, which can get better regulation for the graph. Therefore, the obtained graph signal by such a proposed method is more suitable to describe the aggregated data. In most of the GSP papers, graph smoothness regularization is obtained by optimization of l_1 norm. However, we find l_2 norm is more suitable for the power consumption data. From the above discussion, the paper introduces a new NILM algorithm which is based on graph learning in GSP. The algorithm and results in the paper have not found in the published papers.

Requisite knowledge and basic concepts are described in Sect. 2; these fundamentals set the basis of the GSP-based algorithm. Section 3 describes problem formulation in detail and the proposed NILM algorithm flow. Section 4 describes the details of the proposed method. Section 5 simulation results and conclusions are found in Sect. 6.

2 Fundamentals of graph signals

With the development of the society, many applications of current interests, like social networks, vehicular networks, big data or biological networks, have gained increasing attention in the research community, and thus a very powerful mathematical tool, graph theory, is employed to model and process such data [18]. In graph theory, graphs are used to model relationships and structures hidden in the data. In a graph representation, entities are represented by vertexes while relationship among entities is represented by edges. Graphs can be considered as generic data representation forms, and thus they are very useful to describe the geometric structures of data domains. Thus, there are many applications from social to neural networks [19].

Besides the entities and edges in the graph, there are also data lying on the graph. If the data on these graphs are focus of interests, samples can be used to represent such data, where one sample is corresponding to one data in the graph. In general, attributes and features associated with vertexes can be interpreted as functions or signals defined on graphs [20]. As a whole, these samples can be considered as a graph signal. An example of a graph signal is a community. The dots in the figure are vertexes and may represent residential community. The edges between different vertexes mean that there is relationship between them. If we are interested in population in different residential community, population can be modeled as a graph signal lying on the community graph.

Many examples of graph signals can be found in many engineering and science fields. For example, transportation networks can be modeled as graphs. If epidemiological data are supposed to describe spread of disease, the data can be considered as graph signals lying on the transportation networks [15]. If we are interested in the census data describing human migration patterns, or logistics data describing inventories of trade goods (e.g., gasoline or grain stocks), they are the same and can be treated in a similar way. Some other examples include temperatures measured by a sensor network, ages of the individuals in a social network, Internet traffic in a router network, etc. These are all examples of applications of graph signals [16–18].

In graph signals, an element of a graph signal is placed on a vertex, and scalar values are assigned to every edge between two vertexes as edge weights, and thus graph signals can reveal structural information about signals, such as singularities or irregular structural patterns. That means the intrinsic topology of the underlying graph data domain can be captured by processing the data residing on the vertexes of a weighted graph. Therefore, for the data we are concerned, graph-based data representation, or called graph signals, is flexible and adaptable to incorporate relationships with structures among them, yet remaining sufficiently simple for efficient processing. This is the reason that graph signal processing is attracting great interest in the signal processing community [18].

In graph signal processing (GSP), a dataset $\mathbf{X}$ can be represented by (or transformed into) a signal $\mathbf{S}$ whose index is determined by a weighted undirected graph defined by [19]

$$\mathbf{G} = \{\mathbf{V}, \mathbf{E}, \mathbf{W}\} \tag{1}$$

where $\mathbf{V} = \{v_1, v_2, \ldots, v_m\}$ and $\mathbf{E} = \{e_{ij}\}$ correspond to the graph's node and edge sets, respectively, and

$$\mathbf{W} = [\boldsymbol{\omega}_{\mathbf{ij}}] = \begin{cases} \boldsymbol{\omega}(\mathbf{e}_{\mathbf{ij}}) & \text{if } e_{ij} \in \mathbf{E} \\ \mathbf{0} & \text{if } e_{ij} \notin \mathbf{E} \end{cases} \tag{2}$$

where $\boldsymbol{\omega}_{\mathbf{ij}}$ is each edge's weight with non-negative values in $\mathbf{E}$ with $\omega_{ij} = \omega_{ji}$. We can also rewrite the graph signal $\mathbf{S}$ by a function

$$f : V \to \mathbb{R}^n \tag{3}$$

where each vertex is assigned a scalar value.

In most of research works, Laplacian matrix is used to capture the irregularity of the graph. For graph signal processing, a valid graph topology formula is in the form of Laplacian matrix and thus can support signals [20]. Besides graph's irregularity, generic structure of the measured signals can also be captured by the Laplacian matrix [5]. The Laplacian matrix, therefore, is of importance in the GSP. The combinatorial graph Laplacian matrix $\mathbf{L}$, which is an m by m matrix, is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{4}$$

where $\mathbf{W}$ is the weighted adjacency matrix of the graph and $\mathbf{D}$ is the degree matrix whose diagonal elements are degrees of graph's vertexes. The degree matrix $\mathbf{D}$ is expressed as:

$$\mathbf{D} = \operatorname{diag}(d_{ii}) = \sum_{j=1}^{m} \omega_{ij} \tag{5}$$

Since $\mathbf{L}$ is real and symmetric, there is a complete set of orthogonal eigenvectors and corresponding eigenvalues. Therefore, it can be written as:

$$\mathbf{L} = \boldsymbol{\chi} \Lambda \boldsymbol{\chi} \tag{6}$$

where $\boldsymbol{\chi}$ is the eigenvector matrix, which of each column is eigenvector, and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix where each diagonal element is the corresponding eigenvalue in non-decreasing order. 0 is the smallest eigenvalue with a multiplicity equal to the number of connected components in graph $\mathbf{G}$. Therefore, if graph $\mathbf{G}$ has c connected components, then the rank of $\boldsymbol{\Lambda}$ is m-c [21]. With the help of the Laplacian matrix, graph smoothness f on $\mathbf{G}$ can be obtained in terms of a quadratic form:

$$S(f) = f^T \mathbf{L} f = \frac{1}{2} \sum_{i,j} \omega_{ij} [f(i) - f(j)]^2 \tag{7}$$

where ω_{ij} is the weight assigned to the edge connecting two adjacent vertexes v_i and v_j, and $f(i)$ and $f(j)$ are the signal values associated with these two vertexes. When the graph signal f, a vector here, is expanded to a matrix $\mathbf{X}$, Eq. 7 is expressed as:

$$S(\mathbf{X}) = \frac{1}{2} \sum_{i,j} \omega_{ij} [\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j]^2 = \operatorname{tr}[\mathbf{X}^T \mathbf{L} \mathbf{X}] \tag{8}$$

where matrix $\mathbf{X}$ is expressed as:

$$\mathbf{X} \in \mathbf{R}^{m \times n} = [\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_m]^{\mathrm{T}} \tag{9}$$

where each row $\bar{\mathbf{x}}_i \in \mathbf{R}^n$ resides on one of m nodes of the graph $\mathbf{G}$.

Because weights ω_{ij} are non-negative, we can define graph signal smoothness. The graph signal is assumed as smooth if there is no large difference between two weights of connecting nodes. Some problems, such as regularization and semi-supervised learning, can be solved by graph signal smoothness [5, 22]. It is also playing a central role in our proposed disaggregation algorithm. With the help of the smoothness, we obtain the solutions to learning a graph suitable for the disaggregation data and to event detection for the power consumption changes.

3 Problem formulation based on graph signals

We now formulate the NILM problem to connectivity between different parts of power consumption series by the tool of graph signals and propose an appropriate implementation solution. Before going further, we first define the graph $\mathbf{G}$ lying on the active power measurements. The nodes in $\mathbf{G}$ are time indices of the power measurements, and the edge is the connectivity between these nodes. As for the weight in $\mathbf{G}$, it is the similarity between two nodes according to the power measurements defined on the nodes, which is the focus of the paper, and we will obtain them by the smoothness rules.

3.1 Problem formulation of the NILM problem by the graph signals

As stated above, graph signal has such an ability than can reveal underlying structure in signals, and at the same time graph signals can effectively capture correlation of data samples in time and space. This property of graph signals makes it a very strong method that can solve such as data mining and signal processing problems [23]. In particular, graph signal processing can also be applied to data classification problem if there are short and insufficient periods for modeling. [24]. Bearing this point in mind, we can use graph signals to deal with the NILM problem. From the signal processing point of view, the NILM problem is actually an event detection and data classification problem, where the part containing one usage model of a certain appliance is different with another. Thus, the NILM problem can be cast into event detection and data classification problems in graph signal processing.

In the NILM problem, power consumption data can be modeled as a time series $\bar{x} = [x_1, x_2, \ldots, x_n]$, where x_i, $i = 1, 2, \ldots, n$ is the consumption data at the time index i. For such a signal, according to Eq. 8, graph signal smoothness means that when the weight ω_{ij} is large, the signal values x_i and x_j tend to be related. Conversely, when the weight ω_{ij} is small, the signal values x_i and x_j are not directly related. Relationship between signal values at different times, here, means the data at different times are from the same source, such as a microwave. According to such a feature, with the help of the graph signal processing, we can discriminate which parts of the consumption data are from the same appliance source and thus can also discriminate which parts are not from a certain kind of appliance. Therefore, the data classification of the NILM problem is now casted into a GSP problem. When processing the GSP problem from the data, we use Eq. 8 as a criterion and formulate it as an optimization problem. Actually, Eq. 8 represents a measurement of the graph total variation.

With respect to the graph total variation, the generalized measurement is based on the l_p-norm [25]:

$$\mathrm{S}_p(\mathbf{x}) = \left\| \mathbf{x} - \frac{1}{|\lambda_{\max}(\mathbf{A})|} \mathbf{A}\mathbf{x} \right\|_p^p \tag{10}$$

where $\mathbf{A}$ is the graph's weighted adjacency matrix and $\lambda_{\max}(\mathbf{A})$ denotes the eigenvalue of $\mathbf{A}$ with the largest magnitude. When quantifying the graph total variation, norm l_1 measurement is often used [17]. In our method, however, we use the quadratic form of the graph total variation in Eq. 8 for two reasons. First, it is computationally easier to optimize than the l_1-norm-based graph total variation. Second, the l_1-norm-based graph total variation penalizes less transient changes than the quadratic form in the optimization problem, while our objective is to find the transient changes, and such changes are corresponding to the states changes of the appliances [17].

3.2 Formulation of graph learning from the data

In order to learn a graph from the power consumption data with the help of the graph signal smoothness, as explained above, we propose such an optimization problem as

$$\underset{\mathbf{L}}{\operatorname{argmin}}\ S(\mathbf{X}) = \underset{\mathbf{L}}{\operatorname{argmin}}\ \operatorname{tr}\left(\mathbf{X}^{\mathrm{T}}\mathbf{L}\mathbf{X}\right) \tag{11}$$

Such an optimization problem stated in Eq. 11 is quite difficult to deal with. In the works of regularization of machine learning [26], the similar optimization problem is solved with the form

$$\underset{\mathbf{X}}{\operatorname{argmin}}\ S(\mathbf{X}) = \underset{\mathbf{X}}{\operatorname{argmin}}\ \operatorname{tr}\left(\mathbf{X}^{\mathrm{T}}\mathbf{L}\mathbf{X}\right) + g(\mathbf{X}) \tag{12}$$

Inspired by the works above, we cast further the optimization in Eq. 11 as

$$\underset{\mathbf{L}}{\operatorname{argmin}}\ S(\mathbf{X}) = \underset{\mathbf{L}}{\operatorname{argmin}}\ \operatorname{tr}\left(\mathbf{X}^{\mathrm{T}}\mathbf{L}\mathbf{X}\right) + g(\mathbf{L}) \tag{13}$$

where $g(\mathbf{L})$ is a controlling function that can impose further structure using prior information on $\mathbf{L}$.

We have cast the NILM as an optimization problem with respect to the Laplacian matrix $\mathbf{L}$; however, it is not very easy to deal with. Thus, we rewrite the optimization problem in Eq. 13 as:

$$\underset{\mathbf{W}}{\operatorname{argmin}}\ \frac{1}{2}\sum_{ij} \omega_{ij}\left(\mathbf{x}_i - \mathbf{x}_j\right)^2 + h(\mathbf{W}) \tag{14}$$

For the NILM algorithm, connectivity between the part containing one kind of appliances and the part containing different kind of appliance is our focus. Thus, we set the sparsity constrain to the weight matrix, which can be achieved by the formulation of $\mathbf{W}$ as

$$h(\mathbf{W}) = 2\sigma^2 \sum_{ij} \omega_{ij}\left[\log \omega_{ij} - 1\right] \tag{15}$$

If the weight ω_{ij} is normalized, which means $\sum_{ij} \omega_{ij} = 1$, the weight ω_{ij} can be seen as the probabilities of random variables. Controlling function $h(\mathbf{W})$, therefore, is actually the entropy. According to the information theory [27], the random variables are independent when the entropy is minimized. Such a conclusion is corresponding to our case where the part containing one kind of appliance and the part containing another kind of appliance are considered independent.

According to the discussion above, we finalize the optimization problem as

$$\underset{\mathbf{W}}{\operatorname{argmin}}\ \sum_{ij} \omega_{ij}\left(\mathbf{x}_i - \mathbf{x}_j\right)^2 + 2\sigma^2 \sum_{ij} \omega_{ij}\left[\log \omega_{ij} - 1\right] \tag{16}$$

Because the first term in Eq. 16 is a convex function, we can use primal dual techniques to solve the optimization problem [28].

3.3 Formulation of events detection

As stated above, the basic idea to solve the NILM problem with GSP is to express data through a graph. It can be shown from Eq. 7 that the smoothness of a graph signal can be expressed as $f^{\mathrm{T}}\mathbf{L}f$ [10]. If we arrange $\mathbf{s} = [s_1, s_2, \ldots, s_n]$ as the time series $\bar{x}$, which is the power consumption measurements, the graph signal smoothness in Eq. 7 can be in the following form

$$S(\mathbf{X}) = S(\mathbf{s}) = \mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s} \tag{17}$$

In Fig. 1, there is a graph containing four nodes. Correction between modes is represented by bold edges, where vertical lines' length and direction reflect values in $\mathbf{L}$. If the graph signal $\mathbf{L}$ has N nodes and is also assumed as smooth with respect with underlying graph $\mathbf{G}$, there is a small global smoothness value. Regularization can be assumed as a prior with the smoothness value since changes of the signal are well captured in the Laplacian regularizer matrix $\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s}$ whose elements have relation with weights in $\mathbf{A}$. Therefore, the expected signals with the smoothest value can be written as [5]:

$$\arg\min_{\mathbf{s}} \left\|\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s}\right\|_2^2 \tag{18}$$

As stated above, $\mathbf{s}$ is an vector having n nodes and $\mathbf{L}$ is an matrix with n dimension, we can simplify the global smoothness as [5]:

$$\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s} = s_1 L_{1,1} s_1 + s_1 \mathbf{L}_{1,2:n}\mathbf{s}_{2:n} + \mathbf{s}_{2:n}^{\mathrm{T}}\mathbf{L}_{2:n,1}s_1 + \mathbf{s}_{2:n}^{\mathrm{T}}\mathbf{L}_{2:n,2:n}\mathbf{s}_{2:n} \tag{19}$$

where s_1 represents the first component in $\mathbf{s}$, $\mathbf{s}_{2:n}$ denotes a sub-vector $[s_2, s_3, \ldots, s_n]$, $\mathbf{L}_{2:n,2:n}$ is a sub-matrix of $\mathbf{L}$ whose rows and columns are extracted from 2 to n.

In the papers [29, 30], there is a vector whose elements are training samples in the place of s_1. Here, a random sample s_1 is selected which makes training unnecessary. Because of diagonally symmetric $\mathbf{L}$ and $\mathbf{A}$ is symmetric and diagonal matrix $\mathbf{D}$, the first term in Eq. 19 can be any value because it is not related with the result. Therefore, the following formula holds [5]:

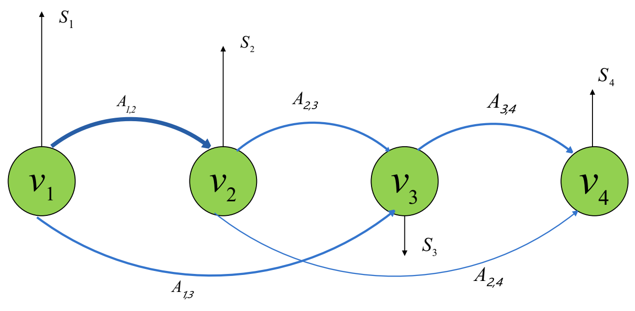

Fig. 1 A GSP model with four nodes

$$\arg\min_{\mathbf{s}} \left\|\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s}\right\|_2^2 = \arg\min_{\mathbf{s}_{2:n}} \left(2\mathbf{s}_{2:n}^{\mathrm{T}}\mathbf{L}_{2:n,1}s_1 + \mathbf{s}_{2:n}^{\mathrm{T}}\mathbf{L}_{2:n,2:n}\mathbf{s}_{2:n}\right) \tag{20}$$

Equation 20 can be solved with following expression [5]:

$$\mathbf{s}^* = \mathbf{L}_{2:n,2:n}^{\#}(-s_1)\mathbf{L}_{1,2:n}^{\mathrm{T}} \tag{21}$$

where $\mathbf{s}^*$ is $\mathbf{L}'s$ sub-matrix's pseudo-inverse matrix. That the solution of Eq. 21 is with $n-1$ element vector.

3.4 Formulation of disaggregation

All known appliances can be grouped into a set M, and the number of elements in M is represented by $|M|$. At time slice t_i, smart meter reading is $P(t_i)$. For simplicity, $P(t_i)$ is replaced by term P_i below. At the same times, energy consumption of appliance $m \in M$ is P_i^m at time slice t_i. Energy disaggregation here is trying to get each appliance's energy consumption P_i^m from only readings P_i at each moment t_i, for $i = 1, \ldots, n$:

$$P_i = \sum_{m=1}^{|M|} P_i^m + n_i \tag{22}$$

In the real environment, there may have unknown appliance working in the house, energy consumption of such devices plus reading noise can be written as n_i at moment t_i.

4 GSP-based disaggregation algorithm

In the section, the proposed method is detailed. There are graph learning, event detection, threshold adaption, data clustering and feature matching in the proposed scheme. Figure 2 illustrates the last 4 steps, because the first step, graph learning, can be separated from these four steps.

4.1 Event detection

With energy consumption reading P_i at moment t_i, for $i = 1, 2, \ldots, n$, changes of energy consumption can be obtained between neighboring moment as $\Delta P_i = P_{i+1} - P_i$, for $i = 1, \ldots, n-1$. A significant change in smart meter readings appears when an event happens corresponding to certain appliances, whose operation modes are switched at the corresponding time slice [4]. In this step, we must set the value of threshold T_0 for detecting events. All values of $\Delta P \in (-\infty, T_0) \cup (T_0, +\infty)$ will be grouped into a set of candidates Q, whose elements can present a series of events [31]. The initial threshold T_0 should be set properly. If it is too large, small power

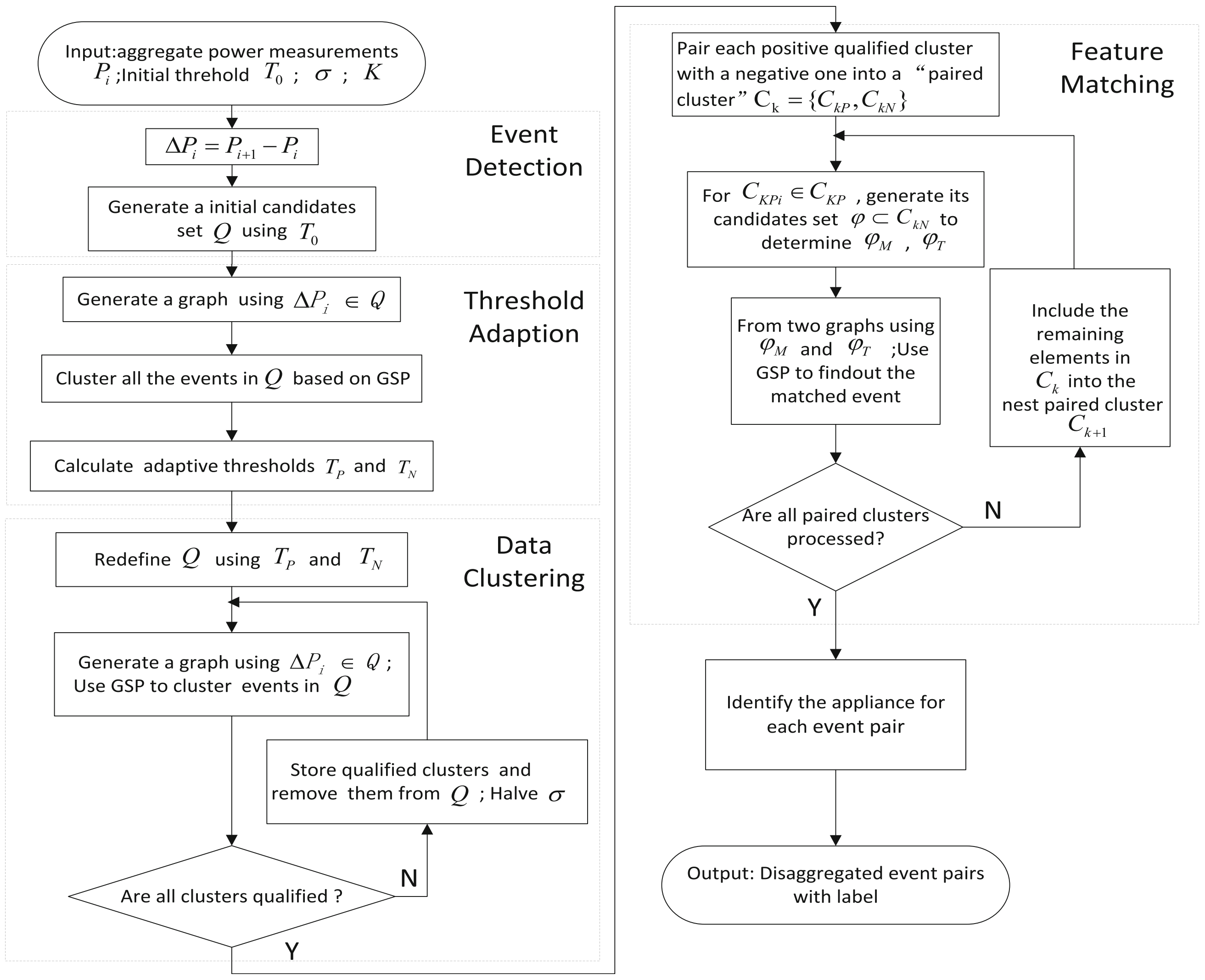
Input:aggregate power measurements P_i ;Initial threhold T_0 ; σ ; K
Event Detection
$\Delta P_i = P_{i+1} - P_i$
Generate a initial candidates set Q using T_0
Threshold Adaption
Generate a graph using $\Delta P_i \in Q$
Cluster all the events in Q based on GSP
Calculate adaptive thresholds T_P and T_N
Data Clustering
Redefine Q using T_P and T_N
Generate a graph using $\Delta P_i \in Q$; Use GSP to cluster events in Q
Store qualified clusters and remove them from Q ; Halve σ
Are all clusters qualified ?
N
Y
Feature Matching
Pair each positive qualified cluster with a negative one into a "paired cluster" $C_k = \{C_{kP}, C_{kN}\}$
For $C_{KPi} \in C_{KP}$, generate its candidates set $\varphi \subset C_{kN}$ to determine φ_M , φ_T
From two graphs using φ_M and φ_T ;Use GSP to findout the matched event
Include the remaining elements in C_k into the nest paired cluster C_{k+1}
Are all paired clusters processed?
N
Y
Identify the appliance for each event pair
Output: Disaggregated event pairs with label

Fig. 2 Flowchart of the GSP-based NILM algorithm

devices are unable to be detected. If it is set too small, reading noises or power fluctuations can cause an event detection, which is not necessary.

4.2 Threshold adaption

In this step, better threshold is supposed to be found with all available energy measurements. Such adaptive thresholds are applied to make candidate set Q more compact.

In particular, a specific graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{W})$ is constructed, where a node $v_i \in \mathbf{V}$ maps an element $\Delta P \in Q$, ω_{ij} denotes the weight of the edge between nodes v_i and v_j calculated by the proposed scheme in Sect. 3.2, where $x_i = \Delta P_i$ and $x_j = \Delta P_j$. We set s_1 to 1 if $\Delta P_1 > T_0$ and -1 otherwise, and initialize all $s_j = 0$, for $j > 1$. Then based on the concepts of GSP, we calculate $\mathbf{s}_{2:N}$ using Eq. 21.

Now, we define q as a constant with value less than 1; if $s_i^* > q$, then the parameter ΔP_i will be set to the first cluster (plus ΔP_1) and will be removed it from Q [4]. After this iteration by detecting the candidate set's element, the first class C_1 is obtained. In the second iteration, the above steps will be repeated in order to compute all elements in Q and to get the second class C_2. If there is no element in set Q, such a procedure will end, and in the end a range of classes will be obtained as results.

In this paper, a positive event will be defined when $\Delta P_i > 0$ while a negative event will be defined $\Delta P_i < 0$. When above iterations finish, there are only positive or negative events in each cluster, which can be called positive cluster and negative cluster, respectively. The mean value of C_i is defined as μ_i, and the standard deviation of C_i is defined as σ_i. For adaptive threshold calculation, relative standard deviation (RSD) R_i is selected to compute cluster's quality of C_i [4]

$$R_i = \left|\frac{\sigma_i}{\mu_i}\right| \tag{23}$$

In the next step, all positive clusters' mean values are to be computed, and the largest RSD value is set to be the updated positive threshold T_{P}. In the similar way, the updated negative threshold T_{N} is defined by the mean value of the negative cluster with largest RSD value. At the end, a more proper candidate set Q can be obtained:

$$Q = \Delta P \in (-\infty, T_{\mathrm{N}}) \cup (T_{\mathrm{P}}, +\infty) \tag{24}$$

4.3 Data clustering

In the similar way of last step, all elements in dataset Q are to be processed by GSP concept. Finally, a consequent set of clusters become available. In the next, those clusters with $R_i > K$ are to be processed further, where K is considered as a constant representing the satisfying quality of a certain class. At the same times, scaling factor σ is decreased to half for increasing the degree of class. Those classes whose values with $R_i \le K$ will be stored as candidate classes and will be computed in the following feature matching step. If all elements in set Q are grouped into a candidate class, the above steps will stop.

If positive candidate classes and negative ones are unequal, then they can be merged into a class group whose elements are greater.

4.4 Feature matching

Now, positive candidate classes and negative ones are equal after above processing; every positive and negative classes will be paired together when mean values are the closest. And thus, there are m paired clusters. As to each paired cluster, there exist two matched elements with one in positive class and the other in the paired negative one by applying GSP concepts. In order to reduce computation load, there are two matching features used in this step.

For $k = 1, 2, \ldots, m$, each paired class is defined as $C_k = \{C_{kP}, C_{kN}\}$, where C_{kP} and C_{kN} denote positive class and negative class, respectively. Those m paired classed are re-arranged in decreasing order of absolute magnitudes. The magnitude-oriented largest paired cluster is defined as C_1. For each $C_{1P_i} \in C_{1P}$, its matched element $C_{1N_j} \in C_{1N}$ will be found then. In particular, two graphs are constructed based on only events in C_{1N}, happening during the time interval after C_{1P_i} and before $C_{1P_{i+1}}$. These selected events in C_{1N} are grouped into set φ. We can set φ_M be absolute magnitude differences set between C_{1P_i} for every candidate in φ, and set φ_T be time intervals set between C_{1P_i} for every candidate in φ. There are two graphs with φ_M and φ_T being formed:

The first graph The first one $G_M = \{V_M, A_M\}$, with $A_{Mi,j}$ calculated by the method in Sect. 3.2 using $x_i = \varphi_{Mi}$ and $x_j = \varphi_{Mj}$. The graph signal s_{M1} is set to be mean value of all elements in φ_M and initialize all $s_{Mj} = 0$, for $j > 1$.

The second graph The second one $G_T = \{V_T, A_T\}$, with $A_{Ti,j}$ calculated by the method in Sect. 3.2 using $x_i = \varphi_{Ti}$ and $x_j = \varphi_{Tj}$. The graph signal s_{T1} is set to be median value of all elements in φ_T and initialize all $s_{Tj} = 0$, for $j > 1$.

Next Eq. 21 is used to compute two graph signals $\mathbf{s}_M^*$ and $\mathbf{s}_T^*$, and then the matched negative event of C_{1P_i} can be obtained when solving the following optimization problem:

$$\arg\max_i \left[\alpha \mathbf{s}_{M_i^*} + \beta \mathbf{s}_{T_i^*}\right] \quad \text{for } i = 1, 2, \ldots, |\varphi| \tag{25}$$

where the number of candidates in φ is defined as $|\varphi|$. Two parameters α and β, with $\alpha + \beta = 1$ are constant coefficients in order to tradeoff two matching features. Equation 25's results are matched negative events' index in C_{1P} for corresponding positive event in C_{1N}.

When each event in C_{1P} has its feature matching, the matched pairs (a positive event and its matched negative event) will be removed from C_1. All remaining elements in C_1 will be added into the next paired cluster C_2. If all paired clusters are processed, such iteration will end.

5 Results and discussion

There are three evaluation metrics to be used for evaluating performance of disaggregation algorithm. These three metrics are precision (PR), recall (RE) and F-Measure (F_{M}) [11] with definitions:

$$\begin{aligned} \mathrm{PR} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ \mathrm{RE} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\ F_{\mathrm{M}} &= 2\frac{\mathrm{PR} * \mathrm{PE}}{\mathrm{PR}+\mathrm{PE}} \end{aligned} \tag{26}$$

To show the competitive performance of the proposed GSP-based NILM algorithm in the paper, we conducted experiments by using two public dataset. One public dataset is REDD dataset [32], and the other is AMPds dataset [33].

5.1 Experiments on REDD dataset

For evaluating feasibility of the proposed algorithm, the publicly available REDD database is used with the proposed GSP-based NILM algorithm. The readings with low frequency in House 2 are down-sampled further to one minute. The acceptable threshold is set $K = 10\%$ for evaluating cluster quality RSD. In order to cluster samples with close distances, parameters are set as $\alpha = \beta = 0.5$, and $q = 0.98$. To avoid detecting stand-by mode, parameter is set as $T_0 = 10\,\mathrm{W}$.

In the first example, measurement data are used from an ideal condition, where there are no noise and unknown appliances. Refrigerator, dishwasher and microwave are selected from House 2, giving rise to the total energy consumption. The corresponding results are shown in Table 1 with high accuracy F_M.

To demonstrate the proposed method's applications in real environment, the data from House 2 from the REDD database are used where there are 6 appliances. Figure 3 shows the changes of smart meter reading during a common day.

Results shown in Table 2 are for REDD data from House 2. No training data are used in the experiment. The disaggregation accuracies are acceptable for most of appliances. However, stove is with the worse performance because of a high FP. Dishwasher's disaggregation results are degraded by low RE caused by a high value of FNs. In addition, there are three different operating modes (around 20 W, 250 W, 1210 W, respectively) for dishwasher. Refrigerator and lighting's high value of FPs are caused by over-smoothing.

For the same data, we also carry out the comparison between the proposed weights computation method and the ordinary Gaussian weights. The Gaussian weights are commonly used in the GSP-based schemes. The corresponding results can be found in Table 3. From Tables 3 and 4, we can see that the proposed weights computation method is with a better performance over the Gaussian weights method, especially for the larger power appliances, such as microwave, dishwasher and stove. However, for the refrigerator with lower power, there is no obvious improvement. Part of the reasons may be that refrigerator is an always-on appliance. For such an appliance, its consumed power always exits in the whole observing duration. Therefore, the entropy of the measurements, which is the base of the proposed weights computation method, cannot result in different weights from the Gaussian weights.

Table 1 Performance of the GSP-based approach in ideal condition

Appliance	TP	FP	FN	PR	RE	FM
Refrigerator	364	15	104	0.96	0.78	0.86
Microwave	86	0	8	1	0.91	0.96
Dishwasher	34	13	6	0.72	0.85	0.78

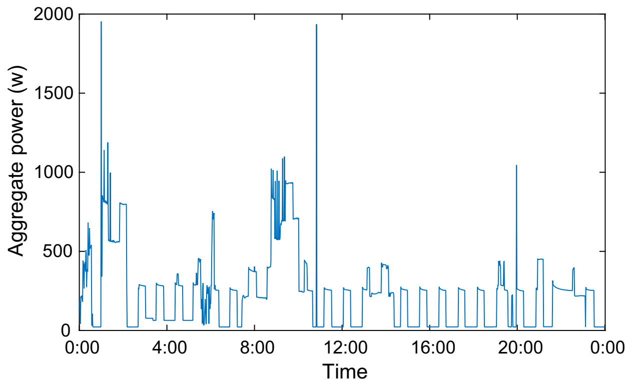

Fig. 3 Aggregate power change in April 19, 2011

Table 2 Performance of the proposed approach for House 2 from the REDD dataset

Appliance	TP	FP	FN	PR (%)	PE (%)	FM (%)
Microwave	85	0	5	100	94	96
Refrigerator	584	116	160	83	78	80
Dishwasher	28	16	20	64	58	60
Stove	35	85	6	29	86	43
Lighting	76	120	11	39	87	54

Table 3 Performance of the GSP-based approach with Gaussian weights

Appliance	TP	FP	FN	PR (%)	PE (%)	FM (%)
Microwave	85	0	7	100	92	96
Refrigerator	580	120	165	83	78	80
Dishwasher	25	19	25	57	50	53
Stove	33	87	10	28	77	40
Lighting	74	122	13	38	85	52

Table 4 Performance of the proposed disaggregation algorithms using AMPds dataset

Load	Precision (%)	Recall (%)	FM (%)
Dryer	99.76	99.79	99.08
Dishwasher	21.1	26.0	23.3
Heat pump	97.43	97.7	97.56
HVDC	98.16	98.19	98.7
Oven	99.65	99.61	99.63

5.2 Experiments on AMPds dataset

Since 2011, NILM methods using HMMs have become a focal point for most researchers. Among those methods based on HMMs, factorial HMMs are becoming a focus. Therefore, the typical factorial HMMs-based method in [34] is used to compare the performances with the proposed algorithm in the paper. Comparison results are listed in Tables 4 and 5. From these two tables, the proposed method has a little better performance than the method based on factorial HMMs. For example, the proposed method gets 99.65% precision for ovens disaggregation while the factorial HMMs-based method gets 98.25% precision.

At the same time, the results in Table 4 also verify that the proposed method can work well on the AMPds dataset. NILM research generally contains real power readings, with the data often being too coarse for more sophisticated analysis algorithms, and often covering too short a time period. We present the Almanac of Minutely Power dataset (AMPds) for load disaggregation research; it contains one year of data that includes 11 measurements at one minute intervals for 21 sub-meters. AMPds also include natural gas and water consumption data. Finally, we use AMPds to present findings from our own load disaggregation algorithm to show that current, rather than real power, is a more effective measure for NILM. There is a point to be noted that precision for dishwasher is very low. Part of the reasons is that dishwasher signature is confused with other appliances when dishwasher is running in deferent states.

6 Conclusions

In this paper, a graph learning-based NILM algorithm is introduced as a solution for smart meter reading data without any training. According to simulation results from REDD dataset and AMPds dataset, the proposed method has the ability to achieve good performance. The comparisons between the proposed algorithm and other commonly used disaggregation methods are also presented, which show the graph learning-based algorithm has better performance to some degree.

In the future, the proposed graph learning based can be further extended to more efficient solvers that can be used in real time. Also, sophisticated models such as hidden semi-Markov model (HSMM), conditional HSMM, difference HMM, and the difference HSMM can be combined with the graph learning approach, because Markov models are actually graphs.

Table 5 Performance of the disaggregation algorithms in [34] using AMPds dataset

Load	Precision (%)	Recall (%)	FM (%)
Dryer	98.38	98.25	98.62
Dishwasher	15.61	18.02	15.25
Heat pump	96.36	96.31	96.47
HVDC	97.45	97.06	97.15
Oven	98.25	98.37	98.14

Compliance with ethical standards

Conflict of interest The authors declared that they have no conflicts of interest to this work.

References

1. Zeifman M, Roth K (2011) Nonintrusive appliance load monitoring: review and outlook. IEEE Trans Consum Electron 57(1):76–84
2. D. of Energy, C. C. UK (2013) Government response to the consultation on the second version of the smart metering equipment technical specifications: part 2
3. Armel KC, Gupta A, Shrimali A, Albert G (2013) Is disaggregation the holy grail of energy efficiency? The case of electricity. Energy Policy 52:213–234
4. Zhao B, Stankovic L, Stankovic V (2016) On a training-less solution for non intrusive appliance load monitoring using graph signal processing. IEEE Access 4:1784–1799. https://doi.org/10.1109/ACCESS.2016.2557460
5. He K, Stankovic L, Liao J, Stankovic V (2017) Non-intrusive load disaggregation using graph signal processing. IEEE Trans Smart Grid. https://doi.org/10.1109/tsg.2016.2598872
6. Giri S, Berges M (2015) An energy estimation framework for event-based methods in non-intrusive load monitoring. Energy Convers Manag 90:488–498
7. Srinivasan D, Ng WS, Liew AC (2006) Neural-network-based signature recognition for harmonic source identification. IEEE Trans Power Deliv 21(1):398–405. https://doi.org/10.1109/TPWRD.2005.852370
8. Kong W, Dong ZY, Hill DJ, Luo F, Xu Y (2016) Improving nonintrusive load monitoring efficiency via a hybrid programing method. IEEE Trans Ind Inform 12(6):2148–2157. https://doi.org/10.1109/TII.2016.2590359
9. Makonin S, Popowich F, Bajic IV, Gill B, Bartram L (2016) Exploiting hmm sparsity to perform online real-time nonintrusive load monitoring. IEEE Trans Smart Grid 7(6):2575–2585. https://doi.org/10.1109/TSG.2015.2494592
10. Egarter D, Bhuvana VP, Elmenreich W (2015) Paldi: Online load disaggregation via particle filtering. IEEE Trans Instrum Meas 64(2):467–477. https://doi.org/10.1109/TIM.2014.2344373
11. Shuman DI, Narang SK, Frossard P, Ortega A, Vandergheynst P (2013) The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process Mag 30(3):83–98. https://doi.org/10.1109/MSP.2012.2235192
12. Romero D, Ma M, Giannakis GB (2017) Kernel-based reconstruction of graph signals. IEEE Trans Signal Process 65(3):764–778. https://doi.org/10.1109/TSP.2016.2620116
13. Mei J, Moura JMF (2016) Signal processing on graphs: causal modeling of unstructured data. IEEE Trans Signal Process. https://doi.org/10.1109/tsp.2016.2634543

14. Perraudin N, Paratte J, Shuman D, Martin L, Kalofolias V, Vandergheynst P, Hammond DK (2014) GSPBOX: a toolbox for signal processing on graphs. arXiv eprints arXiv:1408.5781
15. Irion J, Saito N (2016) Efficient approximation and denoising of graph signals using the multiscale basis dictionaries. IEEE Trans Signal Inf Process Over Netw. https://doi.org/10.1109/tsipn.2016.2632039
16. Lorenzo PD, Barbarossa S, Banelli P, Sardellitti S (2016) Adaptive least mean squares estimation of graph signals. IEEE Trans Signal Inf Process Over Netw 2(4):555–568. https://doi.org/10.1109/TSIPN.2016.2613687
17. Sarand HG, Karimi B (2019) Adaptive consensus tracking of non-square MIMO nonlinear systems with input saturation and input gain matrix under directed graph. Neural Comput Appl 31(7):2171–2182
18. Behjat H, Richter U, Ville DVD, Sornmo L (2016) Signal-adapted tight frames on graphs. IEEE Trans Signal Process 64(22):6017–6029. https://doi.org/10.1109/TSP.2016.2591513.25
19. Huang W, Goldsberry L, Wymbs NF, Grafton ST, Bassett DS, Ribeiro A (2016) Graph frequency analysis of brain signals. IEEE J Sel Topics Signal Process 10(7):1189–1203. https://doi.org/10.1109/JSTSP.2016.2600859
20. Zheng X, Tang YY, Pan J, Zhou J (2016) Adaptive multiscale decomposition of graph signals. IEEE Signal Process Lett 23(10):1389–1393. https://doi.org/10.1109/LSP.2016.2598750
21. Segarra S, Marques AG, Leus G, Ribeiro A (2016) Reconstruction of graph signals through percolation from seeding nodes. IEEE Trans Signal Process 64(16):4363–4378. https://doi.org/10.1109/TSP.2016
22. Tremblay N, Borgnat P (2016) Subgraph-based filterbanks for graph signals. IEEE Trans Signal Process 64(15):3827–3840. https://doi.org/10.1109/TSP.2016.2544747
23. Zhang D, Liang J (2017) Graph-based transform for 2-d piecewise smooth signals with random discontinuity locations. IEEE Trans Image Process. https://doi.org/10.1109/tip.2017.2661399
24. Fracastoro G, Magli E (2017) Steerable discrete fourier transform. IEEE Signal Process Lett. https://doi.org/10.1109/lsp.2017.2657889
25. Sandryhaila A, Moura JMF (2014) Discrete signal processing on graphs: frequency analysis. IEEE Trans Signal Process 62(12):3042–3054. https://doi.org/10.1109/TSP.2014.2321121
26. Xu X, He L, Lu H, Shimada A, Taniguchi RI (2016) Non-linear matrix completion for social image tagging. IEEE Access. https://doi.org/10.1109/access.2016.2624267
27. Bera D, Chakrabarti I, Pathak S, Karagiannidis G (2016) Another look in the analysis of cooperative spectrum sensing over nakagami-*m* fading channels. IEEE Trans Wirel Commun. https://doi.org/10.1109/twc.2016.2633259
28. Komodakis N, Pesquet JC (2015) Playing with duality: an overview of recent primal dual approaches for solving large-scale optimization problems. IEEE Signal Process Mag 32(6):31–54. https://doi.org/10.1109/MSP.2014
29. Stankovic V, Liao J, Stankovic L (2014) A graph-based signal processing approach for low-rate energy disaggregation. In: 2014 IEEE symposium on computational intelligence for engineering solutions (CIES), pp 81–87. https://doi.org/10.1109/cies.2014.7011835
30. Sandryhaila A, Moura JMF (2013) Classification via regularization on graphs. In: 2013 IEEE global conference on signal and information processing, pp 495–498. https://doi.org/10.1109/globalsip.2013.6736923
31. Zhao B, Stankovic L, Stankovic V (2015) Blind non-intrusive appliance load monitoring using graph-based signal processing. In: 2015 IEEE global conference on signal and information processing (GlobalSIP), pp 68–72. https://doi.org/10.1109/globalsip.2015.7418158
32. Welikala S, Dinesh C, Godaliyadda V, Ekanayake MPB, Ekanayake J (2016) Robust non-intrusive load monitoring (nilm) with unknown loads. In: 2016 IEEE international conference on information and automation for sustainability (ICIAfS), pp 1–6. https://doi.org/10.1109/iciafs
33. Makonin S, Popowich F, Bartram L, Gill B, Bajic IV (2013) Ampds: a public dataset for load disaggregation and eco-feedback research. In: 2013 IEEE electrical power energy conference, pp 1–6. https://doi.org/10.1109/epec.2013.6802949
34. Kolter J, Jaakkola T (2012) Approximate interence in additive factorial hmms with application to energy disaggregation. J Mach Learn Res 22(1):1472–1482