Two approaches for synthesizing scalable residential energy consumption data

Xiufeng Liu, Nadeem Iftikhar, Huan Huo, Rongling Li, Per Sieverts Nielsen

PII: S0167-739X(18)31621-2
DOI: https://doi.org/10.1016/j.future.2019.01.045
Reference: FUTURE 4738

To appear in: *Future Generation Computer Systems*

Received date : 7 July 2018
Revised date : 19 December 2018
Accepted date : 22 January 2019

Two Approaches for Synthesizing Scalable Residential Energy Consumption Data

Xiufeng Liu[a,*], Nadeem Iftikhar[b,*], Huan Huo[c], Rongling Li[a], Per Sieverts Nielsen[a]

[a]*Technical University of Denmark*
[b]*University College of Northern Denmark*
[c]*University of Technology Sydney, Australia*

Abstract

Many fields require scalable and detailed energy consumption data for different study purposes. However, due to privacy issues, it is often difficult to obtain sufficiently large datasets. This paper proposes two different methods for synthesizing fine-grained energy consumption data for residential households, namely a *regression-based* method and a *probability-based* method. They each use a supervised machine learning method, which trains models with a relatively small real-world dataset and then generates large-scale time series based on the models. This paper describes the two methods in details, including data generation process, optimization techniques, and parallel data generation. This paper evaluates the performance of the two methods, which compare the resulting consumption profiles with real-world data, including patterns, statistics, and parallel data generation in the cluster. The results demonstrate the effectiveness of the proposed methods and their efficiency in generating large-scale datasets.

Keywords: Energy Consumption, Time series, Synthesize, Simulation, Data generation

1. Introduction

Energy consumption data is one of the most popular datasets used for different studies. Many fields, such as energy, climate, buildings, and software engineering, require considerable energy consumption data for benchmarking or derived research purposes. A good example is about detecting the occupancy of a household through analysis of fine-grained residential energy consumption data, such as [1, 2]. In addition, the analysis of household energy consumption data is considered an effective way to understand consumer behavior for improving energy efficiency [3]. In our recent European H2020 project, ClairCity [4], we build the model for estimating CO_2 emission of cities, which also requires a significant size of household-level energy consumption data as its input. However, most of the studied cities lack such granular energy consumption datasets, and a few only have city or national level data. Moreover, in software engineering, scalable datasets are often used for system benchmarking purposes, including robustness, scalability, and performance [5]. Smart meter datasets have also been used for benchmarking different time-series

[*]Corresponding author
Email addresses: `xiuli@dtu.dk` (Xiufeng Liu), `naif@ucn.dk` (Nadeem Iftikhar)

management systems and the design of analytic algorithms [6, 7]. Nowadays, the ability to handle big datasets is increasingly becoming a mandatory requirement for software systems, including smart energy data management systems. Therefore, there is a high demand for large fine-grained energy consumption data for the above studies.

The challenge is that it is often difficult to obtain a scalable set of realistic energy consumption data, which, among others, are mainly for the following two reasons. One is the availability problem of smart meter data. In the past few years, many countries have begun to use smart meters, mainly in the electricity sector, when traditional grids are upgraded to smart grids. Smart meters record energy consumption in a short time interval, usually every 15 minutes or every hour. However, smart meters are mainly installed in the Western European countries and the Asia Pacific countries [8], while in most other countries, smart meters still have not been deployed. Therefore, high-resolution energy consumption data are not available in many places. The other reason is about data privacy because energy consumption data usually contain sensitive information. For example, the living habits of residents can be revealed through consumption pattern analysis. To date, a few open energy consumption datasets can be found, such as [9, 10, 11, 12, 13]. They are anonymized and limited in size. Anonymization, such as data generalization, can greatly affect the usability of the data. Due to privacy, many countries have restricted the dissemination and the use of personally-relevant data by law, including the Scandinavian countries, Denmark and Sweden. In addition, the recent enforcement of the EU General Data Protection Regulation (GDPR) [14] mandates strong privacy protection for personal data. This makes it difficult to publish any data with a bearing on personal privacy, including energy consumption data. Synthesizing data becomes the only choice for these situations.

Synthetic data generation, however, is a complex task in simulating real-world energy consumption, due to the difficulty of reproducing time series characteristics, including trend, seasonality, and pattern. For example, Figure 1 shows a fragment of an electricity consumption time series for one week, with a regular daily pattern of having a peak in the morning and a peak in the evening. The morning peak appears earlier on a weekday than at the weekend, as the household gets up earlier for work. The second peak of the weekend lasts longer than the working day, probably because the family spends more time at home on weekends and consumes more energy. A synthetic time series should be able to reflect this information.

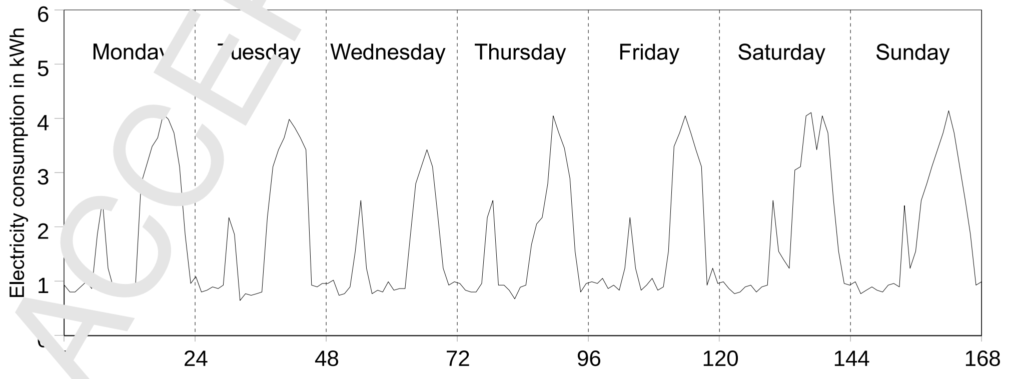


Figure 1: Weekly consumption pattern of a typical household

The objective of this paper is to propose methods and algorithms for synthesizing realistic energy consumption datasets for different needs. We present two distinct methods for generating scalable energy consumption datasets. One is regression-based, and the other is probability-based. They are both supervised machine-learning methods, including a training process and a generation process. The regression-based method allows different auxiliary data for the input, in addition to a small set of time series as the seed, which includes indoor activities, outdoor temperature time series, appliance parameters, building data, and other related data. If these auxiliary data are absent, this method is reduced to the simplest autoregression to synthesize new consumption values by prediction. In addition, we take a number of steps to optimize the data generator in order to reproduce the characteristics of real-world consumption time series as well as possible, including de- and re-seasonalization, clustering, adding base load and white noise. In contrast, the probability-based method requires only the seed as its input. This approach first identifies representative consumption patterns by clustering, then establishes a probability model, and generates new time series. For both methods, we optimize scalable data generation by implementing the program using the memory-based distributed computing framework, *Spark*. This paper is based on our conference paper [15], but with a significant extension: we have added the probability-based method, comprehensively compared the proposed two methods, and discussed several related issues regarding their differences and the selection rules.

To summarize, there are the following contributions made by this paper:

- We propose two distinctly different and novel approaches to generate fine-grained energy consumption data.
- We investigate how to simulate real-world energy consumption time series more effectively, including the preservation of patterns, seasonality, and segmentation groups.
- We propose and implement two data generators that can generate large-scale datasets in parallel in a computing cluster.
- We comprehensively evaluate and compare the proposed data generation methods, their effectiveness in simulating real-world energy consumption data, and their scalability of generating large datasets.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes the two data generation methods. Section 4 describes parallel data generation on Spark. Section 5 evaluates the two data generation methods. Section 6 presents conclusions and suggestions for future work.

2. Related Work

2.1. Energy consumption synthesis approaches

Energy consumption synthesis has raised an increasing research interest in recent decades. There are different models proposed for simulating energy consumption profiles in literature. Refs [16, 17, 18] conduct an exhaustive literature review on methods for energy consumption estimation. In overall, existing methods can be grouped into two broad categories: *top-down* and *bottom-up* approaches.

In order to tackle the lack of detailed data in the domestic energy sector, the top-down modelling approach has been extensively used and implemented. Ref [19] uses statistical data about device penetration and combines it with measurements from a substation to split load profile of the substation into individual household load profiles. Ref [20] uses a graph signal processing based method to dis-aggregate total energy consumption down to individual level, and leverages piece-wise smoothness of the power load signal. Some other top-down based approaches derive approximated load profiles by correlating with weather data or building data. For example, ref [21] generates hourly area-level electric demand profiles by scaling down national-level hourly load profiles [22], and adjusting it according to local weather conditions. However, the top-down modeling approach requires a large amount of historical data, and if there is no ancillary data available to adjust the consumption profile, it will be less accurate. The ability of the top-down approach is also questionable in identifying the improvements of demand-side energy management at the household level [17, 18].

The bottom-up approach, on the other hand, models load profiles by incorporating end-user behaviors and their interaction with, among other things, home appliances and equipment, to achieve better accuracy. Ref [23] is the first bottom-up approach to build a load profile based on a simple probability model of appliance use, for example, by assuming a 90% probability that TV sets are on during the time 19:00–21:00. This approach can approximate energy consumption of individual appliances, but the combined effects for many appliances, used to obtain the overall consumption profile, are distorted. A recent approach [24] makes further development based on this idea, which can generate more accurate realistic load profiles. The improvement is to match appliance use to indoor activities, using them to generate consumption profiles based on a statistical model. This approach, however, requires additional data such as household activities, which are often difficult to obtain. Ref [25] generates load profiles from simulating residential behaviors based on a psychological model, but it requires pre-defined very detailed household templates as its input, such as the information family members, living habits, and home appliances, and many others. More bottom-up based approaches are found for residential electricity consumption simulation, including [26, 27, 28, 29, 30]. Yet, overall, the drawback of the bottom-up based methods requires detailed additional data in order to generate accurate energy consumption time series. The additional data can be residential indoor activities, home appliances and their parameters, and consumption patterns or behaviors of households, which are not easy to obtain.

For the above reasons, some other research, instead, create the methods for generating load profiles only relying on available energy consumption time series, which employ statistics and machine learning techniques. These methods require a certain amount of real-world data to train their simulation models. For example, the studies [31, 32] use statistical averaging based approaches to generate new electrical consumption values. Ref [33] generates electricity load profiles of a household using representative data sample and statistical averages. Most often, these models generate normal, Reighley or Weibull distribution on the seasonal, hourly and social factors. Markov-chain model is used in [34] to synthesize energy consumption time series. A Markov chain is a stochastic process with a number of states, where a state may change to another state with a certain probability depending on one or more of the past states [35]. Inspired by these works, in this paper we design two methods based on data analysis to simulate energy consumption time series. One is regression-based while the other is probability-based. They both only need a small set of real-world energy consumption data as the seed, but they can generate large-scale datasets using the trained models. The generated time series have the characteristics as the training data, including pattern, seasonality, variability, and segmentation.

2.2. Models for energy consumption data prediction

The research that is relevant to the data generation models used in this paper are as follows. The proposed regression-based method uses an autoregressive centred moving average model, which is an improved version of an autoregressive moving average (ARMA) [36]. Ref [37] generates energy consumption time series using the periodic autoregressive moving average model (PARMA) that takes into account seasonality (or period) of a time series. Additionally, ref [38] uses a periodic autoregressive model with exogenous variables (PARX) for short-term energy consumption prediction. This model combines exogenous variables, such as building area, ages, types, outdoor temperatures, and resident characteristics for further improving prediction accuracy. There are many other forecast models that are suitable for fine-grained energy consumption data prediction, which, among others, include the weighted vector quantization (VQ) prediction model [39], the high-order fuzzy time-series forecasting model [40], and the hybrid autoregressive integrated moving average (ARIMA) and neural network model [41].

Ref [42] conducts a survey of time series forecasting, and concludes that regression, stochastic, neural networks, ARIMA and ARIMA variants now play a major role in time series forecasting. However, it is worth noting that other machine learning methods, Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs), are also widely used for energy consumption prediction, such as [43, 44, 45]. In recent years, as Deep Learning (DL) methods [46] have been developed, they are being used for energy prediction, as in [47, 48], which predict consumption values through the characterization of demand profiles based on measured data. DL uses multiple layer computational models to learn representations of data, resulting in a more powerful prediction capability than those obtained by ANNs [49].

3. Methods

This section describes the regression-based and the probability-based methods for synthesizing energy consumption time series. They both are supervised machine learning methods consisting of a training process and a data generation process. The regression-based method generates time series by prediction, while the probability-based method generates by a random walk on a Markov chain. They are detailed as follows.

3.1. The regression-based method

As an energy consumption time series has a number of characteristics, including *trend*, *cyclicity* and *seasonality/periodicity*, to improve simulation accuracy, some pre and post processing of training data are required. Figure 2

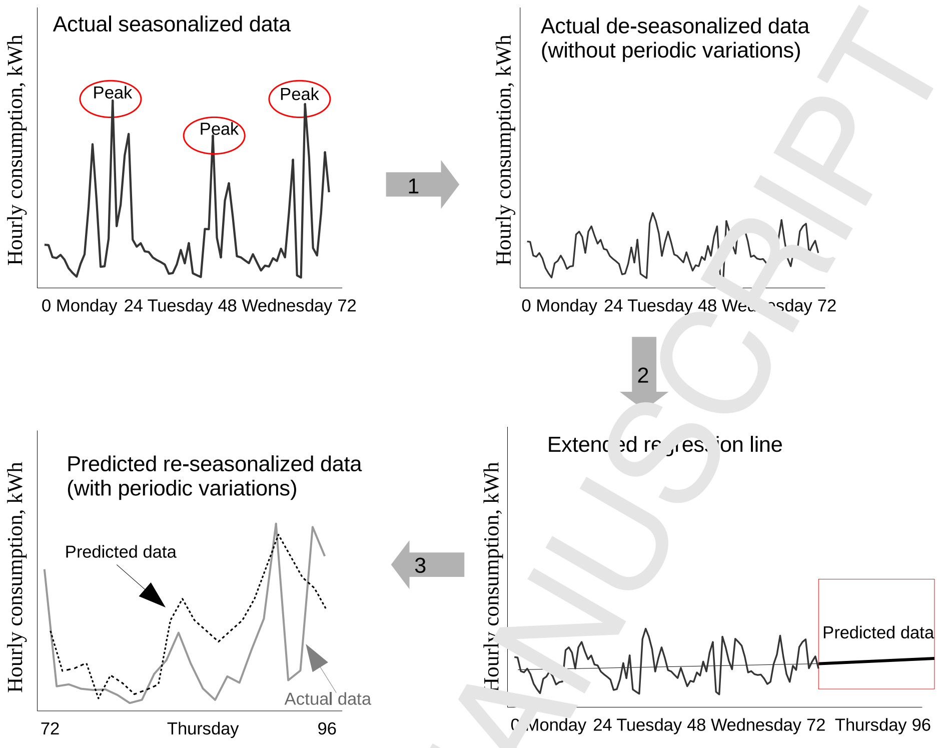

Actual seasonalized data
Peak
Peak
Peak
Hourly consumption, kWh
0 Monday 24 Tuesday 48 Wednesday 72
1
Actual de-seasonalized data
(without periodic variations)
Hourly consumption, kWh
0 Monday 24 Tuesday 48 Wednesday 72
2
Extended regression line
Predicted data
Hourly consumption, kWh
0 Monday 24 Tuesday 48 Wednesday 72 Thursday 96
3
Predicted re-seasonalized data
(with periodic variations)
Predicted data
Actual data
Hourly consumption, kWh
72 Thursday 96

Figure 2: Overview of the regression-based data generation method

shows an overview of the regression-based method. In the preprocessing, we first flatten periodic variations of a training time series, which is called *de-seasonalization*. The reason is that a model trained on a de-seasonalized time series can achieve better prediction accuracy [50]. We then use this model to generate new consumption values. In post-processing, the periodic variations are re-applied to a new time series, adding a base load that represents a fixed consumption for the day. This process is called *re-seasonalization*, which is the last step in the data generation process.

In the following, we provide more details for the regression-based data generation method, including algorithms and optimization techniques.

3.1.1. Model training process

For a time series, $X = < x_1, x_2, ..., x_n >$, the training process consists of three sequential steps, which includes fluctuation flattening, de-seasonalization and autoregressive model training. The output of the training will be used by the generation process to synthesize energy consumption time series. The three steps are described in the following.

Fluctuation flattening. The *Centred Moving Average (CMA) method* [51] is used to flatten periodic fluctuations of a time series. CMA is a sliding-window approach of using the time series mean value within a window to replace the original value in each sliding interval. In this paper, we use a daily load profile in a fixed window size of 24 hours and a sliding interval of 1 hour. The i-th value of a CMA flattened time series, $C(i)$, is defined by Equation 1. Note that as the period of 24 hours is an even number, we use the mean of two adjacent values as the CMA value, i.e.:

$$C(i) = \frac{1}{2}\left(\frac{x_{i-12} + .. + x_i + .. + x_{i+11}}{24}\right) + \frac{1}{2}\left(\frac{x_{i-11} + .. + x_i + .. + x_{i+12}}{24}\right) \tag{1}$$

where x_i represents the i-th observation of a time series.

De-seasonalization. De-seasonalization is used to reduce the periodicity of a time series. The process of de-seasonalization for a time series includes the following step.

First, the so-called *Ratio-to-Moving-Average* or *Raw-index* is defined:

$$\mathcal{R}(i) = \frac{x_i}{C(i)} \tag{2}$$

Then, the *periodic index* for each hour of the day is computed based on the raw index values (see Equation 3). The periodic index for each hour of the day is the mean value of the raw indices at the same hour in all days. For instance, $\mathcal{I}(0)$ is the mean of the values of $\mathcal{R}$ at 0 o'clock in all days. It is, therefore, a total of 24 periodic indices, each of which corresponds to an hour of the day.

$$\mathcal{I}(h) = \frac{1}{n}\sum_{i=0}^{n-1}\mathcal{R}(h + 24i) \tag{3}$$

where n is the number of days in a time series, and h is an hour of the day, i.e., 0 – 23. Due to the floating point issue [52], there is a precision problem for the value of $\mathcal{I}$. It is therefore necessary to normalize $\mathcal{I}$ to ensure that the sum of the periodic is equal to 1.0. Equation 4 performs the normalization, and derives the normalized periodic indices, I':

$$\mathcal{I}'(h) = \frac{24 * \mathcal{I}(h)}{\sum_{h=0}^{23}\mathcal{I}(h)} \tag{4}$$

Finally, the time series is de-seasonalized by using a normalized periodic index, yielding a flattened time series, $X' =< x'_1, ..., x'_i, ..., x'_n >$, in which

$$x'_i = \frac{x_i}{\mathcal{I}'(h)} \tag{5}$$

where $x_i \in X$ and $h = i \; mod \; 24$.

Autoregressive model training. The de-seasonalized time series X' is then used to train the autoregressive (AR) model. As mentioned earlier, the models trained on a flattened time series can obtain better prediction accuracy. The predicted values will be used to construct the final consumption time series. According to [53], the time series values of residential energy consumption are serially co-related, i.e., current consumption is related to past consumption, as verified by the experiment reported in Section 5.3.2. According to previous studies [38, 53, 54], when the order $p = 3$, the autoregression can achieve the best result according to the Bayesian Information Criterion (BIC) value (i.e., the lowest BIC). It is appropriate to choose the same order value for this study.

In summary, the above training process will result in the following outputs, including periodic indices, flattened time series, and AR models. The results will be passed to the data generation process for synthesizing time series. In the present application, we employ a distributed computing system, *Apache Spark*, for a parallel data generation. In our implementation, we therefore write the output directly to the Hadoop distributed file system (HDFS), and organize the output into two separate text files: one for storing periodic indices and the other for storing the AR models and the flattened time series. The records in the two files are linked by unique IDs. The purpose of this implementation is to generate a large number of realistic time series by combining the records of the two files as discussed in the next section.

3.1.2. Time-series generation process

We now describe the time series generation process using Algorithm 1. The time-series generation process uses the periodic indices, the autoregressive data and the flattened time series as the input for generating data. As mentioned earlier, we generated scalable time series on the distributed computing platform, Apache Spark. These parameters are saved in the Hadoop distributed file system, and read into two Resilient Distributed Datasets (RDDs) in Spark, $\mathcal{PI}$ and $\mathcal{AR}$. RDD is an in-memory data structure consisting of a collection of records distributed over one or more nodes so that slave workers can operate in parallel [55]. The two RDD objects have the structures of *<id, periodic-indices>* and *<id, AR-coefs, flattened-time-series>*, respectively. *Theta join* [56] is then applied to them to generate new time-series values (see line 1–13). Theta join is defined as a binary relation function in an analytical query on a database. It can be formalized as $f : wRw' \rightarrow \{1, 0\}$, where w and w' are expressions, R is the operator, $R \in \{<, \leq, =, \geq, >, <>\}$, and the function result is a boolean value, True (1) or False (0). The theta join in this case is the binary relation on the expression on *id* between the two RDDs, $\mathcal{PI}$ and $\mathcal{AR}$. Therefore, through the theta join, a large number of time series can be generated by combining the parameters from the two RDDs tables, but a relatively small dataset is needed as the seed.

The data generation process is discussed as follows, which corresponds to the line 3–12 of Algorithm 1.

(1) *Generate new consumption values:* A new consumption value is generated based on the following autoregressive function:

$$x''_i = c + \sum_{\lambda=1}^{p} \alpha_\lambda x'_{i-\lambda} \tag{6}$$

where c is a constant intercept, α_λ are coefficients, and $x'_{i-\lambda}$ are the flattened time series values (using p values before i);

(2) *Re-seasonalize the time series, and add base load, as well as white noise*, which is expressed by Equation 7.

$$x'''_i = x''_i * \mathcal{I}'(h) + baseLoad + \epsilon_i \tag{7}$$

where $h = i \bmod 24$, $i = 1, ..., n$ and ϵ is white noise. The re-seasonalization is achieved by simply multiplying the flattened time-series value by the corresponding adjusted periodic index. A base load is added to simulate the energy consumption that is independent of the activities of a household, for example, the consumption used by the appliances that are always on, e.g., refrigerators. The base load value can be obtained by averaging the consumption in the middle

Algorithm 1: The process of generated energy consumption time series

```
Input  : The periodic indices $\mathcal{PI}$(id, periodic-indices $I'$), autoregressive models and flattened time series
         $\mathcal{AR}$(id, AR-coefs, flattened-time-series), and the order of auto-regression, $p$
Output: A set of synthetic time series
1  $\mathcal{R} \leftarrow \mathcal{PI} \bowtie_\theta \mathcal{AR}$ ;            /* θ represents the theta-join operation */
2  $O \leftarrow \{\}$ ;                        /* Initialize an empty set of time series */
3  for $r \in \mathcal{R}$ do
4      $\mathcal{X} \leftarrow <>$ ;                       /* Initialize an empty time series */
5      $\alpha, < x_1, ..., x_n >, I \leftarrow$ r(AR-coefs, flattened-time-series, periodic-indices);
6      for $i \in (p + 1)...n$ do
7          $x'' \leftarrow c + \sum_{\lambda=1}^{p} \alpha_\lambda x_{i-\lambda}$ ;
8          $h \leftarrow i \mod 24$ ;
9          $x''' \leftarrow x'' * I(h) + baseLoad + \epsilon_i$ ;
10         $\mathcal{X} \leftarrow \mathcal{X} \oplus x'''$ ;            /* Append new value to the time series */
11     end
12     $O \leftarrow O \cup \{\mathcal{X}\}$ ;
13 end
14 return $O$
```

of the night or the consumption when people are away from home. A more common approach is to use 10% of the average hourly consumption value to represent the base load of a household [57], and in this paper we employ this approach. Finally, we add white noise to simulate a slight variation of each hourly consumption value. The white noise conforms to a standard normal distribution: $\epsilon \sim \mathcal{N}(0, 1.0)$.

(3) *Add new values*. The value x''' generated in the previous step is appended to the time series $\mathcal{X}$. As auto-regression is used, the time series values in $\mathcal{X}$ are all predicted values. The generated time series is added into a time-series set as the final output (line 12).

3.1.3. Optimizing data generation

We now describe the optimization techniques applied to the time series data generator. As discussed in Section 1, residential energy consumption time series have regular time patterns, such as daily, weekly or monthly. In fact, the appearance of these regular patterns is a complex issue as it is related to many factors, such as changes in the weather, building characteristics, and living habits of residents. These patterns may also show spatial and temporal characteristics. For example, the behavioral patterns of the residents in the same neighborhood may be similar, as are patterns within a certain time period. Utilities often use a clustering technique to identify customers with similar patterns in order to provide personalized energy-saving recommendations or better energy services. However, in the generation process, due to the use of theta join, the models and the flattened time series are shuffled to synthesize a time series. This operation will lead to the loss of customer segmentation information from the original time series. In order to preserve segmentation information, we optimize the training process by adding preprocessing step of clustering (see Figure 3). The clustered seeds are then used in remaining training and generating process.

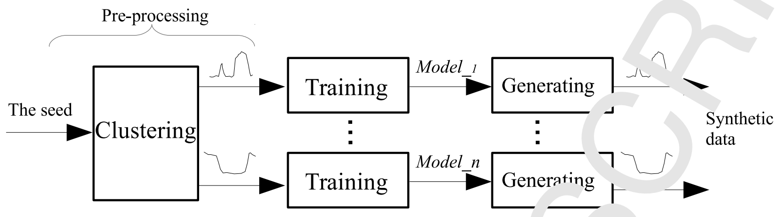


Figure 3: Preprocess the seed by clustering

More specifically, we first cluster the seed based on representative daily consumption patterns of all time series. A representative consumption pattern is the mean pattern of a time series calculated by averaging the consumption values at the same hour in all days. For example, for a time series of i, its representative daily pattern is defined as

$$\bar{X}_i = \{\bar{x}_{i,0}, ..., \bar{x}_{i,h}, ..., \bar{x}_{i,23}\} \tag{8}$$

where $\bar{x}_{i,h}$ is the mean consumption value at the hour of h of all days.

We then apply k-means clustering algorithm [58] to cluster all the representative daily patterns. Usually, k-means clustering uses the Euclidean distance as the metric to quantify the similarity between two vectors, e.g., [59, 60]. For example, the Euclidean distance of two representative daily load profiles, $\bar{X}_i$ and $\bar{X}_j$, is computed by the following equation:

$$euclDist\left(\bar{X}_i, \bar{X}_j\right) = \sqrt{\sum_{h=0}^{23}\left(\bar{x}_{i,h} - \bar{x}_{j,h}\right)^2} \tag{9}$$

However, in this study we choose the Pearson correlation distance metric [61] to optimize the clustering. The correlation distance is defined as follows:

$$corrDist\left(\bar{X}_i, \bar{X}_j\right) = 1 - corr\left(\bar{X}_i, \bar{X}_j\right) \tag{10}$$

where $corr$ is the correlation distance defined as follows:

$$corr\left(\bar{X}_i, \bar{X}_j\right) = \frac{\sum_{h=0}^{23}\left(\bar{x}_{i,h} - \mu_i\right)\left(\bar{x}_{j,h} - \mu_j\right)}{\sqrt{\sum_{h=0}^{23}\left(\bar{x}_{i,h} - \mu_i\right)^2}\sqrt{\sum_{h=0}^{23}\left(\bar{x}_{j,h} - \mu_j\right)^2}} \tag{11}$$

where μ is the mean of the representative daily patterns.

The reason is that the correlation distance is better for measuring the shape or trend of two patterns, while the Euclidean distance is for measuring the difference of attributes in values [62]. For example, the Euclidean distance of Figure 4 (a) and (b) are both $\sqrt{3}$, but we can see that the two patterns in Figure 4 (b) are completely different.

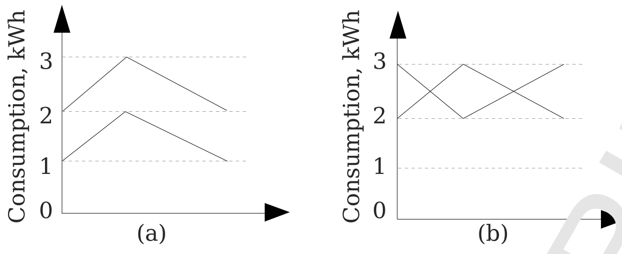


Figure 4: Two identical or opposite patterns

As the correlation value *corr* ranges between -1 and 1, the correlation distance *corrDist* will be between 0 and 2, (i.e., $corrDist = 1 - corr$). Usually, a distance of less than 0.5 represents a good similarity between two vectors. If the distance of two vectors is 0, they have identical patterns, as in Figure 4 (a). If the distance is 2, they have opposite patterns, as in Figure 4 (b).

3.2. *The probability-based method*

We now propose a second method for energy data simulation, which is a probability-based approach. The process of this method consists of two steps, including representative pattern extraction by clustering and new time series generation using a probability model. The probability model is constructed using representative daily patterns (see Figure 5). The procedure is described in the following subsections.

3.2.1. *Extracting representative patterns*

We use the adaptive clustering method [[illegible]3] to extract representative patterns from normalized daily load profiles for each household. The normalization process is defined in the following. For a household, i, the load profile of the d-th day can be represented by $X_i(d)$, where X represents an hourly consumption vector with 24 dimensions, $X \in \mathbb{R}^{24}$ and $d = 1, ..., N$. The normalized daily load profile is defined as follows:

$$X_i^*(d) = \frac{X_i(d)}{S_i(d)} \tag{12}$$

where X^* is the normalized daily load profile and $S_i(d)$ is the total energy consumption of day d. Then, the adaptive clustering is conducted based on all normalized load profiles, and the centroids of the clusters are used as the representative load profiles of a household. As the clustering is based on the normalized data, the representative patterns derived indicate only the shapes of the consumption pattern, without indicating consumption intensity. The shapes can often reflect the consumption habits or activities of a household. Finally, the representative patterns are encoded using ASCII alphabets.

3.2.2. *Time series generation*

The time series generation includes the following procedures. First, based on the derived representative patterns, each time series is converted into a sequence of ASCII alphabets. Then, the sequences are used to train a Markov

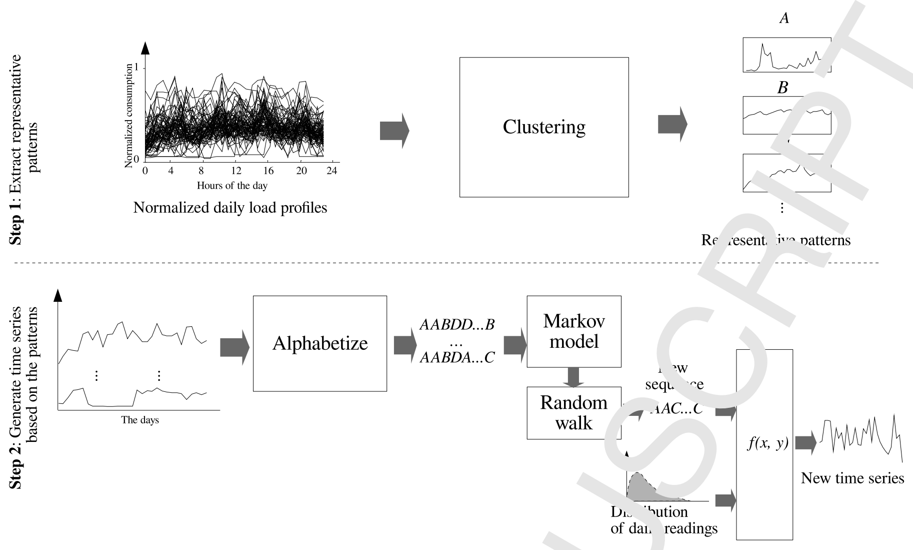


Figure 5: Overview of the probability-based simulation method

chain model, which is then used to generate new sequences by random walks. The new sequence represents the normalized consumption patterns within a series of continuous days. The sequence is then "amplified" to create a consumption time series by multiplying a random number sampled from the daily consumption distribution generated by the training dataset.

Markov chains are often used for sequence generation, e.g. [34, 64, 65]. A discrete-time Markov chain can be defined as a finite set of states, $S = 1, ..., n$, representing the events that occur at every discrete time step. The next state in a Markov chain is conditionally independent of the past states, i.e.:

$$P(S_{t+1} = j|S_t = i, S_{t-1} = i_{t-1}, ..., S_0 = i_0) = P(S_{t+1} = j|S_t = i) \tag{13}$$

where $P(S_{t+1} = j|S_t = i)$ is the probability of the transition between two states i, j. A transition probability matrix (TPM) of the size $n \times n$ is created for each concrete time step. TPM contains all the probabilities of the state transitions.

We compute the transition probability P_{ij} by the following equation:

$$P_{ij} = \frac{n_{ij}}{\sum_{k \in S} n_{ik}} \tag{14}$$

where the numerator represents the number of daily pattern changes from i to j between two continuous days, and the denominator represents the number of pattern changes from i to all states (including state i), $\sum_{k \in S} P_{ik} = 1$. When constructing the TPM, it should be noted that the training datasets may not include the transitions between all states. This will generate a sparse TPM. To address this issue, we use Laplace smoothing to increase the transitions by adding the number 1 such that no zero probability exists in the resulting TPM.

When the TPM for each time step has been derived, we generate new sequences by random walks on the Markov chain. We start from a randomly selected state, then pick each subsequent state according to the TPM corresponding to that particular time step. The resulting alphabet sequence represents a series of synthetic normalized consumption patterns. When a new alphabet sequence has been generated by a random walk, the load profiles of all days are determined, each of which is generated by the normalized daily pattern multiplied by a random number sampled from the corresponding daily consumption distribution. To capture the seasonality of the consumption behaviors, we create a distribution for each day using daily consumption values of all households in order that the sampled random number can reflect energy consumption changes over time. For example, in countries with a widespread use of electric heating, the daily consumption in winter is typically higher than it is in summer, as in Ireland and Canada.

4. Parallel data generation

We use Apache Spark for parallel data generation. Apache Spark is an open source memory-based distributed computing framework, which has implemented the distributed computing primitives, including *map* and *reduce*. Spark is optimized for iterative algorithms and interactive data analysis, which can perform iterative computations on the same dataset. The cluster for deploying Spark typically has the architecture of one master node and multiple slave nodes. The master node assigns jobs to the slave nodes and coordinates the jobs run in a cluster. A job reads the data from a Hadoop distributed file system (HDFS) or a local machine hard driver and performs computations on RDDs. The output is written to HDFS or a local machine. A job can be composed of several steps that are either maps or reduces. All data is split into multiple partitions and the computations are performed on each partition by a separate task. A task is executed by an executor on a slave node.

Using the computation mechanism of Spark, we implement parallel data generation for the proposed two methods, for both the training and the generation programs. It is worth noting that the training program does not have to be implemented using Spark as it is run only once, and the resulting models can be re-used many times to generate data. In the following, we therefore describe only how to parallelize data generation on Spark. For both of the methods, a map-only data generation program is implemented, i.e., no reducer is needed. The models generated by the training process are broadcast to the mappers which generate time series separately without inter-communication. This greatly improves performance when generating large-scale datasets (this will be evaluated by the experiments). For the regression-based method, the broadcast data are periodic indices and auto-regression coefficients. Figure 6 describes the implementation details on Spark. The data process on a mapper includes equal join, projection, theta join and synthesizing time-series values by prediction. Each mapper generates new values based on a partition of the flattened time series. A mapper does an equal join to look up the auto-regression model that corresponds to a flattened time series, and does projection to select periodic indices from the broadcast data. The resulting RDDs both will be as the input for the subsequent time series generation process presented in Algorithm 1. The final results are the new time series written back to HDFS. For the Probability-based method, it has a similar implementation process, but the broadcast data are the transition probability metrics and distribution models of daily readings. The time series generation process is conducted by random walks on a Markov Chain within a mapper. In both methods, the generated time series are then written directly as the map output to HDFS.

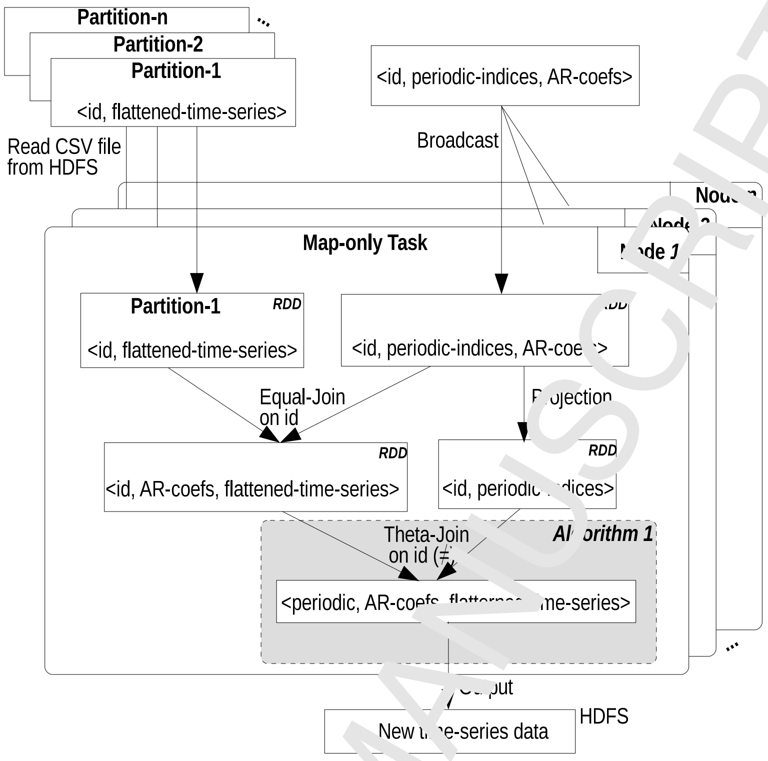


Figure 6: The implementation of regression based data generation method on Spark

5. Evaluation

This section reports an evaluation of the two proposed time series data generation methods. The Irish electricity consumption dataset [10] is used for training the models. The consumption data were collected from July 14, 2009 to December 31, 2010 with over 5,000 residential households and businesses, with a resolution of 30 minutes. We aggregate them into an hourly resolution for the experiments and we consider only residential consumption. Both of the data generation methods use clustering analysis in the training process. There is no rule-of-thumb about the least sample size for clustering analysis [66]. As we intend to maintain a relatively small size of the seed, we randomly sample 30% of the time series as the training dataset.

We evaluate the synthesized data by descriptive and exploratory analysis and compare them with the real-world data. The two proposed data generation methods are compared under different settings.

All the experiments are conducted in a computing cluster of four nodes. All of them are used slave nodes, and one of them is used as the master node. All nodes have an identical configuration: Intel CPU E5-2650 (3.40GHz, 4 Cores) with hyper-threading enabled (2 hyper-threads per core), 8 GB RAM, Hard driver (1TB, 6 GB/s, 32 MB Cache and 7200 RPM), and 64bit-Ubuntu 12.04.

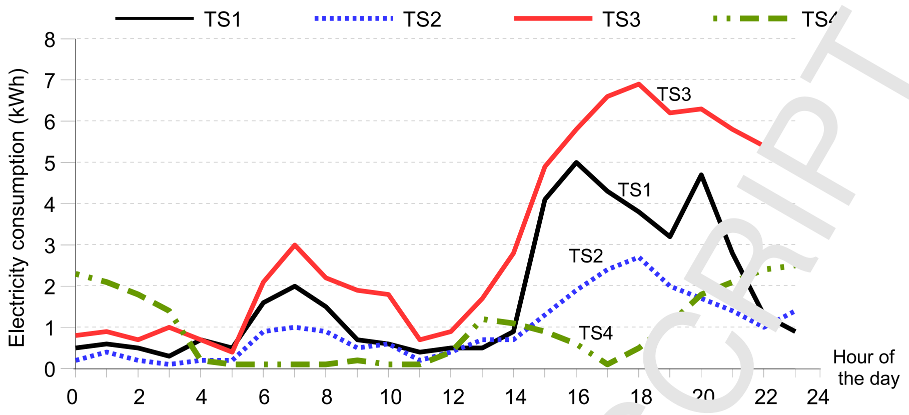


Figure 7: Four typical daily load profiles

Table 1: Pair-wise comparison of the two distance metrics

	(TS_1, TS_2)	(TS_1, TS_3)	(TS_1, TS_4)	(TS_2, TS_3)	(TS_2, TS_4)	(TS_3, TS_4)
euclDist	6.13	9.12	9.64	11.5	4.73	12.4
corrDist	0.12	0.13	1.06	0.12	0.76	1.10

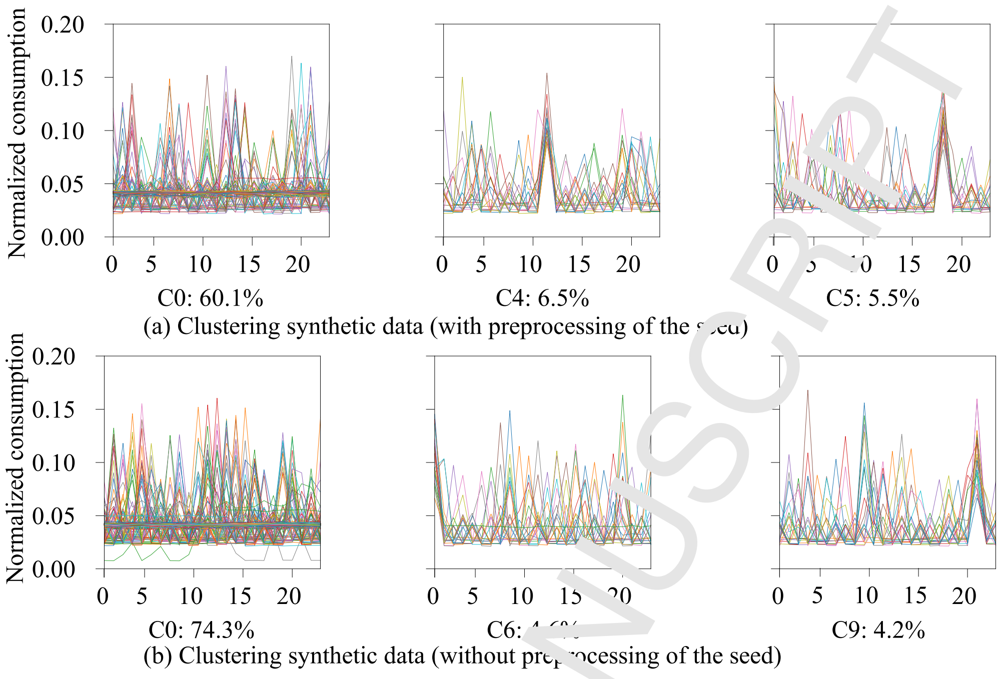
5.1. Regression-based results

As described in Section 3.1.3, we preprocessed the seed by clustering before using it to train models. Also, we used the Pearson correlation distance metric in the clustering. In the following, we will further explain this process by an example before evaluating the generated time series.

This example is shown in Figure 7, which includes typical daily load profiles from four households, denoted by TS_{1-4}. According to energy consumption intensity, TS_3 is the highest, TS_1 is medium, TS_2 and TS_4 are the lowest. According to patterns, TS_1, TS_2 and TS_3 are similar, as they have a morning peak and an evening peak within the same-length time window. In contrast, TS_4 has a different pattern, as it has no morning peak and has a low consumption at roughly 5 o'clock in the afternoon. Based on pattern similarity, TS_1, TS_2 and TS_3 should, therefore, be in the same group, while TS_4 should be in another group.

We now compare the Euclidean distance and the correlation distance for clustering time series according to pattern similarity. Table 1 shows the pair-wise comparison of the distances for the daily load patterns in Figure 7. When the correlation distance metric is used, the distances of the pairs, (TS_1, TS_2) and (TS_1, TS_3), both are smaller than (TS_1, TS_4). In contrast, the distance of the pair, (TS_2, TS_4) is the smallest when the Euclidean distance metric is used. Thus, the load profiles of TS_2 and TS_4 will be assigned to the same group. This suggests that it is more preferable to use the correlation distance metric to cluster consumption patterns or load shapes.

We will now demonstrate the necessity of preserving customer segmentation information using the clustering technique. We generate time series using the models trained by the seed with and without preprocessing, respectively. We perform adaptive clustering on the corresponding daily load profiles, and generate 20 clusters. The top three



(a) Clustering synthetic data (with preprocessing of the seed)

(b) Clustering synthetic data (without preprocessing of the seed)

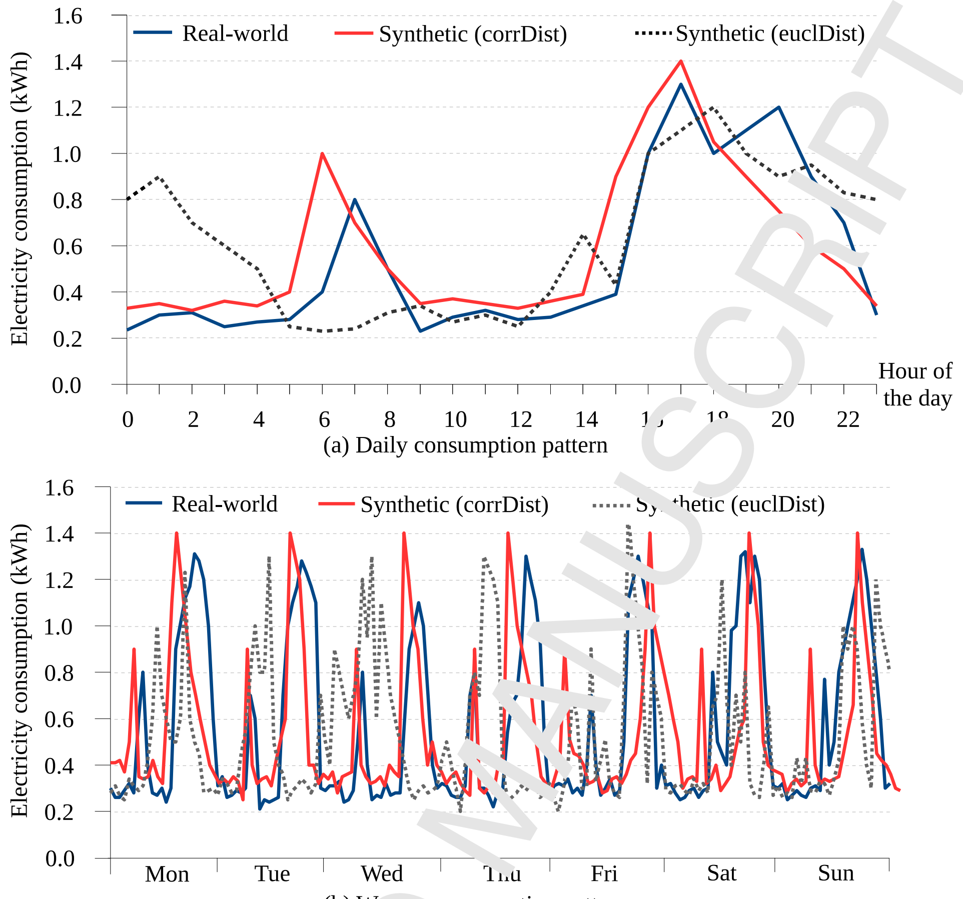
Figure 8: Comparison of pattern preservation when the seed is preprocessed and not preprocessed

clusters are shown in Figure 8. According to the figure, the load profiles in Figure 8 (a) (using a reprocessed seed) are more cohesive than in Figure 8 (b) (using an un-preprocessed seed). This demonstrates the effectiveness of the proposed clustering technique for achieving pattern preservation.

We now compare the synthetic time series with the real-world time series (see Figure 9). The blue line in Figure 9 (a) is the daily load profile of a typical household, while the other two are synthetic load profiles, which result from clustering on the preprocessed seed when correlation distance (corrDist) and Euclidean distance (euclDist) are used respectively. In Figure 9 (a), we can see that the daily load pattern of synthetic data (*corrDist*) matches well with the real-world load pattern: they both have the peaks at the hour of 6–8 and 16–18 (with a slight drift to the left). In contrast, the pattern of the synthetic data (*euclDist*) does not match well with the pattern of the real-world data, as the latter does not have a peak at 1–2 o'clock. Figure 9 (b) shows the average weekly patterns. Here, we can also see that the pattern of synthetic (*corrDist*) matches better than the synthetic (*euclDist*).

5.2. Probability-based results

For the probability-based method, the representative daily consumption patterns must first be identified from the training dataset. We use the seed for training, then use the seed again to validate the results. Adaptive clustering is implemented on the normalized daily load profiles of the seed, resulting in 20 clusters. Figure 10 shows the clustering results ordered by the number of patterns in the clusters. The 20 representative patterns are labelled with the alphabetic characters from A to T. Each time series is then transformed into a sequence of alphabetic characters according to its daily patterns. Based on the sequences, the TPMs of the Markov chain are calculated for all days, for which the



(a) Daily consumption pattern

(b) Weekly consumption pattern

Figure 9: Comparison of consumption patterns

probabilities are calculated according to Equation 14. A resulting alphabet sequence is converted into a synthetic time series by multiplying the random numbers sampled from the corresponding consumption distributions of the days.

We then evaluate the data generator by comparing the real-world and the synthetic data, by examining their statistical properties. We take the daily consumption of June as an example and calculate the average consumption of each hour of the day (see Figure 11). As can be seen, the shapes of the real and synthetic consumption curves are relatively similar, with low consumption in the early morning, becoming higher during the day and the evening. The consumption profiles for the whole month (June) are shown in Figure 12, which reveals the day-to-day patterns and the discrepancies between the time series of real-world and synthetic consumption. The result indicates that the synthetic consumption time series share a very similar pattern with the real-world time series.

Based on the above results, we believe that the probability-based method can produce reasonably realistic consumption data, and can therefore be used to assess building performance or the consumption behaviors according to patterns.

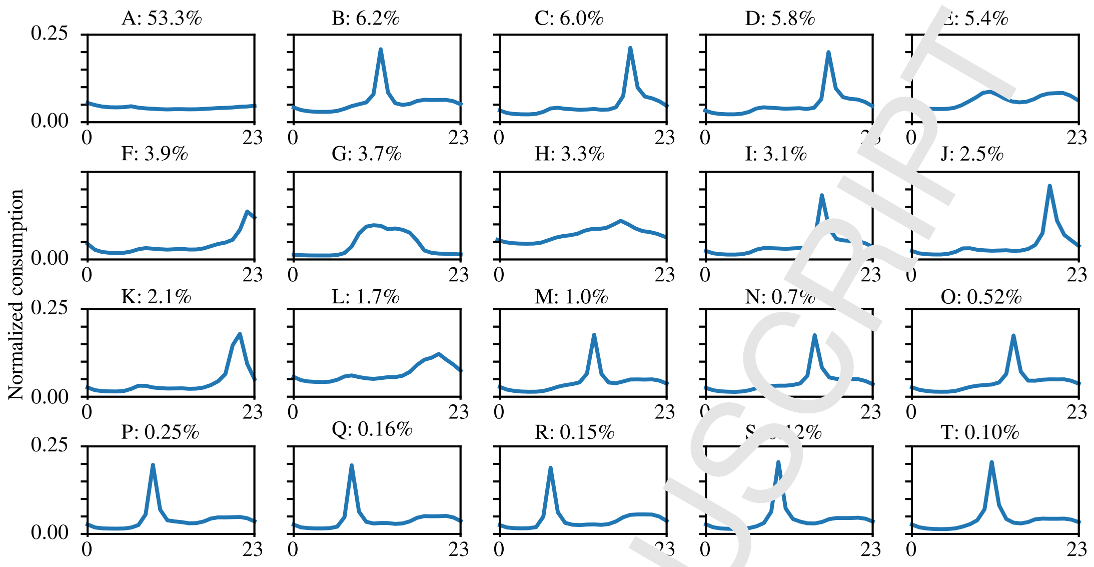


Figure 10: 20 representative patterns identified

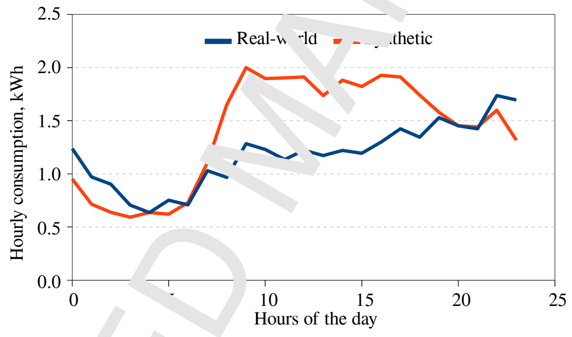


Figure 11: Average daily consumption of June

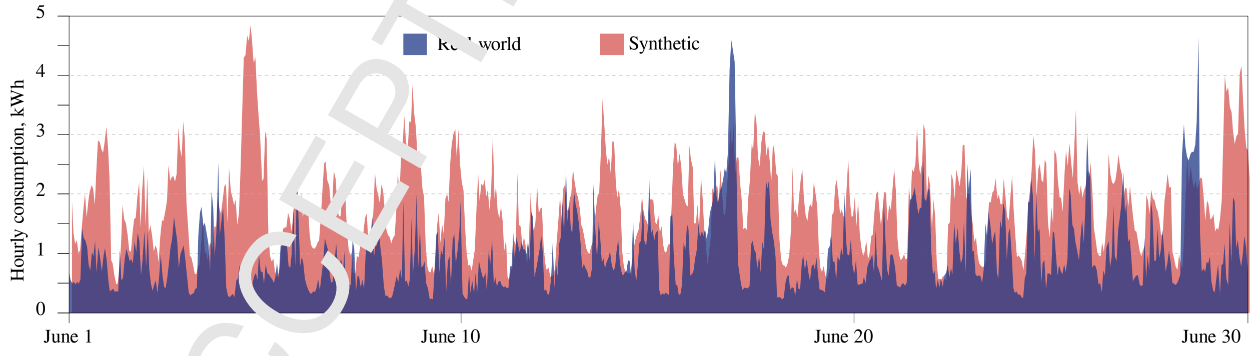


Figure 12: Comparison of energy consumption time series of June

5.3. Comparison

In the previous subsection, we compared the visual patterns or shapes of the synthetic data with the real-world data. In the following, we will examine the statistical parameters of the data and the scalability of generating large datasets using the proposed methods.

5.3.1. *Statistical performance*

Figure 13 shows the probability distributions of the real-world and synthetic hourly consumption during one month (June). The results indicate that the distribution of the data generated by the regression-based method is more similar to the real-world data than the data generated by the probability-based method. This is because the latter is generated from the data independently sampled from the consumption distribution of each day. For further investigation, we present the distribution of the daily consumption of the real-world data for one of the days in Figure 14. As shown in the figure, the data conforms closely to a normal distribution.

Figure 15 shows a quantitative comparison of the real-world and synthetic data with the box-plot method. The box plot shows the summary statistics including minimum, first quartile, median, third quartile, and maximum, with outliers outside the upper limit. The box plot shows the statistical parameters for the twelve months of a year and shows that the three datasets are very similar. The biggest difference is the length of the box. The synthetic data generated by the probability-based method are always slightly higher than the real-world data and the synthetic data generated by the regression-based method. This means that the synthetic data generated by the probability-based method is more distributed over the month. This may be because we use the Laplace method to smooth a zero probability transition between two states in the TPM. This diversifies the transition of patterns in a sequence. On the other hand, the difference can come from the distribution of the daily consumption, from which we sample the random number.

Figure 16 shows the auto-correlation of the real-world and the two synthetic datasets, with a time lag of up to 50 hours. Auto-correlation is calculated based on the normalized patterns, which is a good way to examine the appearance period of repeated patterns. According to the auto-correlation function (ACF) values, the regression-based method provides a better matching with the real-world data. Recall that in the regression-based method, we optimized the model training using Pearson correlation distance to improve the accuracy of pattern recognition, and this experiment verifies that this optimization can yield better results (see also Figure 7 and Table 1). In contrast, the probability-based method generates pattern sequences relying on the TPM, which exhibits sub-optimal matching performance with respect to ACF values.

5.3.2. *System performance*

System performance including the training and data generation processes will be examined in this section. It may be remembered that the training models can be re-used in the data generation process. For each method, the training process includes a number of steps shown in Figure 17 (from bottom to top). This figure also shows the corresponding time required in each step when using the entire set of training data. In the regression-based method, normalization and K-means clustering are optional steps for the optimization purpose, indicated by the dashed-line rectangle. As shown in this figure, clustering is the most time-consuming action in both methods because they use

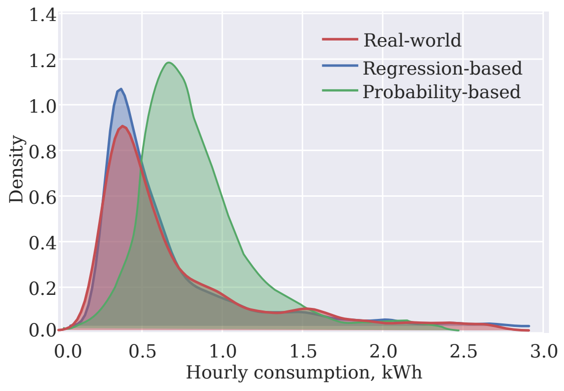


Figure 13: Comparison of hourly reading distribution

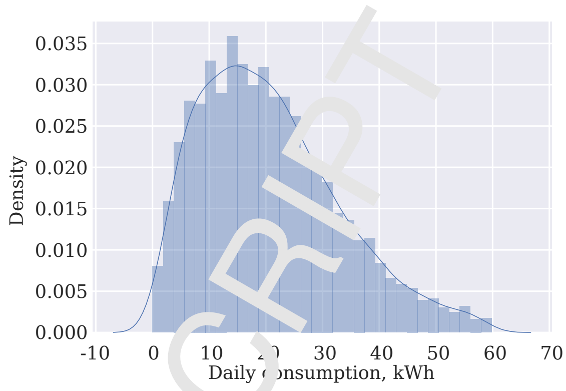


Figure 14: Distribution of daily consumption distribution

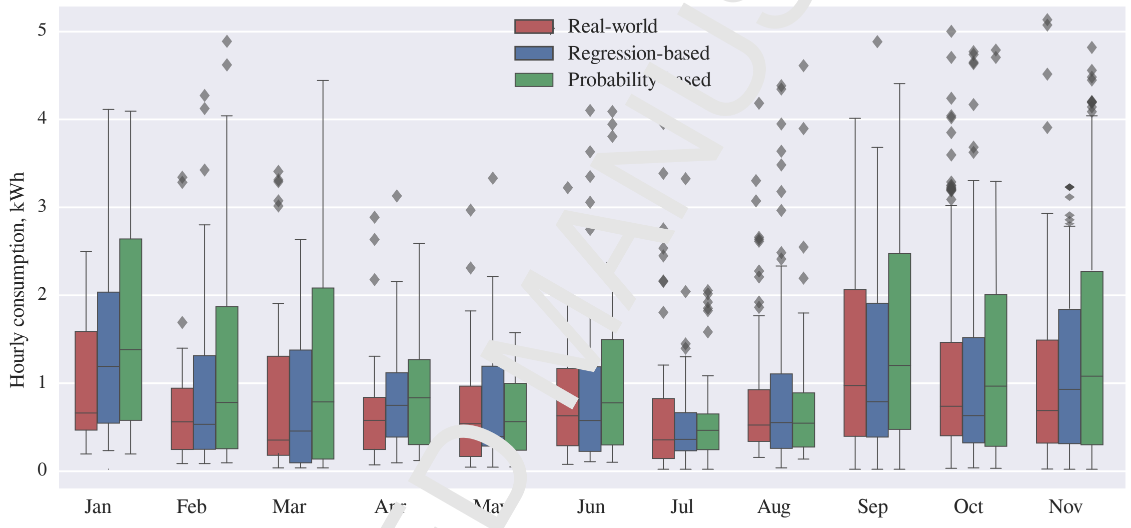


Figure 15: Comparison of summary statistics

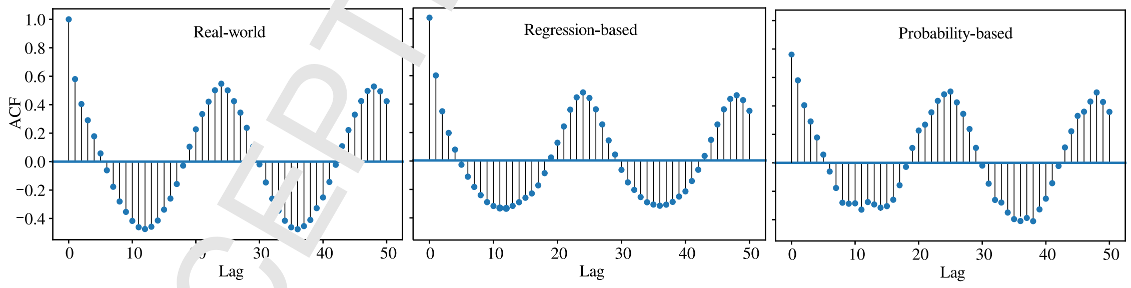


Figure 16: Comparison of normalized time-series auto-correlation

adaptive clustering. It is a two-stage method, which first performs adaptive *K*-means clustering to find the optimal number of clusters using an elbow method, then performs hierarchy-clustering to merge small clusters [63]. Adaptive clustering typically consumes more computer time than normal *K*-means clustering (see Figure 15). The total time of the probability-based method is higher than that of the regression-based method because it consists of five mandatory

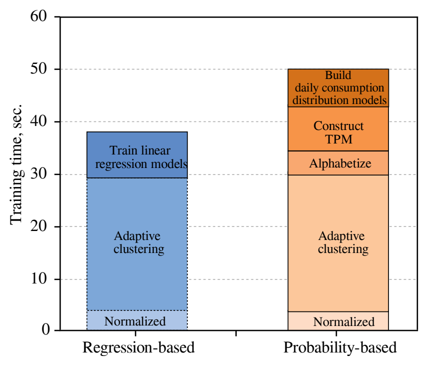


Figure 17: Comparison of training times

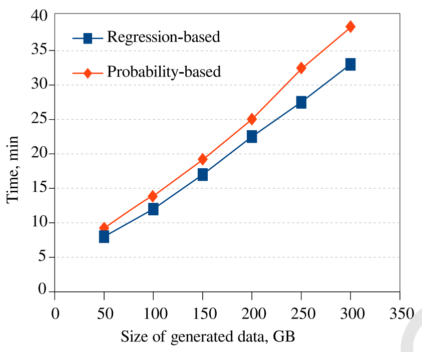


Figure 18: Size-up of generating datasets

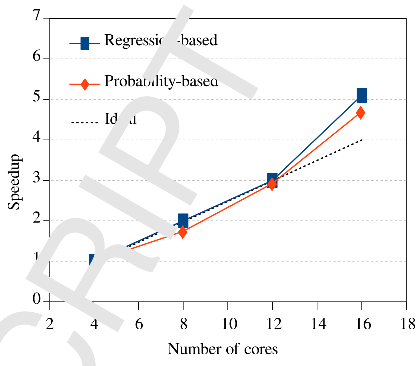


Figure 19: Speedup of generating 100GB data

steps in the entire training process.

In the following, we will evaluate the scalability of generating time series on Spark. It should be remembered that the models generated by the training process are broadcast to the workers in Spark. Map-only tasks are used to generate time series data in parallel. We perform the following two experiments to evaluate scalability, including *size-up* and *speedup*.

In the size-up experiment, we use a total of four nodes (16 cores) to generate data, but scale the generated data from 50 to 300 GB. Figure 18 shows the execution time. The results demonstrate that the time scales well with the amount of data, almost linearly.

In the speedup experiment, we scale the cores from 4 to 16 to generate a fixed-size dataset ($100GB$), and measure the execution time. The speedup is defined by the following equation:

$$speedup = \frac{T_4}{T_n} \tag{15}$$

where T_n is the execution time for n cores ($n = 4, 8, 12$ and 16). Figure 19 shows the results. As shown, both of the proposed methods can achieve good speedup, and the speedup is super linear when the cores increase to 16. In both experiments, the two methods are quite efficient with respect to running time and scalability because they run map-only jobs. The performance of the probability-based method is slightly slower than that of the regression-based method, mainly due to the cost of constructing an alphabet sequence by a random walk. As the size of the TPM is very large, $n^2 * m = 20^2 * 535 = 214,000$ ($n = 20, m = 365 - 1$) where n is the number of states (representative patterns) and m is the number of days in one year, the cost of lookup operations on TPM is substantial.

5.4. Discussion

In summary, the proposed data generators are able to generate realistic time series data with good performance, and the generated data have the characteristics comparable to the real-world data with respect to patterns and statistical information. The two methods are supervised machine learning methods that require real-world datasets as the seed for generating realistic datasets. Our study indicates that clustering is a good way to preserve consumption patterns and segmentation information. The two methods differ in the following ways: One uses prediction to simulate energy consumption data, while the other uses statistics and probability. For the regression-based method, the accuracy of the simulation depends heavily on the prediction model. In reality, it is often difficult to establish an accurate prediction model as it is influenced by many variables, including building type, household characteristics, weather conditions, and more. The models that incorporate these variables are proved to have better prediction accuracy, such as *periodic auto-regression with exogenous variables model (PARX)* [38]. For data availability and/or privacy reasons, the challenge is that data are often difficult to access for these variables. In this paper, we turn to forecasting with a simple autoregressive model that only requires energy consumption time series. In contrast, the probability-based method simulates the real-world time series based on the statistical information of the data, which is the statistic of the representative daily consumption patterns in this experiment. The representative patterns are the centroids of the clusters. Although the use of representative patterns can still produce satisfactory results, the pattern variances of the generated time series are smaller than those generated by the regression-based method, because the peaks are smoothed when using representative patterns. This can be mitigated by multiplying an anomaly factor, for example, to make a peak sharper [67]. The following rules apply to the selection of the data generation methods: The regression-based method should be the first option, as it provides better accuracy and performance for generating a large dataset, especially when socioeconomic data are available. Otherwise, the probability-based method is a good alternative to simulating real-world energy consumption data.

The implementation of these methods includes training and generation programming. The training process in both methods requires several steps. Comparatively, the regression-based method would require less human and computer effort if the optional steps for optimization, normalization, and clustering were omitted. The most time-consuming step is clustering for training, i.e., determining household groups or representative pattern groups. Depending on the size of the seed, the training programming may not have to be implemented as in this paper with a distributed computer programming framework such as Spark. The training process is performed only once, but the resulting models can be used many times. The implementation of the data generation program is relatively simple, as it is a map-only program on Spark. The parameters or models for data generation are also distributed to the mappers during runtime by broadcasting, and kept in memory to generate data for better efficiency. A distributed computing framework makes it possible to generate data in parallel. There are other alternatives for parallel data generation, such as multi-threading. The cluster-based approach is, however, the best way to generate large datasets with an order of tera/petabytes, due to its high scalability. Large datasets are often required for benchmarking big data management systems, e.g., [7].

6. Conclusions and Future Work

Scalable realistic consumption time series are often required for system benchmarking in software engineering and for building performance evaluation in civil engineering. In this paper, we have presented two different data generators that can accurately simulate time series of real-world fine-grained energy consumption. The proposed methods are both supervised machine learning methods that include a training process and a data generation process. However, they are based on different techniques: one is regression-based and the other is probability-based. We have described in detail how to create data models, and how to use the models to generate synthetic datasets. We proposed optimization techniques for a better simulation of real-world energy consumption data, such as the preservation of segmentation, and implemented the data generators on Spark to generate data in parallel. We comprehensively evaluated the proposed methods and compared the two methods. The results have shown that the proposed methods have the ability to simulate realistic energy consumption data, and the implemented data generators have good performance for large-scale data generation.

In future work, we will add more features to improve the data generation models. For example, the regression-based method can use weather conditions (e.g., outdoor temperatures), and a broader seasonality (e.g., the seasons of a year). In addition, we will refine the data generators to make them easy to use for generating various consumption data such as water, gas, or heat.

Acknowledgments. This research is part of the ClairCity project (http://www.claircity.eu) funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 689289, and by UCN-FOU funding (Project-6/2016-17).

References

[1] W. Kleiminger, C. Beckel, T. Staake, S. Santini, Occupancy detection from electricity consumption data, in: Proc. of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings, 2013, 1–8.

[2] L.M. Candanedo, V. Feldheim, D. Deramaix, A methodology based on Hidden Markov Models for occupancy detection and a case study in a low energy residential building. Energy and Buildings, 148(2017) 327–341.

[3] K. Zhou, S. Yang, Understanding household energy consumption behavior: The contribution of energy big data analytics, Renewable and Sustainable Energy Reviews 56(2016) 810–819.

[4] ClairCity: Citizen-led air pollution reduction in cities, Accessed at http://www.claircity.eu as of 2018.11.01.

[5] Smart Meter From Wikipedia, Accessed at `en.wikipedia.org/wiki/Smart_meter` as of 2018.11.01.

[6] X. Liu, L. Golab, W. Golab, I.F. Ilyas, Benchmarking Smart Meter Data Analytics, in: Proc. of the 18th International Conference on Extending Database Technology, 2015, 385–396.

[7] X. Liu, L. Golab, W. Golab, I.F. Ilyas, S. Jin, Smart Meter Data Analytics: Systems, Algorithms, and Benchmarking, ACM Transactions on Database Systems 42(2)(2017).

[8] The Smart Meter Revolution: Towards a smarter future, Accessed at `https://iot.telefonica.com/multimedia-resources/the-smart-meter-revolution-towards-a-smarter-future` as of 2018.11.01.

[9] N. Batra, O. Parson, M. Berges, A. Singh, A. Rogers, A comparison of non-intrusive load monitoring methods for commercial and residential buildings, arXiv preprint arXiv:1408.6595 (2014).

[10] ISSDA, Accessed at `www.ucd.ie/issda/data/commissionforenergyregulationcer` as of 2018.11-01.

[11] S. Makonin, B. Ebert, I.V. Bajic, F. Popowich, Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014, Scientific data 3(2016) 160037.

[12] P. Street, Dataport: the world's largest energy data resource, Pecan Street Inc., 2015.

[13] J. Zico Kolter, M.J. Johnson, REDD: A public dataset for energy disaggregation research, in: Proc. of the SustKDD Workshop on Data Mining Applications in Sustainability, 2011, 59–62.

[14] General Data Protection Regulation (GDPR), Accessed at https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en as of 2018.11.01.

[15] N. Iftikhar, X. Liu, S. Danalachi, F.E. Nordbjerg, J.H. Vollesen, A Scalable Smart Meter Data Generator Using Spark, in: Proc. of OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", 2017, 21–36.

[16] A. Grandjean, J. Adnot, G. Binet, A review and an analysis of the residential electric load curve models, Renewable and Sustainable energy reviews 16(9)(2012) 6539–6565.

[17] M.G. Oladokun, I.A. Odesola, Household energy consumption and carbon emissions for sustainable cities – A critical review of modelling approaches, International Journal of Sustainable Built Environment 4(2)(2015) 231–247.

[18] L.G. Swan, V.I. Ugursal, Modeling of end-use energy consumption in the residential sector: A review of modeling techniques, Renewable and sustainable energy reviews 13(8)(2009) 1819–1835.

[19] M. Stokes, Removing barriers to embedded generation: a fine-grained load model to support low voltage network performance analysis, PhD thesis, De Montfort University, 2005.

[20] K. He, L. Stankovic, J. Liao, V. Stankovic. Non-intrusive load disaggregation using graph signal processing, IEEE Transactions on Smart Grid 9(3)(2018) 1739–1747.

[21] C. Brandoni, A. Arteconi, G. Ciriachi, F. Polonara, Assessing the impact of micro-generation technologies on local sustainability, Energy Conversion and Management, 87(2014) 1281-1290.

[22] A. Franco, P. Salza, Strategies for optimal penetration of intermittent renewables in complex energy systems based on techno-operational objectives, Renewable energy 36(2)(2011) 743–753.

[23] D.J. Aigner, C. Sorooshian, P. Kerwin, Conditional demand analysis for estimating residential end-use load profiles, The Energy Journal 5(3)(1984)81–97.

[24] I. Richardson, M. Thomson, D. Infield, C. Clifford, Domestic electricity use: A high-resolution energy demand model, Energy and buildings 42(10)(2010) 1878–1887.

[25] N. Pflugradt, U. Muntwyler, Synthesizing residential load profiles using behavior simulation, Energy Procedia, 122(2017) 655–660.

[26] M.S. Ahmed, A. Mohamed, R.Z. Homod, H. Shareef, A.H. Sabry, K.B. Khalid, Smart plug prototype for monitoring electrical appliances in Home Energy Management System, in: Proc. of IEEE Student Conference on Research and Development (SCOReD), 2015, 32–36.

[27] S. Bissey, S. Jacques, J.C. Le Bunetel, The Fuzzy Logic Method to Efficiently Optimize Electricity Consumption in Individual Housing, Energies 10(11)(2017) 1701.

[28] A. Marszal-Pomianowska, P. Heiselberg, O.K. Larsen, Household electricity demand profiles – A high-resolution load model to facilitate modelling of energy flexible buildings, Energy 103(2016) 487–501.

[29] M. Pipattanasomporn, M. Kuzlu, S. Rahman, Y. Teklu, Load Profiles of Selected Major Household Appliances and Their Demand Response Opportunities, IEEE Transaction Smart Grid 5(2014) 742–750.

[30] N. Tewathia, Determinants of the household electricity consumption: A case study of Delhi, International Journal of Energy Economics and Policy 4(3)(2014) 337–348.

[31] V. Abeykoon, N. Kankanamdurage, A. Senevirathna, P. Ranaweera, R. Udawalpola, Electrical Devices Identification through Power Consumption using Machine Learning Techniques, International Journal of Simulation–Systems, Science & Technology, 17(2016).

[32] L. Chuan, A. Ukil, Modeling and validation of electrical load profiling in residential buildings in Singapore, IEEE Transactions on Power Systems 30(5)(2015) 2800–2809.

[33] J.V. Paatero, P.D. Lund, A model for generating household electricity load profiles, International Journal of Energy Research 30 (2006) 273–290.

[34] M. Arlitt, M. Marwah, G. Bellala, A. Shah, J. Healey, B. Vandiver, IoTABench: An Internet of Things Analytics Benchmark, in: Proc. of the

6th ACM/SPEC International Conference on Performance Engineering, 2015, 133–144.

[35] L. Kegel, M. Hahmann, W. Lehner, Feature-based comparison and generation of time series, in: Proc. of the 30th International Conference on Scientific and Statistical Database Management, 2018, 20–31.

[36] G.M. Jenkins. Autoregressive–Moving Average (ARMA) Models. Encyclopedia of Statistical Sciences, 1982.

[37] P.L. Anderson, M.M. Meerschaert, K. Zhang, Forecasting with Prediction Intervals for Periodic Autoregressive Moving Average Models, Journal of Time Series Analysis 34(2)(2013) 187–193.

[38] X. Liu, P.S. Nielsen, An ICT-Solution for Smart Meter Data Analytics, Journal of Energy 115(3)(2016) 1710–1722.

[39] A. Lendasse, D. Francois, V. Wertz, M. Verleysen, Vector quantization: a weighted version for time-series forecasting, Future Generation Computer Systems 21(7)(2005) 1056–1067.

[40] M. Chen, A high-order fuzzy time series forecasting model for internet stock trading, Future Generation Computer Systems, 37(2014) 461–467.

[41] G.P. Zhang, Time Series Forecasting using a Hybrid ARIMA and Neural Network Model, Neurocomputing, 50(2003) 159–175.

[42] J.G. De Gooijer, R.J. Hyndman, 25 Years of Time Series Forecasting, International Journal of Forecasting, 22(3)(2006) 443–473.

[43] B. Dong, C. Cao, S.E. Lee, Applying support vector machines to predict building energy consumption in tropical region, Energy and Buildings 37(5)(2005) 545–553.

[44] S. Fan, R.J. Hyndman, Short-term load forecasting based on a semi-parametric additive model, IEEE Transactions on Power Systems 27(1)(2012) 134–141.

[45] Q. Li, P. Ren, Q. Meng, Prediction model of annual energy consumption of residential buildings, in: Proc. of the Advances in Energy Engineering (ICAEE) Conference, 2010, 223–226.

[46] Y. Bengio, Learning deep architectures for AI, Foundations and trends in Machine Learning, 2(1)(2009) 1–127.

[47] C. Fan, F. Xiao, Y. Zhao, A short-term building cooling load prediction method using deep learning algorithms, Applied energy 195(2017) 222–233.

[48] E. Mocanu, P.H. Nguyen, M. Gibescu, W.L. Kling, Deep learning for estimating building energy consumption, Sustainable Energy, Grids and Networks 6(2016) 91–99.

[49] Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature, 321(2015) 436–44.

[50] G.P. Zhang, M. Qi, Neural Network Forecasting for Seasonal and Trend Time Series, European Journal of Operational Research 160(2)(2005) 501–514.

[51] R. Weiers, Introduction to Business Statistics, Cengage Learning, 2010.

[52] K.D. Lawrence, R.K. Klimberg, S.M. Lawrence, Fundamentals of Forecasting using Excel, Industrial Press Inc., 2009.

[53] P. Gianniou, X. Liu, A. Heller, P.S. Nielsen, C. Rode, Clustering-based Analysis for Residential District Heating Data, Journal of Energy Conversion & Management 165(2018) 840–850.

[54] O. Ardakanian, N. Koochakzadeh, R.P. Singh, L. Golab, S. Keshav, Computing Electricity Consumption Profiles from Household Smart Me-ter Data, in: Proc. of the EDBT Workshop on Energy Data Management(EnDM), 2014, 140–147.

[55] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in: Proc. of the 9th USENIX conference on Networked Systems Design and Implementation, 2012, 2–2.

[56] A. Okcan, M. Riedewald, Processing theta-joins using MapReduce, in: Proc. of SIGMOD, 2011, 949–960.

[57] B.J. Birt, G.R. Newsham, I. Beausoleil-Morrison, M.M. Armstrong, N. Saldanha, I.H. Rowlands, Disaggregating categories of electrical energy end-use from whole-house hourly data, Energy and Buildings, 50(2012) 93–102.

[58] J. Wu, Advances in K-means Clustering: A Data Mining Thinking, Springer Science & Business Media, (2012).

[59] T.W. Liao, Clustering of Time Series Data–A Survey, Pattern Recognition, 38(11)(2005) 1857–1874.

[60] M. Parsian, Data Algorithms: Recipes for Scaling Up with Hadoop and Spark, O'Reilly Media, Inc. (2015).

[61] K. Black, Business Statistics: For Contemporary Decision Making, John Wiley & Sons (2011).

[62] P.A. Jaskowiak, R.J. Campello, I.G. Costa, Proximity measures for clustering gene expression microarray data: a validation methodology and a comparative analysis, IEEE/ACM transactions on computational biology and bioinformatics, 10(4)(2013) 8[illegible]5–857.

[63] J. Kwac, J. Flora, R. Rajagopal, Household energy consumption segmentation using hourly data, IEEE Transactions on Smart Grid, 5(1)(2014) 420–430.

[64] B.O. Ngoko, H. Sugihara, T. Funaki, Synthetic generation of high temporal resolution solar radiation data using Markov models, Solar Energy 103(2014) 160–170.

[65] X. Liu, M.C. Papaefthymiou, A markov chain sequence generator for power macromodeling, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23(7)(2004) 1048–1062.

[66] S. Dolnicar, A Review of Unquestioned Standards in Using Cluster Analysis for Data-driven Market Segmentation, in: Proc. of the Australian and New Zealand Marketing Academy Conference, 2002.

[67] N. Iftikhar, X. Liu, F.E. Nordbjerg, S. Danalachi, A prediction-based smart meter data generator, in: Proc. of the 19th International Conference on Network-Based Information Systems, 2016, 173–180.

Dr. Xiufeng Liu, Technical University of Denmark (xiuli@dtu.dk)
Xiufeng Liu is a senior researcher at the Department of Management Engineering, Technical University of Denmark. He received his PhD in Computer Science from Aalborg University, Denmark in 2012. Prior to joining DTU, he worked as a Postdoctoral researcher at Waterloo University, Canada, in 2013-2014. His research includes smart meter data analytics, data warehousing, energy informatics, and big data.

Dr. Nadeem Iftikhar, University College of Northern Denmark (naif@ucn.dk)
Nadeem Iftikhar is an Associate Professor at University College of Northern Denmark. He received his BS degree in Applied Mathematics and Computer Science from Eastern Mediterranean University, Famagusta, Turkish Republic of Northern Cyprus, in 1994, followed by his MS degree in Information Science from University of New South Wales, Sydney, Australia, in 1997 and Ph.D. degree in Computer Science from Aalborg University, Aalborg, Denmark, in 2011. His research includes data warehousing, with a particular emphasis on OLAP and ETL, as well as big data analytics. He is an IEEE member.

Dr. Huan Huo, University of Technology Sydney, Australia (huan.huo@uts.edu.au)
Huan Huo is a senior lecturer at University of Technology Sydner. She received her PhD in Computer Software and Theory from Northeastern University in 2007, then she worked as an associate professor in Optical-Electrical and Computer Engineering School at University of Shanghai for Science and Technology. Her research include cloud data management technology, data stream query optimization, XML data management technology, etc.

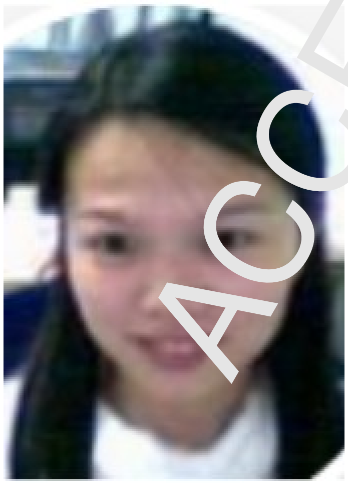
Dr. Rongling Li, Technical University of Denmark (liron@byg.dtu.dk)
Rongling Li an Assistant Professor at the Department of Civil Engineering, Technical University of Denmark. She received her Ph.D. degree in Architecture Engineering from the University of Tokyo, in 2014. She was a Postdoctoral Researcher with Eindhoven University of Technology from 2014 to 2016, then with Technical University of Denmark from 2017 to Nov. 2018. She currently works on two Danish pilot projects of smart energy systems and smart cities: CITIES and EnergyLab Nordhavn. She is an active member of International Energy Agency EBC Annex 67 - Energy Flexible Buildings. Her research includes smart cities, demand flexibility, occupant behavior, energy system modelling, data analysis, building physics and services.

Dr. Per Sieverts Nielsen, Technical University of Denmark (pernn@dtu.dk)
Per Sieverts Nielsen is a senior researcher at the Department of Management Engineering, Technical University of Denmark. He has been working in the energy modelling and sustainability area since 1990. He has participated in a range of international research programmes and multi-disciplinary research programmes. Currently, he is working on various projects in energy systems integration in particular "smart cities" development.

Dr.Xiufeng Liu, Technical University of Denmark (Xiuli@dtu.dk)

Dr. Nadeem Iftikhar, University College of Northern Denmark (naif@ucn.dk)

Dr. Huan Huo, University of Technology Sydney, Australia (huan.huo@uts.edu.au)



Dr. Rongling Li, Technical University of Denmark (liron@byg.dtu.dk)

Dr. Per Sieverts Nielsen, Technical University of Denmark (pernn@dtu.dk)

1. Regression- and probability-based energy consumption simulation methods
2. Scalable residential energy consumption time series generation
3. Fine-grained consumption data generator for simulation
4. Residential consumption data modeling