International Journal of Smart Grid and Clean Energy

An automatic learning framework for smart residential communities

Helia Zandi[a], Michael Starke[a], Jeffrey Munk[a], Teja Kuruganti[a], James Leverette[b], Jens Gregor[c] *

[a] *Oak Ridge National Laboratory, Oak Ridge, TN, USA*
[b] *Southern Company, Birmingham, AL, USA*
[c] *University of Tennessee, Knoxville, Knoxville, TN, USA*

Abstract

Predictability has been foundational to matching supply and demand in the day-to-day operation of the electric power system. Demand predictability is eroding because of the increased use of renewable energy resources and more sophisticated loads, such as electric vehicles and smart appliances. In this paper, an automatic software framework is described which can be used for load forecasting in smart communities. A time-varying clustering-based Markov chain approach is used to predict the energy consumption of residential buildings in a smart community. The training data is based on 1-minute meter data of occupied homes over one month. The data points are first clustered based on the energy consumption and the time of the day. Then, the original data is converted using the Centroids of the clusters. A time-varying Markov chain is subsequently trained to model the energy consumption behavior of residents for each home using the transformed data. The trained model is shown to successfully predict load in 5-minute intervals over a 24 hours period.

Keywords: Smart grid, internet of things, cyber-physical system, Markov chain, load forecast, demand response, clustering

1. Introduction

The internet of things (IoT) has revolutionized the world of intercommunications. In the last decade, numerous smart devices have been introduced ranging from wi-fi connected thermostats to water heaters and electric vehicle chargers to name a few. Most of the intelligent device incorporation has been adopted in the residential sector which is a major energy consumer in the United States (U.S.). Based on a report by the department of energy (DOE), 40% of primary energy consumption in the U.S. is attributable to the building sector, which shows the significance of saving energy in buildings [1].

The integration of smart devices into the residential sector has shed light on a potential future of utility engagement with homeowners: energy consumption can be optimized to support the utility and the homeowner can be compensated. However, communications, control, and optimization supporting software must be developed.

The first step is designing and implementing a stable and secure software framework that can provide seamless system integration, communication, and control. Significant research has been done in this area and the key specifications for an energy management system (EMS) have been identified [2][3][4]. An

* Manuscript received April 4, 2019; revised March 5, 2020.
Corresponding author. *E-mail address*: starkemr@ornl.gov.
doi: 10.12720/sgce.9.3.485-494

EMS should 1) be able to handle communication and control of different devices utilizing different communication protocols, 2) provide forecasting of the resources and integrated systems, 3) be able to handle the computational cost associated with running different optimization algorithms for supporting demand response (DR), and 4) be capable of handling a large amount of data efficiently and securely. All of these features support wide adoption and integration with utility systems in moving to a smart grid.

The smart grid is adopting increased quantities of distributed renewable generation while attempting to be more resilient and reliable. As a result, increased flexibility and better forecasting are becoming more important in day-to-day operations. Hence, building systems that can provide forecasting support with an EMS are crucial to ensuring continued growth.

Because of the importance of accurate load forecasting, extensive research has been carried out for decades to predict the electrical consumption of a geographical location over different time periods [5][6]. These methods can be categorized into short-term, medium-term, and long-term forecasts. Short-term refers to prediction from one hour to one week, medium-term refers to one week to one-year forecast, and any prediction for more than one year is known as a long-term forecast.

Regression methods are one of the most common statistical methods used [7][8][9]. The idea is to model the relationship between energy consumption and other factors such as weather, solar irradiance, seasonal changes, annual load growth, etc. The accuracy of such methods relies on adequate historical data and high correlations between data sets.

Exponential smoothing is another approach used for load forecasting [10]. Historical data is used to build a model that can predict the future load. This method has been enhanced by converting the general exponential smoothing model by a finite Fourier series and conducting an adaptive autoregression modeling technique to account for changes in load component [11].

Support Vector Machines (SVM) is another method used for forecasting. This is done by analyzing the data, finding the pattern, and performing classification and regression analysis. SVM has shown better performance than the direct application of artificial neural networks (ANN) [12]. SVM has been coupled with Simulated Annealing Particle Swarm Optimization (SAPSO) algorithms that parameterize variables within SVM for improved accuracy and convergence [13][14].

Deep neural networks (DNN) have also been employed for load forecasting [15][16]. However, these methods require significant historical data and are computationally expensive to train. Other approaches used for load forecasting are fuzzy logic (FL) algorithms, evolutionary algorithms (EA), and different forms of ANN. In many cases, these systems can be hybridized to create new load forecasting approaches and improved accuracy. In [17], an unsupervised learning is first used to cluster the data and then a supervised learning is used on the clusters for forecasting.

Fig. 1. View of reynolds landing at ross bridge, smart neighborhood[tm]. Image: southern company.

This paper is organized as follows: Section 2 summarizes the objective of the project, Section 3 explains the software architecture used for developing the automatic learning framework, Section 4 describes the load forecast algorithm used and the data it was applied to, Section 5 provides results for applying the algorithm to the data, and Section 6 contains conclusions and future work.

2. Background

In a collaborative project between Oak Ridge National Laboratory (ORNL) and Southern Company, a distributed, residential, cloud-based EMS has been developed and deployed. This system is operating on a community composed of sixty-two occupied single-family homes supported by a microgrid (Fig. 1). Each home is equipped with intelligent wi-fi connected devices that provide data and control to cloud-based systems. For this project, the main controllable loads in the homes are the heating, ventilation, and air conditioning (HVAC) and water heater devices. All electric circuits within the homes are sub-metered, collected, and stored in databases, in real-time, through cloud services via smart metering systems that record at 1-minute resolution. This data is accessible by the EMS through the utility Representational State Transfer (REST) Application Programming Interface (API).

Fig. 2. Microgrid generation resources. Image: Southern Company.

The community also hosts a microgrid for improving electric power delivery. The microgrid consists of photovoltaic panels, a lithium ion-based energy storage system, and a natural gas generator (Fig. 2). With these resources, the microgrid can island the neighborhood during a grid outage. This microgrid controller (MC) has been retrofitted with an optimization routine used to adjust a 24-hour price signal in order to achieve a desired load shape. Likewise, the EMS in the individual homes optimizes the house loads to minimize cost based on the price forecast provided by the MC. This negotiation process with the price signal and the load between the MC and homes is repeated until the MC finds an acceptable point of convergence. The homeowner load forecast prediction algorithm is discussed in Section 4.

3. Software Architecture

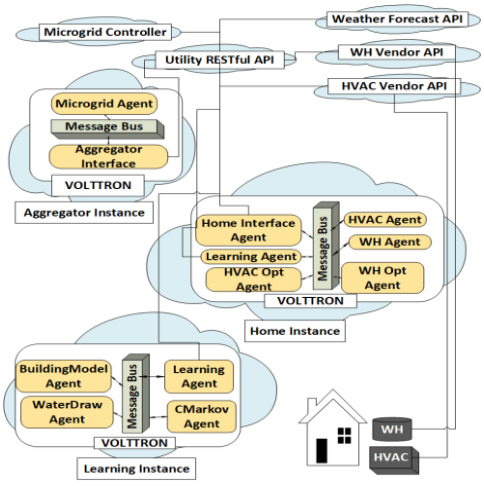

Fig. 3. Framework overall software architecture

A multi-agent system (MAS) architecture has been developed and is used for developing the software framework discussed in this paper. Here, the MAS is a community of cloud-based software agents that

coordinate the decision-making process to satisfy the overall system objective and reduce electricity costs [18].

The developed software framework consists of three main cloud-based instances: 1) Home Instance, 2) Aggregator Instance, and 3) Learning Instance (Fig. 3). Each instance supports some basic functionality which is described in the following paragraphs. A back-bone utility system exists that links and anonymizes resident system credentials through a participant ID with Virtual Machine (VM) name along with device credentials and meter data.

The Home Instance has an association with each home and performs the functions of the residential building EMS optimization and control. To support these features, the Home Instance utilizes a home interface, device interface, optimization, and learning interface agents. Credentials for each home are unique and are automatically pulled from a utility interfacing RESTful API and distributed through local agent communications via the home interface agent. The device agents (HVAC and water heater interfaces) communicate to the devices through the vendor APIs and perform data collection and control dispatch. The optimization utilizes device data such as user schedules and current states and a model predictive control formulation to determine the optimal control dispatch against a 24-hour price signal provided by the MC.

The Aggregator Instance behaves as the neighborhood medium and provides data exchange between each residential EMS and the MC. The Aggregator Instance collects the reported optimal results from each residential level EMS and formulates the output to the MC for decision making. The aggregator also collects data sets from the utility RESTful API and MC for distribution to each EMS for optimization including price signal and weather forecasts (solar irradiance and temperature).

The Learning Instance attempts to tune and predict different data sets associated with the system. These datasets are for two purposes, provide more accurate total load forecasts for the MC optimization and to support local residential EMS optimization decision making. There are four agents associated with this instance: 1) Learning Interface, 2) CMarkov, 3) Building Model, and 4) Water Draw. The agent basic functionality for these agents is described in Table I. This paper will focus on the CMarkov agent which is responsible for non-controllable load forecasting and internal heat load estimation.

The Learning Interface gets the credentials for all the homes so it can query their metering data. At the initialization of the Learning Instance, two weeks of metering data for each home is queried from the utility RESTful API. This data includes the metering of different circuits in the home in 1-minute intervals. CMarkov Agent uses a time-varying clustering-based Markov approach to learn the energy consumption pattern of the home and utilize that for load forecasting. The algorithm is explained in Section 4. The load forecast for the next 24-hours is calculated based on the learning algorithm for each home and the result is sent back to the utility RESTful API, so each Homes Instance can query that and utilize it in its algorithms and forecast. After the learning is done for two weeks of data, another set of data is queried from the utility RESTful API each day and appended to the previously queried data, and a new training is done based on the new data set.

Table 1. Summary of Agents along with their role in the software framework

Learning Instance	
Learning Agent	Queries the utility RESTful API to get the credentials for all the homes in the neighborhood and query the utility RESTful API to access the homes' data. It is also responsible to communicate the learning forecast back to the API so individual homes can access them.
CMarkov Agent	Utilizes a time-varying clustering-based Markov model to forecast the energy consumption and the internal heat load for each home.
BuildingModel Agent	Trains a building model to estimate the building parameters using weather data, metering data, and HVAC data.
WaterDraw Agent	Trains a model to predict the water draw for each home in the neighborhood.

All instances are running on a utility Microsoft Azure© cloud server hosting Ubuntu-based single core machines with 2 GB of random access memory (RAM). Each instance utilizes VOLTTRON™ as the

agent platform which uses a Python-based backbone and enables compiling, packaging, running, and managing different agents [19]. All these instances are communicating with the utility RESTful API for retrieving data and sending control commands.

4. Load Forecast

The development of a load forecast and heat load profile consists of several steps: 1) data retrieval and processing, 2) time-varying clustering-based Markov model development, and 3) application of the trained model for load forecasts. In the following sections each step is discussed.

4.1. Data retrieval and processing

The data used for training and validating the load forecasting is the metered circuit-level data from homes in the neighborhood. This data is available in 1-minute intervals. The circuit-level data includes the measurement for the main circuit, air conditioning, condenser unit, air handler, electric heat, electric vehicle charger, oven, dryer, water heater, dishwasher, outlets, refrigerator, ventilator, lighting, disposal, microwave, and smoke detector.

The data is first processed for anomalies: 1) missing data or erroneous data (high values that were falsely recorded) is replaced with interpolation, and 2) data with incorrect time-stamps is removed. As mentioned in Section 3, the load forecast algorithm should forecast non-HVAC and non-water heater loads (non-controllable loads) and the internal heat load. For performing the learning algorithm for predicting the non-HVAC and non-water heater loads and internal heat load, the metered data input is summed by category as:

$$P^t_{OL} = P^t_{Main} - P^t_{EH} - P^t_{WH} - P^t_{CU} - P^t_{AH} \tag{1}$$

$$P^t_{IHL} = P^t_{Main} - P^t_{Dryer} - P^t_{EV} - P^t_{WH} - P^t_{CU} \tag{2}$$

where P^t_{OL} is the value of non-controllable load for timestep t, P^t_{Main} is the power reading for the main circuit at timestep t, P^t_{EH} is the power of electric heat for current timestep t, P^t_{WH} is the power reading for water heater at timestep t, P^t_{CU} is the power reading for condenser unit at timestep t, P^t_{AH} is the power for air handler at timestep t, P^t_{Dryer} is the power for electric dryer at timestep t , P^t_{EV} is the power for the electric vehicle charger at timestep t, and P^t_{IHL} is the power for the internal heat load at timestep t.

This is done for all the metering data for each minute and then the algorithm is trained over the calculated values. Fig. 4 shows the pattern of the load consumption for a home in the neighborhood based on one month of metered data in May.

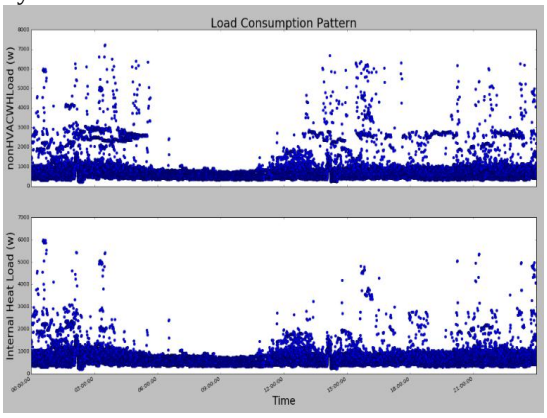

Fig. 4. (upper diagram) The pattern of the energy consumption for non-HVAC and non-water heater loads (lower diagram) Pattern of the energy consumption for internal heat loads.

4.2. Time-varying clustering-based markov chain algorithm

The method for developing 24-hour non-controllable load forecasts and internal heat load estimations in this paper is based on a time-varying clustering-based Markov chain model. A Markov chain is a stochastic model of a sequence of events in which the probability of the next event only depends on the current event. That is,

$$P_r\,(S_{n+1} = s \mid S_1 = s_1, \ldots, S_n = s_n) = P_r\,(S_{n+1} = s \mid S_n = s_n) \tag{3}$$

where S_i denotes a random variable and s_i is the value that it takes on. The latter is also referred to the state of the Markov chain. In a Markov chain model, the transition probability only depends on the current state and it is independent of all the proceeding sequence of states. In our proposed algorithm, the state of the Markov model is determined by applying k-means clustering on the data and group the data based on their similarity in energy consumption for the time of the day. This is an example of unsupervised learning which partitions the data into k clusters by minimizing the following objective function:

$$\arg\min_D \sum_{i=1}^{k} \sum_{x \in D} \|x - s_i\|^2 \tag{4}$$

where $||x - s_i||^2$ is a chosen distance function, D is the set of data points, and s_i is the centroids of the clusters. After the clusters are identified, then each of the data points are transformed by fitting the data to their clusters and transforming them using the centroid of the cluster they belong to.

In a Markov model, the sequence starts with a state and moves to another state in the next step. Let the cluster centroids denote a set of k states, i.e., $S = \{s_1, s_2, .., s_k\}$. The probability of transitioning from one state to another is given by a $k \times k$ transition matrix *where* entry p_{ij} shows the probability of moving from state s_i, to state s_j. In this algorithm, a transition matrix is calculated for each minute of the day using the transformed data. For example, a transition matrix from 0-1, 1-2, …, 1439-1440. The centroids of the clusters are used as the states for the transition matrices. The result would be 1440 matrices of size 50x50. A sample transition matrix for minute 309 can be seen in Fig. 5. Each entry p_{ij} in this matrix shows the probability to move from state i at minute 309 to state j at minute 310.

```
[[ 6.66666667  0.          0.          ...  0.          0.  0.        ]
 [ 0.          0.          0.          ...  0.          0.  0.        ]
 [ 0.          0.          3.33333333  ...  0.          0.  0.        ]
                                       ...
                                       ...
                                       ...
 [ 0.          0.          0.          ...  46.66666667  0.  0.       ]
 [ 0.          0.          0.          ...  0.          0.  0.        ]
 [ 0.          0.          0.          ...  0.          0.  0.       ]]
```

Fig. 5. A sample of a time-varying Markov chain transition matrix for minute 309.

4.3. Using the transition matrix to create a forecast.

The transition matrices computed in the previous section are used to predict the energy consumption for the following minute, given current minute data. For prediction, if the current time step is t, then the t^{th} transition matrix is used, P_t. The current value for the metering data at time t, is first fitted using k-means clustering and the centroid of the cluster is used as its new value, m_t. Then the index of the maximum value in m_t P_t is the value of the predicted power consumption in the next time interval, $t+1$. Depending

on the time horizon of the load forecast, one or multiple transition matrices will be used. For examples, if we are at time t, and the goal is to predict the value in the next m minutes, the following formula can be used:

$$x^{t+m} = m_t P_t P_{t+1} \dots P_{t+m} \quad (5)$$

where m_t is a vector with its t^{th} value be one and the rest be zero, P_t is the transition matrix at time t. The index of the maximum value of x^{t+m} is the next state that the sequence will transition and the forecasted value for the load.

5. Results

5.1. Training based on historical data

For training and validation, one month of data for a home in the neighborhood in the month of May is used as a reference. Based on some initial analysis over accuracy versus number of clusters, *k=50* is used as a candidate for *k* in k-means clustering and identifying the centroids. The result of applying the clustering on a home in the neighborhood is shown in Fig. 6. The red dots in the figure represent the centroid of the clusters and each color represents a different cluster.

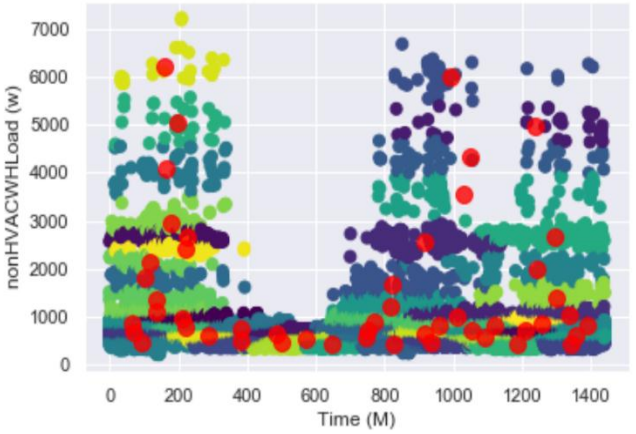

Fig. 6. The result of applying k-means clustering on one month of data for a home in the neighborhood.

The data is then transformed by fitting the data points to clusters. Then, time-varying Markov chain transition matrices are computed based on the transformed data for each minute.

As explained in Section 4.3, for prediction, if the current time step is *t*, then the t^{th} transition matrix is used, P_t. The current value for the metering data at time t is transformed using the k-means clustering model, m_t. Then the index of the maximum value in $m_t P_t$ is the value of the predicted power consumption in the next time interval, *t+1*. The training is done using 74% of the data. Then the trained model is validated using the remaining 26% of the data. The root mean square error (RMSE) and mean error (ME) is computed for a minute prediction for an entire day for each of the 8 days used as part of the validation data for a house in the neighborhood. Only 6.04% of the prediction data have an error above 200 W and 89.02% of them have an error below 100 W for one-minute prediction for a whole day. The result is shown in Fig. **7**.

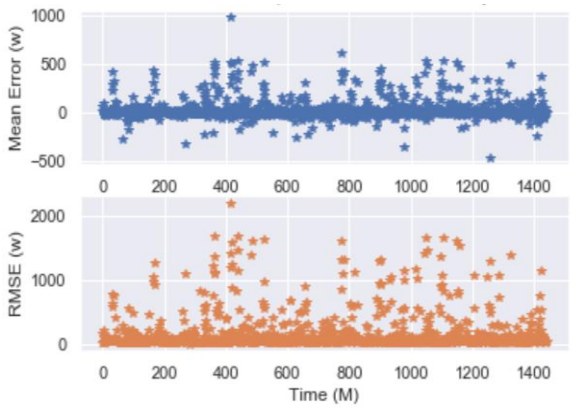

Fig. 7. The RMSE and ME result for the whole day prediction using the time-varying clustering-based Markov algorithm.

The resulting load prediction with 5-minute interval for the whole day is shown in Fig. 8. Based on the result, 3.33% of the data points have an average error of 200 W or more and 90.83% of the data points have an average error less than 100 W.

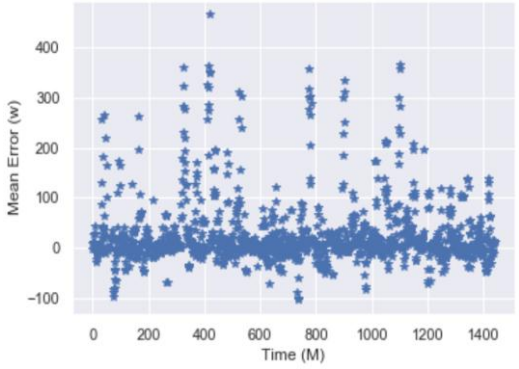

Fig. 8. The value for ME for the 5-minute load prediction for the whole day.

5.2. Field evaluation and testing

The Learning Instance was deployed and running on a single core VM with 2 GB of RAM over the utility cloud server for more than a month. The Learning agent first queries the utility RESTful API to get the credential of all homes in the neighborhood. Then, it queries two weeks of metered circuit-level data from the utility RESTful API for each home. The CMarkov agent uses this data for training the time-varying clustering-based Markov chain model. After the Markov transition matrices are computed for each minute for both internal heat loads and non-controllable loads, then the agent uses the latest metered data and provides a load forecast from midnight to the end of the following day. It then returns the load prediction to the utility RESTful API so the agents in the Home instance can use this prediction result. This includes a day-ahead load prediction for each home in 5-minutes interval, an array of size 288.

The result of forecast for an actual home in the neighborhood can be seen in Fig. 9 and Fig. 10. The time uses for both in the training data and forecast data is based on Coordinated Universal Time (UTC). Note that 91.66% of load forecast for non-controllable loads have an error of less than 400 W. And 92.01% of the forecast for internal heat load have an error less than 400 W. The average error for internal heat load forecast is 185.54 W and the average error for non-HVAC and non-water heater load is 313.18 W.

In Fig. 9 and Fig. 10, the maximum error is between midnight UTC and 1:05 AM UTC. After looking at the actual metered data for this period, we noticed that the dryer was the major load running in that period. We are planning to solve this by performing additional data analysis to find the pattern associated to each activity and utilize the identified patterns for improving the accuracy of the load forecasting algorithm.

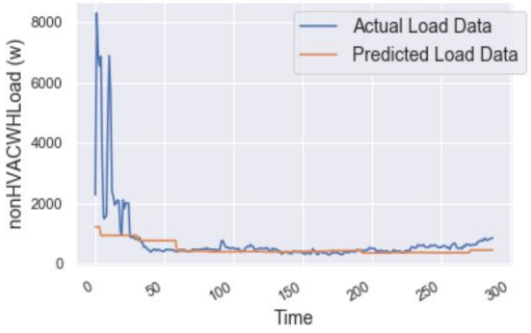

Fig. 9. Day-ahead forecast for non-HVACWH loads for a home in the neighborhood.

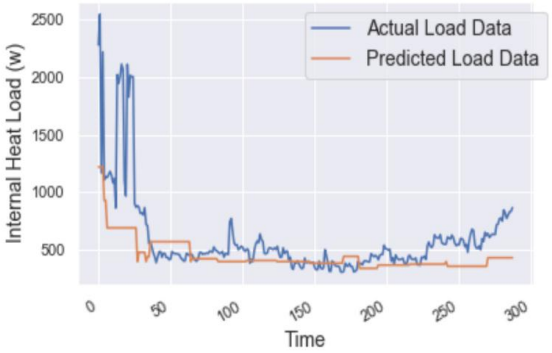

Fig. 10. Day-ahead forecast for internal heat load for a home in the neighborhood.

6. Conclusion and Future Work

In this paper, an automatic learning framework for smart residential communities is described and field evaluated. This framework is automatically collecting data for occupied homes in a smart neighborhood using the utility RESTful API. A time-varying clustering-based Markov chain approach is presented which is applied to the data collected at the circuit-level for each of the homes. The result is a day-ahead load forecast for each home so the optimizer and microgrid controller can utilize this information in their decision-making process.

The accuracy of the forecast for sample homes in 1-minute interval, 5-minutes interval, and a day-ahead forecast is presented. The algorithm shows an error of less than 400 W about 90% of the time for short-term forecasting. However, we are expecting to improve the accuracy by differentiating between the day of week, holidays, and taking weather data into account. Also, for clustering, we will apply other distance functions such as dynamic time wrapping for better time-series clustering and to achieve better accuracy.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Helia Zandi, Michael Starke, and Jeffrey Munk performed the research and data analysis.
Teja Kuruganti, James Leverette, and Jens Gregor provided technical input and reviewed the paper.
All authors helped drafting and reviewing the paper.

Acknowledgments

This work was funded by the U.S. Department of Energy, Energy Efficiency and Renewable Energy, Building Technology Office under contract number DE-AC05-00OR22725.

References

[1] "Buildings Energy Data Book, Chapter 2: Residential Sector, U.S. Department of Energy." [Online]. Available: http://buildingsdatabook.eren.doe.gov/ChapterIntro2.aspx.
[2] Asare-Bediako B, Kling WL, and Ribeiro PF. Home energy management systems: Evolution, trends and frameworks. *Int. Univ. Power Eng. Conf. (UPEC), 47th London*, 2012; 1–5.
[3] Zandi H, Kuruganti T, Vineyard E, and Fugate D. Home energy management systems: An overview," in *9th International Conference on Energy Efficiency in Domestic Appliances and Lighting (EEDAL2017)*, 2017, no. 2018, pp. 605–614.
[4] Paper W. Bringing the smart grid into the home : The value of home energy management for utilities. *Perspective*, 2010; pp. 1–6.
[5] Singh AK, Ibraheem, Khatoon S, Muazzam M, and Chaturvedi DK. Load forecasting techniques and methodologies: A review. *ICPCES 2012 - 2012 2nd Int. Conf. Power, Control Embed. Syst.*, 2012.
[6] Srivastava AK, Pandey AS, and Singh D. Short-term load forecasting methods: A review. *2016 Int. Conf. Emerg. Trends Electr. Electron. Sustain. Energy Syst.*, 2016; 130–138.
[7] Mbamalu GAN and El-Hawary ME. Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation. *IEEE Trans. Power Syst.*, 1993; 8(1): 343–348.
[8] Haida T and Muto S. Regression based peak load forecasting using a transformation technique. *Power Syst. IEEE Trans.*, vol. 9, no. 4, pp. 1788–1794, 1994.
[9] Barakat EH, Qayyum MA, Hamed MN, and Al Rashed SA. Short—term peak demand forecasting in fast developing utility with inherit dynamic load characteristics Part — I application of classical time—series methods. *IEEE Trans. Power Syst.*, 1990; 5, (3): 813–824.
[10] Moghram I and Rahman S. Analysis and evaluation of five short-term load forecasting techniques. *IEEE Trans. Power Syst.*, 1989; 4(4): 1484–1491.
[11] El-Keib AA, Ma X, and Ma H. Advancement of statistical based modeling techniques for short-term load forecasting. *Electr. Power Syst. Res.*, 1995; 35(1): 51–58.
[12] Guo Y, Niu D, and Chen Y. Support vector machine model in electricity load forecasting. *2006 Int. Conf. Mach. Learn. Cybern.*, no. August, 2006; 2892–2896.
[13] Wang J, Zhou Y, and Chen X. Electricity load forecasting based on support vector machines and simulated annealing particle swarm optimization algorithm. *2007 IEEE Int. Conf. Autom. Logist.*, 2007; 2836–2841.
[14] Wang J, Liu Z, and Lu P. Electricity load forecasting based on adaptive quantum-behaved particle swarm optimization and support vector machines on global level. *Proc. 2008 Int. Symp. Comput. Intell. Des. Isc.* 2008; 1: 233–236.
[15] Amarasinghe K, Marino DL, and Manic M. Deep neural networks for energy load forecasting. *2017 IEEE 26th Int. Symp. Ind. Electron.*, 2017; 1483–1488.
[16] Marino DL, Amarasinghe K, and Manic M. Building energy load forecasting using deep neural networks. *IECON 2016 - 42nd Annu. Conf. IEEE Ind. Electron. Soc.*, 2016; 7046–7051.
[17] Babic B and Sobajic DJ. Unsupervised / supervised learning concept for 24-hour load forecasting t t t t. *Electr. Eng.*, 140(4), 1993.
[18] Asare-Bediako B, Kling WL, and Ribeiro PF. Multi-agent system architecture for smart home energy management and optimization. *IEEE PES ISGT Eur.* 2013: 1–5.
[19] Haack J, Akyol B, Carpenter B, Tews C, and Foglesong L. Volttron: An agent platform for the smart grid. *Proc. 2013 Int. Conf. Auton. Agents Multi-agent Syst.*, no. Aamas, 2013; 1367–1368.