Asynchronous Distributed IOT-Enabled Customer Characterization in Distribution Networks: Theory and Hardware Implementation

Andrew Campbell
University of Utah

UUCS-21-017

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

10 August 2021

Abstract

This work proposes and implements an asynchronous distributed IoT (Internet of Things)-enabled customer characterization framework to classify customer's load consumption behaviors in electric distribution networks. More specifically, the proposed framework enables robust fully distributed clustering of customers' electricity consumption habits in a highly scalable and interoperable framework. The proposed clustering method also eliminates the need for hefty synchronization efforts typical to other distributed clustering algorithms. The theoretical foundations of designing the proposed framework are introduced. The orchestration of the layers and the applications integrated in the proposed framework are demonstrated in an experimental implementation on a real-time network adopting a data-centric databus architecture. The results of the experimental implementation on IoT development boards demonstrate that the proposed framework can characterize customer categories with a 95% accuracy compared to classical centralized k-means clustering, while ensuring seamless and interoperable peer-to-peer (P2P) information exchange. This proved to be highly scalable and applicable to the real-world.

ASYNCHRONOUS DISTRIBUTED IOT-ENABLED CUSTOMER CHARACTERIZATION IN DISTRIBUTION NETWORKS: THEORY AND HARDWARE IMPLEMENTATION

by

Andrew Campbell

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

Thomas Henderson, PhD Thesis Faculty Co-Advisor	Mary Hall, PhD Director, School of Computing
MASOOD PARVANIA Masood Parvania, Ph.D Thesis Faculty Co-Advisor	Sylvia D. Torti, PhD Dean, Honors College

July 2021
Copyright © 2021
All Rights Reserved

1 Abstract

This work proposes and implements an asynchronous distributed IoT (Internet of Things)-enabled customer characterization framework to classify customers' load consumption behaviors in electric distribution networks. More specifically, the proposed framework enables robust fully distributed clustering of customers' electricity consumption habits in a highly scalable and interoperable framework. The proposed clustering method also eliminates the need for hefty synchronization efforts typical to other distributed clustering algorithms. The theoretical foundations of designing the proposed framework are introduced. The orchestration of the layers and the applications integrated in the proposed framework are demonstrated in an experimental implementation on a real-time network adopting a data-centric databus architecture. The results of the experimental implementation on IoT development boards demonstrate that the proposed framework can characterize customer categories with a 95% accuracy compared to classical centralized k-means clustering, while ensuring seamless and interoperable peer-to-peer (P2P) information exchange. This proved to be highly scalable and applicable to the real-world.

Table of Contents

1 Abstract **ii**

2 Introduction **1**

3 Background **3**

4 Completed Work **5**

4.1 Introduction to the Developed Framework 5

4.2 Mathematical Formulation 9

4.2.1 Algorithm Pseudo Code 14

4.3 Implementation of the Asynchronous Distributed Clustering Algorithm . . 15

4.3.1 Load Demand Data 15

4.4 Simulation and Experimental Setups 16

4.4.1 Simulation proof of Concept 16

4.4.2 Experimental Setup 17

4.4.3 Hyper-Parameters 18

5 Results and Discussion **19**

5.1 Accuracy and Distance Metrics 19

5.2 Simulation Results 20

5.3 Experimental Validation 21

5.4 Discussion 23

6 Conclusions **24**

7 References **25**

2 Introduction

In power systems, one of the fundamental concerns facing utilities and the grid is ramping. Ramping is defined as the rate of change of the load on the grid. Since the baseload generation of the power grid is slow to scale up or down production, rapid increases or decreases in load can cause problems for the grid as they can lead to dramatic price increases and in some situations lead to blackouts. Additionally, the grid is limited by the frequency and voltage of electricity transmission and rapid increases or decreases in load can destabilize the transmission frequency resulting in blackouts. Utilities can attempt to mitigate ramping by having immediately dispatchable generation units on standby for when the rate of consumption increases. More recently, utilities and Independent Service Operators (ISOs) have begun using demand response programs to incentives customers to decrease their load to reduce the ramping on the grid.[24]

Demand response is the practice of incentivizing electricity consumers to change their load to limit the effects of the demand peak and to reduce ramping [24]. The demand peak is another problem that faces the grid. In some regions, the total amount of generation cannot match the peak demand and can result in power outages. While it is possible for utilities to increase their total available supply by building additional generation units, these generation units are very expensive since they are only needed during a small portion of the day [24]. This increased cost is typically transferred to consumers [24]. Demand response programs on the other hand, while an inconvenience to customers, usually results in cheaper electricity prices [24]. When enough consumers change their consumption habits to reduce the ramping and total peak consumption, utilities do not need to increase their total generation capacity and do not have to rapidly scale up generation, thus resulting in cheaper power production and electricity prices. Additionally, as the grid modernizes, demand response programs will become more effective, increasing their adoption and importance.

As consumers and the grid modernize, more smart devices are being incorporated into the grid [18]. In residential homes, these smart devices are typically internet equipped devices that can schedule their demand loads. A typical example is a smart air conditioning(AC) unit which controls the activation of AC to reduce costs to the owner. In the context of this work, these smart devices are considered Internet of Things (IoT) devices. These IoT devices, when coupled with a demand response program can effectively schedule loads and reduce the peak and ramping on the grid [18]. One strategy that a utility or ISO can employ for demand response is learning customers consumption habits from the customers' IoT devices and using this information for designing a demand response program. In total, understanding customers' electricity consumption behaviors plays a vital role in distribution planning, demand response, market segmentation and management, energy efficiency, and other applications that enhance the reliability and efficiency of distribution networks.

Customer characterization profiles can also be used for planning infrastructure since regions containing customers with a higher power consumption will likely require additional investments as the population grows. Additionally, customer consumption behavior can be directly used for market segmentation which is a valuable tool for ISOs and utilities interested in demand response programs [18]. More specifically, regions with high consumption and ramping are the regions in which an ISO or utility would benefit the most from a demand response program. There are many more other use cases for customer characterization profiles, but it is beyond the scope of this work. Rather, it is assumed that a customer characterization profile is a useful tool for Demand Response programs.

This project proposes a distributed clustering algorithm that is both scalable and robust. More precisely, it proposes an asynchronous distributed IoT-enabled customer characterization framework that provides a fully distributed and asynchronous clustering algorithm to characterize customer consumption profiles. The proposed clustering algorithm utilizes domain specific knowledge to perform feature construction via a proposed function

mapping. The mapping also serves as a dimensionality reduction for clustering. Unlike other distributed clustering approaches, the proposed algorithm is designed for peer-to-peer (P2P) networks and is asynchronous. This work differs from other asynchronous P2P algorithms by abstracting the networking and clustering functionalities. That is, the networking stack is decoupled from the clustering in both implementation and theory. The networking functionalities use a data-centric communication model to ensure scalability and interoperability in real-world deployment. Simultaneously, a hardware implementation of the proposed clustering framework on embedded IoT devices over an Ethernet network utilizing data sets with varying clusterablity [3] demonstrates that the proposed framework produces accurate customer clusters, is efficient in terms of memory, time, and network resources, and is highly scalable for real world applications.

3 Background

Most of the previous works that address customer clustering utilize centralized clustering algorithms with offline data. For instance, authors in [16, 20, 27] presented comparative studies of the various clustering approaches used in the vast literature, such as k-means, k-mediods, and self organizing maps for clustering electricity demand profiles according to daily load patterns. In [7, 8, 13, 21], authors used partition clustering methods, such as k-means, for studying electricity consumption habits of residential customers. In [25], the authors used fast search and find density peaks to obtain the typical dynamics of consumption behavior. The authors in [14, 23] used shape-based clustering approaches based on dynamic time wrapping to categorize household load profiles. In [17], the authors propose a model-based approach using delay coordinate embedding for dynamic load-clustering. In [5], the authors proposed three hierarchical clustering methodologies that allow for the capturing of different characteristics of a time series based on a

set of dissimilarity measures computed over different features. Note that all of the above methods are centralized.

While highly effective in certain situations, centralized clustering algorithms are often difficult to implement in real-world scenarios. For instance, real-time sensor networks, such as smart metering infrastructure, collect huge amounts of data from geographically dispersed locations making centralization impractical (i.e., effective storage and processing of big data). From a system security perspective, centralization usually implies a single point of failure. Considering the importance of grid operation for public security, limiting the occurrence of single points of failure is in the best interest of the general public. In contrast, distributed algorithms rarely have a single point of failure and do not require powerful computers or servers since the computations are distributed among the nodes of the network. Therefore, in the domain of power utilities where big data, security, and computational resources are important considerations, distributed algorithms are advantageous.

The fundamental challenge for distributed clustering algorithms is ensuring, via the algorithmic design, that each node has a global clustering of the whole data set without accessing the entire data set. This can be reduced to three smaller problems: 1) what computations each node performs, 2) what data each node shares, and 3) how each node shares its data. These three questions are usually answered by the algorithm's synchronization requirements. The work in [15, 19] are asynchronous algorithms, and thus each algorithm runs the same computations and uses the same protocol for data sharing. However, in both of these works, the networking protocol was not abstracted from the clustering algorithm. That is, the networking protocol was designed with the clustering algorithm. The work in [6, 9, 10, 11] all use synchronization. I.e., during clustering individuals are required to wait and do not run independently. While [10, 11] require synchronization at every iteration of the clustering algorithm, [6, 9] only require synchronization for the merging of cluster

boundaries.

Finally, most of the literature utilizes simulated networks and offline data for testing and verification. While simulated networks work well for evaluating the theoretical efficacy of a distributed algorithm, they don't capture most of the challenges of real networks, such as heterogeneity, local irregularities, unpredictable congestion and rapid changes, among other difficulties [12]. Additionally, an algorithm's theoretical effectiveness and its real-world effectiveness frequently differ. Therefore, an algorithm that performs well in the idealized controlled environment does not necessarily perform well in a real implementation.

4 Completed Work

4.1 Introduction to the Developed Framework

The framework developed here assumes that each house within a distribution network has an IoT Aggregator, named Intelli-Agg, that collects consumption data of all the customer's loads (e.g. air-conditioning, refrigerator, and all other household appliances). Each Intelli-Agg is capable of predicting its own day ahead demand curve. However, forecasting of demand curves is well studied and beyond the scope of this work [28]. For this manuscript, day ahead curves were generated a priori. The architecture of Intelli-Agg is shown in Fig. 1.

At a high level, each Intelli-Agg is composed of 5 modules. 1) The *Feature Constructor* creates the features needed by the clustering algorithm from the day ahead load profile of the customer. The Intelli-Agg then shares its data with its neighbors. 2) The constructed features, along with neighbor data, are then passed to the *Data Assembler* to

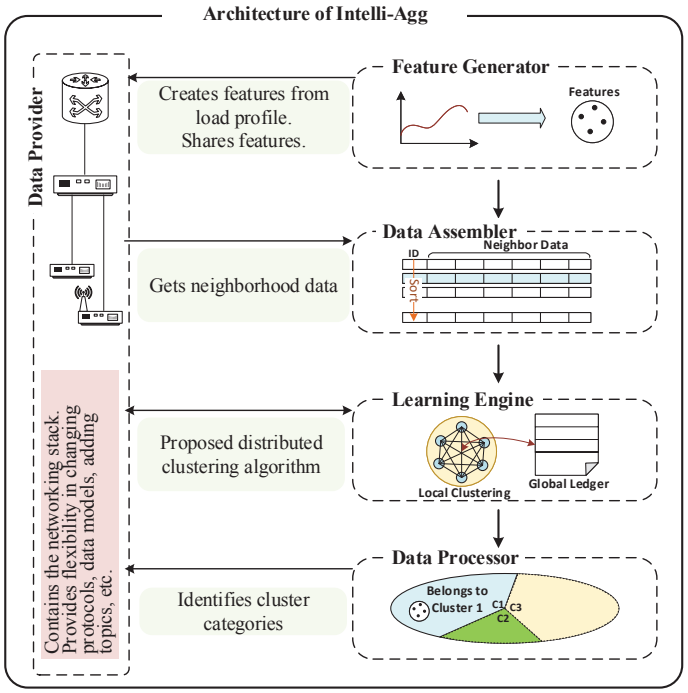

Figure 1: Architecture of the Intelli-Agg.

perform some data pre-processing. 3) The *Learning Engine* performs the proposed distributed clustering algorithm. Throughout this process, data is being shared on a global ledger. Note that the data shared is a summary and the whole data set is not stored in the ledger. 4) The *Data Processor* then returns the learning results (i.e., identifying which cluster the Intelli-Agg belongs to). The end result is a spatio-temporal clustering of electricity consumption as shown in Fig. 2. Finally, 5) all the networking functionalities have been decoupled from the operational functionalities of the proposed algorithm and are handled by the *Data Provider*. That is, any networking schema can be integrated with the proposed clustering algorithm. In this work, the Data Distribution Service (DDS) is used. The steps of the proposed distributed clustering algorithm are summarized in Fig. 3, and are described as follows:

Initialization: At the start of each day each Intelli-Agg is provided with the global average demand ($D_p(t)$) and the ID of the neighborhood it belongs to (N_{ID}). Each Intelli-Agg has its own day-ahead demand curve as well. (Step 1 in Fig. 3).

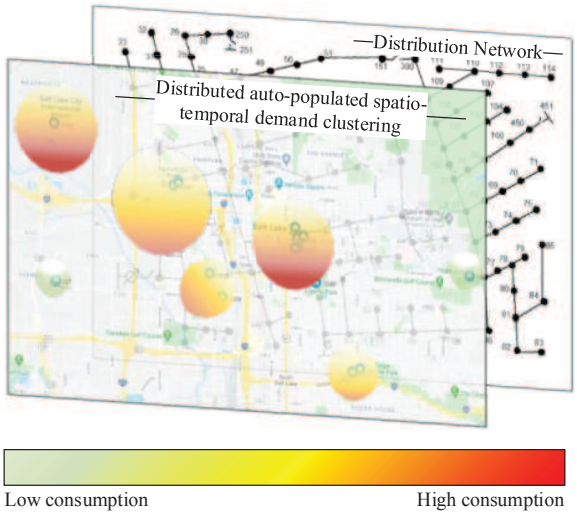

Figure 2: Resulting spatio-temporal demand clustering.

Data Sharing and Feature Mapping: Each Intelli-Agg performs a feature construction and dimensionality reduction with its own day ahead profile. Then, it shares those features with its neighbors (Step 2.a in Fig. 3). That is, every individual in a neighborhood will have the constructed features of all other individuals in the neighborhood.

Distributed K-Means Algorithm: After all the data has been shared with Intelli-Aggs in the same neighborhood, each Intelli-Agg performs one iteration of local clustering (Step 2.b in Fig. 3). The local clustering is based on k-means. Traditionally, k-means is centralized and converges to the centers by repeated iterations and random initialization. In this work, the proposed algorithm requires that clustering be fully distributed, efficient, and scalable to real-world uses.

This distributed k-means algorithm consists of 2 major components: 1) local clustering, and 2) a global ledger. On each iteration, an Intelli-Agg in the network reads weighted centers from the ledger and adds the weighted centers to its local data set. The "weight" of the added center is equivalent to the number of elements whose nearest cluster is that center. For clustering, the weight is equivalent to multiple copies of the center (i.e., if center1 has a weight of 2, for clustering purposes, there are 2 copies of center1 in the

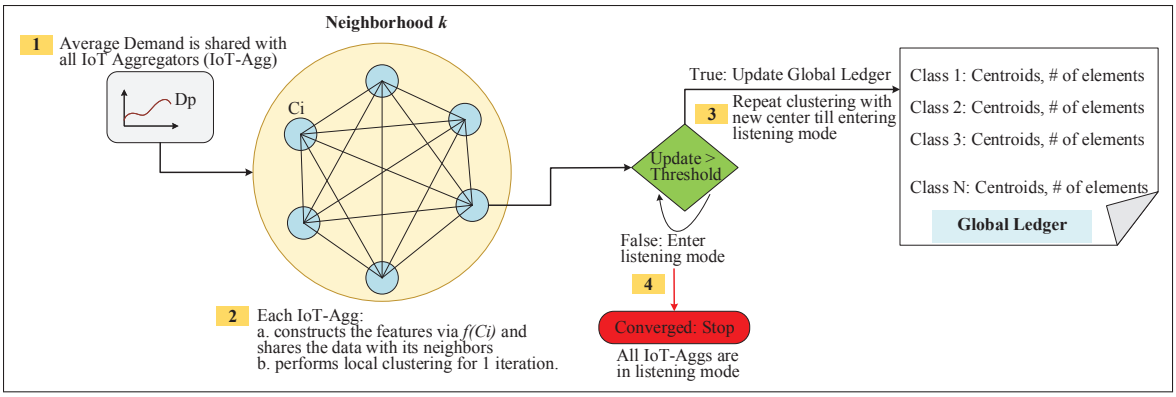

Figure 3: Proposed asynchronous distributed clustering algorithm.

data set). This weighting ensures that local abnormalities do not disproportionately affect the centers during local clustering. After reading the weighted centers, the algorithm then performs a traditional centralized k-means clustering on the appended data set locally. It uses the global centers for initializing the centers. That is, each Intelli-Agg, after reading the centers from the ledger, appends it local data with the global centers and runs k-means. Next, it compares the locally clustered centers to those on the ledger. If the Euclidean distance (L_2 norm) between the centers on the ledger and the local centers is greater than a specified threshold, the Intelli-Agg pushes its local centers to the ledger along with updated weights (Step 3 in Fig. 3). That is, if the Intelli-Agg, after clustering with the new weighted centers, changes which centers its data points belong to, it will increase or decrease the weighting accordingly. It also increases the threshold value by the specified step size after every update. If the Intelli-Agg cannot make an update larger than the threshold, it enters listening mode (Step 4 in Fig. 3).

Convergence and Cluster Identification: The algorithm converges once all Intelli-Aggs enter listening mode and the global ledger is no longer updated. Each Intelli-Agg then reads the final centers on the ledger and identifies the cluster it belongs to based on those centers. Convergence is guaranteed since the step size increases the threshold after each update.

4.2 Mathematical Formulation

Time series data provides a challenge for clustering-based learning approaches as it increases the required computation and complexity for learning. Particularly with Euclidean distance, as dimensionality increases, the distance between points becomes indistinguishable, thus devaluing the clustering results. Furthermore, as data collection and prediction capabilities increase their granularity and accuracy, time-series will become more computationally intensive and the curse of dimensionality will become more prevalent. To address the former challenges, in this work, a function mapping from the time series data to a single point in a lower dimensional vector space is proposed. This mapping ensures that data can easily be clustered and is scalable. Note that the proposed mapping moves a function $f(t) : \mathbb{R} \to \mathbb{R}$ to a point $\in \mathbb{R}^{4n}$, but it is typical to represent a demand curve as a vector in $\mathbb{R}^{24x}$, where x is the number of samples per hour. Using this representation, then indeed the proposed mapping is a dimensionality reduction. Here, n is the number of partitions over the time interval. If $n = 1$ then the full 24-hour time period is mapped into $\mathbb{R}^4$. If $n = 2$ the time period would be partitioned into two smaller intervals, and so on.

Many of the current feature construction strategies in the literature rely on Fourier transformations, Markov models, and neural networks[4, 22, 26]. In the case of time-series, it is normal to perform a signal deconstruction as a dimensional reduction. This work appeals to simplicity and uses expert domain knowledge to perform feature construction. The following discussion will present the features that were constructed to represent the time-series demand data along with the rationale of each feature.

Four features were constructed for dimensionlaity reduction. These features allow for both a robust clustering and preservation of valuable information from the demand profiles. Let $C_i(t) : \mathbb{R} \to \mathbb{R}$ represent the demand profile for customer i at a given time $t \in [0, 1]$. For any t, $C_i(t)$ is the day ahead demand for customer i at time t. A time period is

partitioned by t_a and t_b such that $t_0 = 0, t_n = 1$ and $t_0 \leq t_a < t_b \leq t_n$. The four features are: 1) the cost function, 2) peak consumption, 3) peak consumption time, and 4) ramping.

1. ***The Cost Function:*** $\int_{t_a}^{t_b} Cost(C_i(t))$

   Let $D_p(t)$ be the average demand curve for the specified population and p be the maximum value of the curve, thus $p = max(D_p(t))$. Note that the choice of population size is an implementation decision dependent on the topology of the distribution network. The cost function is defined as:

$$Cost(C_i(t)) = \frac{C_i(t) - D_p(t)}{p - D_p(t) + 1} \tag{1}$$

   Here, the difference between the customer's demand and the average population demand is calculated, and then it is scaled down by the distance to the peak. That is, $C_i(t) - D_p(t)$ is scaled by some $s \in [\frac{1}{p+1}, 1]$, where $s \to 1$ as $D_p(t) \to p$. This ensures that a reduction or increase in consumption of power is weighted more when it occurs closer to peak demand. By integrating this feature, the total cost difference over the specified time period for a customer compared to the average consumption of the population is computed. This should aid in forming 3 demand clusters: Low, less than average; Base, average; and High, more than average demand. Although the scaling is not the real monetary cost of electricity, the proportional weighting for demand during peak hours will help ensure that consumption at peak hours is treated as a higher cost. For example, two customers could have identical total demand, but the associated cost of supplying a customer whose peak demand coincides with the global peak time is much higher than that of a customer whose peak consumption does not occur during global peak hours. Using cost in this way will aid in separating consumption habits.

   Claim: $Cost(C_i(t))$ is a homeomorphism.

Proof. Let C be the set of possible customer demand curves restricted by physical constraints. More precisely, for some $c \in C$, $c(t) \in [P_{min}, P_{max}]$ and where $|P_{min}| \leq |P_{max}|$. Now, let $g_{D_p}(c) = c - D_p$ where $D_p \in C$. Notice that for $c_i, c_j \in C$ such that $c_i \neq c_j$ then $g_{D_p}(c_i) \neq g_{D_p}(c_j)$ and that for $d_i, d_j \in g_{D_p}(c)$ where $d_i \neq d_j$, then $g^{-1}(d_i) \neq g^{-1}(d_j)$. Therefore, g_{D_p} is bijective. Also notice that since $D_p, c \in C$ are continuous, the operation $c - D_p$, and its inverse, are also continuous. Therefore g_{D_p} is a homeomorphism.

Let $h(d \in g_{D_p}(c)) = \dfrac{d}{p - D_p + 1}$ where $p = max(D_p)$. Since D_p is continuous and $p+1$ is a constant, $p - D_p + 1$ is continuous and since $p - D_p + 1 \neq 0$, $\dfrac{d}{p - D_p + 1}$ is also continuous, by similar argument h^{-1} is also continuous. Also, notice that if $d_i, d_j \in g_{D_p}(C)$ such that $d_i \neq d_j$ then $h(d_i) \neq h(d_j)$ and if $r_i.r_j \in h(g_{D_p}(c))$ such that $r_i \neq r_j$ then $h^{-1}(r_i) \neq h^{-1}(r_j)$. Therefore since g and h are both homeomorphisms and the composition of homeomorphisms is also a homeomorphism, then the cost function must also be a homeomorphism. In other words, imagine each demand curve as a path in $\mathbb{R}^2$ restricted over $[0,1] \times [-1,1]$, where the path must start at $(0, y_0)$ and end at $(1, y_f)$. The first mapping shifts and bends the path, while the second mapping fixes a point (the maximum) and stretches the path about that point and the endpoints. Here, the function space was scaled by $\dfrac{1}{max(|C_i(t) - D_P(t)|)}$. □

2. ***Peak Consumption:*** $P(C_i(t))$

   P calculates the peak consumption value over the time interval $[t_a, t_b]$. That is, $P(C_i(t)) = max_{t \in [t_a, t_b]}\{C_i(t)\}$ A consumer who has a larger than average demand peak is likely a more expensive consumer, just as a consumer with a smaller than average peak is likely a less expensive consumer. Also, due to physical limitations on distribution lines, the magnitude of consumption is valuable, irrespective of the associated cost of a consumer.

3. ***Peak Consumption Time:*** $PT(C_i(t))$

$PT(C_i(t))$ returns the time value of the peak over $[t_a, t_b]$. That is, $PT(C_i(t))\Big|_{t_a}^{t_b} = P^{-1}(p)$, where $p = max_{t\in[t_a,t_b]}\{C_i(t)\}$ Similar to the magnitude of the peak, knowing when the consumer has maximum demand will aid in determining the consumer's consumption profile. A consumer who has a peak time significantly before or after the average peak will be a less expensive customer. Like peak consumption, peak consumption time, irrespective of cost, is a useful metric in managing loads in power distribution systems.

4. ***Ramping:*** $C_i'(t_p)$

   This is the slope at the peak on the interval $[t_a, t_b]$. More precisely, let $t_p = PT(C_i(t))\Big|_{t_a}^{t_b}$ and $C_i'(t_p)$ be the derivative. Then, we are evaluating for:

$$C_i'(t_p) = \lim_{h\to 0^-}\left(\frac{C_i(t_p+h) - C_i(t_p)}{h}\right) \tag{2}$$

   It is important to know the rate at which a consumer is increasing their power consumption. A consumer who increases there rate of consumption as they approach their peak is a more expensive consumer. Not only that, ramping is an important metric as power systems are limited in their ability to rapidly increase and decrease available electricity both from a distribution and generation point of view.

Accordingly, the purposed feature-construction function mapping is defined as:

$$f(C_i(t))\Big|_{t_a}^{t_b} = \left(\left(\int_{t_a}^{t_b} Cost((C_i(t))\right), \left(P\left(C_i(t)\Big|_{t_a}^{t_b}\right)\right), \left(PT\left(C_i(t)\Big|_{t_a}^{t_b}\right)\right), \left(C_i'(t_p)\right)\right) \tag{3}$$

Algorithm 1: Proposed Clustering Algorithm

Input: $\mathrm{D}_p(t), N_{IDs}, C_i, \{t_0, ..., t_n\}$

Output: c_i^g

1 Construct Features of C_i

for $t_a, t_b \in \{t_0, ...t_n\}$ where $t_0 \leq t_a < t_b \leq t_{n-1}$:

$$v_j = f(C_i(t))\Big|_{t_a}^{t_b}$$

2 Share v_j with all $C_i \in N_{IDs}$ where $j \neq i$

3 Get C^g and W^l from ledger:

if C^g, W^l are empty, return zero vectors

4 $C, W =$ Distro-kmeans(C^g, W^g)

5 if $C \neq C^g$:

wait γ seconds repeat steps 3,4

6 Enter ListeningMode

7 if all $C_i ==$ ListeningMode:

Read ledger: $C^g = L_c$:

$c_i^g =$ DetermineClusterMembership(v_j)

return c_i^g

8 repeat step 7

Algorithm 2: Distro-kmeans

Input: C^g, W^g

Output: C^g, W^g

1 Update V: $V = V \bigcup C^g$

2 Cluster V: $C^l, W^l =$ Clustering results

3 Compute new centers:

for $i \in [1,3] \in \mathbb{Z}$

$$c_i^l = \frac{(c_i^l * w_i^l) + (c_i^g * w_i^g)}{w_i^l + w_i^g}$$

$$w_i^l = \frac{w_i^l + w_i^g}{2}$$

4 for $i \in [1,3] \in \mathbb{Z}$, if $\left\| c_i^l - c_i^g \right\| > \varepsilon$:

$C^g = C^l$ and $W^g = W^l$

$\varepsilon += \delta$

break

5 $V = V_0$

6 return C^g, W^g

4.2.1 Algorithm Pseudo Code

Let C_i be an individual Intelli-Agg in neighborhood k_n where $|k_n| = j$. Let $V_i = \{v_1, ..., v_j\}$ be the set of points in $\mathbb{R}^4$ where $f(C_i) = v_i \; \forall C_i \in k_n$, where f is Equation 3. Each C_i has its own local V_i. Let $C^g = \{c_1^g, c_2^g, c_3^g\}$ represent the global centers for Low, Base, and High clusters, and $W^g = \{w_1^g, w_2^g, w_3^g\}$ the global weights corresponding to each center. Let ε be the threshold value, and δ be the step size. At each step of the algorithm C_i will read from the ledger and update V_i so that $V_i = V_i \bigcup C^g$. C_i, then, performs local clustering on V_i but it weights each $c_m^g \in V_i$ by the corresponding w_m^g. The clustering returns local centers labeled $C^l = \{c_1^l, c_2^l, c_3^l\}$ and the corresponding weights are labeled $W^l = \{w_1^l, w_2^l, w_3^l\}$. Each weight is the number of v_i in the corresponding center, i.e., let $d(x, y)$ be the Euclidean distance between x and y, then $w_m^l = |\{x \in V_i : \min_{\forall c^l \in C^l} (d(v_i, c^l)) = c_m^l\}|$.

Then, C^l is updated using the formula:

$$c_m^l = \frac{(c_m^l * w_m^l) + (c_m^g * w_m^g)}{w_m^l + w_m^g} \tag{4}$$

and the weights are updated according to:

$$w_m^l = \frac{w_m^l + w_m^g}{2} \tag{5}$$

If the distance from C^l and C^g is greater than the threshold ε, then C^g is set to C^l and W^g is set to W^l:

$$\begin{aligned} &\text{if } \exists c_i^l \text{ such that } \left\| c_i^l - c_i^g \right\| > \varepsilon : \\ &\qquad\qquad C^g = C^l \text{ and } W^g = W^l \\ &\qquad \text{else:} \\ &\qquad\qquad C^g = C^g \text{ and } W^g = W^g \end{aligned} \tag{6}$$

The clustering process is summarized in Algorithms 1 and 2.

4.3 Implementation of the Asynchronous Distributed Clustering Algorithm

4.3.1 Load Demand Data

The Topical Meteorological Year (TMY3) data set was utilized to generate load profiles representing residential houses. The distribution network considered in this work is the IEEE 33-node feeder distribution network. The TMY3 data set contains hourly load profile data for residential buildings based on the Building America House Simulation Protocols. This data set also uses the Residential Energy Consumption Survey (RECS) for statistical references of building types by location. Hourly load profiles are available for all TMY3 locations in the United States [1]. The following assumptions are made: 1) Each node in the the 33-node feeder represents a neighborhood with only residential houses. 2) There are three classes of houses classified according to their consumption patterns as follows: $L(Low)$ are houses with power consumption less than average demand, $B(Base)$ are houses with average demand, and $H(High)$ are houses with more than average demand. 3) Average demand is defined as the average of the aggregated demands of all nodes in the 33-node test feeder.

Two populations were generated from the TMY3 data set for residential houses in Salt Lake City, Utah: 1) P_{rep}, where each node contains at least one house from each category (i.e., L, B, and H), and therefore, each node is representative of the entire population; 2) P_{nonrep}, where each node contains houses randomly selected from only one category, and therefore, each node is non-representative of the entire population. To guarantee that the proposed distributed clustering algorithm generalizes well for real world scenarios, 13

variations of P_{rep} and P_{nonrep} were generated. Each variation set was labeled iP_{rep}, where i represents the level of variation. That is, each iP_{rep} had a random $\frac{1}{3}$ of the demand curves in P_{rep} shifted by a random value in $[-i,i]$ where $i \in \{0,1.5,2,2.5,3,3.5,4,5,6,7,8,9,10\}$. The remaining $\frac{2}{3}$ of the curves are unchanged.

The three clusters (low, base, and high) can be clearly seen in the original set. However, this is less intuitive to identify as the variation level increases reaching 10. Testing the proposed clustering algorithm on high variation levels ensures that the proposed algorithm generalizes well for real world scenarios. *Mathematical Formulation:* Let $P \in \{P_{rep}, P_{nonrep}\}$, $C_i(t) \in P$ be a demand curve, and $v \in V = \{0,1.5,2,2.5,3,3.5,4,5,6,7,8,9,10\}$. Then let $\frac{1}{3}P$ be a random subset of P containing $\frac{1}{3}$ of the curves from P. Then $\forall C_i(t) \in \frac{1}{3}P$, pick a random value from $[-v,v] \in \mathbb{R}$, call it r and generate vP by

$$vP = \bigcup_{\forall i} \{C(t) | C(t) = C_i(t) + r\} \tag{7}$$

4.4 Simulation and Experimental Setups

4.4.1 Simulation proof of Concept

A proof of concept experiment was conducted in order to test the validity of the proposed clustering algorithm. The algorithm was implemented in Python and tested on the two data sets P_{rep} and P_{nonrep} with all their variations. However, the P2P network was simulated on a single device. That is each, Intelli-Agg was represented by an object, and each object performed synchronous communications between other objects.

4.4.2 Experimental Setup

The proposed clustering framework was implemented in C++11 using the standard libraries and the RTI DDS API [2]. A total of 7 embedded IoT-devices were used. These included 5 Odroid X-U4 devices with an Octa core ARM Cortex-A15 Quad 2Ghz 32-bit processor running lightweight Ubuntu Linux, 1 Beaglebone Black with an AM3358 ARM Cortex-A8 32-bit processor running Debian Linux, and 1 laptop with 2.4Ghz intel i7 64-bit processor running full Ubuntu Linux. All the devices were connected to a local area network (LAN) via a NetGear NetHawk router and a 24-port Cisco Ethernet switch.

Multiple threads on each device were executed, where each thread represented an individual Intelli-Agg. In order not to exceed the network card capacity of the IoT-devices while conducting the experiments, no more than 15 threads (i.e., codes for Intelli-Aggs for 15 customers) were compiled and executed on the Odriod X-U4 devices, 20 on the laptop, and 5 on the Beaglebone. Therefore, subsets of iP_{rep} and iP_{nonrep} of sizes 67 and 64, respectively, were considered. Call the subsets iP^s_{rep} and iP^s_{nonrep}. For each iP^s_{rep} and iP^s_{nonrep}, the individuals across the devices were distributed conforming to their respective networking constraints. These subsets were randomly generated from iP_{rep} and iP_{nonrep}.

The networking in the proposed clustering framework follows a data-centric communication scheme. More specifically, the Data Distribution Service (DDS) middleware is utilized to establish a data-centric messaging scheme, utilizing a databus that follows a P2P publish-subscribe (Pub-Sub) communication model. The Pub-Sub model enables information streams (i.e., instances) to be reliably disseminated to several applications, while maintaining adequate network performance. In this implementation, the DDS middleware establishes a global dataspace that is accessed by all Intelli-Aggs, as shown in Fig. 4. The circles in this dataspace represent topics. Each topic provides an identifier to data items within the global data space. Topics have names, data-types, and quality of service (QoS)

profiles related to the data itself. The arrows connecting the Intelli-Aggs to these topics in the data space represent the contributions to and the data-retrievals from them by the publishers and subscribers of the involved Intelli-Aggs, respectively. An individual shares its mapped data with its neighborhood (i.e., Step 2.a Fig. 3) by updating the data in the topic "Nbd_Data." Here, all Intelli-Aggs update the data in the same topic, however each Intelli-Agg uses its assigned C_{ID} as a key representing its unique identifier. Next, Each IoT_Agg reads the data of its neighbors from "Nbd_Data" topic by filtering for the keys of the other Intelli-Aggs in the neighborhood. Similarly, when updates are made to the global ledger (i.e., Step 3 in Fig. 3) an individual updates the "Centers" topic. The "Centers" topic contains the centers location and the corresponding weight.

4.4.3 Hyper-Parameters

There are 3 hyper-parameters for the proposed clustering algorithm. The first two are the update threshold denoted by ε and the step size denoted by δ. The third hyper-parameter is the pause time between local clustering iterations denoted by γ. While intuition suggests smaller ε and δ values, in practice it was found $\varepsilon = 0.101$ and $\delta = 0.041$ were an adequate compromise between accuracy and overall clustering duration. When ε, $\delta \to 0$, accuracy did improve, but total clustering duration increased as well. It was noted that δ affected clustering duration considerably more than epsilon. It was also observed that after reaching a sufficiently small ε and δ values, marginal decreases provided insignificant accuracy improvements. The correct choice of γ is dependent on the networking and computation speeds of the Intelli-Aggs. For networks with slower communication speeds and/or lower computational resources, a higher γ value is required. In general, increasing γ did provide greater accuracy, but the clustering duration also increased considerably. In this work, $\gamma \in [1,2]$ yielded good results. The choice of $\varepsilon, \delta, \gamma$ is a logistical decision about the acceptable duration of clustering over the network.

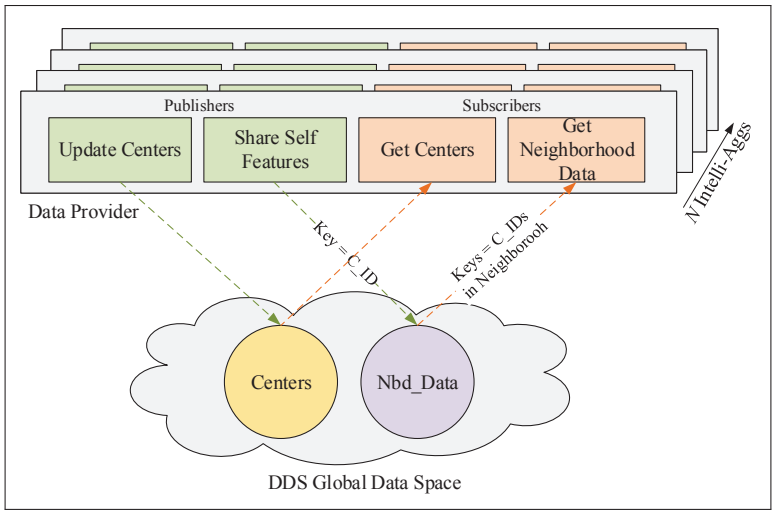

Figure 4: DDS network architecture.

5 Results and Discussion

This section presents the simulation results, and the experiments designed and implemented to validate the efficacy and accuracy of the proposed asynchronous distributed clustering framework.

5.1 Accuracy and Distance Metrics

To evaluate the efficacy of the clustering algorithm, we defined two metrics for success: Accuracy and Approximate distance. For both of these metrics, we considered the data produced by a centralized k-means algorithm to be the ground truth. Let G be the ground truth and D be the set of labels produced from the algorithm. Accuracy is defined by the proportion of labels that were the same for centralized and distributed:

$$\forall \text{ labels, } l_g \in G \text{ and } l_d \in D \text{ Accuracy} = \frac{|\{l : l_g = l_d\}|}{|l_g|} \tag{8}$$

Approximate distance is defined by a normalized Euclidean distance of each center pro-

duced by the algorithm from each center produced by the centralized k-means. That is, for Ground truth = $\begin{pmatrix} c_1 & c_2 & c_3 \end{pmatrix}$ and produced centers = $\begin{pmatrix} c'_1 & c'_2 & c'_3 \end{pmatrix}$ we have:

$$Distance = \frac{\frac{\|c_1 - c'_1\|}{\|c_1\|} + \frac{\|c_2 - c'_2\|}{\|c_2\|} + \frac{\|c_3 - c'_3\|}{\|c_3\|}}{3} \tag{9}$$

Defining the distance metric as shown in Equation 9 ensures that individual centers are scaled by the ground truth, thus, removing any skewing caused by the magnitude of the ground truth centers. Therefore, this metric provides the average radius of where the distributed centers are from the ground truth. A radius of 0 means the produced center matches the ground truth, while a radius of 1 means the produced center is as far from the ground truth as the magnitude of the ground truth.

5.2 Simulation Results

As shown in Fig. 5 (a), the proposed clustering algorithm achieved an average distance (i.e., Equation 9) of 0.028 for iP_{rep} and 0.023 for iP_{nonrep} across all variation levels. This shows that the resulting centers from the proposed algorithm clusters are almost identical to those produced with traditional centralized k-means.

Fig. 5 (b) shows the accuracy of labeling electricity consumption in the right category. For both iP_{rep} and iP_{nonrep}, the recorded average accuracy was 0.9944 for all variation levels. For iP_{rep}, 84.6% of the entire data had and accuracy of 1, and the remaining had a minimum accuracy of around 0.98, while for iP_{nonrep}, 70% of the entire data had and accuracy of 1, and the remaining had a minimum accuracy of around 0.96. This implies that the proposed algorithm generalizes well for real world scenarios.

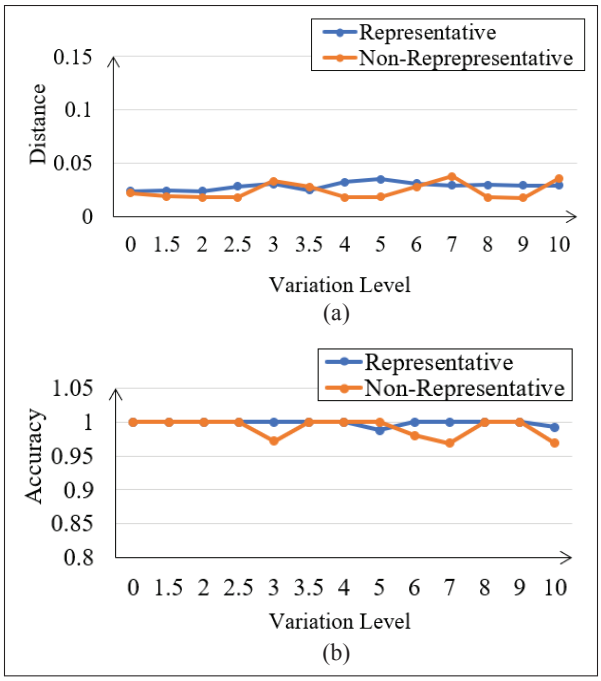

Figure 5: Simulation results for distance and accuracy versus variation.

5.3 Experimental Validation

Four iterations of the proposed clustering were performed for each variation level per subset. The approximate distance and the accuracy for each iteration were computed and the averages were recorded. The results for the accuracy and distance metrics are summarized in Fig. 6 (a) and (b), respectively.

For all iP^s_{rep}, our proposed distributed clustering algorithm produced an average accuracy of 0.9699 with a standard deviation of 0.0268 and an average distance of 0.0541 with standard deviation of 0.0268. For all iP^s_{nonrep} an average accuracy of 0.9491 was observed with standard deviation of 0.0303 and an average distance of 0.0685 with a standard deviation of 0.0226. See Table 1. This suggests that the implementation is very robust as both data sets and metrics produced low standard deviations. If the Rep and Non-Rep results are aggregated an average accuracy and standard deviation of 0.9596 and 0.0294 are achieved. If it is assumed that the accuracy of the proposed algorithm follows a normal

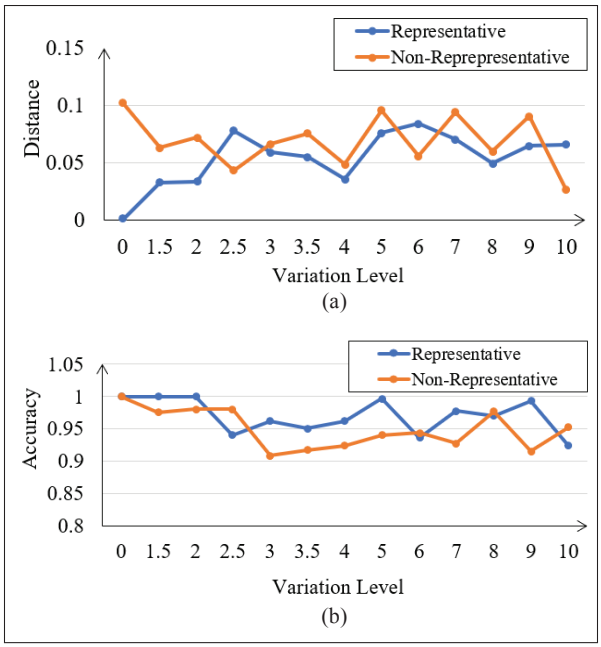

Figure 6: Experimental results for distance and accuracy versus variation.

distribution, then the results suggest that 95% of computed results will have a clustering accuracy no worse than 0.9007, since 0.9007 is two standard deviations away from the mean. Similarly, aggregating approximate distance yields an average of 0.0614 with a standard deviation of 0.0229. Again, if a normal distribution for the approximate distance is assumed, then 95% of produced results will have distance no larger than 0.1071. This means that in real world scenarios, the proposed clustering algorithm will still perform well, and only in rare occasions, (5% of the time) will the algorithm perform below 90% accuracy. Therefore the clustering algorithm is robust for real applications.

In summary, the proposed clustering framework can correctly classify approximately 95% of all individuals in a network. This accuracy is consistent which is evident from the low corresponding standard deviations. Additionally, since 13 levels of variance were used and both representative and self-similar neighborhoods were tested, the proposed asynchronous distributed clustering algorithm generalizes well for real world environments and applications.

Table 1: Standard deviation of experimental results

	Accuracy	Std	Distance	Std
Rep	0.97	0.0268	0.054	0.0268
Non-rep	0.95	0.0303	0.068	0.0226

5.4 Discussion

Synchronization: The proposed algorithm does not require any form of synchronization. As shown in Algorithms 1 and 2, there are no messages between individuals specifying when to perform clustering or what information should be shared. Each individual Intelli-Agg performs the same algorithm. The proposed algorithm clusters asynchronously because of the threshold and step size ε, δ, and because the weights of the centers are included on the ledger and updated at each step. That is, for an individual Intelli-Agg C_i, after making an update, for a subsequent update to occur, C_i's centers must be at least ε distance from the ledger's centers. This locally prevents an individual from continuously updating the ledger, thus, not requiring consensus from all individuals before making an update. The weights handle late joiners. If an individual joins late, although their threshold to update is low, the existing centers are weighted more heavily than its own. Therefore, an individual will either not make an update since they cannot shift the centers by ε, or their update will be sufficiently small not to affect the existing centers. Similarly, if an individual C_i does not have data, the algorithm defaults to updating a zero vector with zero weights. If this situation occurs, then individuals with missing data will be unable to make an update since there update magnitude is always less than ε. These two practices, the threshold and weighted centers ensure that no consensus or other synchronization is required for effective clustering. Since synchronization is not required, the proposed algorithm is more robust to

real world uncertainties, such as disconnected devices or intermittent network connectivity.

Low Computational Resources: The function mapping is only required once and is not computationally intensive. The k-means algorithm, however, is a computationally intensive algorithm, but the number of elements is restricted by the neighborhood size. Each Intelli-Agg performs m-means on its local neighborhood only. Therefore, neighborhoods can be restricted in size to allow for quick computations on devices lacking computational resources. Additionally, since the proposed clustering algorithm restricts the number of local k-means iterations to 1 per update, the local computational complexity for the algorithm is linear with respect to the neighborhood size.

Scalability: The proposed algorithm and its implementation can easily be scaled to larger areas. This is because of the P2P DDS networking, asynchronous clustering, and low local computational resources. The P2P networking means network communication does not require a powerful server that can collectively handle large amounts of packets. Therefore, increased latency from larger networks will not negatively affect clustering performance. Asynchronous algorithms are usually scalable since the network or population size should have a minimal effect on individual nodes. This is true for this framework. That is, the size of the network has a minimal effect on the required computations that each Intelli-Agg must perform, therefore, larger networks will not overload the Intelli-Aggs.

6 Conclusions

This work presented an asynchronous distributed clustering framework, which accomplishes demand curve clustering over a P2P network in a fully distributed manner. This algorithm was implemented in both simulation and in a real-world scenario with IoT devices over a real network. The clustering algorithm in the proposed framework, using

domain knowledge, performs a feature construction that serves as a dimensionality reduction for clustering. It then performs a fully distributed modified k-means algorithm over the P2P network asynchronously. It accomplished this by requiring a threshold, step size, and weighting on the modified k-means algorithm. Intelli-Aggs in the proposed framework communicate over a P2P network using a data-enctric databus network architecture. In simulation, the algorithm was able to cluster datas ets with 0.9942 accuracy when compared to a centralized k-means algorithm. In a real-world application with single board computers representing neighborhoods within a distribution network, the proposed algorithm achieved an accuracy of 0.9595. The data sets ranged from low variance (i.e., easy to cluster) to high variance (i.e., no "natural" clusters), proving that the algorithm, even with highly variant data, can perform well compared to a centralized k-means algorithm. The data sets also varied in composition. Some neighborhoods were representative of the whole population, while other neighborhoods were entirely self-similar. This implies that the proposed algorithm will perform well with real data and is robust to the distribution and composition of load profiles over a real distribution network. Because of the data-centricity, the algorithm is highly scalable and can handle large numbers of connected devices.

7 References

[1] Commercial and Residential Hourly Load Profiles for all TMY3 Locations in the United States. `https://openei.org`.

[2] RTI - The Largest Software Framework Provider for Smart Machines and Real-World Systems. `www.rti.com`.

[3] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 1–8, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR.

[4] Reem Al-Otaibi, Nanlin Jin, Tom Wilcox, and Peter Flach. Feature construction and calibration for clustering daily load curves from smart-meter data. *IEEE Transactions on Industrial Informatics*, 12(2):645–654, 2016.

[5] A. M. Alonso, F. J. Nogales, and C. Ruiz. Hierarchical clustering for smart meter electricity loads based on quantile autocovariances. *IEEE Transactions on Smart Grid*, 11(5):4522–4530, 2020.

[6] M. Bendechache and M. Kechadi. Distributed clustering algorithm for spatial data mining. In *2015 2nd IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM)*, pages 60–65, 2015.

[7] A. Bosisio, A. Berizzi, A. Vicario, A. Morotti, B. Greco, G. Iannarelli, and D. D. Le. A method to analyzing and clustering aggregate customer load profiles based on pca. In *2020 5th International Conference on Green Technology and Sustainable Development (GTSD)*, pages 41–47, 2020.

[8] Kushan Ajay Choksi, Sonal Jain, and Naran M. Pindoriya. Feature based clustering technique for investigation of domestic load profiles and probabilistic variation assessment: Smart meter dataset. *Sustainable Energy, Grids and Networks*, 22:100346, 2020.

[9] M. N. Vrahati D. K. Tasoulis. Unsupervised distributed clustering. In *IASTED International Conference on Parallel and Distributed Computing and Networks*, 2004.

[10] S. Datta, C. Giannella, and H. Kargupta. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1372–1388, 2009.

[11] A. Elgohary and M. A. Ismail. Efficient data clustering over peer-to-peer networks. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 208–212, 2011.

[12] D. Gross, J. F. Shortle, M. J. Fischer, and D. M. B. Masi. Difficulties in simulating queues with pareto service. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 407–415 vol.1, 2002.

[13] Tuong Le, Minh Thanh Vo, Tung Kieu, Eenjun Hwang, Seungmin Rho, and Sung Wook Baik. Multiple electric energy consumption forecasting using a cluster-based strategy for transfer learning in smart building. *Sensors*, 20(9), 2020.

[14] Chang Liu, Xiaodi Wang, Yuan Huang, Youbo Liu, Ran Li, Yang Li, and Junyong Liu. A moving shape-based robust fuzzy k-modes clustering algorithm for electricity profiles. *Electric Power Systems Research*, 187:106425, 2020.

[15] H. Mashayekhi, J. Habibi, T. Khalafbeigi, S. Voulgaris, and M. van Steen. Gdcluster: A general decentralized clustering algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 27(7):1892–1905, 2015.

[16] Fintan McLoughlin, Aidan Duffy, and Michael Conlon. A clustering approach to domestic electricity load profile characterisation using smart metering data. *Applied Energy*, 141:190–199, 2015.

[17] Omid Motlagh, Adam Berry, and Lachlan O'Neil. Clustering of residential electricity customers using load time series. *Applied Energy*, 237:11–24, 2019.

[18] T. R. Olorunfemi and N. Nwulu. A review of demand response techniques and operational limitations. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pages 442–445, 2018.

[19] Cheng Qiao and Kenneth N. Brown. Asynchronous distributed clustering algorithm for wireless sensor networks. In *Proceedings of the 2019 4th International Conference on Machine Learning Technologies*, ICMLT 2019, page 76–82, New York, NY, USA, 2019. Association for Computing Machinery.

[20] Amin Rajabi, Mohsen Eskandari, Mojtaba Jabbari Ghadi, Li Li, Jiangfeng Zhang, and Pierluigi Siano. A comparative study of clustering techniques for electrical load pattern segmentation. *Renewable and Sustainable Energy Reviews*, 120:109628, 2020.

[21] Joshua D. Rhodes, Wesley J. Cole, Charles R. Upshaw, Thomas F. Edgar, and Michael E. Webber. Clustering analysis of residential electricity demand profiles. *Applied Energy*, 135:461–471, 2014.

[22] Seunghyoung Ryu, Hyungeun Choi, Hyoseop Lee, and Hongseok Kim. Convolutional autoencoder based feature extraction and clustering for customer load analysis. *IEEE Transactions on Power Systems*, 35(2):1048–1060, 2020.

[23] T. Teeraratkul, D. O'Neill, and S. Lall. Shape-based approach to household electric load curve clustering and prediction. *IEEE Transactions on Smart Grid*, 9(5):5196–5206, 2018.

[24] Alexandra von Meier. *Electric Power Systems: A Conceptual Introduction*. John Wiley & Sons, Ltd, 2006.

[25] Y. Wang, Q. Chen, C. Kang, and Q. Xia. Clustering of electricity consumption behavior dynamics toward big data applications. *IEEE Transactions on Smart Grid*, 7(5):2437–2447, 2016.

[26] Yi Wang, Qixin Chen, Chongqing Kang, and Qing Xia. Clustering of electricity consumption behavior dynamics toward big data applications. *IEEE Transactions on Smart Grid*, 7(5):2437–2447, 2016.

[27] S. Yilmaz, J. Chambers, and M.K. Patel. Comparison of clustering approaches for domestic electricity load profile characterisation - implications for demand side management. *Energy*, 180:665–677, 2019.

[28] D. Zhang, X. Han, and C. Deng. Review on the research and practice of deep learning and reinforcement learning in smart grids. *CSEE Journal of Power and Energy Systems*, 4(3):362–370, 2018.

Name of Candidate:	Andrew Campbell
Birth date:	May 20, 1999
Birth place:	St. Croix, US Virgin Islands
Address:	803 E Sherman Ave. Salt Lake City, Utah, 84105