# Scalable Prediction of Energy Consumption using Incremental Time Series Clustering

Yogesh Simmhan and Muhammad Usman Noor
University of Southern California
Los Angeles, CA 90089
{simmhan, mnoor}@usc.edu

*Abstract*—**Time series datasets are a canonical form of high velocity Big Data, and often generated by pervasive sensors, such as found in smart infrastructure. Performing predictive analytics on time series data can be computationally complex, and requires approximation techniques. In this paper, we motivate this problem using a real application from the smart grid domain. We propose an incremental clustering technique, along with a novel affinity score for determining cluster similarity, which help reduce the prediction error for *cumulative* time series within a cluster. We evaluate this technique, along with optimizations, using real datasets from smart meters, totaling ~700,000 data points, and show the efficacy of our techniques in improving the prediction error of time series data within polynomial time.**

***Keywords-velocity; time series; predictive analytics; clustering; smart power grids***

## I. Introduction

The unprecedented growth in the ability to collect data from physical and digital systems has led to a heightened focus on management and analysis of Big Data. One of the driving forces in this data explosion has been the pervasive deployment of sensing instruments within various spheres of our lives, ranging from personal monitoring devices like FitBit to environmental monitoring of, say, pollution levels [1]. These "always on" devices generate large *volumes* of data and with high *velocity* – two of the three pillars of Big Data characteristics [2]. In particular, the emergence of *Cyber-Physical Systems (CPS)* such as smart power grids and smart transportation is leading to extensive digital sampling of power and road networks [3]. Datasets generated by such CPS have a time series notion that is valuable when performing data mining and analytics to help control and optimize such systems for the societal good, be it to improve energy efficiency or to ease traffic congestion. However, the time series nature also makes analytics challenging due to the high dimension of the data and causes traditional machine learning and data mining techniques to be computationally intractable [4]. This motivates the need for domain-driven approximate algorithms that offer heuristics for making Big Data analytics computationally tractable while meeting the domain needs [5].

Consider the power grid domain, where millions of power consumers are being upgraded with Smart Meters that can sample energy consumption and report them back to the utility at 15-min granularities, *in real-time* [6]. This is a 3000x jump in the volume of data traditionally collected by power utilities. This also offers unique opportunities for intelligent management of the power grid to improve efficiency and reliability. One canonical smart grid application is demand-response (DR) [7], whereby the utility uses real-time power consumption data from individual customers in the service area to *forecast the future demand* and initiate *energy curtailment programs* that can avoid a supply-demand mismatch. These curtailment programs target individual customers whose power usage is expected to increase in the near future and offer them incentives to shift their impending demand in a bid to relieve stress on the power grid.

Time series forecasting models such as ARIMA can be trained on historical power consumption data to predict the near future based on recent data available from the real-time smart meters [8]. However, two domain challenges emerge when applying such predictive analytics to a large utility service area such as the Los Angeles Power Grid [9] that has over a million consumers: (i) training a prediction model per-customer is computationally costly, and (ii) the variability in the energy usage per consumer causes the prediction models to have high errors. Aggregating data from multiple customers into a single "virtual consumer" helps address both these issues. Besides reducing the number of (virtual) customers to train and prediction for, this can also reduce the temporal variability of the virtual customer, thereby increasing the prediction accuracy [19].

A key goal of this aggregation is to reduce the prediction error of the virtual customer represented by aggregating each cluster of customers. *Hence, the novel distance metric used for clustering is, effectively, the degree to which prediction error is reduced by that aggregation, i.e., if adding a customer to a cluster reduces the prediction error, the distance is smaller.* This innovative distance measure, unlike Euclidian distances, is valuable when clustering time series data not based on similarity in patterns – which is a very high-dimension problem – but based on their *cumulative ability to reduce the prediction error for the cluster*. Tradition techniques such as k-means clustering are not directly usable here since there is no distance metric to find the centroid (or medoid) customer for subsequent iterations [10]. So we turn to *incremental clustering* that offers an approximate but scalable approach.

In this paper, we propose the use of an incremental clustering technique to group customers together into virtual customers, with the goal of minimizing the *cumulative* consumption prediction error. This is distinct from traditional time series clustering that intend to find *similarity* between data points. Our novel distance metric selects the affinity of a customer (i.e. time series) to a cluster based on their ability to reduce the cumulative prediction error of the cluster (Sec. III). This approach has polynomial time complexity that scales for large utility service areas. We examine factors such as number of clusters and initial seeding that impact the clustering (Sec. IV), and analyze these empirically for clustering and prediction of real time series data from Smart Meters for ~85 smart meters collected over 3 months (Sec. V). Our work, while validated for this vital CPS application domain, is applicable to scalable clustering of other large-scale time series datasets, and ties together the volume and velocity of Big Data with the scalable analytics required to make meaningful use of them.

## II. Related Work

Predictive analytics over time series data is well-studied, though less so from a scalability perspective that Big Data entails. Several time series forecasting techniques have been used in domains varying from weather [20] to finance [21]. One of the widely used time series forecasting models is Autoregressive integrated moving average (ARIMA), with many variations. An ARIMA model is initially trained (or fitted) over historical time series data to help determine its coefficients. Subsequently, this model can predict the future time series values that follow these historical values. For e.g., [12] analyzes ARIMA for forecasting global temperature for a ten year period from 2001 to 2010 after training the model for data from 1880 to 2000. The prediction accuracy can be measured using statistics such as Mean Absolute Percentage Error (MAPE). ARIMA has also been used to study household electricity consumption for making daily, weekly and monthly predictions [13]. In [8], the authors use regression-based time series models for hourly home peak load prediction. They observe that variations of ARIMA, like SARMA, have a high prediction power, offering 30% higher accuracy than Support Vector Regression (SVR). Our own prior work has also examined ARIMA for forecasting energy consumption for buildings in a campus micro-grid [14], and it is shown to be better for making near term predictions than techniques like regression trees [15]. Hence, we adopt ARIMA as a standard prediction model whose errors we aim to further reduce through the proposed clustering.

Much work has gone into clustering, which maximizes the similarity between items within a cluster and dissimilarity between clusters using Euclidean or Gaussian distance measures and clustering algorithms such as partitioning, hierarchical and evolutionary methods [22]. Distance measures used in partitioning and hierarchical algorithms have been developed for static data [16] but do not scale to data streams due to their high dimensionality [23]. Data streams are a type of Big Data which have ordered data points which, due to the large volume, can only be scanned once or a few times [24]. Data streams can be clustered using multilevel algorithms that divide the streams data into sub-streams to extract feature summaries from them and subsequently clusters the sub-streams on these features using partitioning/hierarchical clustering [25, 26].

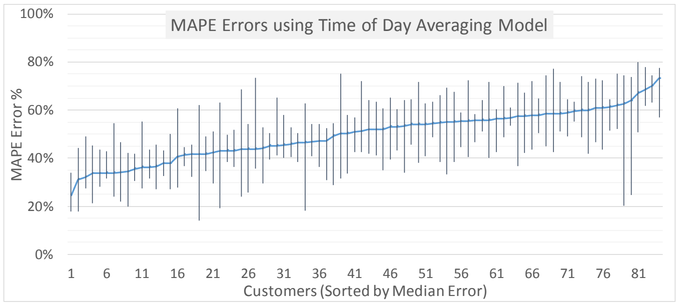
Time series is another form of Big Data, with high dimension since each interval is treated as a dimension. Static techniques such as relocation and agglomerative hierarchical clustering have been applied on time series data but as the number of dimensions grow with the length of the time series, these techniques become computationally expensive. Feature-based techniques attempt to reduce the problem space by extracting motifs from the time series and replacing the time series with (a much smaller number of) motifs, thus achieving dimensionality reduction. For e.g., wavelets that represent data in terms of average and differences of a prototype function has been proposed for motif discovery to approximate a time series [17]. The initial assignment of the time series using k-means is done at the lower wavelet resolution and the algorithm moves to a higher resolution after determining a good initial cluster center. Innovative distance measures have been proposed for time series data, like Dynamic Time Warping (DTW) [18] that allows non-linear alignment between two time series, unlike Euclidian. DTW clustering is cast as a generalizable anytime algorithm with approximations to trade-off execution time against quality of results.

These methods treat clustering as an end in itself and also require the identities of the individual time series within a cluster to be retained. In fact, k-means needs to identify a time series as the centroid within each cluster to perform subsequent iterations. However, we propose to use clustering as a means to reduce the prediction errors of the cumulative time series within a cluster. Or, in other words, *rather than reduce the dimension of each time series, we retain these dimensions and sum the values of all time series within a cluster to offer one* virtual *time series per cluster.* This is a unique application of clustering and requires the introduction of a *novel distance measure that reduces the prediction error for this aggregated time series* (for a model like ARIMA) rather than use distance measures based on similarities between time series.

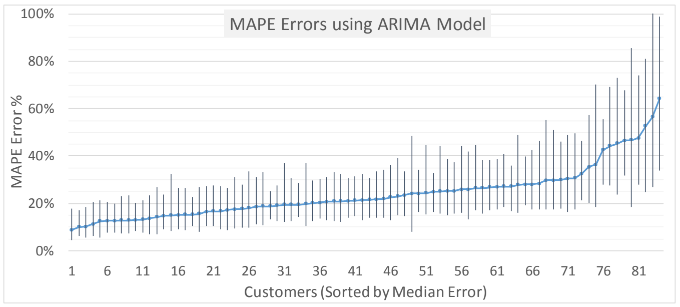
Incremental clustering algorithms [11] process one data object in each iteration and assigns it to a cluster based on a similarity function. The goal is to perform a single scan of the data and hence reduce the time complexity of clustering. In particular, this can help new data that arrives to be clustered as it is available. We adopt an incremental clustering approach in this paper in combination with our novel distance metric for clustering the time series data.

## III. Motivation & Approach

Smart Grids are facing a data explosion, with millions of customers getting upgraded to smart meters and power utilities contending with over 100 energy consumption data points per customer, per day, sampled every 15-mins that they need to analyze and use for intelligent grid operations. Predicting the consumption demand for customers in the next few hours or days helps a utility plan for additional generation, say, by starting a dormant power generation unit, purchasing power from the energy market, or targeting customers for demand reduction. In the latter case, customers are given incentives to shift their non-critical loads to off-peak hours.



(a) Time of Day Averaging Model



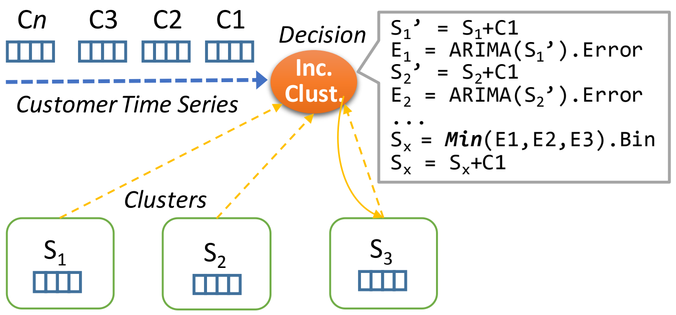
(b) ARIMA Model

**Figure 1**. Median MAPE Forecasting Errors over 30 days for 84 smart meters. Q1 and Q3 error deviations are shown in whiskers.

ARIMA is one of the better performing time series forecasting models [14] compared to baseline averaging models. For e.g., Figures 1(a) and 1(b) show the median MAPE prediction errors of a Time of the Day averaging model (Y Axis) and an ARIMA model for 84 smart meters in a utility service area (X Axis) when predicting 1-hour ahead KWh consumption at 15-min intervals over a 30 day period, compared to the observed KWh during that period. Two months of consumption data (Feb-Mar, 2013) were used to train the models and the predictions were made for Apr, 2013. The ARIMA function is run using the `autoArima()` function in the R statistical package. The customers are sorted based on the average prediction error, and the error bars show the first and third quartiles of error deviations.

As can be seen, ARIMA is significantly better than the averaging model, offering a mean of 32% MAPE across customers compared to 51% error for the time of day model. However, the 32% error is still higher than what is acceptable for utility decisions, which is on the order of 10% or lower. This higher error is due to the fine time granularity of predictions at every 15-mins (i.e., a 1-hour forecast of energy consumption predicts four KWh values at 15-min intervals), as also the size of the customers. For a residential customer, even switching on an incandescent bulb or a washing machine can cause a jump in the energy use within a 15-min interval relative to their overall consumption. Such rapid changes are hard to detect for time series models.

Since the time granularity of 15-mins is required by the utility for decision making, the "granularity" of customers can be relaxed in order to improve the accuracy of ARIMA, following the law of large numbers [19]. Furthermore, the time



```
S1' = S1+C1
E1 = ARIMA(S1').Error
S2' = S2+C1
E2 = ARIMA(S2').Error
...
Sx = Min(E1,E2,E3).Bin
Sx = Sx+C1
```

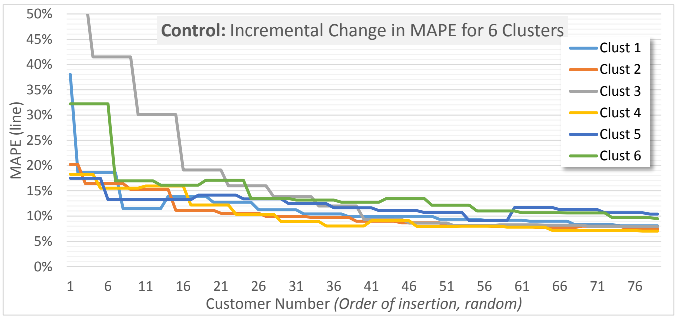
**Figure 2.** Incremental Clustering Approach for Energy Consumption Time Series Data. The *Min* function calculates the affinity score.

taken to train the ARIMA model for a single customer is ~2 mins. When scaling to a million customers in a utility region such as Los Angeles, this is time consuming (despite the embarrassing parallelism), given that the model has to be retrained every few weeks when fresh data is available. This is an additional motivation for clustering customers so that the retraining is done only for aggregated clusters of virtual customers rather than for individuals.

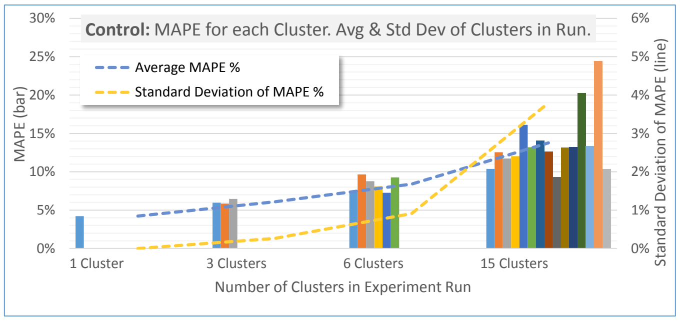
Hence, the goal for our clustering is to group customers whose cumulative energy consumption behaves in a more predictable way and minimizes the ARIMA prediction error for the cluster as a whole. *A non-goal, compared to other clustering techniques, is for customers to have "similar" consumption patterns based on distance measures. In fact, customers who have complementary consumption time series patterns may help smoothen the cumulative consumption time series for a cluster.* In this process of clustering, we lose information of the predicted consumption for individual customers, which can coarsen the granularity of, say, messaging customers who are good targets for reducing energy demand. Rather than message individual customers, the messages will have to be broadcast to all customers in the cluster. *So a secondary goal of our clustering is to reduce the number of customers per cluster*. This, paradoxically, means that for a given number of customers, having more clusters is better (without increasing the error per cluster) as it will provide greater decision making power to target fine-grained customers for energy curtailment.

This can be achieved by clustering the customers through supervised or unsupervised learning. These clustering techniques have two components: the *clustering algorithm* and a *similarity measure*. In order to computationally scale the problem, we propose to use an incremental clustering algorithm (Fig. 2). Here, each customer's time series data is scanned once and it's suitability with each cluster determined using an affinity score. The affinity score is computed as follows. For a cluster 'i', let $S_i$ be the cumulative time series vector that contains as many items as the length of the training data (e.g., 60 days x 96 intervals). The value of $S_i$ is calculated by summing the time series for each customer in that cluster.

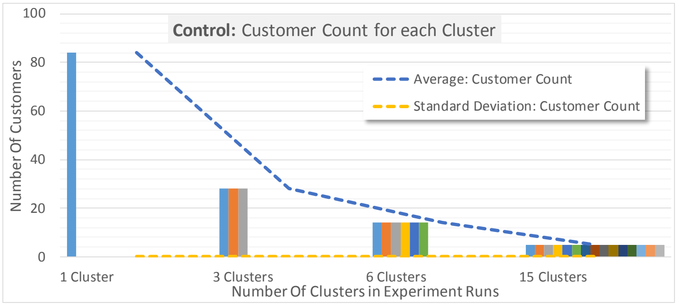
$$S_i[j] = \sum_{k=1}^{p} C_i^k[j]$$



(a) Incremental reduction in MAPE as each new customer is added across 6 clusters.



(b) MAPE for each cluster (bar). Experiment is run with different numbers of clusters: 1, 3, 6 & 15. Average and Std. Deviation of MAPE (line) for each clusters is also shown.



(c) # of customers in each cluster (bar). Experiments run with different numbers of clusters: 1, 3, 6 and 15. Average and Std. Deviation of # of customers per cluster (line) is also shown.

**Figure 3.** Incremental Clustering with *Round Robin* Assignment.

where ‘i’ is the cluster number, ‘p’ is the number of customers in the cluster, ‘j’ is the index of the vector, and $C_i^k$ is the time series vector for customer ‘k’ in cluster ‘i’.

Also, an ARIMA model is built using this cumulative time series as training data and let $E_i$ be the MAPE for predicting its future values. This error can be calculated given the future values for the constituent customers of the cluster, or by using 2/3rds of the cumulative time series data for training and 1/3rd for comparing predictions against observed. For each new customer time series $C^z$ that needs to be assigned to a cluster, we first calculate the updated cumulative time series vector for each cluster upon adding the new customer to that cluster, i.e., $S_i'[j] = S_i[j] + C^z[j]$. We also estimate the new MAPE error by training an ARIMA model using $S_i'$ and calculating its error, say, $E_i'$. The affinity score is calculated as a function of the old and new errors, $E_i$ and $E_i'$. We use the difference between the two as the affinity score, $\boldsymbol{a_i = E_i - E_i'}$, for a candidate customer with a cluster, and the customer is assigned to the cluster that offers the highest affinity score, i.e., the candidate customer is placed in the cluster that has the best reduction in its current MAPE prediction error upon adding that customer.

There are several factors that impact this broad approach, such as initializing the cluster, selecting the number of clusters, balancing the number of customers in a cluster, order of the candidate customers, and so on. These will be discussed in the next section. The time complexity of this approach, for ‘n’ customers and ‘m’ clusters is given by polynomial time of O(α.n.m), where α is the (constant) time taken to build a single ARIMA model and estimate its MAPE error.

## IV. Optimizations to Incremental Clustering

In the simplest manifestation of the above approach, we can preset the number of clusters to a static value and perform a round-robin assignment of customers to each cluster. Here, we do not consider any distance function. This offers a “control” experiment that can illustrate the value of introducing the affinity score. Figure 3(a) shows the MAPE error value for 6 clusters change as customers are added to each, in turn. In this and all other experiments, the order in which customers are considered is randomized at the beginning to avoid its influence. As expected, there is an overall drop in the MAPE error % as the number of customers in a cluster increases. This bears out the intuition that variability reduces (and hence predictability using ARIMA increases) as we accumulate more customers into a cluster’s time series to form a larger “virtual customer”. Hence, even a naïve incremental assignment of customer to clusters has its benefits. However, we also see that the drop in MAPE is not monotonic as we add customers. There are many instances where adding a customer to a cluster increases the MAPE, though the overall trend is downwards. Figure 3(b) shows the result of such experiments with 1, 3, 6 and 15 clusters. Here, we plot the MAPE per cluster as bar plots on the primary Y axis, and the average across all clusters along with their standard deviation as line plots on the primary and secondary Y axes. We notice that as the number of clusters in a run increases, the *average error across clusters also increases*. This is a consequence of having less customers per cluster as the number of clusters increase. We also see that the *standard deviation of MAPE increases* across clusters in a run. This shows that a round-robin assignment while reducing average error across clusters can lead to individual clusters with high errors. For e.g., the highest and lowest errors in the 15-cluster case range between 24% and 13%. This offers scope for improvement. Lastly, in Figure 3(c), we see one of the positive aspects of the round-robin technique: the number of customers in each cluster is (near) equal and thus the *cluster sizes are balanced*. This meets one of the secondary goals of the domain.

There may be cases where the performance of the round-robin technique is adequate, e.g., with few customers and a small number of clusters. However, the value of using the affinity score and other optimizations proposed below emerge as these bounds are relaxed. The heuristics are discussed here while their detailed empirical results presented in the next section.

### *A. Cluster Seeding*

The customer assignment based on affinity score uses the relative improvement in errors as a metric. Hence, it addresses cases where there is already one or more customers in each cluster, thereby giving it a prior prediction error to compare against. However, the initial customer – also called the *seed customer* – that is placed in each cluster can influence the clustering quality, and this process of selecting the initial customer for a cluster is called *cluster seeding*.

In the most basic form, a random customer is selected as the seed customer for each cluster. This is called *random seeding* and it is elegant in its simplicity, similar to the random order in which customers are incrementally considered for cluster assignment. There is also no overhead in this process as any customer can be chosen, and this is an *unsupervised* technique. This is also similar to the random partition used as in the initial iteration of k-means clustering, except that k-means has the opportunity to select the centroid as the seed in subsequent iterations. Our incremental clustering does just a single pass.

Another hypothesis is to order the list of customers based on some *measure*, evenly partition this ordered space into the total number of clusters, and pick the medial customer within each partition as the initial seed for each cluster. We call this *supervised* approach as *midpoint seeding*. The measure used for ordering is domain specific and can help offer a more balanced number of customers in each cluster while also reducing their MAPE error %. In our case, we propose three measures: the average of the KWh in the time series for each customer, the standard deviation of the KWh time series for each customer, and the MAPE error % for each customer's individual ARIMA prediction. The former uses the intuition that customers with similar load sizes (i.e. small, medium and large uses of energy) may better complement the magnitude of variation in energy consumption. The latter offers customers with different variability (hence error %) across the clusters for future customers to compare against

### *B. Elasiticy of Number of Clusters*

The number of clusters that are created can have an impact on the MAPE, as also on the domain needs. By default, one can use a static number of clusters that is specified by the user, similar to the value of "k" in k-means clustering. This can be picked either because of the need for a specific number of cluster in the domain or the need to control the number of customers in a cluster (assuming a balanced distribution of customer counts, though not guaranteed). We have considered 1 (trivial), 3, 6 and 15 as the static number of clusters in our experiments.

However, when there is no strong need to pin the number of clusters to a static value, it may be possible to grow the number of clusters elastically, based on the need to optimize some metric. We propose a heuristic for elastic number of clusters that has a lower and an upper bound on the number of clusters, $m_{lo}$ and $m_{hi}$. The number of clusters is incremented by one in case an incoming customer does not improve the MAPE for any existing cluster by more than a threshold value, *t*. This offers flexibility to the domain application, both in the range of cluster counts as well as the rate at which the errors improve as customers are added. It also allows the algorithm to grow the number clusters based on need rather than be bound to an initial set of static clusters with a static seed customer in each. This can also help to capture customers who are "outliers" in a separate cluster rather than force them into an existing cluster, and thereby make its error worse. For our experiments, we use three pairs of lower-upper cluster bounds, 1-15, 3-6, and 6-15, and use two different thresholds, *t=0%* and *t=1%*. The former forces the errors for existing clusters to monotonically decrease as candidate customers are added, adding a new cluster otherwise. The latter expects an even faster drop in error, by 1% or more, for every new customer added.

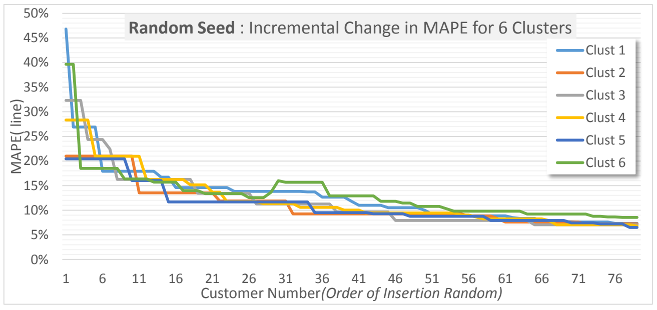
## V. Empirical Results

For the experimental evaluation of the incremental clustering technique and its optimizations we use a sample of 84 smart meters from a utility service area as a representative population. The smart meter's kWh consumption data is recorded for 3 month period at 15-min interval or ~3x30x96 = 8,600 intervals, for a total of ~725,000 data points across all smart meters on which the results were validated.

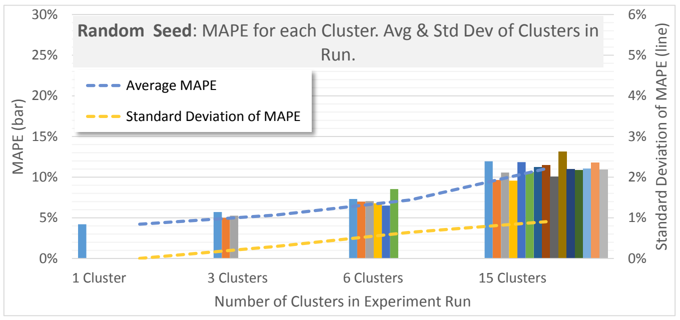
We used 2 months of data per smart meter as a training for the ARIMA and the last one month for calculating the MAPE of the ARIMA model. Besides the training, other optimizations such as midpoint seed also used the 2-month time series data for their heuristics. All experiments were repeated multiple times and the representative results presented. The order of the smart meters was randomized before each experimental run. All experiments were scripted using the R statistical package and run on a server with 16-core Opeteron processors and 64GB RAM. Each experimental run took between 5-40 hours to complete using a single threaded process, depending on the number of clusters. We observed that the total clustering time was a linear function of the number of clusters, thereby bearing out our time complexity estimate of O(α.n.m), for n=84 smart meters, a constant α as the time taken to build a single ARIMA model, and the number of clusters 'm' being the variable.

### *A. Impact of Cluster Seeding*

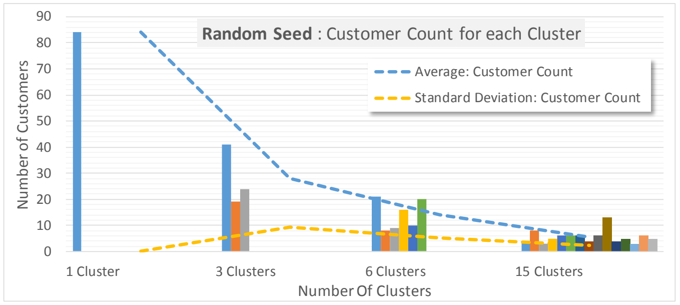
In our initial experiment, we use the random seeding to assign a different random initial smart meter (i.e., customer) as seed to each of the 4 static cluster sizes we consider: 1, 3, 6 and 15 clusters. Note that the 1-cluster case is a trivial scenario where all customers are clustered into one, and this gives a nominal "best case" scenario for MAPE from clustering and acts a lower bound for our clustering experiment. On the other hand, we use 15 as the upper bound of the number of clusters, with each cluster having on an average 4 customers – as it can be observed from figure 3(a), the cluster's MAPE drop is significant till 4 customers after which it starts flattening out. Clusters of 3 and 6 give us reasonable midpoints for comparison between the upper and low bounds. We use the standard affinity score for incremental clustering of each customer that looks for the best error reduction in each cluster. Figure 4 shows the results of this experiment.



(a) Incremental reduction in MAPE as each new customer is added across 6 clusters



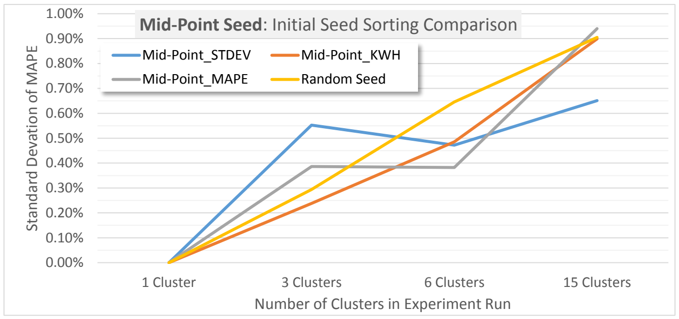
(b) MAPE for each cluster (bar). Experiment is run with different numbers of clusters: 1, 3, 6 & 15. Average and Std. Deviation of MAPE (line) for each clusters is also shown.



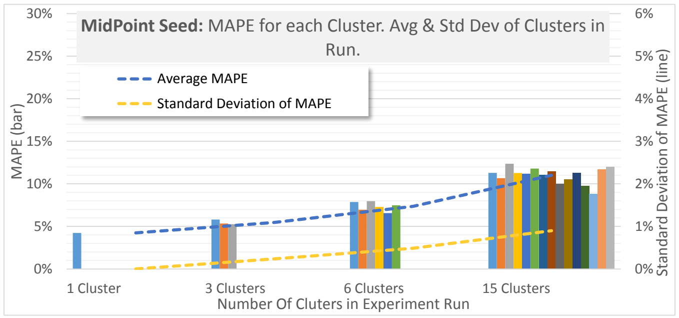
(c) # of customers in each cluster (bar). Experiments run with different numbers of clusters: 1, 3, 6 and 15. Average and Std. Deviation of # of customers per cluster (line) is also shown.

**Figure 4.** Incremental Clustering with *Random Seed* and *Affinity Score* based Customer Assignment.

It is worthwhile comparing these plots against the equivalent plots from Figure 3, which used a round-robin assignment. In Figure 4(a), we see just one instance where adding a customer actually increased the MAPE of a cluster using the affinity score, as opposed to 6 instances when this happened in the round-robin assignment in Figure 3(a). Furthermore, we also see in Figures 3(b) and 4(b) (blue dotted line) that there is a tangible drop in the average MAPE for each cluster, ranging from 12-20% improvement in *relative* MAPE. This is a 1-2% improvement in absolute MAPE, e.g., 11% vs. 13.8% for the 15-cluster case for affinity-score vs. round-robin. The improvement is even more significant when considering the standard deviation of MAPE between clusters in an experiment (yellow dotted line). Here, the relative reduction in standard deviation is as high as 76% for the affinity score assignment compared to



(a) Standard deviation of MAPE is shown for different midpoint seed ordering methods (dotted/dashed), and compared with random seed (solid line). Experiments are run for different cluster sizes. KWh seed has uniformly better standard deviation.



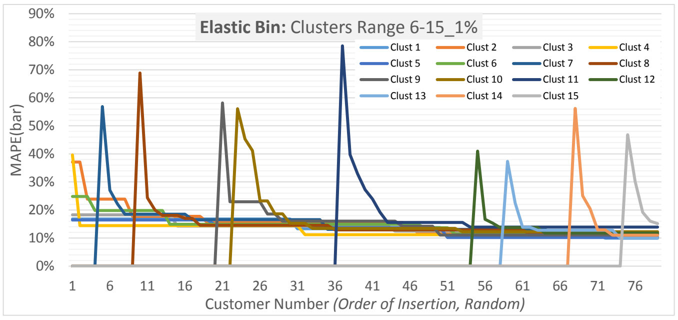
(b) MAPE for each cluster (bar) when using *customer's KWh* as the ordering for midpoint selection. Results for different numbers of clusters, and their Average and Std. Deviation of MAPE (line) are shown.

**Figure 5.** Incremental Clustering with *Midpoint Seed* and *Affinity Score* based Customer Assignment.

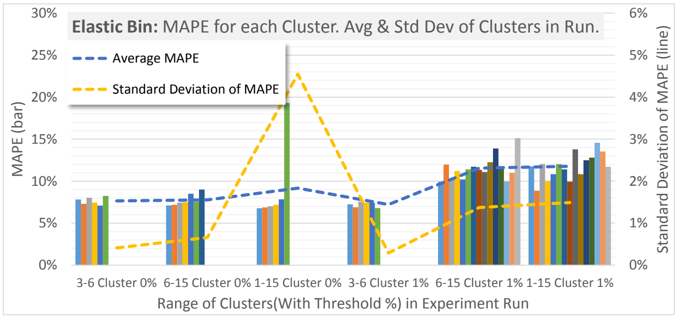
round-robin. This shows the value of our proposed similarity metric in not just reducing the average MAPE for clusters but achieving an even error % across clusters. The obvious advantage of the round-robin technique is of course in its ability to balance the number of customers in a cluster (Figures 3(c) and 4(c)), whereas the affinity score method has a more imbalanced 19, 24 and 41 customers in the 3-cluster case.

We next compare the random seed method against the *midpoint seed optimization*. Three different customer ordering approaches were used for determining the midpoint for the partitions, and hence the initial seed for each cluster. The customers were ordered based on their average KWh for the 2 month time series data, the standard deviation of this 2 month time series, and the MAPE when predicting their consumption using an ARIMA model built for each customer. We observed that there was not tangible difference in the MAPE for the midpoint seed approach compared to the random seed approach for the different static cluster sizes. However, there was a noticeable different in the absolute standard deviation of their MAPE, depending on the ordering chosen for the midpoint seed compared to the random seed.

Figure 5(a) shows the absolute standard deviations in the MAPE for the different seed approaches for the different static cluster sizes of 1, 3, 6 and 15. We see that midpoint seed using KWh ordering offers a uniform lower standard deviation



(a) Incremental reduction in MAPE as each new customer is from among 6 clusters, initially, and among 15 cluster finally. Spikes indicate a new cluster is created.



(b) MAPE for each cluster (bar) for elastic numbers of clusters. Results for different numbers of cluster upper and lower bounds, along with thresholds, as well as their Average and Std. Deviation of MAPE (line) are shown.

**Figure 6.** Incremental Clustering with *elastic number of clusters. Random Seed* and *Affinity Score* based Customer Assignment.

compared to the random seed. While the axis range may seem small (1%), this translates to a 10% relative range, given that this deviation is on MAPE that are ~10%. The other midpoint seeds do not show a consistent improvement. It is understandable that the midpoint seed helps even out the MAPE errors across the different clusters (though not reducing their errors). By ordering customers based on their KWh, each cluster is seeded with a customer whose average KWh size falls in a different partition. It is likely that subsequent candidate customers were placed in clusters whose seed was closer to that customer's average KWh since they would be in a better position to balance out the magnitude of variation in KWh. However, the impact of this is unlikely to carry over beyond the initial number of candidate customers since the cumulative KWh value of the clusters will start to dominate beyond a certain number of customers. The MAPE using the midpoint seed (KWh ordering) is shown in Figure 5(b), and is similar to Figure 4(b), but for the reduced standard deviation.

### B. *Impact of Cluster Size Elasticity*

In this experiment we set the number of clusters to fall between an upper and lower bound, and the number of clusters grow elastically when an incoming customer is unable to improve the MAPE error % of any existing cluster by a certain threshold. We evaluate lower and upper bounds of 3-6, 6-15 and 1-15 clusters, with thresholds of 0% and 1%. Figure 6(a) shows this incremental clustering strategy in action for elastic clusters with lower bound of 6 clusters and upper bound of 15 clusters, and a 1% threshold. We see that there are initially 6 clusters, shown by 6 initial MAPE values, and occasionally, an additional MAPE line springs up, denoting the creation of a new cluster. Since we have a threshold of 1% here, each addition of a customer has to either reduce the MAPE for a cluster by 1% or cause a new cluster to be created.

Figure 6(b) shows the MAPE as bar plots per cluster for each of these elastic combinations, as also the average and standard deviations of MAPE error % as line plots for each combination. The merits of the elastic cluster strategy is in automatically trading-off the number of clusters against the average MAPE error % for the clusters, without having the users pick an unsuitable static number of clusters that can cause either a high MAPE error % (if more clusters are statically chosen) or fewer clusters without any improvement in MAPE. Note that in our domain, more clusters are better. In some cases, such as 3-6 clusters, we do not see any clear improvement of this method. For e.g., the static 3-cluster, 3-6 elastic cluster (0% and 1%), and static 6-cluster have average MAPE error % of 5.33%, 7.65%, 7.22%, and 7.20% respectively. So picking an elastic number of cluster here is as bad (or marginally worse) than picking 6 clusters statically. However, when we consider the MAPE error % for static 6-cluster, 6-15 elastic cluster (0% and 1%), and static 15-cluster, with values of 7.20%, 7.76%, 11.54%, and 11.04%, respectively, we see that 6-15 elastic (0%) actually performed nearly as well as the static 6-cluster case while offering more number of clusters (7). If the user had instead stuck to just static 6 or 15, this trade-off in improving the number of clusters with minimal gain in error may have been missed. One of the side effects of this elasticity is that there is a higher standard deviation in the number of customers per cluster. This is understandable. Since clusters can be created very late, there is a greater skew in the number of customers present in clusters that were added early vs. those that were added late.

## VI. Conclusion

In this paper, we introduced a novel similarity measure for performing incremental clustering of time series datasets, with the goal of using the *cumulative* values of the clusters for performing time series predictions. This is a novel application of clustering as a form of coarsening different time series entities to perform more accurate predictions, and for which existing distance-based clustering techniques do not suffice. This is of unique value for predictive analytics over high velocity Big Data that often have a time series structure. We have validated our proposed clustering and optimization techniques for the Smart Grid domain using real datasets, and in addition to the qualitative improvements in the prediction errors and cluster balancing, we observe a poly time complexity with the number of time series datasets and clusters.

## Acknowledgment

This material is based upon work supported by the United States Department of Energy under Award Number DEOE0000192, and the Los Angeles Department of Water and Power (LA DWP). The views and opinions of authors expressed herein do not necessarily state or reflect those of the United

States Government or any agency thereof, the LA DWP, nor any of their employees.

## REFERENCES

[1] Mitra, U; Emken, B. A.; Sangwon, L.; Ming, L.; Rozgic, V.; Thatte, G.; Vathsangam, H.;Zois, D.; Annavaram, M.; Narayanan, S.; Levorato, M.; Spruijt-Metz, D.; Sukhatme, G.KNOWME: "a case study in wireless body area sensor network design". *IEEE Communications Magazine* . 50(5): 116-25. 2012.

[2] L. Douglas, "3d data management: Controlling data volume, velocity, and variety," Gartner, Tech. Rep., 2001.

[3] X.Yu, C.Cecati,, T.Dillon, M.G. Simoes,. "The new frontier of smart grids" *Industrial Electronics Magazine*, IEEE, 5(3), 49-63. 2011.

[4] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. West-over. "Exact discovery of time series motifs." in 9th SIAM InternationalConference on Data Mining SDM, 2009.

[5] K. Wagstaff. "Machine learning that matters" in International Conference on Machine Learning (ICML), 2012

[6] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. "Putting the 'smarts' into the smart grid: A grand challenge for artificial intelligence." Comm. of the ACM, 55(4):86–97, 2012

[7] J. L. Mathieu, D. S. Callaway, and S. Kiliccote. "Variability in automated responses of commercial buildings and industrial facilities to dynamic electricity prices." Energy and Buildings 43.12 3322-3330, 2011.

[8] R.P. Singh , P. X. Gao, and D. J. Lizotte. "On Hourly Home Peak Load Prediction." in Smart Grid Communications (SmartGridComm), IEEE, 2012.

[9] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou and V. Prasanna, "Cloud-based Software Platform for Data-Driven Smart Grid Management", Computing in Science and Engineering , July/August , pp. 1-11 , IEEE and AIP, 2013.

[10] S.S.Singh, N. C. Chauhan. "K-means v/s K-medoids: A Comparative Study" in National Conference on Recent Trends in Engineering & Technology, 13-14 May 2011.

[11] Y. Liu, Q. Guo, L. Yang, and Y. Li. "Research on incremental clustering." In Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on, pp. 2803-2806. IEEE, 2012

[12] Babu, C. N., and B. E. Reddy. "Predictive data mining on Average Global Temperature using variants of ARIMA models." Advances in Engineering, Science and Management (ICAESM), 2012

[13] P.Chujai, N.Kerdprasop and K. Kerdprasop. "Time Series Analysis of Household Electric Consumption with ARIMA and ARMA Models." In Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1. 2013.

[14] S. Aman, Y. Simmhan, and V. K. Prasanna, "Holistic Measures for Evaluating Prediction Models in Smart Grids" (under review, 2013)

[15] S. Aman, Y. Simmhan, and V. K. Prasanna "Improving Energy Use Forecast for Campus Micro-grids using Indirect Indicators Saima Aman, Yogesh Simmhan and Viktor K. Prasanna" IEEE ICDM International Workshop on Domain Driven Data Mining (DDDM), Vancouver, Canada, 2011

[16] T. Warren Liao, "Clustering of time series data—a survey." Pattern Recognition38.11 1857-1874, 2005.

[17] J. Lin,, M. Vlachos, E. Keogh, and D.Gunopulos. "Iterative incremental clustering of time series." In Advances in Database Technology-EDBT 2004, pp. 106-122. Springer Berlin Heidelberg, 2004.

[18] Rakthanmanon, Q. Z. G. B. Thanawin, and E. Keogh. "A Novel Approximation to Dynamic Time Warping allows Anytime Clustering of Massive Time Series Datasets." *http://siam.omnibooksonline.com/2012datamining/data/papers/088.pdf*: Date Visited: 9/10/2013

[19] Law of Large Numbers, *http://www.encyclopediaofmath.org/index.php/Law_of_large_numbers:* Date Visited: 9/10/2013

[20] X. Chen, Y. Liu, H. Liu, J.G. Carbonell. "Learning Spatial-Temporal Varying Graphs with Applications to Climate Data Analysis". *AAAI*, 2010.

[21] Schwert, G. William. "Why does stock market volatility change over time?." The Journal of Finance 44.5 (1989).

[22] A.K.Jain, M.N.Murty, P.J. Flynn "Data Clustering: A Review,", ACM Computing Surveys, 31(3), 1999.

[23] M.Verleysen, D. François. "The curse of dimensionality in data mining and time series prediction." Computational Intelligence and Bioinspired Systems. 758-770. 2005.

[24] S. Guha, A. Meyerson, N. Mishra, R. Motwani, R., & O'Callaghan, L. "Clustering data streams: Theory and practice". Knowledge and Data Engineering, IEEE Transactions on, 15(3), 515-528, 2003

[25] M.M.Gaber, A.Zaslavsky, and S. Krishnaswamy. "Mining data streams: a review." ACM Sigmod Record 34.2 18-26, 2005

[26] T. Zhang, R. Ramakrishnan, and M. Livny. "BIRCH: an efficient data clustering method for very large databases." ACM SIGMOD Record. Vol. 25. No. 2. ACM, 1996.