Challenge
Online Time Series Clustering For Demand Response
A Theory to Break the 'Curse of Dimensionality'

Ranjan Pal, Charalampos Chelmis, Saima Aman, Marc Frincu, Viktor Prasanna
Viterbi School of Engineering
University of Southern California
{rpal, chelmis, saman, frincu, prasanna}@usc.edu

ABSTRACT
The advent of smart meters and advanced communication infrastructures catalyzes numerous smart grid applications such as dynamic demand response, and paves the way to solve challenging research problems in sustainable energy consumption. The space of solution possibilities are restricted primarily by the huge amount of generated data requiring considerable computational resources and efficient algorithms. To overcome this Big Data challenge, data clustering techniques have been proposed. Current approaches however do not scale in the face of the "increasing dimensionality" problem where a cluster point is represented by the entire customer consumption time series. To overcome this aspect we first *rethink* the way cluster points are created and designed, and then design an efficient online clustering technique for demand response (DR) in order to analyze high volume, high dimensional energy consumption time series data at scale, and on the fly. Our online algorithm is randomized in nature, and provides optimal performance guarantees in a computationally efficient manner. Unlike prior work we (i) study the consumption properties of the whole population simultaneously rather than developing individual models for each customer separately, claiming it to be a 'killer' approach that breaks the "curse of dimensionality" in online time series clustering, and (ii) provide tight performance guarantees in theory to validate our approach. Our insights are driven by the field of sociology, where collective behavior often emerges as the result of individual patterns and lifestyles.

Keywords
time series, clustering, demand response, online algorithm

1. INTRODUCTION
With the increased penetration of smart meters and advanced wireless network infrastructures, smart grids are becoming ubiquitous. Numerous practical smart grid applications, including Demand Response (DR) [15], are catalyzed by Advanced Metering Infrastructure. As a result, utilities need to dynamically adjust their DR strategy; this leads to Dynamic DR (D^2R) programs according to which the target, duration, and depth of curtailment is a dynamically changing function of customers' responsiveness, particularly in incentive-based DR programs. However, the capability of the cyber-infrastructure to support such applications efficiently is primarily limited by the Big Data deluge (i.e., high volume, velocity, variety, and veracity) coming from sensors. The large amount of high speed data coming from smart meters and other smart appliances such as thermostats, luminosity sensors, etc., pose significant challenges to real-time data processing and decision making by impacting the efficiency and speed at which information can be extracted from the data stream.

Traditional solutions for predicting energy consumption and curtailment analyze either individual or groups of customers by applying prediction techniques on the historical energy consumption time series. Individual customer predictions are challenging because different prediction methods give different results in terms of accuracy for distinct consumption trends [7]. Predicting consumption for each customer can become a computational bottleneck especially for large smart grids with millions of customers. i.e., the response time for consumer data processing is on the order of a few hours [7]. Clearly, such performance guarantees are inefficient for D^2R applications. In this regard, clustering techniques have the advantage of making predictions easier by (a) reducing the noise in the aggregate customer energy consumption time series [17], and (b) reducing the customer prediction time complexity by not being required to running prediction algorithms on individual customers. A good customer clustering can provide utilities with an optimal set of customers to target during D^2R.

While customer clustering has its benefits it does not come without a trade-off. Traditionally, each point in the cluster is represented by the time series of a single customer. In an online scenario such as D^2R, where the time series grows linearly, clustering techniques are faced with the problem of increasing dimensionality. Increasing dimensionality is a major issue to the space-time performance of existing time series clustering algorithms; such algorithms are designed to perform well for fixed-dimensional data sets [13], and give bad performance for online cases, where clustering needs to be recomputed on the fly as the number of dimensions increase. While one solution to the problem is to keep dimensionality constant by disregarding stale data, this approach may not always work because most consumption time series data show some periodicity, and that might get lost by dropping older data points.

In this paper we challenge the use of traditional approaches towards clustering consumption time series data for data *continually* coming from a massive number of smart sensors. We instead propose a novel clustering approach where we fix the dimensions to the number of customers. In our approach, each point denotes the energy consumption values for all customers at a given point in time. This enables the formation of clusters of points in time which encompass the consumption values of all smart meters simultaneously. The advantage of our proposed approach is two-fold. First, clustering can be performed incrementally on "data-in-motion", i.e., new data points can be incrementally integrated as they arrive in a stream allowing the clustering configuration to be computed once and updated incrementally with the arrival of new observations. In con-

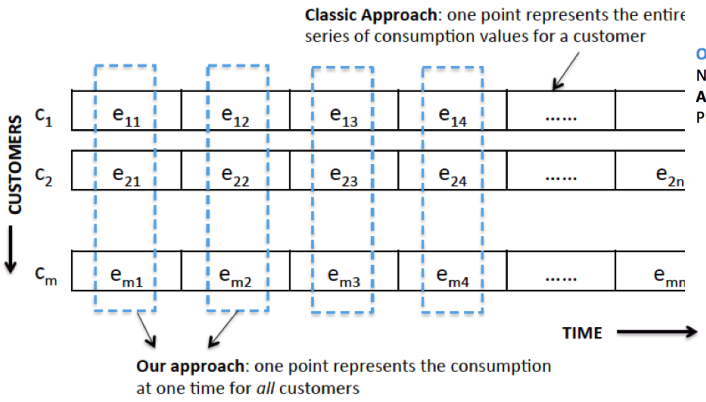

Figure 1: Classic vs. our proposed point representation.

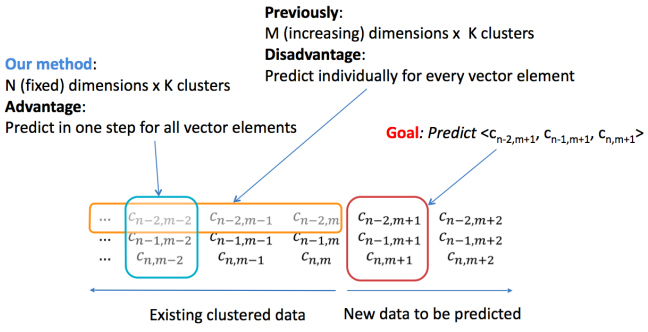

Figure 2: Advantage of our approach w.r.t clustering goal

trast, traditional approaches require clustering points (time series) whose dimensionality increases with time; thus with increasing dimensions, clustering has to be recomputed from scratch. Second, by co-integrating customers' consumption values in a vector it is possible for emerging patterns to be mined.

Figure 1 depicts the two approaches. The consumption value for customer i at time j is denoted by c_{ij}. Our representation can be naturally obtained by *transposing* the original collection of time series. Instead of new columns been added with the arrival of new data points, the matrix grows in the rows dimension in our representation. The clustering goal and the advantages of our approach in this context is depicted in Figure 2

We make the following research contributions in this paper.

- We propose an intuitive idea to the 'dynamic customer time series segmentation' problem that is driven by the field of sociology, where collective behavior emerges as a result of individual patterns (see Section 2.).
- We model this problem as an online clustering of time series data, which in turn is intractable, i.e., computationally expensive (see Section 2.3.3.). To address this issue, we propose an online approximation algorithm with provable performance guarantees to dynamically cluster energy consumption data points on the fly (see Section 3.1.). Based on our online approximation algorithm, we design an online randomized algorithm with better provable performance guarantees to online clustering compared to the pure approximation version (Section 3.2.).

The rest of the paper is organized as follows: in Section 2, we describe the challenges and drawbacks of traditional clustering techniques for consumption time series segmentation, and state our model preliminaries that include problem formulation. We design and analyze our online clustering algorithm in Section 3. We conclude our paper in Section 4.

2. PROBLEM SETTING AND BACKGROUND

In this section, we first state the importance of mining emerging behavior from individual consumer dynamics. We then describe our problem setting, which is followed by a description of the model preliminaries.

2.1 Mining Emerging Behavior

Intuitively, energy consumption is expected to be periodic, as it is governed by human activities that usually follow some schedule (e.g., daily or weekly). For example, in workplace and even residential settings, it is very likely that people are at the same place on Monday mornings, and therefore it is also likely that an emerging behavior can be recorded. In this case the energy consumption of a building will be similar on Monday mornings even if occupied by multiple tenants with different schedules or hosts hundreds of office spaces. As an example, we present daily consumption observations over a course of a year for four buildings of different types in Figure 3. From Figure 3, it can is seen that, despite the differentiation between consumption patterns among individual buildings, consumption is relatively stable for each of the buildings individually at a specific point in time over the course of a day throughout the year. Some variation is to be expected depending on the function of buildings (e.g., the second building from the left demonstrates a significant drop in consumption during summer). Similarly, usage is likely to differ by few half hours earlier or later due to natural irregularities in behavior (e.g., someone returned home at 6:30 p.m. instead of 6 p.m.). In our study, we are focusing on 15-minute intervals which even though can provide fine details on consumption, can be affected by small shifts in behavior (e.g., a tenant who overslept or worked at home on a Monday) can significantly impact the expected periodicity. Our premise is that such patterns can be detected and utilized efficiently both for consumer behavior analysis and load prediction. Typically utilities develop personalized models for each customer or rely on customer segmentation techniques, where individual models are made for each customer segment, to reduce their modeling and prediction uncertainty. *Our hypothesis is that using our representation of time series can lead to significant insights about customers' emerging behavior, and more importantly, to efficient very-short-term and medium-term prediction algorithms for electricity consumption forecasting.*

So, how does a utility go about uncovering such patterns for hundreds of thousands or millions of customers? In this work, we are venturing to address this question by appropriately arranging fine-grained streaming energy data and examining it holistically. Our approach is based on social theory, according to which individual human behavior often results in emerging collective behavior. Our assumption is that collective patterns should emerge as a result of individual patterns (as shown in Figure 3).

2.2 Problem Setting

We consider a fixed large number of customers in a metropolitan area, whose time series data of energy consumption for a given pe-

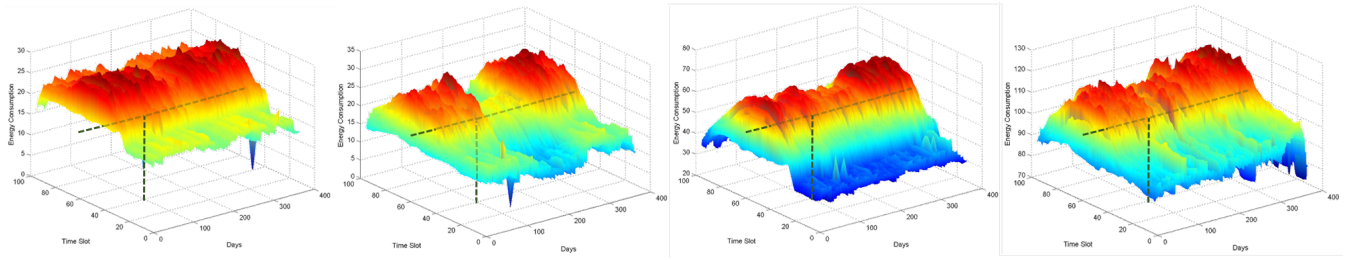

Figure 3: Smart meter data for four buildings of diverse functions, measured over a period of one year.

riod of each day (e.g., starting at 12 AM) and sampled every 15 minutes or less is known to the local utility. The utility wants to make energy consumption predictions with the goal of achieving consumption reduction during DR. As such, as a first step, the utility adopts a time series clustering technique to group customers together based on consumption trends. This not only lowers the prediction error per cluster but also allows utilities to treat each cluster independently by customizing the DR program. Traditional clustering techniques require re-running the clustering every time a new consumption data point is available. This is however time consuming and unfeasible in a D^2R scenario. As a result a clustering technique that can update itself with the advent of new data points for a given day without *re-evaluating* the segmentation from scratch is preferred. The objective is to *dynamically* update the customer clusters efficiently with respect to space (memory) and time complexity. In this paper we propose an online time-series clustering approach with provable performance guarantees. Here, the term "performance guarantee" refers to the quality of clustering with respect to the optimal clustering possible on the data points available currently.

2.2.1 Related Work in Brief

Several recent approaches have been proposed in the literature for time series clustering related to human patterns on different activities. These are driven by different end goals, such as to summarize information conveyed in temporal data, and to find representative consumption pattern for each cluster of time series. Chua et al. [3] have performed segmentation and clustering of time series of sensor data collected in smart homes for unsupervised learning of human behaviors. Hino et al. [10] have clustered daily household electricity patterns to find representative customer patterns. Martinez-Alvarez et al. [14] have performed time series clustering using similarity of pattern sequences for prediction.

Regarding the mechanism behind time-series clustering, many different approaches have been used. These include approaches based on Euclidean distance, Manhattan distance, shapelets, and dynamic time warping (DTW), etc. For more detailed information on time series clustering mechanims, see [13] [4]. All time series clustering approaches require appropriate selection of numerous variables, for e.g., the number of clusters, appropriate window length for time series data,), are computationally expensive, etc., - but the common underlying properties characterizing all these approaches are that they are (a) *heuristics*, and not *provably optimal,* (b) static in nature, and (c) does not scale well to high dimensions (unless accompanied by dimensionality reduction techniques; one exception being the approach in [4])

In this work, we derive time-series clustering approaches that are provably optimal, dynamic, and suited for high-dimensional data.

2.2.2 Challenging a Conventional Mechanism

In this section, we first provide the rationale of why our given problem setting is subject to the 'increasing dimensionality' problem of the conventional application of time series clustering algorithms. We then provide a brief intuition of how this challenge can be resolved in an effective way so as to harness the power and simplicity of clustering mechanisms.

Challenge: Assume a time series clustering algorithm exists to dynamically update groups of users for efficient D^2R. Such an algorithm would need to deal with data that is increasing in dimensionality. Intuitively, the length of each vector $c(i,:), \forall i$ (i.e., vector of consumption values for customer i in matrix C) increases with time because new data points are added as they are recorded. This strictly increasing dimensionality is a major road block to the space-time performance of time series clustering algorithms; such algorithms are designed to perform well for fixed-dimensional data sets. Windowing techniques can be applied to keep dimensionality constant, however this approach may lead to accuracy degradation; as important consumption patterns might be overlooked in favor of more recently added data points.

Solution Insight: Our main intuition behind solving the above-mentioned challenge is to change the conventional view of looking at time series data. More precisely, given a pre-specified number of customers, instead of considering each consumption value in a time series as a dimension, we fix the number of dimensions to be the number of customers. Then for each time point, we have a vector of consumption values of customers, where the length of the vector equals the number of consumers (see Figure 1). We obtain a data point which is not an increasing time series but a vector of fixed dimension. Our goal is to efficiently cluster these data points in an online fashion as they arrive with time.

2.3 Model Preliminaries

In this section we describe the principle behind our proposed clustering mechanism, formulate our clustering problem, and comment on its complexity.

2.3.1 Clustering Principle

Our clustering mechanism is based on the principle of *Hierarchical Agglomerative Clustering* (HAC) [5] [18]. The basic idea is: *initially assign n points to n distinct clusters; repeatedly merge pairs of clusters until their number is sufficiently small.* HAC computes hierarchy trees of clusters whose leaves are individual points and internal nodes correspond to clusters formed by merging clusters at the children. The *primary advantage* of HAC-based algorithms is that (i) in a dynamic setting (such as ours, where there is the advent of new data points, and we need to update the clustering accordingly in an efficient manner), it is desirable to retain the hierarchical structure while ensuring efficient update and high-quality clustering, and (ii) experience shows that HAC performs extremely well

both in terms of efficiency and cluster quality [16] [19].

2.3.2 Problem Formulation

Assume a general arbitrary metric space M, e.g., $\mathbb{R}^{\geqslant}$. Consider a set of n_1 points in M that have *already* been clustered into k clusters so as to minimize the maximum cluster diameter. Here, each point is a vector of energy consumption values for a consumer at a particular time instant, the *diameter* of a cluster is defined to be the maximum inter-point distance in it, and the the *distance* between points in M is given by a distance function on the metric space, e.g., l_2 distance in $\mathbb{R}^m$. Now consider a set of n_2 points in M that are yet to arrive. For each point arrival, we need to design an algorithm (say A) that maintains a collection of k clusters such that either the input point is assigned to one of the current k clusters, or it starts off as a new cluster while two existing clusters are merged into one. Clearly A in *online* in nature. We define the *performance ratio* of A as the maximum over all update sequences of the ratio of its maximum cluster diameter to that of the optimal clustering for the input points. By formulating our problem in this way, we enforce the requirement that *at all times* algorithm A will maintain a HAC for the points presented up to that time. *Our main objective in this paper is to design A such that it is efficient in both computational time and space, and at the same time providing the best performance guarantee.*

We note here that it could have been the case that a newly arrived point could start off from a new cluster and we could allow the points of one old cluster to be re-distributed among the remaining clusters, rather than two clusters to be merged together. The problem with such formulations is that they do not lead to HACs.

2.3.3 Problem Intractability

The static version of our clustering problem falls into the group of problems known as *pairwise clustering* or *k-center* problems [2] [11]. Both these problem types are NP-Hard in nature [8] [12], and in fact hard to approximate to within factor 2 for arbitrary metric spaces. Even if we consider the specific case of Euclidean metric spaces, the problem types are NP-Hard for data point dimensionality greater than or equal to 2 (such as in our case), and for arbitrary distance metrics. It is evident that with the static clustering problem being hard, the online version is at least harder. Thus, in our work we will look to design efficient approximation algorithms for the online clustering problem. In this regard, we borrow techniques from [6] [9] to come up with an algorithm whose time complexity is solely a function of k.

3. ONLINE CLUSTERING ALGORITHM

In this section, we design our proposed online time series clustering algorithm. As mentioned earlier, due to to the inherent intractability of our clustering problem, we need to resort to the design of efficient approximation algorithms. In order to ensure strong performance guarantees, our final aim is to construct a randomized online algorithm for our clustering problem, the structure of which lies embedded in an approximation online algorithm we describe next.

3.1 Approximation Algorithm Design

Our approximation online algorithm is mainly based on two parameters, α and β (to be described later), and thus we will term it as the '(α, β) - online time series clustering algorithm', or simply (α, β) - OTSC. The algorithm works in phases: at the start of phase i, it has a collection of $k+1$ clusters $C_1, C_2, ..., C_{k+1}$ and a lower bound d_i on the optimal clustering's diameter (denoted as OPT). Each cluster C_i has a center c_i which is one of the points in the cluster. The following algorithm *invariants* are assumed at the start of phase i: (a) for each cluster C_j, the radius of C_j defined as $max_{p\,\epsilon\,C_j} d(c_j, p)$ is at most αd_i; (b) for each pair of clusters C_j and C_l, the inter-center distance $d(c_j, c_l) \geq d_i$; and (c) $d_i \leq$ OPT.

Algorithm 1: (α, β)-OTSC finds a time series clustering

Input: (a) *Dynamic* point set $S \subset \mathbb{R}^n$ of consumer energy consumption data at time instants. Let $n = |N|$ - number of consumers, (b) Number of desired clusters, k, (c) Given *dynamic* set T of k clusters of currently observed data points, each cluster having at least one point, (d) Given (α, β) pair such that $\frac{\alpha}{\alpha - 1} \leq \beta$, and (e) d - smallest interpoint distance in T.

Output: A k clustering configuration, T

```
Repeat forever
while |T| ≤ k do
    Get new point x; S ← S ∪ {x}
    if D(x, T) > βd then
        T ← T ∪ {x}
T' ← {}
while ∃z ϵT such that D(z, T') > βd do
    T' ← T' ∪ {z}
T ← T'
d ← βd
return T
```

Each phase consists of two stages: the first is the *merging stage* in which the algorithm reduces the number of clusters by merging certain pairs; the second is the *update stage* in which the algorithm accepts new updates and tries to maintain at most k clusters without increasing the radius of the clusters or violating the invariants. A phase ends when the number of clusters again exceeds k. We now explain in detail the merging and update stages of our algorithm.

3.1.1 Merging Stage

The merging stage works as follows: Define $d_{i+1} = \beta d_i$, and let G be the d_{i+1} - *margin* graph on the k + 1 cluster centers, $c_1, c_2, ..., c_{k+1}$. We define a d-margin graph on a set of points $P = \{p_1, p_2, ..., p_n\}$ as the graph $G = (P, E)$ such that $(p_i, p_j)\,\epsilon\,E$ if and only if $d(p_i, p_j) \leq d$. The graph G is used to merge clusters by repeatedly performing the following steps while the graph is non-empty: pick an arbitrary cluster C_i in G and merge all neighbors into it; make c_i the new cluster's center; and remove C_i and its neighbors from G. Let $C'_1, C'_2, ..., C'_m$ be the new clusters at the end of the merging stage. Note that it is possible that $m = k + 1$ when the graph G has no edges, in which case the algorithm will be forced to declare the end of phase i without going through the update stage. *We have the following lemma regarding the merging stage, the proof of which is in the Appendix.*

LEMMA 1. *The pairwise distance between the cluster centers after the merging stage of phase i is at least d_{i+1}, and the radius of the clusters after the merging stage of phase i is at most $d_{i+1} + \alpha d_i \leq \alpha d_{i+1}$.*

3.1.2 Update Stage

The update stage continues while the number of clusters is at most k. When a new data point arrives, the algorithm attempts to place

it in one of the current clusters without exceeding the radius bound αd_{i+1}: otherwise a new cluster is formed with the update as the cluster center. When the number of clusters reaches k + 1, phase i ends and the current set of k +1 clusters along with d_{i+1} are used for the $(i+1)th$ phase. We have the following lemma on the invariant preservation after every phase of our deterministic clustering algorithm. *The proof of the lemma is in the Appendix.*

LEMMA 2. *The k + 1 clusters at the end of the i^{th} phase satisfy the following conditions: (i) the radius of the clusters is at most αd_{i+1}, (ii) the pairwise distance between cluster centers is at least d_{i+1}, and (iii) $d_{i+1} \leq OPT$, where OPT is the diameter of the optimal clustering for the current set of points.*

Algorithm 1 provides our algorithmic steps. We have the following theorem regarding the computational complexity of (α, β)-OTSC, *the proof of which is in the Appendix.*

THEOREM 1. *Algorithm (α, β)-OTSC has an optimal performance ratio of 8 in any metric space when both $\alpha = \beta$ equals 2, and can be implemented to run in $O(k \log k)$ amortized time per update.*

As mentioned above, the performance ratio of (α, β)-OTSC is 8, but we can do significantly better if we use this algorithm as the backbone to design a randomized algorithm, as shown next.

3.2 Randomized Algorithm Design

The randomized algorithm remains essentially the same as the deterministic one, the main change being the value of d_1, which is the lower bound for phase 1. In the deterministic case we choose d_1 to be the minimum pairwise distance of the first $k + 1$ points, say x. We will now choose a random value r from the closed interval $[\frac{1}{e}, 1]$ according to the probability density function $\frac{1}{r}$. We will also set d_1 to rx, redefine $\beta = e$, and force α to be equal to $\frac{e}{e-1}$. We now state our randomized algorithm that we name as (α, β) - ROTSC.

The following theorem regarding the computational complexity of (α, β)-ROTSC, *the proof of which is in the Appendix.*

THEOREM 2. *Algorithm (α, β)-ROTSC has an optimal performance ratio of 2e, i.e., approximately a factor of 5.43, in any metric space with $(\alpha, \beta) = (\frac{e}{e-1}, e)$, and can be implemented to run in $O(k \log k)$ amortized time per update.*

3.2.1 Initial Results.

To test the performance our proposed randomized algorithm, we ran a simple test on a synthetic dynamic time-series data set generated using techniques in [1]. We compared the clustering efficiency result with the results obtained by *re-running* existing static clustering techniques (k-means with l_2 metric, and DTW) with every new data point arrival. Compared to k-means and DTW, our algorithm performed at least 20 times faster, and at least 15% better in terms of clustering performance as measured by *normalized mutual information*, for a given value of k. We plan to run more tests as part of future work.

Algorithm 2: (α, β)-ROTSC finds a time series clustering

Input: (a) *Dynamic* point set $S \subset \mathbb{R}^n$ of consumer energy consumption data at time instants. Let $n = |N|$ - number of consumers, (b) Number of desired clusters, k, (c) Given *dynamic* set T of k clusters of currently observed data points, each cluster having at least one point, (d) Given $(\alpha = \frac{e}{e-1}, \beta = e)$ pair such that $\frac{\alpha}{\alpha-1} \leq \beta$, and (e) d - $r\times$ smallest interpoint distance in T, where r is a random value chosen from $[\frac{1}{e}, 1]$ with probability density function $\frac{1}{r}$.

Output: A k clustering configuration, T

```
Repeat forever
while |T| ≤ k do
    Get new point x; S ← S ∪ {x}
    if D(x, T) > βd then
        T ← T ∪ {x}
T' ← {}
while ∃z ∈T such that D(z, T') > βd do
    T' ← T' ∪ {z}
T ← T'
d ← βd
return T
```

4. CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of dynamically clustering consumer energy consumption data for demand response purposes, as it arrives over time in a stream. Given that time series data is high dimensional, it is a big challenge in the Smart Grid to dynamically cluster consumer energy usage on the fly, with the continuous increase in streaming data dimensions. We resolve this challenge with an idea stemming from the field of sociology: *collective behavior often emerges as the result of individual patterns and lifestyles.* This idea motivated us to look at the time series clustering problem in an inverted manner, where each data point is a fixed dimensional vector of the energy consumption of all the consumers in the system, at a particular time instant. As a result the clustering problem is reduced to an online task of dynamically clustering points of fixed dimensions. In this regard, we designed an online algorithm that is randomized in nature, and provides optimal performance guarantees in a computationally efficient manner. Unlike prior work, (i) we studied the consumption properties of the whole population simultaneously rather than developing individual models for each customer separately, claiming it to be a 'killer' approach that breaks the 'curse of dimensionality' in online time series clustering, and (ii) we provided tight performance guarantees for the quality of our approach.

As part of future work, we plan to use our insights for developing a single predictive model for all customers in unison instead of learning individual models for each customer in isolation. Specifically, we plan to leverage our clustering methodology to perform both short-term and long-term prediction for the entire population of consumers using real-world smart grid data.

5. REFERENCES

[1] R. J. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *Hellenic Conference on Informatics*, 1999.

[2] M. Bern and D. Eppstein. *Approximation Algorithms for Geometric Problems*. PWS Publishing Company, 1996.

[3] S-L. Chua, S. Marsland, and H. Guesgen. Unsupervised

learning of human behaviors. In *AAAI*, 2011.

[4] R. Ding, Q. Wang, Y. Dang, Q. Fu, H. Zhang, and D. Zhang. Yading: Fast clustering of large-scale time series data. In *VLDB*, 2015.

[5] B. Everitt. *Cluster Analysis*. Heinemann Educational, 1974.

[6] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *STOC*, 1988.

[7] Marc Frincu, Charalampos Chelmis, Muhammad Usman Noor, and Viktor K. Prasanna. Accurate and efficient selection of the best consumption prediction method in smart grids. In *Proc. IEEE International Conference on Big Data*, page in print. IEEE, 2014.

[8] M. R. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[9] T. E. Gonzalez. Clustering to minimize the maximum inter-cluster distance. *Theoretical Computer Science*, 38, 1985.

[10] H. Hino, H. Shen, N. Murata, S. Wakao, and Y. Hayashi. A versatile clustering method for electricity consumption pattern analysis in households. *IEEE Transactions on Smart Grid*, 2013.

[11] D. Hochbaum. *Various Notions of Approximations: Good, Better, Best, and More*. PWS Publishing Company, 1996.

[12] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. *SIAM Journal of Applied Mathematics*, 37, 1979.

[13] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11), 2005.

[14] F. Martinez-Alvarez, A. Troncoso, J. C. Riquelme, and J. S. Ruiz. Energy time series forecasting based on pattern similarity. *IEEE Transactions on Knowledge ad Data Engineering*, 2011.

[15] Farrokh Rahimi and Ali Ipakchi. Demand response as a market resource under the smart grid paradigm. *Smart Grid, IEEE Transactions on*, 1(1):82–88, 2010.

[16] G. Salton and M. J. Gill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Compnay, 1983.

[17] Y. Simmhan and M.U. Noor. Scalable prediction of energy consumption using incremental time series clustering. In *Big Data, 2013 IEEE International Conference on*, pages 29–36, Oct 2013.

[18] C. J. van Rijsbergen. *Information Retrieval*. Buttersworth, 1979.

[19] P. Willet. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management*, 24, 1988.

6. APPENDIX

In this section, we provide detailed proofs of the lemmas and theorems proposed in Section 3.

Proof of Lemma 1. Prior to merging, the distance between two clusters which are adjacent in the margin graph is at most d_{i+1}, and their radius is at most αd_i. Therefore, the radius of the merged cluster is at most

$$d_{i+1} + \alpha d_i \leq (1 + \frac{\alpha}{\beta}) d_{i+1} \leq \alpha d_{i+1},$$

where the last inequality follows from our assumption choice that $\frac{\alpha}{\alpha-1} \leq \beta$. Now the distance between the cluster centers after the merging stage is d_{i+1}, and a new cluster is created only if a request point is at least d_{i+1} away from all current clusters. Therefore the cluster centers have pairwise distance at least d_{i+1}. Thus, we have proved Lemma 1. ■

Proof of Lemma 2. We have $k+1$ clusters at the end of the phase since that is the terminating condition. From Lemma 1, the radius of the clusters after the merging stage is at most αd_{i+1}, and from the definition of the update stage this bound is not violated by the insertion of new points. Now the distance between the cluster centers after the merging stage is d_{i+1}, and a new cluster is created only if a request point is at least d_{i+1} away from all current clusters. Therefore the cluster centers have pairwise distance at least d_{i+1}. Since at the end of the phase we have $k+1$ cluster centers that are d_{i+1} apart, the optimal clustering is forced to put at least two of them in the same cluster. It follows that $d_{i+1} \leq$ OPT. Thus, we have proved Lemma 2. ■

Proof of Theorem 1. Based on Lemmas 1 and 2, the algorithm (α, β)- OTSC ensures the invariant that $d_i \leq$ OPT at the start of phase i. The radius of the cluster during phase i is at most αd_{i+1}. Thus, the performance ratio at any time during phase i is at most $\frac{2\alpha d_{i+1}}{OPT} \leq \frac{2\alpha\beta}{OPT} \leq 2\alpha\beta$. Now the values of α, β that minimize $2\alpha\beta$ and at the same time satisfies the condition $\frac{\alpha}{\alpha-1} \leq \beta$, are $\alpha = 2, \beta = 2$. Thus, algorithm (α, β)- OTSC has a performance ratio of 8 in any metric space, and the ratio is tight.

Regarding the computational complexity of the algorithm, we first assume that there is a black-box for computing the distance between two points in the metric space, in unit time, and this is a reasonable assumption. We maintain the edge lengths of the complete graph induced by the current cluster centers in a heap. Since there are at most k clusters, the space requirement is $O(k^2)$. When a new point arrives, we compute the distance of this point to each of the current cluster centers, which requires $O(k)$ time. If the point is added to one of the current clusters, we are done. If, on the other hand the new point initiates a new cluster, we insert into the heap edges labeled with the distances between this new center and the existing cluster centers. This step takes $O(k \log k)$ time. For accounting purposes in the amortized analysis, we associate $\log k$ credits with each inserted edge. We will show that it is possible to charge the cost of implementing the merging stage of the algorithm to the credits associated with the edges. This implies the desired time bound.

We assume without loss of generality that the merging stage merges at least two clusters. Let d be the margin used during the phase. The algorithm extracts all the edges from the heap which have length less than d. Let m be the number of edges deleted from the heap. This deletion step costs $O(m \log k)$ time. The d-margin graph on the cluster centers is exactly the graph induced by these m edges. It is evident that finding new cluster centers usign the margin graph takes tie linear in the number of edges of the graph, assuming the edges are given in the form of an adjacency list. Forming the adjacency list from the edges takes linear time. Thus, the total cost of the merging phase is bounded by $O(m \log k + m) = O(m \log k)$ time. The credit of $\log k$ is placed with each edge when it is inserted into the heap accounts for this cost. Thus, we have proved Theorem 1. ■

Proof of Theorem 2. Let σ be the sequence of updates, and let the optimal cluster diameter for σ be γx, where x is the minimum pairwise distance of the first $k+1$ points. The optimal value is at least x, so $\gamma \geq 1$. Now suppose we choose $d_1 = rx$ for some $r \in (\frac{1}{e}, 1]$. Let ρ_r be the maximum radius of the clusters created for σ with this value of r. Using arguments similar to those in the proof of Theorem 1, we can show that ρ_r is at most $d_{i+1} + \alpha d_i = \frac{e^{i+1} d_1}{e-1}$, where i is the largest integer such that

$$d_i = e^{i-1} d_1 = e^{i-1} rx \leq OPT = \gamma x.$$

Let i^* be the integer such that $e^{i^*-1} \leq \gamma < e^{i^*}$, and $\delta = \frac{\gamma}{e^{i^*}}$. Then we have

$$\rho_r \leq \frac{rex\gamma}{(e-1)\delta}; r > \delta,$$

and

$$\rho_r \leq \frac{re^2 x\gamma}{(e-1)\delta}; r \leq \delta.$$

Let X_r^- and X_r^+ be the indicator variables for the events $[r \leq \delta]$ and $[r > \delta]$ respectively. We claim that the expected value of ρ_r is bounded by

$$E[\rho_r] \leq \int_{\frac{1}{e}}^{1} \frac{re\gamma x(eX_r^- + X_r^+)}{\delta r(e-1)} dr,$$

or

$$E[\rho_r] \leq \frac{eOPT}{\delta(e-1)} \int_{\frac{1}{e}}^{1} (eX_r^- + X_r^+) dr,$$

or

$$E[\rho_r \leq \frac{eOPT\delta(e-1)}{\delta(e-1)} = eOPT.$$

Therefore, the expected diameter is at most $2eOPT$. Thus, we have proved Theorem 2. ■