2016 IEEE International Conference on Smart Grid Communications (SmartGridComm): Data Management and Grid Analytics and Dynamic Pricing

Time-Series Clustering for Data Analysis in Smart Grid

Akanksha Maurya* Alper Sinan Akyurek* Baris Aksanli** Tajana Simunic Rosing**
*Electrical and Computer Engineering Department, University of California San Diego (UCSD)
**Computer Science and Engineering Department, University of California San Diego (UCSD)

***Abstract*—The increasingly pervasive deployment of networked sensors in the Smart Grid for monitoring energy consumption has resulted in an unprecedentedly large amount of data generation. Efficient methods are required to understand this high volume and high dimensional data on an embedded platform, which has many challenges due to memory, processing and power constraints. One of the popular methods to analyze time-series data is clustering. In this paper, we discuss D-Stream II, a common time-series clustering algorithm, and demonstrate that it fails to obtain clusters in sample Smart Grid applications. Then, we propose an enhanced version of this algorithm, which handles the scenarios where the original algorithm fails. We show the effectiveness of our algorithm using real residential power consumption data from Pecan Street database. Our enhanced algorithm efficiently handles the high volume energy consumption data and captures the everyday energy consumption patterns of each residential home, which allow the consumer to compare energy consumption with their neighbors or detect any abnormality such as a defective appliance. The consumers can also be clustered into different groups, which can be effectively used to enhance the demand response policies. Our algorithm can perform clustering on approximately half a million energy consumption data points using only 5KB max memory, 360J max overhead in energy consumption and can complete in 8 mins on a resource limited embedded platform (Raspberry Pi 2).**

Keywords— time-series, clustering, smart grids, density-based clustering, embedded platform

I. Introduction And Related Work

Smart Grid leverages a large number of sensors to collect information about the surroundings, e.g. energy, voltage, current of each appliance, temperature data, etc. This generates a huge volume of data and the need for faster processing [1]. Such continuously increasing data provide tremendous opportunities in understanding the dynamics of the consumer as well as the utility to optimize the Smart Grid. Data can be generated at a high temporal resolution (e.g. every second), but frequently, relatively low-resolution data (e.g., every 15 mins) is transmitted and used, as the amount of storage space required for high-resolution data is prohibitively large. For utility companies serving millions of customers, storing this high-resolution data can sum up to petabytes [12]. Similarly, transmission of such high-resolution data can cause serious congestion issues in the transmission network. However, high-resolution data is extremely beneficial for many analytical applications such as detailed visualization [13], energy disaggregation [14], monitoring of residential power quality [15], and short-term forecasts of energy generation/load [16, 17] or energy prices [18]. Effective techniques are required for analyzing this data closer to the source (e.g., a smart meter) so that relatively low volume of data is stored and transmitted. In the Smart Grid the lower-end devices closer to where data is generated are typically embedded devices, which has constraints in memory, processing speed and power.

Time-series clustering on sensor data can be highly beneficial for Smart Grid applications, such as analyzing everyday energy demand patterns. Both users and utilities can take advantage of the results of such analysis. The former group can optimize their energy consumption by adjusting their demand based on electricity price and/or alternative energy resource availability. The latter can identify potential user groups by clustering, which can be effectively used to enhance the Demand Response policies [3] for a real-time automatic control in the Smart Grid.

Time-series clustering methods are classified into five categories [4]: Partitioning Clustering, Hierarchical Clustering, Density-based Clustering, Grid-based Clustering, and Model-based Clustering. An effective clustering algorithm for an embedded platform is one which performs clustering accurately, using limited input from the user with the memory, processing and power constraints of the embedded system. CluStream [5] is a partitioning based clustering algorithm for time-series data. It uses k-means as the base method and forms spherical shaped clusters. This algorithm fails to form clusters of arbitrary shape and also requires the user to pre-set the number of clusters. Thus, it is not appropriate for our application. CluTree [6] is a hierarchical clustering based algorithm for time-series data. This algorithm has no backtracking capability, i.e. once a cluster is merged or split, it cannot be undone. Similar to the previous one, it has very limited adaptability. In hierarchical clustering, cluster information is stored for every hierarchy level, which makes CluTree very expensive in terms of memory utilization. SWEM [7] is a model-based clustering algorithm that clusters data in a time-based sliding window with expectation maximization technique. This algorithm optimizes the fit between the data and a mathematical model. The accuracy of the algorithm depends heavily on the accuracy of the pre-selected mathematical model. The algorithms that are based on partitioning, hierarchical clustering and model-based clustering are not appropriate for embedded platforms due to their non-adaptive behavior [4].

DenStream [8] is a density-based clustering algorithm. It has the ability to form arbitrary shape clusters, detect outliers and automatically determine the number of clusters. Often

978-1-5090-4075-9/16/$31.00 ©2016 IEEE

density-based clustering is combined with grid-based clustering, which is known as density-grid-based clustering [4]. D-Stream [9] is a density-grid-based time-series clustering algorithm. This algorithm has fast processing time and bounded memory utilization since it depends on the number of grids, which is fixed, instead of the number of data points. D-Stream II [10] is an extension of the original D-Stream algorithm. It improves the clustering accuracy by considering the attraction of grids, which characterizes the spatial information of the data in each grid. Due to low memory utilization, fast processing time and clustering accuracy [4][10] as compared to other algorithms, D-Stream II is an appropriate fit for clustering time-series data on an embedded platform such as Raspberry Pi 2 (RPi2) in a Smart Grid. RPi2 provides a low-cost platform which interconnects and controls various devices/sensors and presents a computing environment that suits well for embedded Smart Grid applications.

Despite its advantages, there are three scenarios which occur frequently in real time series, where the D-Stream II algorithm fails or is inaccurate. For example, D-Stream II algorithm fails to generate any clusters for 135 real time-series datasets of one-year energy consumption of 15 residential house from Pecan Street, which results in significant information loss and leads to an incorrect interpretation of energy consumption data. In this paper, we propose a modified D-Stream II algorithm, which performs clustering in scenarios where the original algorithm fails while maintaining the performance in other scenarios. Our algorithm performs clustering on approximately half a million energy consumption data points using only 5KB max memory, 360J max energy consumption overhead and can complete in 8 minutes on RPi2 which is within the memory, processing and power constraints of RPi2.

The remainder of this paper is organized as follows: in the next section, we introduce the original D-Stream II algorithm and its issues. In section III, we introduce our clustering algorithm. In section IV, we analyze one-year energy consumption data of 15 residential houses from Pecan Street with our algorithm. Finally, in section V, we conclude the time-series clustering for analysis in Smart Grid.

II. D-STREAM II ALGORITHM

The D-Stream II algorithm has online and offline phases. In the online phase as shown in Algorithm 1, lines 5-7, each multi-dimensional input data is mapped to a corresponding discretized grid and the characteristic vector of the grid is updated. In the offline phase, which is performed after every time interval gap as shown in Algorithm 1, lines 8-14, the clusters are adjusted dynamically i.e. new clusters are created and existing clusters are disintegrated. The D-Stream II algorithm adopts a density decaying technique to capture the dynamic changes of a data stream and an attraction-based mechanism to accurately generate cluster boundaries. It clusters two neighboring grids only if they are strongly correlated. Two grids are considered as strongly correlated if their attractions in both directions are higher than a threshold value. During the offline phase, it only adjusts grids whose density attributes changed since the last time the grids were adjusted.

Each grid stores a characteristic vector which captures the evolving nature of the data stream. The characteristic vector of a grid *g* is a tuple *(tg, tm, C, D, label)* [17], where *tg* is the last time when *g* is updated, *tm* is the last time when *g* is removed from grid list as a sporadic grid, *C* is a 2D-vector denoting the attraction from *g* to its neighbors, *D* is the grid density at the last update, and label is the class label of the grid. Whenever a data point is mapped to the grid its characteristic vector is updated.

Algorithm 1: Main Procedure of D-Stream II

```
1.  tc = 0;
2.  initialize an empty red-black tree for grid list;
3.  while data stream is active do
4.      read record x = (x1, x2, . . . , xd);
5.      determine the density grid g that contains x;
6.      if (g not in grid list) insert g to grid list;
7.      update the characteristic vector of g;
8.      if tc == gap then
9.          call initial clustering(grid list);
10.     end if
11.     if tc mod gap == 0 then
12.         detect and remove sporadic grids from grid list;
13.         call adjust clustering(grid list);
14.     end if
15.     tc = tc + 1;
16. end while
```

The algorithm takes into consideration the attraction of two grids while merging the grids into a cluster. As shown in Fig 1, the data in grid 1 is located at the upper left corner and the data in grid 5 is located at the lower half of the grid and both the grids have density above the threshold value, while the density of grids 2, 3, 4 and 6 is less than dense threshold. The other algorithm will cluster grid 1 and grid 5 and consider other grids as transitional or sparse. The D-Stream II algorithm will cluster grid 1 with grid 2, 3, 4 and grid 5 with grid 6 based on the attraction of the neighboring grids which based on the spatial locality of the data looks more accurate.

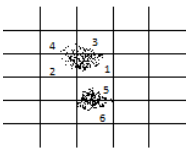


Fig 1 Attraction between different grids[17]

Algorithm 2 : Procedure for initial clustering in D-Stream II (*grid list*)

```
1.  update the density of all grids in grid list;
2.  assign each dense grid to a distinct cluster;
3.  label all other grids as NO CLASS;
4.  repeat
5.      foreach cluster c
6.          foreach outside grid g of c
7.              foreach neighboring grid h of g
8.                  if (g and h are strongly correlated) and (h
                        belongs to cluster c')
9.                      if (|c| > |c'|) label all grids in c' as in c;
10.                     else label all grids in c as in c';
11.                 else if (g and h are strongly correlated) and
                            (h is transitional) label h as in c;
12. until no change in the cluster labels can be made
```

Algorithm 2 shows the first component of offline phase i.e. initial clustering which is called when the time interval is equal to the gap time. The initial clustering procedure marks all the dense grids as separate clusters and for each cluster checks if a strongly correlated neighbor grid exists or not. If a strongly correlated neighbor is found then the neighboring grid is merged with the cluster.

Algorithm 3: Procedure for adjust clustering in D-Stream II (*grid list*)

```
1. update the density of all grids in grid list;
2. foreach grid g whose attribute
        (dense/sparse/transitional) is changed since last
         call to adjust clustering()
3.   if (g is a sparse grid)
4.      delete g from its cluster c, label g as NO CLASS;
5.      if (c becomes unconnected) split c into two clusters;
6.   else if (g is a dense grid)
7.      among all neighboring grids of g that are strongly
        correlated to g, find out the grid h whose cluster
        ch has the largest size;
8.      if (h is a dense grid)
9.         if (g is labeled as NO CLASS) label g as in ch;
10.        else if (g is in cluster c and |c| > |ch|)
11.              label all grids in ch as in c;
12.        else if (g is in cluster c and |c| ≤ |ch|)
13.              label all grids in c as in ch;
14.     else if (h is a transitional grid)
15.        if ((g is NO CLASS) and (h is an outside grid
                if g is added to ch)) label g as in ch;
16.        else if (g is in cluster c and |c| ≥ |ch|)
17.              move h from cluster ch to c;
18.  else if (g is a transitional grid)
19.     among neighboring grids of g that are dense and
        strongly correlated to g, find the grid whose
        cluster c' has the largest size;
20.     label g as in c';
21. end for
```

The second component of offline phase i.e. adjusts clustering procedure is shown in Algorithm 3. Adjust clustering procedure is called whenever the time interval is a multiple of the gap time. The algorithm clusters two neighboring grids only if they are strongly correlated where two grids are considered as strongly correlated if their attractions in both directions are higher than a threshold value. The threshold value is calculated by averaging the total sum of attraction between each pair of grids which is $\frac{1}{|P|(1-\lambda)}$, where P is the total number of grid pairs and λ is decaying factor. During the offline phase, the algorithm only adjusts grids whose density attributes changed since the last time the grids were adjusted. For example, at time *t1* grid *g* is transitional and after *tgap* time, the grid becomes dense then the algorithm adjust the grids and clusters the grid *g* with its strongly correlated neighbor. But, if even after *tgap* time the grid *g* is still transitional, the algorithm does not adjust the grids.

The D-Stream II algorithm determines, this time, interval gap so that the dynamic nature of the data is captured and the time interval is not too small or large. The D-Stream II algorithm handles outliers by detecting the sporadic grids from the sparse grid and removing the grid from the grid list. The sparse grid is considered as sporadic grid if the density of the grid is less than the density threshold function which is determined such that a transitional or dense grid will never be falsely deleted due to the removal of the sporadic grid. This approach improves the time complexity of the algorithm since all grids are not adjusted every time gap interval.

Although the original D-Stream II algorithm performs accurate and efficient time-series data clustering in most scenarios, it fails or is inaccurate in three scenarios: 1) no isolated dense clusters are formed after initial clustering as the original D-Stream II algorithm generates a newly isolated clusters only once in offline phase i.e. only in the initial clustering procedure, 2) dense grids are never merged if initially they are not correlated as the original D-Stream II algorithm does not consider correlation as an attribute and 3) the original D-Stream II algorithm also considers the diagonal grids along with the neighboring grids for calculation attraction with neighboring grids.

III. Modified D-Stream Algorithm

Our modification of D-Stream II clustering algorithm handles all three scenarios mentioned above while maintaining the performance in other scenarios and performs effective and accurate clustering as compared to the original D-Stream II algorithm. It generates a new isolated dense grid in every offline phase call, it also considers the correlation between the two grids as clustering criteria and only considers the neighboring grids for attraction calculation. The algorithm is also capable of detecting and removing sporadic grids mapped by outliers in order to dramatically improve the space and time efficiency of the system.

Algorithm 4 : Main procedure for our modified algorithm

```
1.  tc = 0;
2.  initialize an empty red-black tree for grid list;
3.  while data stream is active do
4.      read record x = (x1, x2, . . . , xd);
5.      determine the density grid g that contains x;
6.      if (g not in grid list) insert g to grid list;
7.      update the characteristic vector of g;
8.      if (tc mod gap == 0) or (tc == gap )then
9.         detect and remove sporadic grids from grid list;
10.        call adjust clustering(grid list);
11.     end if
12.     tc = tc + 1;
13. end while
```

A. Isolated dense grids:

The original D-Stream II algorithm considers an isolated dense grid as a newly generated cluster only in the initial clustering procedure. During the subsequent call of offline phase, D-Stream II algorithm checks if a dense grid has any strongly correlated neighboring grids which belong to an already generated cluster or not. If such a strongly correlated neighbor is found then the grid is added to the neighboring cluster. If no such neighboring grid is found, then the D-Stream II algorithm does not categorize the dense grid into a newly generated cluster which over the time can grow into a larger cluster. Since D-Stream II algorithm does not consider such isolated dense grids the clustering result are not very accurate. In our modified D-Stream II algorithm, we consider an isolated dense grid as a newly generated cluster if no strongly correlated neighbors are found. Algorithms 4 and 5 show the

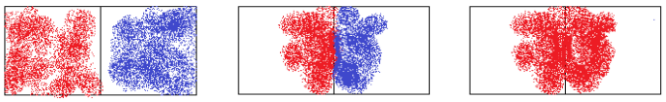
main and adjust clustering procedures of the modified D-Stream II algorithm. In the modified D-Stream II algorithm, adjust clustering procedure for every time interval gap, if for a dense grid *g* a strongly correlated neighbor does not exist and the grid does not already belong to any cluster then a newly isolated cluster is formed as shown in Algorithm 5 in line 9 and 10. Since in the modified algorithm we always generate an isolated cluster for a dense grid in adjust clustering procedure, we do not have to explicitly call the initial clustering procedure. Therefore, in our modified D-Stream II clustering algorithm the call for the initial clustering procedure is removed from the main procedure as shown in Algorithm 4 on line 8-10.

Algorithm 5: Adjust clustering procedure in our algorithm (*grid list*)

1. update the density of all grids in *grid list*;
2. **foreach** grid *g* whose attribute (dense/sparse/transitional) is changed since last call to adjust clustering() or the grid becomes strongly correlated to at least one neighbor
3. **if** (*g* is a sparse grid)
4. delete *g* from its cluster *c*, label *g* as NO CLASS;
5. **if** (*c* becomes unconnected) split *c* into two clusters;
6. **else if** (*g* is a dense grid)
7. among all neighboring grids of *g* that are strongly correlated to *g*, find out the grid *h* whose cluster *ch* has the largest size;
8. **if** (*h* is not found) and (*g* does not belong to any cluster)
9. assign grid *g* to a distinct cluster;
10. **else if** (*h* is a dense grid)
11. **if** (*g* is labeled as NO CLASS) label *g* as in *ch*;
12. **else if** (*g* is in cluster *c* and $|c| > |ch|$)
13. label all grids in *ch* as in *c*;
14. **else if** (*g* is in cluster *c* and $|c| \leq |ch|$)
15. label all grids in *c* as in *ch*;
16. **else if** (*h* is a transitional grid)
17. **if** ((*g* is NO CLASS) and (*h* is an outside grid if *g* is added to *ch*)) label *g* as in *ch*;
18. **else if** (*g* is in cluster *c* and $|c| \geq |ch|$)
19. move *h* from cluster *ch* to *c*;
20. **else if** (*g* is a transitional grid)
21. among neighboring grids of *g* that are dense and strongly correlated to *g*, find the grid whose cluster c *'* has the largest size;
22. label *g* as in *c'*;
23. **end for**

B. Dense grid merging attributes:

D-Stream II algorithm adjusts the grid list only when the grid attributes change since the last time the grid was adjusted. The algorithm only considers the whether a grid is dense, transitional or sparse as an attribute. It does not take into consideration the evolving attraction of the two grids. It is possible that at time *t* two grids are dense but are not strongly correlated as shown in Fig 2(a). However, over time, the two grids become strongly correlated. The D-Stream II algorithm only adjusts cluster when an attribute changes, it does not merge the two dense grids as the attribute of both the grids has not changed as shown in Fig 2(b). Thus, in our modified D-Stream II algorithm, we also consider the correlation between the grids as an attribute as shown in line 2 of algorithm 5 to improve the clustering accuracy as shown in Fig 2(c).

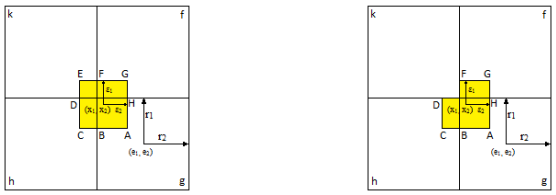


(a)Non-correlated grids (b) D-Stream II (c) Our algorithm

Fig 2 Correlation between two grids

C. Attraction Calculation:

In D-Stream II algorithm the attraction of a grid with its neighboring grids is calculated as shown in Fig 3(a) where the attraction of grid *g* with grid *f* is defined as the ratio of the volume of intersection of the hypercube (ACEG) yellow region and grid *f* i.e FG*GH to the volume of the hypercube. However, the D-Stream II algorithm defines the neighboring grids of grid *g* as a set of grids whose center differ from *g* in at most one direction. Therefore, a grid *g* will have 2 neighbors in i^{th} dimension, one whose center in i^{th} dimension is greater than the center of grid *g* in i^{th} dimension and other with center in i^{th} dimension less than the center of grid *g* in i^{th} dimension and the center of grid *g* and its neighbor is equal in all another dimension except i^{th} dimension. In our modified D-Stream II algorithm, while calculating the attraction of grid *g* to its neighboring grids, we do not consider the attraction of grid *g* with grid *k* as only grid *f* and grid *h* are neighbors of grid *g* as shown in Fig 3(b) which improves the clustering accuracy.



(a) (b)

Fig 3 Grid attraction calculation

IV. RESULTS

We use our modified D-Stream II time-series clustering to analyze everyday energy demand patterns. We analyze the real time-series energy consumption data of 15 different residential houses from Pecan Street which represent various people, demographics, and energy consumption patterns. We used three different time resolutions: 1 hour, 15 minutes and 1 minute and three different energy consumption resolutions: 0.125kWh, 0.25kWh, and 0.5kWh, where energy resolution is equal to the width of each grid. Higher time and energy resolution give more granular information about the energy consumption pattern but utilize more memory. Lower time and energy resolution give coarse information about the energy consumption patterns but utilize less memory. We use different time and energy resolution to analyze the trend between granularity and memory utilization and observed that our D-Stream II algorithm perform clustering within the memory constraints of time and energy resolution as 1min and 0.125kWh respectively.

We implemented the modified D-Stream II clustering algorithm on RPi2 which is a low processing embedded platform and is very frequently used for home automation in a Smart Grid environment. We analyzed a total of 135 different

data sets of real energy consumption 9 datasets per house for different time and energy resolutions. The original D-Stream II algorithm was not able to converge and provide clusters on any of 135 data sets. However, our modified D-Stream II algorithm obtained clusters for all 135 data sets of 15 different residential house.

A. *Comparison with original D-Stream II:*

Fig 4 shows the number of grids used and the number of clusters formed over time for clustering one-year energy consumption data of house number 1 using original D-Stream II and modified D-Stream II clustering algorithm for 1-hour time resolution and 0.25kWh energy resolution. The number of grids populated is the same for both the algorithm as it represents the mapping of data which is same for both algorithms. Fig 4 shows that the number of the clusters formed is always zero for original D-Stream II but the modified D-Stream II algorithm generates dynamic clusters over time where the number of clusters first increases as the grid is being populated for the first time and then decreases as correlated clusters are merged and less frequently used ones are deleted.

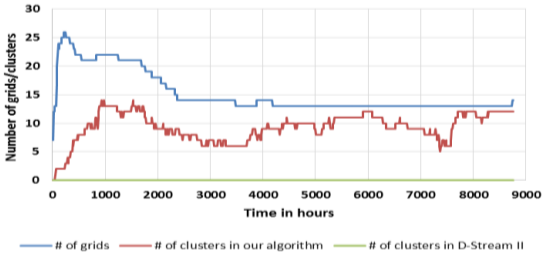


Fig 4 Number of grids used and clusters formed over time

B. *Applications:*

The dynamic clustering can be used for context analysis in many ways. One way is to analysis the frequency of different energy consumption values over time. Fig 5 shows the number of times a grid with particular energy consumption value is updated over a time period of one-year for five representative houses for the time resolution of 15 minutes and energy resolution of 0.125kWh. Most frequent energy consumption for house no 1, 2, 3, 4 and 5 is around 0.625kWh, 0.375kWh, 0.375kWh, 0.5kWh, and 0.25kWh respectively. The range of most frequent energy consumption for house no 59, 68, 86, 93 and 94 is around 0.375-2.25 kWh, 0.25-1 kWh, 0.25-1.75 kWh, 0.25-1.5 kWh and 0.125-0.75 kWh respectively.

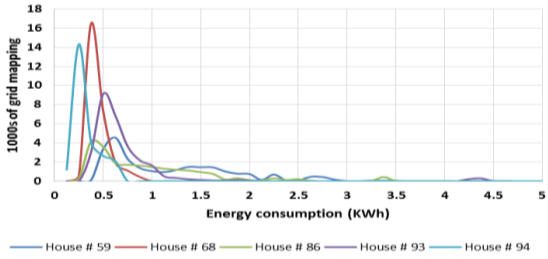


Fig 5 Frequency of energy consumption

Consumers can detect defective appliances by using dynamic clustering. If different clusters with higher energy values are formed for similar energy consumption pattern which this means an appliance is consuming more energy because the appliance is defective. They can also compare their energy consumption pattern with their neighbor by comparing the clusters formed by different houses. When customers received information on the energy consumption of their neighbors, average energy use declined [19].

C. *Memory utilization on RPi2:*

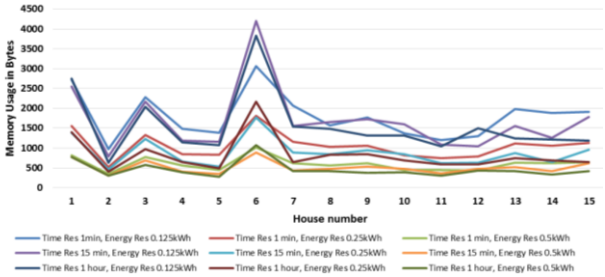
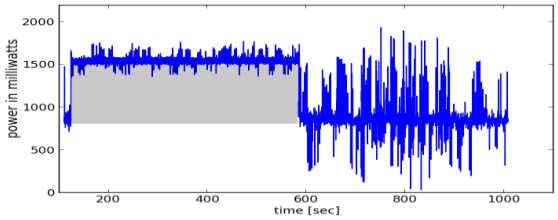


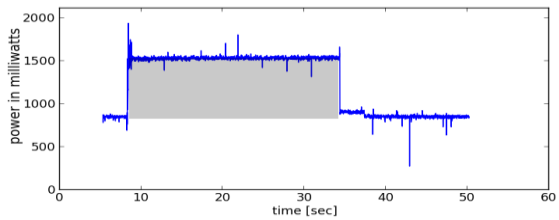
Fig 6 Max memory in bytes for different time resolutions

In Fig 6, we show the maximum memory usage in bytes while clustering energy consumption data of each house for different time and energy resolution. We can see that the maximum memory utilization among all the houses is 4196 bytes (< 5KB) which occur for house no. 6 for energy resolution of 0.125kWh and time resolution of 15 mins. Also, we can see that the memory utilization increases with increase in the time resolution as we will have more data points since the sampling rate is increased and decreases with a decrease in energy resolution as we will have less number of grids.

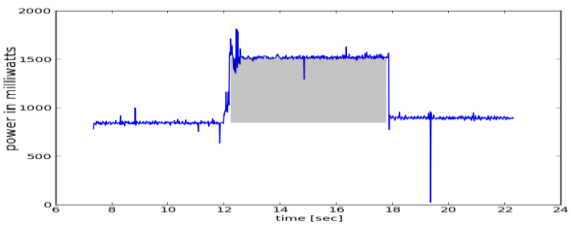
D. *Power/Energy Consumption on RPi2:*



a) Time resolution of 1 min



b) Time resolution of 15 mins



c) Time resolution of 1 hour

Fig 7. Power consumption on Raspberry Pi 2

Fig 7 shows the power consumption for different time resolution while performing clustering using our algorithm for house no 7 which has the maximum memory utilization among 15 houses. The power consumption when RPi2 is idle is 800mW and it increases to 1500mW when our clustering algorithm is performed. Therefore, the energy consumption overhead increase (shown as the shaded area in Fig 7) on RPi2 for clustering using our algorithm for house no 6 is 360J, 20J, and 4J for time resolution 1min, 15mins and 1 hour respectively. Average energy consumption overhead increase on RPi2 for 15 houses is 291.6J, 16.84J and 3.78J for time resolution 1min, 15mins and 1 hour respectively. The energy consumption increases with increase in the granularity of time resolution as the clustering is performed on a larger set of data points and takes more execution time. The max energy consumption overhead increase (360J) in RPi 2 due to our algorithm is reasonably low for a real-time embedded application.

E. Performance on RPi2:

We also found that for the given one-year energy consumption of five different house the maximum number of grids used were 75 out of 200 and the maximum number of clusters formed were 26. Fig 8 shows the execution time in logarithmic scale for three different houses for different time resolutions. The total execution time for clustering one-year energy consumption data for house no 6 using our algorithm on RPi2 is 520sec, 25sec, and 5sec for the time resolution of 1 min, 15 mins and 1 hour respectively. The average execution time on RPi2 for all 15 houses is 421sec, 21sec, and 4.5sec for the time resolution of 1 min, 15 mins and 1 hour. The execution time increased with increase in the granularity of time resolution as we had more data points as shown in Fig 8. Execution time for the time resolution of 1 min is more than execution time for the time resolution of 15 mins and 1 hour (60mins). The max execution time (520sec) on RPi2 for processing nearly half a million data points is within the processing constraint of a real-time embedded system.

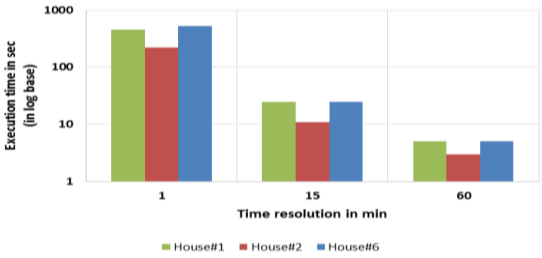


Fig 8 Execution time vs time resolution

V. Conclusion

In this paper, we proposed a modified D-Stream II density grid-based clustering algorithm for context analysis of time-series data in Smart Grid. We show the D-Stream II fails to cluster in three different scenarios while our algorithm converges and perform clustering. Based on our analysis of energy consumption of 15 residential houses, we showed that modified D-Stream II clustering algorithm performs the clustering within memory and processing constraints of RPi2. The modified D-Stream II algorithm perform clustering on approximately half a million energy consumption data values within a memory utilization of 5KB and execution time of 8min and energy consumption overhead of 360J.

Acknowledgement

This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This work was funded also by ARPA-E NODES project.

References

[1] X. Yu, C. Cecati, T. Dillon, and M. G. Simoes. The new frontiers of smart grids. IEEE Industrial Electronics Magazine, 5(3), 2011.

[2] S. D. Ramchurn, P. Vyetelingum, A. Rogers, and N. R. Jennings. Putting the smarts into the smart grid: A grand challenge for artificial intelligence. Communications of the ACM, 55(4), 2012

[3] J. L. Mathieu, D. S. Callaway, and S. Kilicote. Variability in automated responses of commercial buildings and industrial facilities to dynamic electricity prices. Energy and Buildings, 43(12), 2011.

[4] A Amini, TY Wah, H Saboohi On Density-Based Data Streams Clustering Algorithms: A Survey. Journal of Computer Science and Technology 29 (1), 116-141

[5] Aggarwal C C, Han J, Wang J, Yu P S. A framework for clustering evolving data streams. In Proc. the 29th International Conference on Very Large Data Bases, Sept. 2003, pp.81-92.

[6] Kranen P, Assent I, Baldauf C, Seidl T. The clustree: Indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst., 2011, 29(2)

[7] Dang X, Lee V, Ng W K et al. An EM-based algorithm for clustering data streams in sliding windows. In Proc. the Int. Conf. Database Systems for Advanced Applications, Apr. 2009, pp.230-235

[8] Cao F, Ester M, Qian W, Zhou A. Density-based clustering over an evolving data stream with noise. In Proc. the 2006 SIAM Conference on Data Mining, April 2006, pp.328-339

[9] Chen Y, Tu L. Density-based clustering for real-time stream data. In Proc. the 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Aug. 2007, pp.133-142

[10] Tu L, Chen Y. Stream data clustering based on grid density and attraction. ACM Transactions on Knowledge Discovery Data, 2009, 3(3)

[11] https://www.eia.gov/totalenergy/data/monthly/pdf/mer.pdf

[12] Eichinger F., Pathmaperuma D., Vogt H., Muller, E.: Data ¨ Analysis Challenges in the Future Energy Domain. In: Intelligent Data Analysis for Sustainable Development, chap. 7, pp. 181–242 (2013)

[13] Nga D., See O., Do Quang, C., Chee, L.: Visualization Techniques in Smart Grid. Smart Grid and Renewable Energy 3(3), 175–185 (2012)

[14] Kolter J.Z., Johnson M.: REDD: A Public Data Set for Energy Disaggregation Research. (SustKDD) (2011)

[15] Ilic, D., Karnouskos, S., Goncalves Da Silva, P.: Sensing in Power Distribution Networks via Large Numbers of Smart Meters. (ISGT), pp. 1–6 (2012)

[16] Dannecker, L., Bohm, M., Fischer, U., Rosenthal, F., Hacken- ¨ broich, G., Lehner, W.: State-of-the-Art Report on Forecasting – A Survey of Forecast Models for Energy Demand and Supply. Deliverable 4.1, The MIRACLE Consortium, Dresden, Germany (2010)

[17] Ramanathan, R., Engle, R., Granger, C.W., Vahid-Araghi, F., Brace, C.: Short-run Forecasts of Electricity Loads and Peaks. International Journal of Forecasting 13(2), 161–174 (1997)

[18] Aggarwal, S.K., Saini, L.M., Kumar, A.: Electricity Price Forecasting in Deregulated Markets: A Review and Evaluation. International Journal of Electrical Power and Energy Systems 31(1), 13–22 (2009)

[19] http://www.nber.org/digest/feb10/w15386.html