An Intelligent System for Mining and Maintaining Correlation Patterns among Appliances in Smart Home

Yi-Cheng Chen[1] and Julia Tzu-Ya Weng[2,3]

[1]*Department of Computer Science and Information Engineering, Tamkang University, Taiwan*

[2]*Department of Computer Science and Engineering, Yuan Ze University, Taiwan*

[3]*Innovation Center for Big Data and Digital Convergence, Yuan Ze University, Taiwan*

ycchen@mail.tku.edu.tw julweng@saturn.yzu.edu.tw

Abstract

Recently, due to the great advent of sensor technology, residents can collect the usage data of appliances in a house easily. However, with data progressively generating, it is still a challenge to visualize how these appliances are used. Thus, a mining and maintaining system is needed to incrementally discover appliance usage patterns. Most previous studies on usage pattern discovery are mainly focused on analyzing the patterns of single appliance and do not consider the incremental maintenance of mining results. In this paper, a novel system, namely, *Dynamic Correlation Mining System* (*DCMS*) is developed to capture and maintain the correlation patterns among appliances incrementally. The experimental results indicate that proposed system is efficient in execution time and possesses scalability. Furthermore, we apply DCMS on a real-world dataset to show the practicability.

Keywords: *sensor data analysis; smart home; correlation pattern; intelligent system; incremental mining*

1. Introduction

Concerns over global climate changes have motivated significant efforts in reducing the electricity usage in residence which is a significant contributor of greenhouse gas emissions. However, electricity conservation is difficult for the residents since the lack of detailed electricity usage. With the advance of sensor technology, an increasing number of smart power meters, which facilitates data collection of appliance usage, have been deployed.

With the appliance usage data, residents could supposedly visualize how the appliances are used. Nonetheless, with a huge amount of usage data progressively generated, subtle information may exist but hidden. Therefore, it is necessary to design a system not only to capture appliance usage patterns but also maintain the mining results. These patterns can help users to better understand how they use the appliances at home.

Most prior studies focus on knowledge extraction for a single appliance instead of the correlation among appliances in a house. In our daily life, we usually use different appliances simultaneously. For example, air conditioner and television in the living room may be turned on in the evening, as shown in Fig. 1. The correlation among the usage of some appliances can provide valuable information to assist residents better understand how they use appliances.

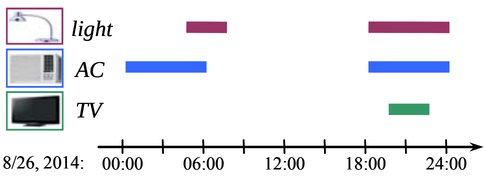

Fig. 1: An example of daily appliance usage sequence.

In real applications, the usage data usually are generated progressively, i.e., new data have been inserted and appended in database. Obviously, incremental mining of correlation patterns is complex and arduous, and requires a different approach from patterns extracted from single appliance. To the best of our knowledge, little attention has been paid to this issue, partly because of the complex relationship among usage intervals. When appending an interval, the complex relations may lead to the generation of a greater number of possible candidates.

Allen's 13 temporal relations [1] are usually adopted to describe the complex relations among usage intervals, as shown in Table 1. However, Allen's temporal logics are binary relations and may be problematic when describing relationships among more than three intervals. An appropriate representation is crucial for this circumstance. Various representations [5, 15, 18] have been proposed; however, most of them have a restriction on either ambiguity or scalability and do not consider the processing of incremental maintenance.

Table 1: Allen's 13 relations between two intervals.

Temporal Relation	Inversed Relation	Pictorial Example	Endpoint sequence
A before *B*	*B* after *A*	A B	$A^+ A^- B^+ B^-$
A overlaps *B*	*B* overlapped-by *A*	A B	$A^+ B^+ A^- B^-$
A contains *B*	*B* during *A*	A B	$A^+ B^+ B^- A^-$
A starts *B*	*B* started-by *A*	A B	$(A^+ B^+) A^- B^-$
A finished-by *B*	*B* finishes *A*	A B	$A^+ B^+ (A^- B^-)$
A meets *B*	*B* met-by *A*	A B	$A^+ (A^- B^+) B^-$
A equal *B*	*B* equal *A*	A B	$(A^+ B^+) (A^- B^-)$

In this paper, we develop an intelligent system, *Dynamic Correlation Mining System* (*DCMS*), to incrementally mine correlation patterns in smart home. The contributions of our proposed system are as follows:

– First, we develop a new representation, *dynamic representation*, to express a pattern nonambiguously. We use the arrangement of endpoints of all intervals to simplify the processing of complex relation among intervals, and consider the time information to facilitate incremental mining.

– Second, based on the dynamic representation, an algorithm, *I*ncremental *C*orrelation *P*attern *Miner* (*ICPMiner*), is proposed to incrementally discover correlation patterns in usage database. Experimental studies indicated that, in incremental environment, ICPMiner is efficient and outperforms other state-of-the-art algorithms.

– Third, we employ some pruning strategies to reduce the search space and avoid non-promising database process. The experimental results reveal that pruning strategies can improvement the runtime performance of ICPMiner efficiently.

– Finally, we applied DCMS on real datasets to demonstrate the practicability of incremental maintenance of the correlation patterns.

The rest of the paper is organized as follows. Section 2 provides the related works. Section 3 introduces the system architecture and preliminary. Section 4 describes the ICPMiner algorithm. Section 5 gives the experiments and performance study, and we conclude in Section 6.

2. Related Work

In this section, we discuss some previous works extracted useful knowledge and patterns of a single device applying on energy disaggregation [3, 7, 11, 14, 17] or appliance recognition [2, 4, 6, 8, 9, 10, 13].

Suzuki et al. [17] use a new NIALM technique based on integer programming to disaggregate residential power use. Matthews et al. [14] use a dynamic Bayesian network and filter to disaggregate the data online. Kim et al. [11] investigate the effectiveness of several unsupervised disaggregation methods on low frequency power measurements collected in real homes. They also propose a usage pattern which consists of on-duration distribution of all appliances. Goncalves et al. [7] explore an unsupervised approach to determine the number of appliances in the household, including their power consumption and state, at any given moment. Chen et al. [3] disaggregate utility consumption from smart meters into specific usage associated with certain human activities. They propose a novel statistical framework for disaggregation on coarse granular smart meter readings by modeling fixture characteristic, household behavior, and activity correlations.

Ito et al. [8] extract features from the current (e.g., amplitude, form, timing) to develop appliance signatures. For appliance recognition, Kato et al. [10] use Principal Component Analysis to extract features from electric signals and classify them using Support Vector Machine. Aritoni et al. [2] develop a software prototype to understand the behaviors of household appliances. Chen et al. [4, 6] introduce two types of usage patterns to describe users' representative behaviors. Lin et al. [13] apply power meters for appliance recognition on the electric panel. Jakkula et al. [9] propose an Apriori-based algorithm for activity prediction and anomaly detection from sensor data in a smart home. All aforementioned studies focus on knowledge extraction for a single appliance and ignore the concept of incremental maintenance of mining results in a smart home.

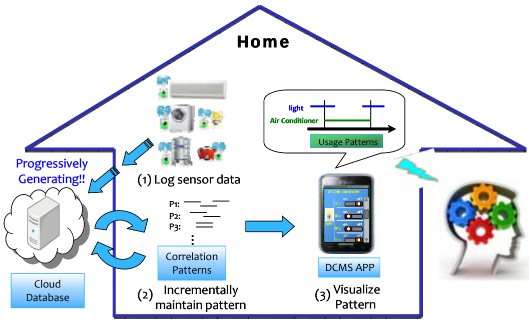

Fig. 2: System framework of DCMS

3. System Architecture & Preliminary

We develop an intelligent system, called *Dynamic Correlation Mining System* (abbreviated as ***DCMS***), not only to capture the correlation patterns among appliances but also to maintain the discovered pattern when usage data are progressively generated in the smart home. The framework of DCMS is shown in Fig. 2.

We first attach a smart meter to each appliance in smart home environment. The smart meter will transmit the usage data of the appliance to a cloud server. Since the data are generated progressively, an efficient algorithm, named *I*ncremental *C*orrelation *P*attern *Miner* (*ICPMiner*), is proposed to incrementally mine and maintain the correlation patterns among appliances. Finally, we develop an APP to visualize the discovered correlation patterns for residents. Before introducing the ICPMiner, we give some definition first.

Definition 1 (usage sequence and database) Let $E = \{e_1, e_2, \ldots, e_k\}$ be the set of appliances. We say the triplet $(e_i, s_i, f_i) \in E \times N \times N$ is a usage interval, where $e_i \in E$, $s_i, f_i \in N$ and $s_i < f_i$. The s_i and f_i are called the starting time and the finishing time, respectively. An usage sequence q is a series of usage intervals $\langle(e_1, s_1, f_1), \ldots, (e_n, s_n, f_n)\rangle$. The time information of q is the starting time of first interval and the finishing time of last interval in q, i.e., s_1 and f_n. A database $DB = \{r_1, r_2, \ldots, r_m\}$ is called a usage database where each record r_i is a pair of sequence-id (*SID*) and usage sequence, i.e., $r_i = \langle SID_i, q_i\rangle$.

Definition 2 (dynamic representation) Given a usage sequence $q = \langle(e_1, s_1, f_1), \ldots, (e_i, s_i, f_i), \ldots, (e_n, s_n, f_n)\rangle$, $T_q = \{ s_1, f_1, \ldots, s_i, f_i, \ldots, s_n, f_n \}$ is a set of all endpoints in q. After sorting T in non-decreasing order, an endpoint sequence $q_e = \langle t_1, t_2, \ldots, t_{2n}\rangle$ can be derived by representing s_i and f_i as e_i^+ and e_i^-, respectively. We use the parenthesis to form an endpointset to indicate the times of endpoints are the same. The corresponding endpoint sequences of 13 Allen's temporal relations are shown in Table 1. The dynamic representation of q includes the corresponding endpoint sequence q_e and time information $[s_1, f_n]$ of q. For example, given a usage sequence $\langle(A, 1, 3), (B, 5, 9)\rangle$, its time set is $\{1, 3, 5, 9\}$; hence, the corresponding endpoint sequence is $\langle A^+A^-B^+B^-\rangle$. The dynamic representation of q is $\langle A^+A^-B^+B^-\rangle$ [1, 9]. Without loss of generality, for the rest of this paper, we suppose all the sequences in a usage database have been transformed into dynamic representation.

Definition 3 (correlation pattern and frequent pattern tree) Given a usage database *DB*, a record $\langle SID, q_e, [s, f]\rangle$ is said to contain an endpoint sequence α, if α is a subsequence of q_e (represented as $\alpha \sqsubseteq q_e$). The support of α in *DB* is the number of records containing α, i.e., $support(\alpha) = |\{ \langle SID, q_e, [s, f]\rangle \in DB) \mid \alpha \sqsubseteq q_e\}|$. Given a positive integer *min_sup* as the support threshold, the set of correlation patterns includes all frequent endpoint sequences whose supports are no less than *min_sup*. A frequent pattern tree (*FPT*) *T* is a tree that represents the set of correlation patterns in database. A node *d* in *T* stores an endpoint corresponding to a correlation pattern that starts from the root node to *d*. Each node also preserves two information, say ***support_value*** and ***sequence_list***. The support_value represents the support count of the correlation pattern. The sequence_list stores a list of *SID*s to represent the sequences containing this correlation pattern.

Actually, two types of incremental updates for usage database are used: 1) inserting new usage sequences into database, denoted as INSERT; 2) appending new usage intervals to existing usage sequences, denoted as APPEND. An application may include all types of updates. When the database is updated with a combination of INSERT and APPEND, we can regard the INSERT as a special case of APPEND, for inserting a new sequence is equivalent to appending a new sequence to an empty sequence, as shown in Fig. 3.

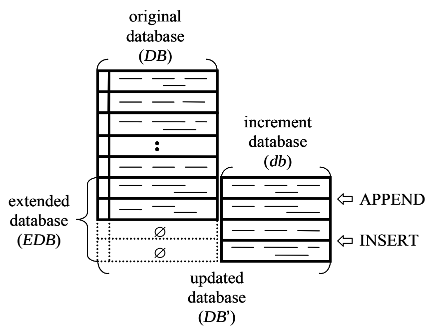

Fig. 3: Concept of incremental update in usage database.

With three usage sequences q, q' and q'', $q'' = q \diamond q'$ means q' is the concatenation of q. q' is called the **appended sequence** of q. q'' is an **updated sequence** of q appended with q'. To facilitate the presentation of this paper, we define increment and update databases. Given a temporal database, *DB*, truncated and appended with a few event sequences after a period, *DB* is called **original database**.

Definition 4 (increment and updated database) The increment database *db* is referred to as the set of newly appended sequences. The *SID*s of the appended sequences in *db* may already exist in *DB*. A database *DB* combining all the event sequences in *db* is referred to as the updated database *DB'*, as shown in Fig. 3.

4. ICPMiner Algorithm

When a usage database *DB* is updated to *DB'*, there are three possible cases for the correlation patterns in *DB'*:

– Case1: A pattern is frequent in *DB'*, and also frequent in *DB*.
– Case2: A pattern is frequent in *DB'*, and infrequent in *DB* but has a frequent pattern in *DB* as a prefix.
– Case3: A pattern is frequent in *DB'*, and infrequent in *DB* and has no any frequent patterns in *DB* as a prefix.

Case1 is easy to handle since we have already stored the information of previous mining results into FPT_{DB}. We can obtain the correlation patterns in Case1 by checking and adjusting the support of every pattern in FPT_{DB} in *DB'*. In Case2, although we have not preserved any information of infrequent sequences in *DB*, all correlation patterns have at least one prefix subsequence which is frequent in *DB*, i.e., the frequent prefix is stored in FPT_{DB}. Hence, we can utilize the correlation patterns in FPT_{DB} as prefix to recursively discover the correlation patterns in Case 2. Since, in Case 3, the correlation patterns have no information stored in previous mining results, FPT_{DB}, we need to scan *DB'* for all new frequent endpoints, and then use each new frequent endpoint as prefix to construct projected database and recursively mine all correlation patterns in Case 3.

In order to calculate the support of all patterns which are infrequent in *DB* but frequent in *DB'*, the naïve method may keep the information of all possible candidate set, i.e., mining *EDB* with *min_sup* = 1. This awkward approach may consume large memory and many non-promising database projection. To remedy this problem, we propose an algorithm, ICPMiner, with two optimization techniques to reduce unnecessary space searches.

Definition 5 (search reduction) Given a temporal pattern α in *DB* (node α in FPT_{DB}), when *DB* is updated to *DB'*, *incre_sid* is defined as a set of all SIDs in increment database *db* and $incre\_endpoint_{|\alpha}$ is defined as a set of all event slices in $db_{|\alpha}$. We have two search space reductions,

i) sequence-reduction: If {α' s sequence list} $\cap$ *incre_sid* = $\emptyset$, then $DB_{|\alpha}$ is identical to $DB'_{|\alpha}$. The support of α and all temporal patterns prefixed with α, i.e., node α and all child nodes of α in FPT_{DB}, are unchanged in *DB'*. Hence there is no temporal pattern which is infrequent in *DB* but becomes frequent in *DB'* with α as prefix. We can stop searching α and all α's child nodes in FPT_{DB}.

ii) endpoint-reduction: If α' s parent node in in FPT_{DB} does not insert any node as child node when *DB* is updated to *DB'*, and the set of {α and all α' s sibling nodes} $\cap$ $incre\_endpoint_{|\alpha} = \emptyset$, then the support of α and all temporal patterns prefixed with α, i.e., node α and all child nodes of α in FPT_{DB}, are unchanged in *DB'*. Hence there is no temporal pattern which is infrequent in *DB* but becomes frequent in *DB'* with α as prefix. We can stop searching α and all child nodes of α in FPT_{DB}

The search space reduction in Definition 5 plays an important role in ICPMiner. When the minimum support goes lower and the maintained patterns turn to be longer, many unnecessary searches can be avoided effectively. As observed in our experiments, the search space reduction can skip more than 60% nodes in FPT_{DB}, especially when minimum support is extremely low. This is also the main reason why ICPMiner not only outperforms other algorithms in runtime performance, but also consumes less memory space. The pseudo code is shown in Algorithm 1.

Algorithm 1: *ICPMiner* (*DB'*, *min_sup*, FPT_{DB})

Input: *DB'*: updated temporal database, *min_sup*: the minimum support, FPT_{DB}: frequent pattern tree of original *DB*
Output: $FPT_{DB'}$: frequent pattern tree of updated database *DB'*

// **initial Phase**
$FPT_{DB'} \leftarrow \emptyset$; determine *EDB*;
transform *DB'* into dynamic presentation and find all frequent endpoints concurrently;
NFS ← new frequent endpoints in *DB'* ; // frequent endpoints in *DB'* $\notin FPT_{DB}$

// **mining phase**
for each endpoint *b* in *NFS* **do**
  insert *b* into $FPT_{DB'}$;
  call ***CPrefixSpan*** ($DB'_{|b}$, *b* , *min_sup*, $FPT_{DB'}$);

// **extending phase**
scan *DB'* for update the support of node in FPT_{DB} ;
for each node α in FPT_{DB} **do**
  FPT_{DB} ← ***CPrefixSpan*** (*DB'*, α, *min_sup*, FPT_{DB});
  for each node α in $FPT_{DB} \geq$ *min_sup* **do**
    insert α into $FPT_{DB'}$;
  if ***search_reduction*** (α, $DB'_{|\alpha}$) = "false"
    call ***CPrefixSpan*** ($DB'_{|\alpha}$, α , *min_sup*, $FPT_{DB'}$);
Output $FPT_{DB'}$;

There are three phases in ICPMiner, initial phase, mining phase and extending phase. Initial phase first uses the interval extension to transform all sequences into dynamic representation (line 2, algorithm 1), and scans *db* once to discover all new frequent endpoints in *DB'*. Notice that, if we store previous infrequent endpoints in *DB*, we can find the complete set of new frequent endpoints in *DB'* by just scan *EDB* without rescanning *DB* again (line 3, algorithm 1). Then, in mining phase, we use each new frequent slice as prefix to construct projected database and call *CPrefixSpan* to discover the temporal patterns (lines 4-6, algorithm 1).

CPrefixSpan extends the concept of projected database from [16] and employs two optimization strategies to reduce the search space. Since the starting endpoints and finishing endpoints definitely occur in pairs in a sequence, we only project the frequent finishing endpoints which have the corresponding starting endpoints in their prefixes (lines 3-5, procedure 1). We can prune off non-qualified patterns before constructing projected database.

Procedure 1: *CPrefixSpan* ($DB_{|\alpha}$, α, *min_sup*, FPT_{DB})

Input: $DB_{|\alpha}$: projected database, α: a temporal pattern, *min_sup*: the minimum support, FPT_{DB}: frequent pattern tree of original *DB*

01: scan $DB_{|\alpha}$ once and find all frequent endpoints *c*;
02: **for each** frequent endpoint *c* **do**
03:   **if** *c* is a "finishing endpoint" **then**
04:     **if** exist corresponding starting endpoint in α **then**
05:       append *c* to α to form β;
06:   **if** *c* is a "starting endpoint" **then**
07:     append *c* to α to form β;
08: **for each** β **do**
09:   construct projected database $DB_{|\beta}$ with significant postfix;
10:   **if** $|DB_{|\beta}| \geq$ *min_sup* **then**
11:     insert β into FPT_{DB};
12:   **if** *search_reduction* (β, DB'$|\beta$) = "false"
13:     call ***CPrefixSpan*** ($DB_{|\beta}$, β, *min_sup*, FPT_{DB});

Moreover, when constructing a projected database, some endpoints in postfixes need not be considered. With respect to a prefix $\langle p\rangle$, a finishing endpoint in a projected postfix is called significant, if it has corresponding starting endpoint in $\langle p\rangle$. We construct the projected database $DB_{|\langle p\rangle}$ by collecting significant endpoints only (line 9, procedure 1). All insignificant endpoints are eliminated since they can be ignored in the discovery of temporal patterns. Note that the *search_reduction* technique in Definition 5 can be used in CPrefixSpan when we call it recursively. We utilize *search_ reduction* to check whether growing can stop (line 12, procedure 1). If not, we recursively call CPrefixSpan to discover the temporal patterns.

Finally, in extending phase, ICPMiner updates the support of every frequent pattern in *DB*. If a pattern is still frequent in *DB'*, we also use *search_reduction* to check if we can stop growing. If not, *CPrefixSpan* is called to discover the temporal patterns (lines 12-13, algorithm 1).

5. Experimental Results

To evaluate the performance of ICPMiner, we implement CTMiner [5], TPrefixSpan [18], IEMiner [15] for comparison. All algorithms were implemented in C++ language and tested on a computer with Pentium D 3.0 GHz with 2 GB of main memory. The performance study has been conducted on both synthetic and real world datasets. First, we compare the execution time and memory usage using synthetic datasets at extreme low minimum support. Then, we use a real dataset [12] to show the performance and the practicability of incremental mining for correlation patterns.

The synthetic datasets are generated using synthetic generation program [18]. Since the original data generation program was designed to generate static database, the generator requires modifications on incremental scenario accordingly. The parameter setting of temporal data generator is shown in Table 2. We partition the updated database *DB'* into the original database *DB* and increment database *db*, as the example in Fig. 1. Different settings of three parameters are used to reflect different updating scenarios.

Table 2: Parameters of synthetic data generator.

parameters	description
$\|D\|$	Number of event sequences
$\|C\|$	Average size of event sequences
$\|S\|$	Average size of potentially frequent sequences
N_S	Number of potentially frequent sequences
N	Number of event symbols
R_{inc}	Ratio of the number of sequences in increment database *db* to updated database *DB'*
R_{ext}	Ratio of the number of existed sequences extended to new sequences inserted in increment database *db*
R_{app}	Ratio of the number of intervals of an existed sequence appearing in original database *DB* to increment database *db*

5.1 *Execution time and memory usage*

In all the following experiments, two parameters are fixed, i.e., the average size of potentially frequent sequences, $|S| = 4$, and the number of potentially frequent sequences, N_S = 5,000. We set R_{inc} = 10%, R_{ext} = 50% and R_{app} = 20% to model common database updating scenario.

The first experiment for comparison of four algorithms is on the dataset *D*10k–*C*10–*N*1k with the minimum support thresholds varying from 0.01 % to 0.005 %. Obviously, re-mining from scratch with non-incremental algorithm is less efficient than using incremental maintaining algorithm, as illustrated in Fig. 4(a). When we continue to lower the minimum threshold, the

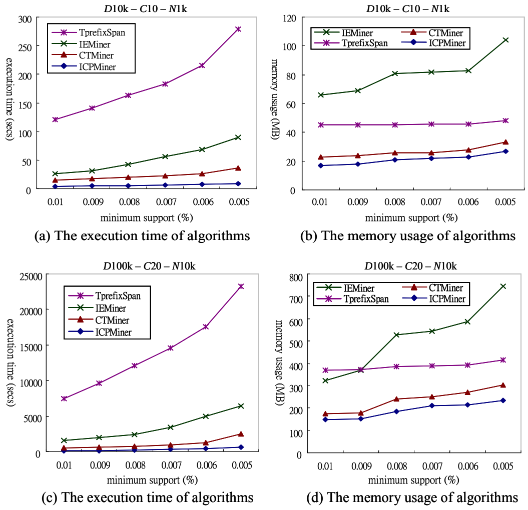

Fig. 4: Performance and memory usage on synthetic datasets.

runtime of ICPMiner outperforms the other algorithms. The memory usages of algorithms are showed as in Fig. 4(b). We can see that ICPMiner consumes less memory than the other algorithms. The second experiment is performed on data set *D*100k–*C*20–*N*10k, which contains 100,000 usage sequences, average length 40 and 10,000 usage intervals with common database updating scenario. The execution time of different algorithms is shown in Fig. 4(c). We can see that when the support is 0.005%, ICPMiner is significantly faster than other methods. Fig. 4(d) shows the memory usages of four algorithms with different minimum support thresholds. Obviously, ICPMiner consumes less memory than the other algorithms.

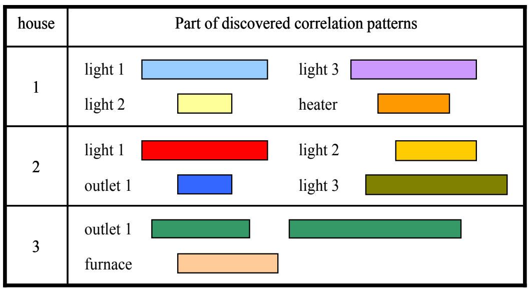

Fig. 5: Part of correlation patterns from REDD dataset

5.2 *Real World Dataset Analysis*

In addition to using synthetic datasets, we also have performed an experiment on real-world dataset to indicate the applicability of correlation pattern mining. The dataset REDD [12] used in the experiment is the power reading of appliances collected from six different houses. Each house has about 15 appliances. We convert the raw data into the usage interval with turn-on time and turn-off time. Fig. 5 shows the part of mining result applying ICPMiner on REDD dataset with *min_sup* = 0.3.

6. Conclusion

Recently, considerable concern has arisen over the electricity conservation due to the issue of greenhouse gas emissions. In this paper, we propose an intelligent system, *DCMS*, which not only could capture the usage correlation among appliances in a house, but also dynamically maintain the mining results with progressive data generation. The experimental studies indicate that DCMS is efficient and scalable. Furthermore, DCMS is applied on a real-world dataset to show the practicability of correlation pattern mining.

Reference

[1] J. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of ACM*, vol.26, issue 11, pp.832-843, 1983.

[2] O. Aritoni and V. Negru. A Methodology for Household Appliances Behavior Recognition in AmI Systems Integration. *7th International Conference on Automatic and Autonomous Systems (ICAS'11)*, pp. 175-178, 2011.

[3] F. Chen, J. Dai, B. Wang, S. Sahu, M. Naphade and C. Lu. Activity Analysis Based on Low Sample Rate Smart Meters. *17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pp. 240-248, 2011.

[4] Y. Chen, C. Chen, W. Peng, W Lee. Mining Correlation Patterns among Appliances in Smart Home Environment. *18th Pacific-Asia Conference in Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining (PAKDD'14)*, pp. 222-233, 2014.

[5] Y. Chen, J. Jiang, W. Peng and S. Lee. An Efficient Algorithm for Mining Time Interval-based Patterns in Large Databases. *19th ACM International Conference on Information and Knowledge Management (CIKM'10)*, pp. 49-58, 2010.

[6] Y. Chen, Y. Ko, W. Peng and W. Lee. Mining Appliance Usage Patterns in a Smart Home Environment. *17th Pacific-Asia Conference in Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining (PAKDD'13)*, pp. 99-110, 2013.

[7] H. Goncalves, A. Ocneanu and M. Berges. Unsupervised Disaggregation of Appliances using Aggregated Consumption Data. *KDD workshop on Data Mining Applications in Sustainability (SustKDD'11)*, 2011.

[8] M. Ito, R. Uda, S. Ichimura, K. Tago, T. Hoshi and Y. Matsushita. A Method of Appliance Detection Based on Features of Power Waveform. *Proceedings of 4th IEEE Symposium on Applications and the Internet (SAINT'04)*, pp. 291-294, 2004.

[9] V. Jakkula and D. Cook. Using Temporal Relations in Smart Environment Data for Activity

Prediction. *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, pp. 1-4, 2007.

[10] T. Kato, H. Cho, D. Lee, T. Toyomura and T. Yamazaki. Appliance Recognition from Electric Current Signals for Information-energy Integrated Network in Home Environments. *Ambient Assistive Health and Wellness Management in the Heart of the City*, vol. 5597, pp. 150-157, 2009.

[11] H. Kim, M. Marwah, M. Arlitt, G. Lyon and J. Han. Unsupervised Disaggregation of Low Frequency Power Measurements. *Proceedings of 11th SIAM International Conference on Data Mining (SDM'11)*, pp. 747-758, 2011.

[12] J. Kolter, M. Johnson. REDD: A Public Data Set for Energy Disaggregation Research. *KDD workshop on Data Mining Applications in Sustainability (SustKDD'11)*, 2011.

[13] G. Lin, S. Lee, J. Hsu and W. Jih. Applying Power Meters for Appliance Recognition on the Electric Panel. *Proceedings of 5th IEEE Conference on Industrial Electronics and Applications (ISIEA'10)*, pp. 2254-2259, 2010.

[14] H. Matthews, L. Soibelman, M. Berges and E. Goldman. Automatically Disaggregating the Total Electrical Load in Residential buildings: a profile of the required solution. *Intelligent Computing in Engineering*, pp. 381-389, 2008.

[15] D. Patel, W. Hsu and M. Lee. Mining Relationships Among Interval-based Events for Classification. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 393-404, 2008.

[16] J. Pei, J. Han, B. Mortazavi-Asl, H. Pito, Q. Chen, U. Dayal and M. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proceedings of 17th International Conference on Data Engineering (ICDE'01)*, pp. 215-224, 2001.

[17] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura and K. Ito. Nonintrusive Appliance Load Monitoring Based on Integer Programming. *International Conference on Instrumentation, Control and Information Technology (ICIT'08)*, pp. 2742-2747, 2008.

[18] S. Wu and Y. Chen. Mining Nonambiguous Temporal Patterns for Interval-Based Events. *IEEE Transactions on Knowledge and Data Engineering*, vol.19, issue 6, pp. 742-758, 2007.