2013 IEEE 13th International Conference on Data Mining Workshops

A Novel System for Extracting Useful Correlation in Smart Home Environment

[1]Yi-Cheng Chen, [2]Wen-Chih Peng and [3]Wang-Chien Lee
[1]*Department of Computer Science and Information Engineering, Tamkang University, New Taipei City, Taiwan*
[2]*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*
[3]*Department of Computer Science and Engineering, The Pennsylvania State University, Pennsylvania, USA*
E-mail: ycchen@mail.tku.edu.tw wcpeng@cs.nctu.edu.tw wlee@cse.psu.edu

***Abstract*—Owing to the great advent of sensor technology, the usage data of appliances in a house can be logged and collected easily today. However, it is a challenge for the residents to visualize how these appliances are used. Thus, mining algorithms are much needed to discover appliance usage patterns. Most previous studies on usage pattern discovery are mainly focused on analyzing the patterns of single appliance rather than mining the usage correlation among appliances. In this paper, a novel system, namely, *Correlation Pattern Mining System* (*CPMS*), is developed to capture the usage patterns and correlations among appliances. With several new optimization techniques, CPMS can reduce the search space effectively and efficiently. Furthermore, the proposed algorithm is applied on a real-world dataset to show the practicability of correlation pattern mining.**

Keywords- correlation pattern; smart home; sequential pattern; time interval-based data; usage representation

I. INTRODUCTION

Recently, many researchers focus on the reduction of electricity usage in residence because it is a significant contributor of greenhouse gas (GHG) emissions. However, electricity conservation is an arduous task for the residential users due to the lack of detailed electricity usage. Due to the advance of sensor technology, the electricity usage data of in-house appliances can be collected easily. In particular, an increasing number of smart power meters, which facilitates data collection of appliance usage, have been deployed. With the usage data, one could supposedly visualize how the appliances are used.

With an anticipated huge amount of appliance usage data, subtle information may exist but hidden. Therefore it is necessary to devise data mining algorithms to discover appliance usage patterns in order to make representative usage behavior of appliances explicit. Appliance usage patterns can not only help users to better understand how they use the appliances at home but also detect abnormal usages of appliances. Moreover, it facilitates appliance manufacturers to design intelligent control of smart appliances.

Many prior studies discuss how to extract useful knowledge regarding usage patents of a single appliance via energy disaggregation [3, 6, 7, 13, 16, 24] or appliance recognition [2, 5, 8, 12, 18, 22, 24]. Farinaccio et al. [6] use some patterns, such as number of *ON-OFF* switches, to disaggregate the whole-house electricity consumption into a number of major end-uses. Suzuki et al. [24] use a new NIALM technique based on integer programming to disaggregate residential power use. Lin et al. [16] use a dynamic Bayesian network and filter to disaggregate the data online. Kim et al. [13] 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.

Prudenzi [22] utilize an artificial neural network based procedure for identifying the electrical signatures of residential appliances. Ito et al. [8] extract features from the current (e.g., amplitude, form, timing) to develop appliance signatures. For appliance recognition, Kato et al. [12] 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. Matthews et al discuss some of these works and characterize workable solutions [18]. Chen et al. [5] introduce two types of usage patterns to describe users' representative behaviors. Based on these two types of patterns, an intelligent system, Jakkula et al. [9, 10, 11] propose an Apriori-based algorithm for activity prediction and anomaly detection from sensor data in a smart home. HAUBA [5], is developed to analyze the usage status of all appliances in a smart home environment.

All aforementioned 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, while the night, 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.

Moreover, it is difficult to discover useful knowledge from a huge set of generated patterns. Too many patterns sometimes hinder users from understanding their actual

978-0-7695-5109-8/13 $31.00 © 2013 IEEE
DOI 10.1109/ICDMW.2013.15

behaviors. Hence, we aim to derive compact and meaningful patterns in this study.

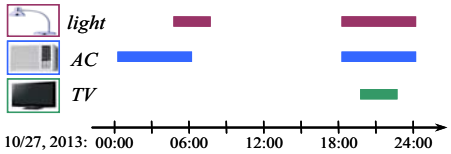


Fig. 1: An example of daily usage sequence.

So far, little attention has been paid to the issue of mining correlation among appliances, which undoubtedly is more complex and arduous than mining the usage patterns of an appliance alone, and thus requires new mining techniques. In this paper, a new framework fundamentally different from previous work is proposed to discover the usage correlation patterns. The contributions of our work are as follows:

- We define the notion of *correlation pattern* based on time interval-based sequence. Since the usage of a device can be regarded as a usage interval (time duration between *turn-on* and *turn-off*), interval-based sequence can depict users' daily behaviors unambiguously.
- The relation between any two usage intervals is intrinsically complex. This complex relation is really crucial for designing a correlation pattern mining algorithm with high efficiency and effectiveness, since it may lead to more candidate sequences and heavier workload for computing the support. We propose a method, called *usage representation*, to simplify the processing of complex relations among intervals by considering the global information of intervals in the sequence.
- We develop an intelligent system, called *Correlation Pattern Mining System* (abbreviated as ***CPMS***), to capture the usage patterns implying the correlations among appliances with several optimized techniques to reduce the search space effectively.
- To demonstrate the practicability of correlation pattern mining, we apply CPMS on a real dataset and analyze the results to show the discovered patterns are not just an anecdote.

The rest of the paper is organized as follows. Section 2 provides the preliminaries. Section 3 introduces the proposed CPMS system. Section 4 reports the experimental results in a performance study, and finally Section 5 concludes the paper.

II. PRELIMINARIES

Definition 1 (Usage-interval and usage-interval sequence) Let $A = \{a_1, a_2, \ldots, a_k\}$ be a set of k appliances. Let the triplet $(a_i, o_i, f_i) \in A \times N \times N$ denote a usage-interval of a_i, where $a_i \in A$, $o_i, f_i \in N$ and $o_i < f_i$. The two time points o_i, f_i denote the *using times*, where o_i and f_i are the turn-on time and the turn-off time of appliance a_i, respectively. The set of all usage-intervals over A is denoted by I. A usage-interval sequence is a series of usage-intervals $\langle(a_1, o_1, f_1), (a_2, o_2, f_2), \ldots, (a_n, o_n, f_n)\rangle$, where $o_i \le o_{i+1}$, and $o_i < f_i$.

Take Fig. 1 as an example. Suppose there are three appliances, *light*, *air conditioner* (*AC*), *television* (*TV*). Each appliance has its interior location in the house. (*light*, 18:00, 24:00) is a usage-interval and ⟨(*AC*, 00:00, 06:00), (*light*, 05:00, 08:00), (*light*, 18:00, 24:00), (*AC*, 18:00, 24:00), (*TV*, 20:00, 22:00)⟩ is a daily usage-interval sequence on Oct. 27, 2013.

Definition 2 (Usage-interval database) Considering a database $DB = \{r_1, r_2, \ldots, r_m\}$, each record r_i, where $1 \le i \le m$, consists of a date, a usage-interval (i.e. an appliance symbol, a turn-on time, and a turn-off time, where turn-on time < turn-off time). *DB* is called a *usage-interval database*. If all records in *DB* with the same date are grouped together and ordered by nondecreasing turn-on time, turn-off time and appliance symbol, actually, *DB* can be transformed into a collection of daily usage-interval sequences. Fig. 2 shows a usage database which consists of 17 usage intervals and 4 daily usage-interval sequences.

Processing usage-interval sequence is a difficult task. Since the relation among usage intervals is intrinsically complex. Allen's 13 temporal logics [1], in general, can be adopted to describe the relations among intervals. However, Allen's logics are binary relation. When describing relationships among more than three intervals, it may suffer several problems. In this paper, we modify the coincidence representation [4] and propose a new expression, called *usage representation*, to address the ambiguous and scalable problem of Allen's temporal logics.

Given a usage-interval sequence $Q = \langle(a_1, o_1, f_1), (a_2, o_2, f_2), \ldots, (a_n, o_n, f_n)\rangle$, The set $TS_Q = \{o_1, f_1, o_2, f_2, \ldots, o_i, f_i, \ldots, o_n, f_n\}$ is a *time set corresponding to Q*, where $1 \le i \le n$. If we order all the elements of TS_Q in nondecreasing order, we can derive a sequence $T_Q = \langle t_1, t_2, \ldots, t_{2n}\rangle$ where $t_i \in TS_Q$, $t_i \le t_{i+1}$. T_Q is called a *time sequence corresponding to Q*.

Definition 3 (Usage-point and usage sequence) Given a usage-interval sequence $Q = \langle(a_1, o_1, f_1), (a_2, o_2, f_2), \ldots, (a_i, o_i, f_i), \ldots, (a_n, o_n, f_n)\rangle$, where $(a_i, o_i, f_i) \in I$ and corresponding $T_Q = \langle t_1, t_2, \ldots, t_j, \ldots, t_{2n}\rangle$, a function Φ that maps a usage interval (a_i, o_i, f_i) into two usage-points a_i^+ and a_i^- is defined as follows.

$$\Phi(t_j, Q) = \begin{cases} a_i^+ & \text{if } t_j = o_i \\ a_i^- & \text{if } t_j = f_i \end{cases}, \qquad (1)$$

where a_i^+ and a_i^- are called *on*-point and *off*-point of interval (a_i, o_i, f_i), respectively. The usage-points $a_k^*, \ldots, a_\ell^*$ (* can be $^+$ or $^-$) are collected in brackets as a pointset if they occur at the same time in T_Q, denoted as $(a_k^*, \ldots, a_\ell^*)$. A usage sequence S_Q of Q is denoted by $\langle s_1, \ldots, s_i, \ldots, s_{2n}\rangle$ where s_i is a usage-point.

For example, in Fig. 2, the database collects 4 daily usage-interval sequences. The usage sequence of date 2 is $\langle B^+B^-D^+(E^+F^+)(E^-F^-)D^-\rangle$, and (E^+F^+) and (E^-F^-) are two pointsets because they occur at the same time, respectively.

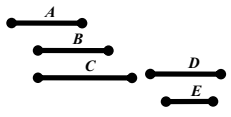
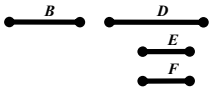
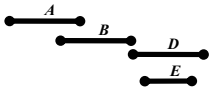
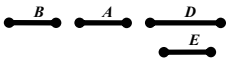
date	appliance symbol	turn-on time	turn-off time	pictorial example	usage representation (usage sequence)
1	*A*	02:10	07:30		$\langle A^+ (B^+ C^+) A^- B^- C^- D^+ E^+ E^- D^- \rangle$
1	*B*	05:20	10:00		
1	*C*	05:20	12:30		
1	*D*	16:10	22:40		
1	*E*	18:00	20:00		
2	*B*	00:40	05:30		$\langle B^+ B^- D^+ (E^+ F^+) (E^- F^-) D^- \rangle$
2	*D*	08:00	14:00		
2	*E*	10:20	13:10		
2	*F*	10:20	13:10		
3	*A*	06:00	12:20		$\langle A^+ B^+ A^- (B^- D^+) E^+ E^- D^- \rangle$
3	*B*	07:20	14:00		
3	*D*	14:00	20:30		
3	*E*	17:30	19:00		
4	*B*	08:30	10:00		$\langle B^+ B^- A^+ A^- D^+ E^+ E^- D^- \rangle$
4	*A*	13:20	16:00		
4	*D*	20:00	23:30		
4	*E*	21:30	22:40		

Fig. 2: An example of usage database.

Definition 4 (Usage representation) Given a usage-interval sequence $Q = \langle (a_1, o_1, f_1), \ldots, (a_n, o_n, f_n) \rangle$ and corresponding time sequence $T_Q = \langle t_1, \ldots, t_i, \ldots, t_{2n} \rangle$, by Definition 3, we can derive a usage sequence $S_Q = \langle s_1, \ldots, s_i, \ldots, s_{2n} \rangle$. S_Q is also called the usage representation of Q.

By Definition 4, we can transfer a usage-interval database into usage representation. Take the database in Fig. 2 as an example. The usage representation of *DB* is shown in the last column in Fig. 2. For the rest of this paper, we assume the usage database has already been transformed into usage representation.

Let $S_1 = \langle x_1, \ldots, x_i, \ldots, x_n \rangle$ and $S_2 = \langle x_1', \ldots, x_j', \ldots, x_m' \rangle$ be two usage sequences, where x_i, x_j' are pointsets and $n \le m$. S_1 is called a subsequence of S_2, denoted as $S_1 \sqsubseteq S_2$, if there exist integers $1 \le k_1 \le k_2 \le \ldots \le k_n \le m$ such that $x_1 \subseteq x_{k_1}', x_2 \subseteq x_{k_2}', \ldots, x_n \subseteq x_{k_n}'$. Given a usage-interval database *DB* in usage representation, the tuple $(date, S) \in DB$ is said to contain a usage sequence S' if $S' \sqsubseteq S$. The support of a usage sequence S' in *DB*, denoted as $support(S')$, is the number of tuples in the database containing S'. More formally,

$$support(S') = |\{ (date, S) \in DB \mid S' \sqsubseteq S \}|. \quad (2)$$

Obviously, the support count decides the significance of a usage sequence. We use a support threshold, *min_sup*, to filter out insignificant usage sequences. A usage sequence $S = \langle s_1, \ldots, s_n \rangle$ in *DB* is called a *frequent sequence*, if $support(S) \ge min\_sup$.

Definition 5 (Correlation pattern) Given a usage-interval database *DB* in usage representation and a threshold, *min_sup*, a usage sequence is called frequent if its support is no less than *minsup*. A frequent usage sequence is called a ***correlation pattern*** if all usage-points in the sequence appear in pairs, i.e., every on(off)-point has a corresponding off(on)-point.

Again, take the database in Fig. 2 as an example. Given $min\_sup = 2$, $\langle A^+ B^+ A^- B^- \rangle$ is a correlation pattern since its support is $3 \ge 2$ and all usage-points in sequence appear in pairs. However, although $\langle A^+ B^+ A^- \rangle$ is a frequent usage sequence, it is not a correlation pattern due to on-point B^+ having no corresponding off-point.

III. CPMS SYSTEM

We focus our study on correlation pattern mining in smart home due to its wide applicability and the lack of research on this topic. In this paper, we develop a new system, called *Correlation Pattern Mining System* (abbreviated as **CPMS**), to discover correlation patterns effectively and efficiently. CPMS utilizes the arrangement of endpoints to accomplish the mining of correlation among appliances' usage. We also propose two pruning strategies to effectively reduce the search space and speedup the mining process. In Section 3.1, we discuss several advantages of usage representation. In Section 3.2, we detail the mining system and proposed pruning mechanisms.

3.1 Advantages of usage representation

Extracting correlation patterns from data collected in smart homes can provide resident useful information to better understand the relation among usage of appliances. Given a correlation pattern, as defined in Definition 5, a user can know the relation between appliances.

Consider the correlation pattern $\langle A^+ B^+ A^- B^- \rangle$ in aforementioned example. Suppose appliances *A* and *B* are the light and the coffee machine, respectively. Given the correlation of light and coffee machine, we can know the relation between them. This information is very useful for several applications, such as abnormal detection and activity prediction. For example, a user forgets to turn off the coffee machine when she goes to supermarket. The home management system (HMS) detects that the coffee machine is still turn-on at a time. Sine the pattern represents the representative behavior (i.e., turning off coffee machine after turning off light), the probability that coffee machine is still on is very low. Thus, the HMS sends a message to the user's smart phone to notify this anomaly. Activity prediction also can be realized by discovering correlation patterns. From the

example pattern, we can observe that the coffee machine is usually turned on after the light is turned on. If we detect the light is turned on at a given time, the HMS may automatically turns on the coffee machine if probability of the aforementioned correlation pattern is high.

temporal relation	inversed relation	pictorial example	usage representation
A before B	*B after A*	A B	$A^+A^-B^+B^-$
A overlaps B	*B overlapped-by A*	A B	$A^+(A^-B^+)\,B^-$
A contains B	*B during A*	A B	$A^+B^+B^-A^-$
A starts B	*B started-by A*	A B	$(A^+B^+)\,B^-A^-$
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	$(A^+B^+)\,(A^-B^-)$

Fig. 3: The usage representation of Allen's relations between two intervals.

Obviously, the correlation pattern mining is an arduous task. Since the time period of the two usage-intervals may overlap, the relation between them is intrinsically complex. Allen's 13 temporal logics [1], in general, can be adopted to describe the relations among intervals, as shown in Fig. 3. However, Allen's logics are binary relations. When describing relationships among more than three intervals, Allen's temporal logics may suffer several problems.

A suitable representation is very important for describing a correlation pattern. In this paper, a new expression, called *usage representation*, is proposed to effectively address the ambiguous and scalable issue [25] for describing relationships among intervals. Given two different usage-intervals A and B, the usage representation of Allen's 13 relations between A and B is categorized as in Fig. 3. Several merits of usage representation are discussed as follows,

- **Compactness:** Since each usage-interval has two usage-points, we only use $2k$ space for expressing a k-interval sequence. The usage representation scales well even with plenty of usage-intervals appearing in a sequence.
- **Nonambiguity:** According to [25], we can find that the usage representation has no ambiguous problem. First, by Definition 3, we can transform every usage-interval sequence to a unique usage sequence. In other words, the temporal relations among intervals can be mapped to a usage sequence. Second, in a usage sequence, the order relation of the starting and finishing endpoints of A and B can be categorized as shown in Fig. 3. Hence, we can infer the original temporal relationships between intervals A and B nonambiguously.
- **Simplicity:** Obviously, the complex relations between intervals are the major bottleneck of correlation pattern mining. However, the relation between two usage points is simple, just "*before*," "*after*" and "*equal*." The simpler the relations, the less number of intermediate candidate sequences are generated and processed.

3.2 CPMS

Before introducing the proposed system, we modify the idea in [21] and define the projected database first. Let α be a usage sequence in a database DB with usage representation. The α - projected database, denoted as $DB_{|\alpha}$, is the collection of postfixes of sequences (including usage sequences and corresponding time sequence) in DB with regards to prefix α.

Fig. 4 shows the system framework of CPMS. We first collect the usage data of all appliances from the attached smart meters and store them in cloud server (i.e., usage database). Then we extract the usage patterns implying the correlations among appliances from usage database. Algorithm 1 illustrates the correlation pattern mining algorithm, CoPMiner, of CPMS.

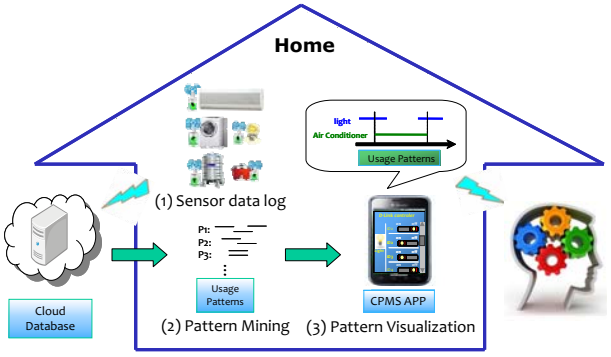


Fig. 4: The system framework of CPMS.

We first transform the usage database to usage representation and calculates the count of each usage-point concurrently (line 2, algorithm 1). CoPMiner removes infrequent usage-points under given support threshold, *min_sup* (line 3, algorithm 1). Then, for each frequent usage-point, CoPMiner constructs the projected database and calls ***UPrefixSpan*** recursively to obtain all correlation patterns (lines 5-7, algorithm 1). Note that we only consider the on-point here (line 4, algorithm 1). Finally, we output all discovered correlation patterns.

Algorithm 1: CoPMiner (*DB*, *min_sup*, *min_sim*)

Input: a usage-interval database *DB*, the support threshold *min_sup*, the similarity threshold *min_sim*
Output: all correlation patterns *P*

01: $P \leftarrow \varnothing$;
02: transform *DB* into usage presentation by Definition 4;
03: find all frequent usage-points and remove infrequent usage-points in *DB*;
04: *FS* ← all frequent "**on-points**";
05: **for** each $s \in FS$ **do**
06: construct $DB_{|s}$;
07: ***UPrefixSpan***($DB_{|s}$, *s*, *min_sup*, *P*);
08: output all correlation patterns *P*;

By borrowing the idea of the PrefixSpan [21], UPrefixSpan is developed with two search space pruning methods. The pseudo code is shown in Algorithm 2. For a prefix α, UPrefixSpan scans its projected database $DB_{|\alpha}$ once to discover all local frequent usage-points and remove infrequent ones (line 1, algorithm 2). For frequent usage-point s, we can append it to original prefix to generate a new frequent sequence α' with the length increased by 1. As such, the prefixes are extended (lines 3-4, algorithm 2). If all usage-points in a frequent sequence appear in pairs, i.e., every on(off)-point has corresponding off(on)-point, we can output this frequent sequence as a correlation pattern (lines 5-6, algorithm 2). Finally, we can discover all correlation patterns by constructing the projected database with the frequently extended prefixes and recursively running until the prefixes cannot be extended (lines 7-8, algorithm 2).

Algorithm 2: UPrefixSpan ($DB_{|\alpha}$, α, *min_sup*, P)

Input: a projected database $DB_{|\alpha}$, an usage sequence α, the support threshold *min_sup*, and a set of correlation patterns P
Output: a set of correlation patterns P

01: scan $DB_{|\alpha}$ once, remove infrequent usage-points and find every frequent usage-point v such that:
(i) v can be assembled to the last pointset of α to form a frequent sequence; or
(ii) $\langle v \rangle$ can be appended to α to form a frequent sequence;
01: $FS \leftarrow$ all frequent usage-points;
02: $FS \leftarrow$ ***point_pruning***(FS, α); // **point-pruning strategy**
03: **for** each $s \in FS$ **do**
04: append s to α to form α';
05: **if** α' is a correlation pattern **then**
06: $P \leftarrow P \cup \alpha'$;
07: $DB_{|\alpha'} \leftarrow$ ***DB_construct***($DB_{|\alpha}$, α'); // **prefix-pruning strategy**
08: ***UPrefixSpan***($DB_{|\alpha'}$, α', *min_sup*, P);

Procedure *point_pruning* (FS, α)
09: $temp\_point \leftarrow \varnothing$;
10: **for** each $s \in FS$ **do**
11: **if** s is a "**off-point**" **then** // **point-pruning strategy**
12: **if** exist corresponding "**on-point**" in α **then**
13: $temp\_point \leftarrow temp\_point \cup s$;
14: **if** s is a "**on-point**" **then**
15: $temp\_point \leftarrow temp\_point \cup s$;
16: **return** *temp_point*;

Procedure *DB_construct* ($DB_{|\alpha}$, α')
17: $temp\_seq \leftarrow \varnothing$;
18: find all postfix sequences of α' in $DB_{|\alpha}$ to form $DB_{|\alpha'}$;
19: **for** each postfix sequence $q \in DB_{|\alpha'}$ **do**
20: eliminate the "off-points" in q which has no corresponding "on-point" in α'; // **prefix-pruning strategy**
21: $temp\_seq \leftarrow temp\_seq \cup q$;
22: **return** *temp_seq*;

Taking into account the property of usage-point, we propose two pruning strategies, point-pruning and postfix-pruning to reduce the searching space efficiently and effectively. Firstly, the on-points and the off-points definitely occur in pairs in a usage sequence. We only require projecting the frequent on-points or the frequent off-points which have the corresponding on-points in their prefixes. For example, if we scan the projected database $DB_{|\langle A^+ \rangle}$ with respective to prefix $\langle A^+ \rangle$ and find three frequent local usage-points, A^-, B^+ and B^-. We only require extending prefix $\langle A^+ \rangle$ with A^- and B^+ (i.e., $\langle A^+A^- \rangle$ and $\langle A^+B^+ \rangle$), since B^- has no corresponding on-points in its prefix. It is because that sequence $\langle A^+B^- \rangle$ has no chance to grow to a frequent sequence. This strategy is called **point-pruning strategy** (line 2 and lines 9-16, algorithm 2) which can prune off non-qualified patterns before constructing projected database

Second, when we construct a projected database, some usage-points in postfix sequences need not be considered. With respect to a prefix sequence $\langle \alpha \rangle$, an off-point in a projected postfix sequence is insignificant, if it has no corresponding on-points in $\langle \alpha \rangle$. Hence, when collecting postfix sequences to construct $DB_{|\langle \alpha \rangle}$, we can eliminate all insignificant off-points since they can be ignored in the discovery of correlation patterns. This pruning method is called **postfix-pruning strategy** which can shrink the length of postfix sequence and further reduce the size of projected database effectively (line 7 and lines 17-22, algorithm 2).

IV. Experimental Result

To best of our knowledge, CPMS is the first method discussing the usage pattern implying the correlation among appliances. For performance discussion, we compare the mining algorithm of CPMS (i.e., CoPMiner) with three interval-pattern mining algorithms, *CTMiner* [4], *IEMiner* [20] and *TPrefixSpan* [25]. All algorithms were implemented in Java language and tested on a workstation with Intel i7-3370 3.4 GHz with 8 GB main memory. The performance study has been conducted on both synthetic and real world datasets. First, we compare the execution time using synthetic datasets at different minimum support. Second, we conduct an experiment to observe the memory usage and the scalability on execution time of CoPMiner. Finally, CoPMiner is applied in real-world dataset to show the performance and the practicability of mining correlation patterns.

The synthetic datasets in the experiments are generated using synthetic generation program modified from [21]. Since the original data generation program was designed to generate time point-based data, the generator for correlation pattern mining algorithm requires modifications on interval events accordingly. The parameter setting of usage data generator is shown in Fig. 5.

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

Fig. 5: Parameters of synthetic data generator

We create a set of potentially frequent sequences used in the generation of event sequences. The number of potentially frequent sequences is N_S. A potentially frequent sequence is generated by first picking the size of sequence from a Poisson distribution with mean equal to $|S|$. Then, the event intervals in potentially frequent sequence are chosen from N event symbols randomly. All the duration times of event intervals are classified into three categories: long, medium and short, which are normally distributed with an average length of 12, 8 and 4, respectively.

For each event interval, we first randomly decide its category and then determine its length by drawing a value. The temporal relations between consecutive intervals are selected randomly to form a potentially frequent sequence. Since we adopt normalized temporal patterns [19], the temporal relationships can be chosen from the set {*before*, *meets*, *overlaps*, *is-finished-by*, *contains*, *starts*, *equal*}. After all potentially frequent sequences are determined, we generate $|D|$ event sequences. Each event sequence is generated by first deciding the size of sequence, which was picked from a Poisson distribution with mean equal to $|C|$. Then, each event sequence is generated by assigning a series of potentially frequent sequences. Finally, we assign the on-time of each usage-interval with discrete uniform distribution on {1, 2, …, 100}. The off-time is the on-time plus the interval length. The location information attached to each appliance is uniformly chosen on {1, …, 10}×{1, …, 10}×{1, 2, 3}.

4.1 Performance and Scalibility on Synthetic Datasets

In all the following experiments, two parameters are fixed, i.e., $|S| = 4$ and N_S = 5,000. The other parameters are configured for comparison. Fig. 6 shows the running time of the four algorithms with minimum supports varied from 1% to 5% on the dataset *D*10k–*C*20–*N*1k. Obviously, when the minimum support value decreases, the processing time required for all algorithms increases. We can see that when we continue to lower the threshold, the runtime for IEMiner and TPrefixSpan increase drastically compared to CTMiner and CoPMiner. This is partly because these two algorithms still process interval-based data with complex relationship. The complex relationship may lead to generate more number of intermediate candidate sequences.

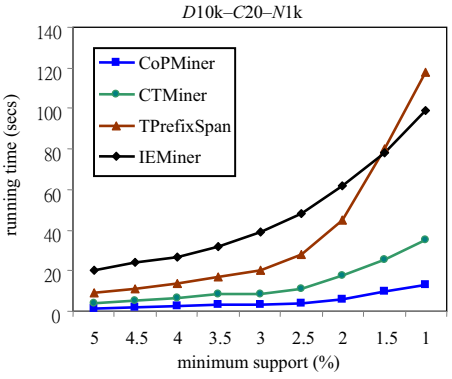


Fig. 6: Runtime performance testing on *D*10k–*C*20–*N*1k dataset

Fig. 7 shows the execution time of the four algorithms with minimum supports varied from 1% to 5% on the dataset *D*100k–*C*20–*N*10k, which is much larger since it contains 100,000 event sequences and 10,000 event intervals. From the figure, we can observe that CoPMiner has the best runtime performance. Note that, although CTMiner also simplify the complex relation among intervals, the segmentation strategy of representation consumes more processing time. On the contrary, the proposed usage presentation only requires capturing two endpoints of an interval. Furthermore, three pruning strategies also play an important role for the efficiency of CoPMiner. We will discuss these in details later.

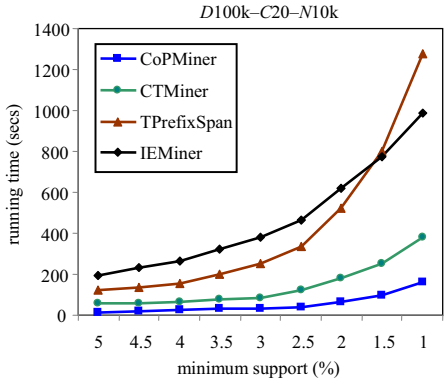


Fig. 7: Runtime performance testing on *D*100k–*C*20–*N*10k dataset

Then, we study the scalability of CoPMiner. Here, we use the data set C = 20, N = 10k with varying different database size. Fig. 8 shows the results of scalability tests of four algorithms with the database size growing from 100K to 500K sequences. We fix the min_sup as 1%. Fig. 9 depicts the results of scalability tests of CoPMiner under different database size growing with different minimum support threshold varying from 1% to 5%. As the size of database increases and minimum support decreases, the processing time of all algorithms increase, since the number of patterns also increases. As can be seen, CoPMiner is linearly scalable with different minimum support threshold. When the number of generated patterns is large, the runtime of CoPMiner still increases linearly with different database size.

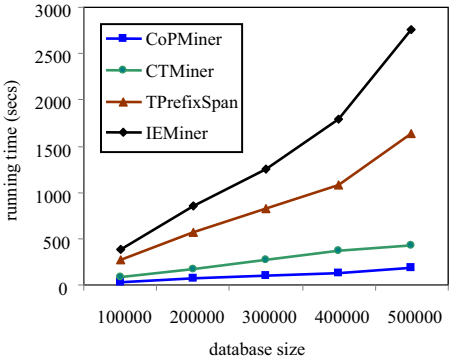


Fig. 8: Scalability testing of different algorithms on different database size

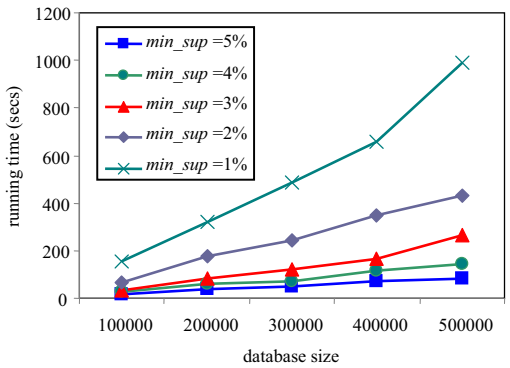


Fig. 9: Scalability testing of different *min_sup* on different database size

In summary, performance study shows that CoPMiner has the best overall performance among the algorithms tested. The scalability study also depicts that CoPMiner scales well even with large databases and low thresholds.

4.2 Influence of Proposed Pruning Strategies

To reflect the speedup of proposed pruning methods, we measure CoPMiner with pruning strategies and without pruning strategy on time performance. We compare four algorithms, *CoPMiner* (includes all pruning strategies), *CoP_Point* (only point-pruning strategy), *CoP_Postfix* (only postfix-pruning strategy) and *CoP_None* (without any pruning strategy). The experiment is performed on the data set *D*100k–*C*20–*N*10k. Fig. 10 is the results of varying minimum support thresholds from 0.5% to 1%. As shown in figure, CoP_Point can improve 23.4% to 27.9% of the performance of CoP_None. That means point-pruning can improve about 25% performance of CoPMiner. Because of removing non-qualified usage-points before database projection, point-pruning can efficiently speedup the execution time.

The impact of the postfix-pruning is also presented in Fig. 10. As can be seen from the graph, postfix-pruning can improve about 11% performance of *CoP_None*. We can find that postfix-pruning can improve the performance of CoPMiner by effectively eliminating all useless usage-points for correlation pattern construction.

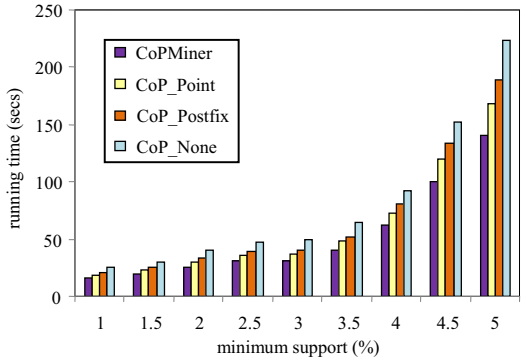


Fig. 10: Influence testing of three pruning strategies on different *min_sup*

In summary, the pruning strategies constantly improve 32% runtime performance of CoPMiner. Consequently, the proposed pruning strategies not only effectively reduce the searching space but also efficiently improve the performance of CoPMiner.

4.3 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 [14] 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. 11 lists the information of six different houses in REDD dataset after transformation.

house	number of appliances	number of sequences	number of intervals
1	16	36	1107
2	8	15	536
3	19	26	1361
4	17	31	1655
5	19	10	179
6	11	19	526

Fig. 11: Information of six different houses in REDD dataset

Fig. 12 shows the part of mining result with *min_sup* = 0.3. Obviously, from the proposed CPMS system, residents can know the correlation among the appliances in their house easily. This useful information can not only help users to better understand how they use the appliances at home but also detect abnormal usages of appliances.

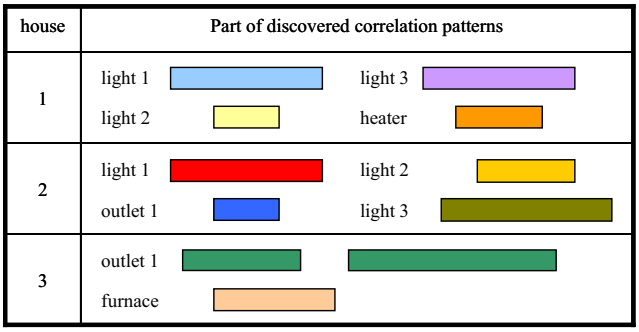


Fig. 12: Part of discovered correlation patterns from REDD dataset

V. CONCLUSION

Recently, considerable concern has arisen over the electricity conservation due to the issue of greenhouse gas emissions. If representative behaviors of appliance usages are available, residents may adapt their usage patterns to conserve energy effectively. However, previous studies on usage pattern discovery are mainly focused on analyzing single appliance and ignore the usage correlation among appliances. In this paper, we introduce a new concept,

correlation pattern, to capture the representative usage behaviors implying correlations among appliances. An intelligent system, CPMS is developed to discover patterns based on proposed usage representation. We also introduce several pruning strategies to improve the performance of the mining algorithm. The experimental studies indicate that CPMS is efficient and scalable. Furthermore, CPMS is applied on a real-world dataset to show the practicability of correlation pattern mining.

ACKNOWLEDGE

Wen-Chih Peng was supported in part by the National Science Council, Project No. 100-2218-E-009-016-MY3 and 100-2218-E-009-013-MY3.

REFERENCES

[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. *Proceedings of 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. *Proceedings of 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*, pp. 240-248, 2011.

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

[5] 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.

[6] L. Farinaccio and R. Zmeureanu. Using a Pattern Recognition Approach to Disaggregate the Total Electricity Consumption in a House into the Major End-uses. *Energy and Buildings*, vol. 30, no. 3, pp. 245-259, 1999.

[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. Learning Temporal Relations in Smart Home Data. *Proceedings of the Second International Conference on Technology and Aging*, 2007.

[10] 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.

[11] V. Jakkula, D. Cook and A. Crandall. Temporal pattern discovery for anomaly detection in a smart home. *Proceedings of the 3rd IET Conference on Intelligent Environments (IE'07)*, pp. 339-345, 2007.

[12] 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.

[13] 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.

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

[15] P. Liao, T. Chen and P. Chung. A Fast Algorithm for Multilevel Thresholding. *Journal of Information Science and Engineering, Institute of Information Science, Academia Sinica, 17*, pp. 713-727, 2001.

[16] 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.

[17] B. Liu, Y. Yang, G. Webb and J. Boughton. A Comparative Study of Bandwidth Choice in Kernel Density Estimation for Naive Bayesian Classification. *13th Pacific-Asia Conference in Knowledge Discovery and Data Mining, Advances in Knowledge Discovery and Data Mining, (PAKDD'09)*, pp. 302-313, 2009.

[18] 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.

[19] P. Papapetrou, G. Kollios, S. Sclaroff and D. Gunopulos. Discovering Frequent Arrangements of Temporal Iintervals. *International Conference on Data Mining (ICDM'05)*, pp. 354-361, 2005.

[20] 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.

[21] 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.

[22] A. Prudenzi. A Neuron Nets Based Procedure for Identifying Domestic Appliances Pattern-of-use from Energy Recordings at Meter Panel. *IEEE Power Engineering Society Winter Meeting*, vol. 2, pp.491-496, 2002.

[23] B. Silverman. Density Estimation for Statistics and Data Analysis. *CHAPMAN and HALL*, 1986.

[24] 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.

[25] 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.