# Mining Correlation Patterns among Appliances in Smart Home Environment

Yi-Cheng Chen[1], Chien-Chih Chen[2], Wen-Chih Peng[2], and Wang-Chien Lee[3]

[1] Department of Computer Science and information engineering, Tamkang University, Taiwan
[2] Department of Computer Science, National Chiao Tung University, Taiwan
[3] Department of Computer Science and Engineering, The Pennsylvania State University, USA
ycchen@mail.tku.edu.tw, {flykite,wcpeng}@cs.nctu.edu.tw,
wlee@cse.psu.edu

**Abstract.** Since 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 algorithm, namely, *Correlation Pattern Miner* (*CoPMiner*), is developed to capture the usage patterns and correlations among appliances probabilistically. With several new optimization techniques, CoPMiner 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.

## 1 Introduction

Recently, 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, residents could supposedly visualize how the appliances are used. Nonetheless, 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 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.

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, while the night, air conditioner and

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 222–233, 2014.

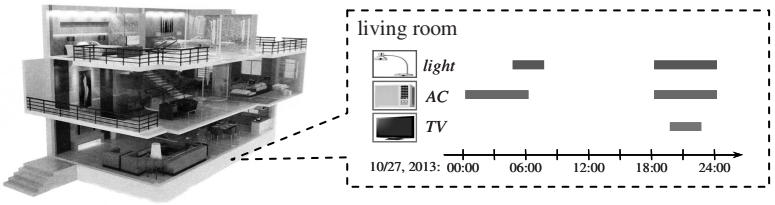



**Fig. 1.** An example of daily usage sequence

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.

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: (1) We define the notion of *correlation pattern* based on time interval-based sequence including probability concept. Since the usage of a device can be regarded as a usage interval (duration between *turn-on* and *turn-off*), interval-based sequences can depict users' daily behaviors unambiguously. (2) The relation between any two usage intervals is intrinsically complex which may lead to more candidate sequences and heavier workload for computation. 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. (3) We develop an efficient algorithm, called *Correlation Pattern Miner* (abbreviated as ***CoPMiner***), to capture the usage patterns implying the correlations among appliances with several optimized techniques to reduce the search space effectively. (4) The readability of patterns is also an essential issue. A large number of patterns may become an obstacle for users to understand their actual behaviors. A spatial constraint is introduced to prune off non-promising correlation and reduce the number of generated correlation patterns. (5) To demonstrate the practicability of correlation pattern mining, we apply CoPMiner 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. Sections 2 and 3 provide the related works and preliminaries, respectively. Section 4 introduces the proposed CoPMiner algorithm. Section 5 reports the experimental results in a performance study, and finally Section 6 concludes the paper.

## 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, 6, 11, 13, 18] or

appliance recognition [2, 5, 7, 10, 18]. Suzuki et al. [18] use a new NIALM technique based on integer programming to disaggregate residential power use. Lin et al. [13] 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. [6] 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. [7] 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. [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. [8, 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 instead of the correlation among appliances in a house. In this paper, we propose a mining algorithm to extract patterns including correlation among appliances and probability concept.

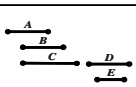
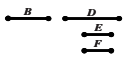
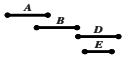
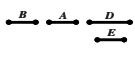
| date | appliance symbol | turn-on time | turn-off time | interior location | pictorial example | usage representation (usage sequence, time sequence) |
|---|---|---|---|---|---|---|
| 1 | *A* | 02:10 | 07:30 | (1, 1, 1) |  | $\begin{pmatrix} A^+ & (B^+ & C^+) & A^- & B^- & C^- & D^+ & E^+ & E^- & D^- \\ 2 & 5 & 5 & 7 & 10 & 12 & 16 & 18 & 20 & 22 \end{pmatrix}$ |
| 1 | *B* | 05:20 | 10:00 | (1, 2, 1) | | |
| 1 | *C* | 05:20 | 12:30 | (3, 4, 2) | | |
| 1 | *D* | 16:10 | 22:40 | (1, 3, 1) | | |
| 1 | *E* | 18:00 | 20:00 | (3, 4, 1) | | |
| 2 | *B* | 00:40 | 05:30 | (1, 2, 1) |  | $\begin{pmatrix} B^+ & B^- & D^+ & (E^+ & F^+) & (E^- & F^-) & D^- \\ 0 & 5 & 8 & 10 & 10 & 13 & 13 & 14 \end{pmatrix}$ |
| 2 | *D* | 08:00 | 14:00 | (1, 3, 1) | | |
| 2 | *E* | 10:20 | 13:10 | (3, 4, 2) | | |
| 2 | *F* | 10:20 | 13:10 | (2, 2, 1) | | |
| 3 | *A* | 06:00 | 12:20 | (1, 1, 1) |  | $\begin{pmatrix} A^+ & B^+ & A^- & (B^- & D^+) & E^+ & E^- & D^- \\ 6 & 7 & 12 & 14 & 14 & 17 & 19 & 20 \end{pmatrix}$ |
| 3 | *B* | 07:20 | 14:00 | (1, 2, 1) | | |
| 3 | *D* | 14:00 | 20:30 | (1, 3, 1) | | |
| 3 | *E* | 17:30 | 19:00 | (3, 4, 1) | | |
| 4 | *B* | 08:30 | 10:00 | (1, 2, 1) |  | $\begin{pmatrix} B^+ & B^- & A^+ & A^- & D^+ & E^+ & E^- & D^- \\ 8 & 10 & 13 & 16 & 20 & 21 & 22 & 23 \end{pmatrix}$ |
| 4 | *A* | 13:20 | 16:00 | (1, 1, 1) | | |
| 4 | *D* | 20:00 | 23:30 | (1, 3, 1) | | |
| 4 | *E* | 21:30 | 22:40 | (3, 4, 1) | | |

**Fig. 2.** An example of usage database

## 3 Preliminaries

**Definition 1 (Usage-interval and usage-interval sequence).** Let $A = \{a_1, a_2, \ldots, a_k\}$ be a set of $k$ appliances. Without loss of generality, we define a set of uniformly spaced location and time points based on natural numbering $N$. A function, $Loc$: $A \rightarrow N^3$, specifies the location of each appliance in $A$. 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. 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 $o_i < f_i$. $Loc(a_i)$ is the interior location of appliance $a_i$ in a smart home environment.

**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 and an interior location of appliance. $DB$ 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. Note that the location information can be viewed as attachment to appliances. Fig. 2 shows a usage database which consists of 17 usage intervals and 4 daily usage-interval sequences.

**Definition 3 (Usage-point and usage sequence).** Given a usage-interval sequence $Q = \langle(a_1, o_1, f_1), (a_2, o_2, f_2), \ldots, (a_n, o_n, f_n)\rangle$, the set $TS_Q = \{o_1, f_1, o_2, f_2, \ldots, o_i, f_i, \ldots, o_n, f_n\}$ is called a *time set corresponding to* $Q$. By ordering 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* $Q$. A function $\Phi$ that maps a usage interval $(a_i, o_i, f_i)$ into two usage-points $a_i^+$ and $a_i^-$ is defined as follows,

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

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.

**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 $S_Q = \langle s_1, \ldots, s_i, \ldots, s_{2n}\rangle$. The usage representation of $Q$ is defined as a pair,

$$(S_Q, T_Q) = \begin{pmatrix} s_1 & \ldots & s_i & \ldots & s_{2n} \\ t_1 & \ldots & t_i & \ldots & t_{2n} \end{pmatrix}. \tag{2}$$

Note that the using time of usage point $s_i$ in $S_Q$ is $t_i$ in $T_Q$. Take the database in Fig. 2 as an example. Without leading into ambiguity, we consider the turn-on and turn-off times by hour. 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 $x_1 \subseteq x_{k1}', x_2 \subseteq x_{k2}', \ldots, x_n \subseteq x_{kn}'$. Given a usage-interval database *DB* in usage representation, the tuple $(date, S, T) \in DB$ 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, $support(S') = |\{ (date, S, T) \in DB \mid S' \sqsubseteq S \}|$. (3)

As mentioned above, each appliance in a house has its own location. For an appliance $a$ in $A$, the function, $Loc: A \to N \times N \times N$, gives the locations $(a_x, a_y, a_z)$ of $a$. The similarity between two appliances $a_1$ and $a_2$ is defined as follows:

$$similarity(a_1, a_2) = \begin{cases} 1 & \text{if } Loc(a_1) = Loc(a_2) \\ \dfrac{1}{Loc(a_1) - Loc(a_2)} & \text{if } Loc(a_1) \ne Loc(a_2) \end{cases}, \text{ where } Loc(a_1) - Loc(a_2) = |a_{1x} - a_{2x}| + |a_{1y} - a_{2y}| + |a_{1z} - a_{2z}|. \quad (4)$$

For example, in Fig. 2, the similarity of appliances *B* and *C* is $\frac{1}{2+2+1} = \frac{1}{5} = 0.2$. We use a support threshold, *min_sup* and *min_sim*, 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 $support(S) \ge min\_sup$ and $\forall\, s_i, s_j$ in $S$ where $i, j \le n$, $similarity(s_i, s_j) \ge min\_sim$.

**Definition 5 (Correlation pattern).** Given *DB* in usage representation and two thresholds, *min_sup* and *min_sim*, the set of frequent sequences, *FS*, includes all frequent usage sequences in *DB*. A correlation pattern *P* is defined as,

$$P = (S, f(S)) = \begin{pmatrix} s_1 & \ldots & s_i & \ldots & s_n \\ f_1 & \ldots & f_i & \ldots & f_n \end{pmatrix}, \text{ where } S = \langle s_1, \ldots, s_n \rangle \in FS \text{ and } f_i \text{ is the probability function of } s_i \text{ in } DB. \quad (5)$$

We modify the idea of multivariate kernel density estimation [14, 17] to estimate the probability function of each $s_i$ in $S$. Suppose the time information of $s_i$ in *DB* is $\{t_{i1}, t_{i2}, \ldots t_{im}\}$, the probability function is defined as,

$$f_i(x) = (K(x), h, \{t_{i1}, t_{i2}, \ldots, t_{im}\}) = \frac{1}{mh} \sum_{j=1}^{m} K\left(\frac{x - t_{ij}}{h}\right), \text{ where } K \text{ is Gaussian Normal,}$$

$$\text{i.e., } K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \text{ and } h = \frac{range(\{t_{i1}, t_{i2}, \ldots, t_{im}\})}{\sqrt{m}}. \quad (6)$$

For example, in Fig. 2, with *min_sup* = 2 and *min_sim* = 0.3, $\langle A+A-D-D- \rangle$ is a frequent sequence since its support is $3 \ge 2$ and similarity $(A, D) = 0.5 \ge 0.3$. The correlation pattern with respective to $\langle A+A-D-D- \rangle$ is $\begin{pmatrix} A^+ & A^- & D^+ & D^- \\ f_{A^+} & f_{A^-} & f_{D^+} & f_{D^-} \end{pmatrix}$.

We only discuss $f_{A+}$ as an example. The time information of $A+$ is {2, 6, 13}; hence $f_{A+}(x)$

$= \frac{1}{3h\sqrt{2\pi}}(e^{-\frac{1}{2}(\frac{x-2}{h})^2} + e^{-\frac{1}{2}(\frac{x-6}{h})^2} + e^{-\frac{1}{2}(\frac{x-13}{h})^2})$ with $h = \frac{range(\{2,6,13\})}{\sqrt{3}} = \frac{13-2}{\sqrt{3}} =$ 6.35.

## 4 Mining Appliance Usage Patterns

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 algorithm, called *Correlation Pattern Miner* (abbreviated as **CoPMiner**), to discover correlation patterns effectively and efficiently. CoPMiner utilizes the arrangement of endpoints to accomplish the mining of correlation among appliances' usage. We also propose four pruning strategies to effectively reduce the search space and speedup the mining process.

### 4.1 Merits of Correlation Pattern and 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 distribution of usage time of appliances. With a turn-on/off time of an appliance, we can derive the usage probability of other appliances. Consider the correlation pattern in aforementioned example. Suppose appliances *A* and *D* are the light and the coffee machine, respectively. Given the turn-on/off times of light and coffee machine, we can derive the usage probability for them, i.e., the probability for the light and coffee machine to be on/off at that time. This probability information is very useful for several applications, such as abnormal detection and activity prediction.

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. 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 [19] 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: (1) **Lossless:** Usage representation not only implies the temporal relation among intervals, but also includes the accurate usage time of each interval. This concept can achieve a lossless representation to express the nature of the interval sequence. (2) **Nonambiguity:** According to [19], 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 depicted easily. Hence, we can infer the original temporal relationships between intervals *A* and *B* nonambiguously. (3) **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.

### 4.2 CoPMiner Algorithm

Before introducing the algorithm, we modify the idea in [16] and define the projected database first. Let $\alpha$ be a usage sequence in a database *DB* with usage representation. The $\alpha$ - 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 $\alpha$.

Algorithm 1 illustrates the main framework of CoPMiner. It first transforms 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). For each frequent starting usage-point *s*, we find all its time information $\{t_{s1}, t_{s2}, \ldots t_{sm}\}$ in *DB* and estimate the probability function $f_s$ by Definition 5 (lines 6-7, algorithm 1).

**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 ← ∅;
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 ∈ FS do
06:     find all corresponding usage time information of s in DB;
07:     f_s ← calculate the probability function of s by Definition 5;
08:     construct DB_|s only with each usage-point v, where
        similarity(s, v) ≥ min_sim;   // spatial-pruning strategy
09:     UPrefixSpan(DB_|s, s, f_s, min_sup, P);
10: output all correlation patterns P;
```

As mentioned above, the spatial distance may conflict with the correlation dependency between two appliances. When building the projected database $DB_{|s}$, CoPMiner collects the postfixes by using **spatial pruning strategy**. We eliminate the usage-points which have the similarity with regard to *s* smaller than *min_sim* in collected postfix sequences (line 8, algorithm 1). Finally, CoPMiner calls UPrefixSpan recursively and output all correlation patterns (lines 9-10, algorithm 1).

By borrowing the idea of the PrefixSpan [16], UPrefixSpan is developed with two search space pruning methods. The pseudo code is shown in Algorithm 2. For a prefix $\alpha$, 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 $\alpha'$ with the length increased by 1. We also use the time information of $s$ in $DB_{/\alpha}$ to estimate the probability function $f_s$ by Definition 5, and then include $f_s$ into $f(\alpha')$. As such, the prefixes are extended (lines 3-7, 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 and its probability function as a correlation pattern (lines 8-9, 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 10-11, algorithm 2).

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 12-19, 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 14 and lines 20-25, algorithm 2).

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

Input: a projected database $DB_{|\alpha}$, an usage sequence $\alpha$, the support threshold *min_sup*, a similarity threshold *min_sim*, 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 $\alpha$ to form a frequent sequence; or
(ii) $\langle v\rangle$ can be appended to $\alpha$ to form a frequent sequence;
02: $FS \leftarrow$ all frequent usage-points;
03: $FS \leftarrow$ ***point_pruning***($FS$, $\alpha$); // **point-pruning strategy**
04: **for** each $s \in FS$ **do**
05: find all corresponding usage time information of $s$ in $DB_{|\alpha}$;
06: $f_s \leftarrow$ calculate the probability function of $s$ by Definition 5;
07: append $s$ to $\alpha$ to form $\alpha'$;
08: $f(\alpha') \leftarrow f(\alpha) + f_s$;
09: **if** $\alpha'$ is a correlation pattern **then**
10: $P \leftarrow P \cup (\alpha', f(\alpha'))$;
11: $DB_{|\alpha'} \leftarrow$ ***DB_construct***($DB_{|\alpha}$, $\alpha'$); // **prefix-pruning strategy**
12: ***UPrefixSpan***($DB_{|\alpha'}$, $\alpha'$, $f(\alpha')$, *min_sup*, $P$);

**Procedure *point_pruning* ($FS$, $\alpha$)**
13: $temp\_point \leftarrow \varnothing$;
14: **for** each $s \in FS$ **do**
15: **if** $s$ is a "**off-point**" **then** // **point-pruning strategy**
16: **if** exist corresponding "**on-point**" in $\alpha$ **then**
17: $temp\_point \leftarrow temp\_point \cup s$;
18: **if** $s$ is a "**on-point**" **then**
19: $temp\_point \leftarrow temp\_point \cup s$;
20: **return** $temp\_point$;

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

## 5 Experimental Results

To best of our knowledge, CoPMiner is the first algorithm discussing the correlation among appliances included probability concept. Three interval-pattern mining algorithms, *CTMiner* [4], *IEMiner* [15] and *TPrefixSpan* [19] have been implemented for performance discussion. For fair comparison, when comparing the execution time of CoPMiner with other interval-pattern mining algorithms, we only discuss the part of usage sequence mining (i.e., exclusive of computation of probability function). All algorithms were implemented in Java language and tested on a workstation with Intel i7-3370 3.4 GHz with 8 GB main memory. 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 [12] to show the performance and the practicability of mining correlation patterns. The synthetic datasets in the experiments are generated using synthetic generator in [4] and the parameter setting is shown in Fig. 3.

| Parameters | Description |
|---|---|
| $\mid D \mid$ | Number of event sequences |
| $\mid C \mid$ | Average size of event sequences |
| $\mid S \mid$ | Average size of potentially frequent sequences |
| $N_S$ | Number of potentially frequent sequences |
| $N$ | Number of event symbols |

**Fig. 3.** Parameters of synthetic data generator

### 5.1 Performance and Scalability on Synthetic Dataset

In all the following experiments, two parameters are fixed, i.e., $\mid S \mid = 4$ and $N_S$ = 5,000. The other parameters are configured for comparison. Note that, for fair comparison, when comparing the performance of CoPMiner with other interval-pattern mining algorithms, we only discuss the part of usage sequence mining (i.e., exclusive of computation of probability function). Fig. 4(a) shows the running time of the four algorithms with minimum supports varied from 1 % to 5 % on the dataset *D*100k–*C*20–*N*10k. 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 which may lead to generate more number of intermediate candidate sequences.

Then, we study the scalability of CoPMiner. Here, we use the data set $C = 20$, $N$ = 10k with varying different database size. Fig. 4(b) 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. 4(c) 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.

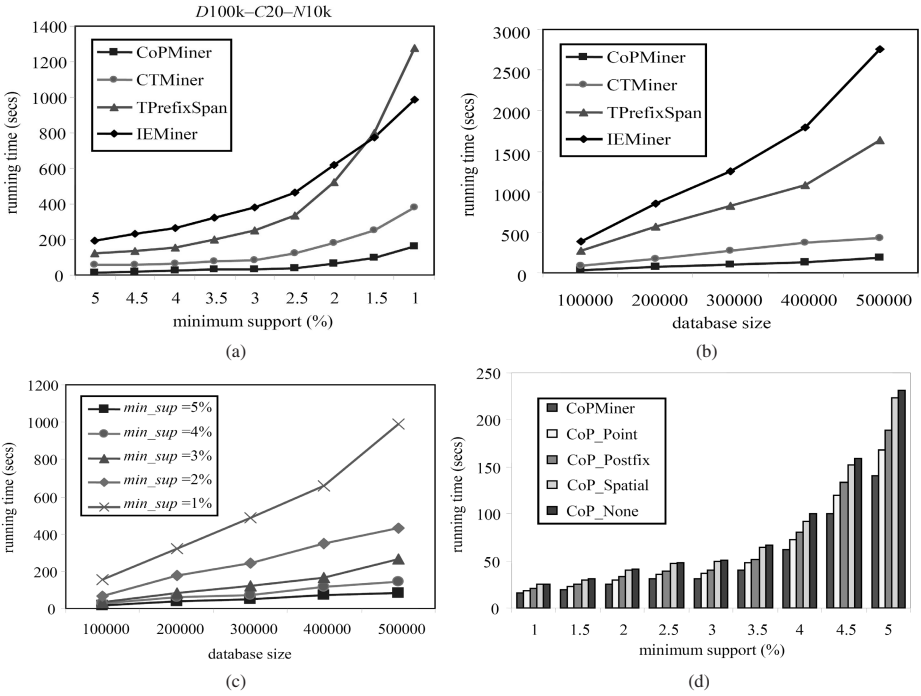



**Fig. 4.** Experimental results on synthetic datasets

## 5.2 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 five algorithms, *CoPMiner* (includes all pruning strategies), *CoP_Point* (only point-pruning strategy), *CoP_Postfix* (only postfix-pruning strategy), *CoP_Spatial* (only spatial-pruning strategy) and *CoP_None* (without any pruning strategy). The experiment is performed on the data set *D*100k–*C*20–*N*10k. Fig. 4(d) is the results of varying minimum support thresholds from 0.5% to 1%. As shown in figure, point-pruning can improve about 25% performance. Because of removing non-qualified usage-points before database projection, point-pruning can efficiently speedup the execution time. As can be seen from the graph, postfix-pruning can improve about 11% performance. Postfix-pruning can improve the performance by effectively eliminating all useless usage-points for correlation pattern construction. We also can observe that spatial-pruning constantly ameliorate the performance about 2.5.

### 5.3 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 [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 with *min_sup* = 0.3 and *min_sim* = 0.1. The probability function of each usage-point in pattern is listed below.

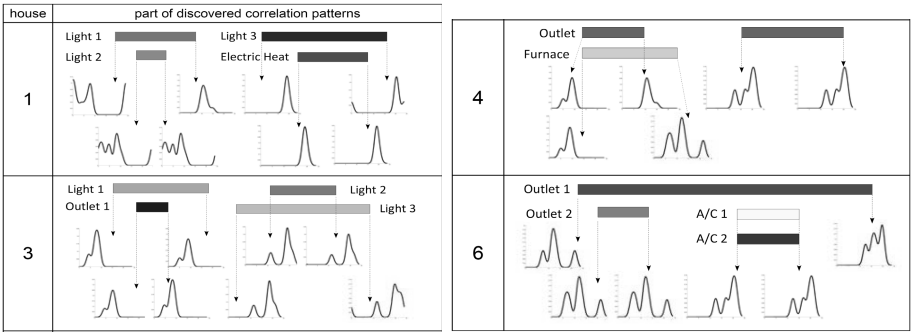



**Fig. 5.** Part of discovered correlation patterns from REDD dataset

## 6 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. In this paper, we introduce a new concept, correlation pattern, to capture the usage patterns and correlations among appliances probabilistically. An efficient algorithm, CoPMiner is developed to discover patterns based on proposed usage representation. The experimental studies indicate that CoPMiner is efficient and scalable. Furthermore, CoPMiner is applied on a real-world dataset to show the practicability of correlation pattern mining.

## References

1. Allen, J.: Maintaining Knowledge about Temporal Intervals. Communications of ACM 26(11), 832–843 (1983)
2. Aritoni, O., Negru, V.: A Methodology for Household Appliances Behavior Recognition in AmI Systems Integration. In: 7th International Conference on Automatic and Autonomous Systems (ICAS 2011), pp. 175–178 (2011)
3. Chen, F., Dai, J., Wang, B., Sahu, S., Naphade, M., Lu, C.T.: Activity Analysis Based on Low Sample Rate Smart Meters. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2011), pp. 240–248 (2011)

4. Chen, Y., Jiang, J., Peng, W., Lee, S.: An Efficient Algorithm for Mining Time Interval-based Patterns in Large Databases. In: Proceedings of 19th ACM International Conference on Information and Knowledge Management (CIKM 2010), pp. 49–58 (2010)
5. Chen, Y.-C., Ko, Y.-L., Peng, W.-C., Lee, W.-C.: Mining Appliance Usage Patterns in a Smart Home Environment. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) PAKDD 2013, Part I. LNCS (LNAI), vol. 7818, pp. 99–110. Springer, Heidelberg (2013)
6. Goncalves, H., Ocneanu, A., Bergés, M.: Unsupervised disaggregation of appliances using aggregated consumption data. In: KDD Workshop on Data Mining Applications in Sustainability, SustKDD 2011 (2011)
7. Ito, M., Uda, R., Ichimura, S., Tago, K., Hoshi, T., Matsushita, Y.: A method of appliance detection based on features of power waveform. In: 4th IEEE Symposium on Applications and the Internet (SAINT 2004), pp. 291–294 (2004)
8. Jakkula, V., Cook, D.: Using Temporal Relations in Smart Environment Data for Activity Prediction. In: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), pp. 1–4 (2007)
9. Jakkula, V., Cook, D., Crandall, A.: Temporal pattern discovery for anomaly detection in a smart home. In: Proceedings of the 3rd IET Conference on Intelligent Environments (IE 2007), pp. 339–345 (2007)
10. Kato, T., Cho, H.S., Lee, D., Toyomura, T., Yamazaki, T.: Appliance recognition from electric current signals for information-energy integrated network in home environments. In: Mokhtari, M., Khalil, I., Bauchet, J., Zhang, D., Nugent, C. (eds.) ICOST 2009. LNCS, vol. 5597, pp. 150–157. Springer, Heidelberg (2009)
11. Kim, H., Marwah, M., Arlitt, M., Lyon, G., Han, J.: Unsupervised disaggregation of low frequency power measurements. In: 11th SIAM International Conference on Data Mining (SDM 2011), pp. 747–758 (2011)
12. Kolter, J.Z., Johnson, M.J.: REDD: A public data set for energy disaggregation research. In: KDD Workshop on Data Mining Applications in Sustainability, SustKDD 2011 (2011)
13. Lin, G., Lee, S., Hsu, J., Jih, W.: Applying power meters for appliance recognition on the electric panel. In: 5th IEEE Conference on Industrial Electronics and Applications (ISIEA 2010), pp. 2254–2259 (2010)
14. Liu, B., Yang, Y., Webb, G.I., Boughton, J.: A Comparative Study of Bandwidth Choice in Kernel Density Estimation for Naive Bayesian Classification. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 302–313. Springer, Heidelberg (2009)
15. Patel, D., Hsu, W., Lee, M.: Mining Relationships Among Interval-based Events for Classification. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), pp. 393–404 (2008)
16. Pei, J., Han, J., Mortazavi-Asl, B., Pito, H., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In: Proceedings of 17th International Conference on Data Engineering (ICDE 2001), pp. 215–224 (2001)
17. Silverman, B.: Density Estimation for Statistics and Data Analysis. Chapman and Hall (1986)
18. Suzuki, K., Inagaki, S., Suzuki, T., Nakamura, H., Ito, K.: Nonintrusive appliance load monitoring based on integer programming. In: International Conference on Instrumentation, Control and Information Technology, pp. 2742–2747 (2008)
19. Wu, S., Chen, Y.: Mining Nonambiguous Temporal Patterns for Interval-Based Events. IEEE Transactions on Knowledge and Data Engineering 19(6), 742–758 (2007)