Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, concerns over global climate changes have motivated significant efforts in reducing the emissions of CO_2 and other greenhouse gases (GHGs). With the increasing requirement for more electric power, the development of sustainable energy like hydroelectricity, solar and wind energy attract many researchers to engage in. The manufacturing process of industry contributes large amount of greenhouse gases while the business and residential area counts on steady and continuous power supply. For the residential or industrial area, it is not an easy task to indicate that where we have wasted our precious electricity power for unnecessary appliances uses. Many researchers focus on the reduction of electricity usage in residence because it is a significant contributor of greenhouse gas emissions.

However, electricity conservation is an arduous task for the residential users due to the lack of detailed electricity usage. For the residential or industrial area, it is not an easy task to indicate that where we have wasted our precious electricity power for unnecessary appliances uses. Therefore we expect that we can use electricity more efficiently by discovering the appliances behaviors. If the useful inspiration of our behaviors and representative patterns of appliance electricity usage are available, residents can adapt their appliance usage behaviors to conserve the energy effectively.

Due to the advance of sensor technology, the electricity usage data of in-house appliances can be collected easily. For example, it is no longer an obstacle that we manufacture the electrical sensors with capability of collecting the voltage, current and apparent power consumption information. Collecting the appliances or environment status will be easy [3, 18]. 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. Nonetheless, with an anticipated huge amount of appliance usage data, subtle information may exist but hidden. We observed that in the smart environment the appliances behaviors of the residents have the spatial relation. For example, people watching television in the living room of first floor will also turn on the lights in the living room, rather than turn on the other appliances in the second or third floor. We also observed that the appliances ON-OFF data in the smart environment can be regarded as the time interval-based database. 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 cannot 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.

In our daily life, we usually use different appliances simultaneously. For example, in the morning, coffee machine and toaster in the kitchen are often used together to prepare for breakfast, while the light, air conditioner and television in the living room may be turned on in the evening (as shown in Figure 1). The correlation among the usage of some appliances can provide valuable information to assist residents better understand how they use appliances.

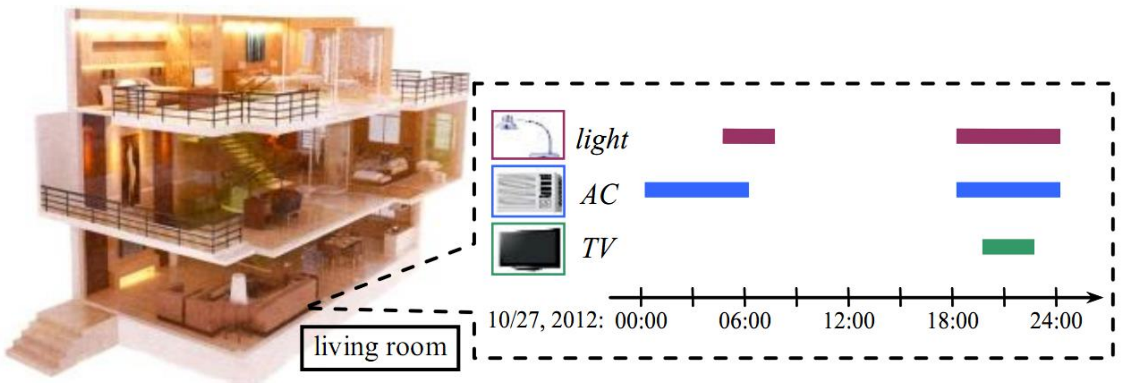


Figure 1: An example of daily usage sequence

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 behaviors. Hence, we aim to derive compact and meaningful patterns in this study. Obviously, the locations of appliances in a house provide good hints. For example, as shown in Fig. 1, the correlation between television and light in the same living room on the first floor may reveal a high dependency between these two devices. Moreover, the co-occurrence of turning on the television in living room on the first floor and turning on the light in bathroom on the third floor may be merely a coincidence.

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:

- Probabilistically, 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 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.
- The readability of patterns is also an essential issue. Sometimes, a large number of patterns may become an obstacle for users to understand their actual behaviors. To generate compact, expressive and meaningful patterns, we propose a method that takes into account the interior of a smart home. A

spatial constraint is introduced to prune off non-promising correlation and reduce the number of generated correlation patterns.

- 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.

Now let us list the application domain of mining correlation patterns. That is, we will mention who, where and when will people need the pattern mining algorithm. We first enumerate the application domain are Researchers in Library and Information Science [5], in Information and Electrical Engineering (Smart Sensors and Environment related) [7, 8], Economy, Banking and Finance [30], Medical Information [31] and Mobile Sensors Applications [32, 33].

As the experiments in CTMiner [5], the lending/borrowing behaviors of books can be our applications. Different from CTMiner explores what lending/borrowing behaviors of books are tends to appear, our algorithm introduce the spatial constraint, which can be regarded as the general similarity of items, can explore more specific the relations of patterns and along with its happening time. That is, CTMiner may tell us that many people borrowing a book of C++ Programming Language (Science Class with class number 300), a book of Philosopher (Philosophy Class with class number 100) and a book of Eastern Architecture (Arts Class with class number 900). This lending behavior is not easy to recommend people to borrow what kind of related C++ programming book. However, our algorithm especially focuses on the relations of items and happening time and may tell people to lend another C++ programming books which many people recommend. For the managers of library, they can see the lending time of these books to decide which part of books are frequently lend and pay more attention on managing or introducing new books into these area.

In banking and finance, the spatial constraint in correlation pattern can be seen as the relations between different stocks, future contracts, bonds, funds and other derivatives in financial instrument. Financial data has much worthy implicit knowledge and the Chief Finance Officer (CFO) of company may be interested in the stock interactive behaviors in Stock Market to make proper financial decisions for company.

Medical Information will also record many laboratory variables and disease for patients. Especially, the medical data like electronic health record (EHR) is multivariate and contains class labels [31]. For mobile users, the user's temporal profile is very crucial for App prediction [32, 33]. Our algorithm may explore more interval-based sensors features for each user and can be used in prediction.

The rest of the paper is organized as follows. Section 2 reviews some related works. Section 3 provides the preliminaries and the framework of our system. Section 4 introduces the detail definitions of usage representation correlation pattern, and probability-based projected database and finally proposed CoPMiner algorithm. Section 5 proposed three pruning strategies for discovering correlation patterns. Section 6 gives the detailed applications on anomaly detection by using correlation patterns. Section 7 reports the experimental results in a performance study and the correlation patterns mined in real-world dataset, and finally Section 8 concludes this paper.

Chapter 2

Related Work

During the past several decades, many researchers focus on sequential pattern mining, temporal pattern mining and discover the behaviors in smart environment. We provide a very brief survey on these domains and summarize the domains into two parts. First, we discuss some literatures about sequential pattern mining and temporal pattern mining. Then, we will discuss some works mainly apply to smart environment. Some applications like behaviors discovery, anomaly detection and activity prediction are also listed.

2.1 Research Works in Smart Environment

Many prior studies discuss how to extract useful knowledge regarding usage patents of a single appliance via energy disaggregation [4, 9, 10, 17, 20, 29] or appliance recognition [2, 7, 8, 12, 16, 22, 27, 29].

In the domain of energy disaggregation, Chen et al. [4] 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. Farinaccio et al. [9] use some patterns, such as number of ON-OFF switches, to disaggregate the whole-house electricity consumption into a number of major end-uses. Goncalves et al. [10] explore an unsupervised approach to determine the number of appliances in the household, including their power consumption and state, at any given moment. Kim et al. [17] 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. Lin et al. [20] use a dynamic Bayesian network and filter to disaggregate the data online. Suzuki et al. [29] use a new NIALM technique based on integer programming to disaggregate residential power use.

For appliance recognition, Aritoni et al. [2] develop a software prototype to understand the behaviors of household appliances. Ito et al. [12] extract features from the current (e.g., amplitude, form, timing) to develop appliance signatures. Kato et al. [16] use Principal Component Analysis to extract features from electric signals and classify them using Support Vector Machine. Prudenzi [27] utilize an artificial neural network based procedure for identifying the electrical signatures of residential appliances. Matthews et al. [22] discuss some of these works and characterize workable solutions.

For usage pattern of appliances, Chen et al. [8] introduce two types of usage patterns to describe users' representative behaviors. HAUBA [7] is developed to analyze the usage status of all appliances in a smart home environment. An intelligent system, Jakkula et al. propose an Apriori-based algorithm [13] for activity prediction [14] and anomaly detection [15] from sensor data in a smart home. However, Jakkula et al. use the Allen's representation for time interval-based data, which still has ambiguity, and mining algorithm is candidate generation, which is not efficient enough.

All aforementioned studies focus on knowledge extraction for a single appliance instead of the correlation among appliances in a house.

2.2 Sequential/Temporal Pattern Mining

The sequential pattern mining originally focuses on the time point-based database [11, 26]. Han et al. [11] propose an efficient sequential pattern mining method, named **FreeSpan**. The general idea of **FreeSpan** is to integrate the mining of frequent sequences with that of frequent patterns and use projected sequence databases to confine the search and the growth of subsequence fragments. Pei et al. [26] propose an efficient sequential mining algorithm, named **PrefixSpan**, based on divide-and-conquer inspiration. **PrefixSpan** explores prefix-projection in sequential pattern mining, which substantially reduces the size of projected databases.

The time point-based data denotes the transactions happened at the specific time. However, the time interval-based data records the event both with start time and finish time. Therefore, a proper representation for time interval-based is a great issue. Allen’s logic [1] gives the 13 relations between any two intervals. Figure 2 lists all the Allen’s relations. However, Allen’s relation will suffer the disambiguity when representing three or more intervals. To conquer the problem, researchers usually use a relation matrix to maintain.

Some algorithms are proposed to discover the temporal patterns in time interval-based database [5, 6, 23, 25, 30]. Morchen et al. [23] propose the Time Series Knowledge Representation(TSKR) representation and mining algorithm TSKM. Patel et al. [25] propose the Augmented Hierarchical Representation (abbreviate as AHR) to explore temporal pattern for classification. Wu et al. [30] propose the Temporal Sequence (abbreviate as TS) to mining temporal patterns. Chen et al. [5] propose a more compact representation, named Coincidence Representation (abbreviate as CR). Chen et al. [6] propose an Endpoint Representation (abbreviate as ER). To fairly compare the advantages and disadvantage, we use the example of Figure 3 to show how these temporal representation works. Table 1 lists the above five representations for the example in Figure 3.

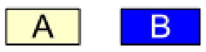
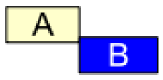
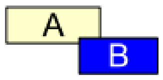
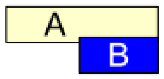
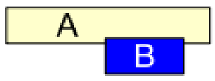
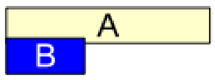
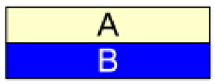
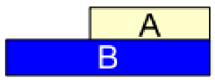
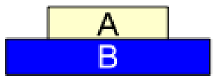
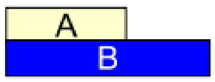
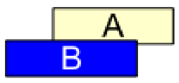
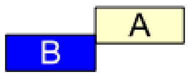
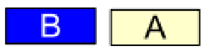
Temporal Relation	Pictorial Example	Endpoints constraints
A *before* B		$A.s < A.f < B.s < B.f$
A *meets* B		$A.s < A.f = B.s < B.f$
A *overlaps* B		$A.s < B.s < A.f < B.f$
A *finished-by* B		$A.s < B.s < A.f = B.f$
A *contains* B		$A.s < B.s < B.f < A.f$
A *started-by* B		$A.s = B.s < B.f < A.f$
A *equals* B		$A.s = B.s < A.f = B.f$
A *finishes* B		$B.s < A.s < A.f = B.f$
A *during* B		$B.s < A.s < A.f < B.f$
A *starts* B		$A.s = B.s < A.f < B.f$
A *overlapped-by* B		$B.s < A.s < B.f < A.f$
A *met-by* B		$B.s < B.f = A.s < A.f$
A *after* B		$B.s < B.f < A.s < A.f$

Figure 2: Allen's 13 relations between two intervals

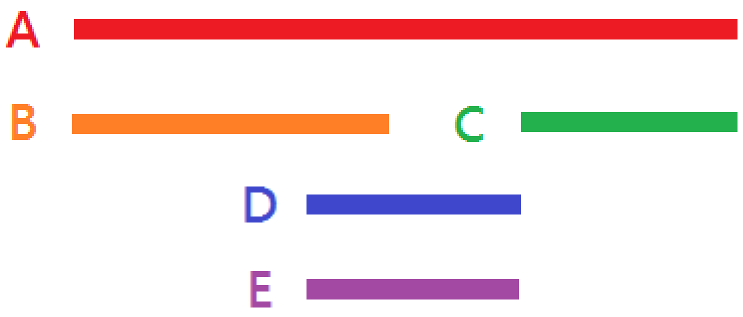


Figure 3: An example contains 13 Allen's relation

Table 1: Five distinct representations for temporal relations in Figure 3

Method	**Representation**
TSKR	AB $\rightarrow$ ABDE $\rightarrow$ ADE $\rightarrow$ AC
AHR	(((A *Started-by* [0,0,0,0,1] B) *Contains* [1,0,0,1,0] D) *Contains* [2,0,0,2,0] E) *Finished-by* [2,0,2,2,0] C
TS	$A^+ = B^+ < D^+ = E^+ < B^- < C^+ = D^- = E^- < A^- = C^-$
CR	$(A^+ B^+)\ (D^+ E^+ B^-)\ (D^- E^-)$ @ $(C\ A^-)$
ER	$(A^+ B^+)\ (D^+ E^+)\ B^-\ (D^- E^- C^+)\ (A^- C^-)$

Chapter 3

System Overview

In section 3.1 Preliminaries, we will first give formal definitions of some terms we use in this paper and then the definition of our problem. In section 3.2 System Framework, we will give a concrete overview of our system framework and also describes how our system works.

3.1 Preliminaries

Definition 1 (Usage-interval and usage-interval sequence)

Let $A = \{a_1, \ldots, a_k\}$ 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 $\mathbb{N}$. A function, $Loc: A \rightarrow \mathbb{N}^3$, specifies the location of each appliance in A. Let the triplet $(a_i, o_i, f_i) \in A \times \mathbb{N} \times \mathbb{N}$ denote a **usage-interval** of a_i, where $a_i \in A, o_i, f_i \in \mathbb{N}$ and $o_i < f_i$. The two time endpoints o_i and f_i are the turn-on time and the turn-off time of appliance a_i respectively. This indicates the interval of appliance a_i starts at time o_i and finishes at time f_i with $o_i < f_i$. The set of all usage-intervals over A is denoted by I_A. A **usage-interval sequence** Seq is a series of usage-intervals triples. Formally, $Seq = \langle (a_1, o_1, f_1), (a_2, o_2, f_2), \ldots, (a_n, o_n, f_n) \rangle$ with $o_i \leq o_{i+1}$ and $o_i < f_i$. Also all the *n* elements in *Seq* are ordered by (1) start time o_i (2) finish time f_i and finally (3) appliance symbol a_i. $Loc(a_i)$ is the interior location of appliance a_i in a smart home environment. We can either use the logical location or real location to define Loc.

Let us take Figure 1 as an example. Suppose that 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, 2012.

Definition 2 (Usage-interval database)

The set $DB = \{r_1, r_2, \ldots, r_m\}$ is said to be a **usage-interval database** if *D* collects the records $r_i = \langle sid_i, v_i \rangle$ with the sequence date $sid_i \in \mathbb{N}$ and its corresponding daily usage-interval sequence v_i. Also, an interior location of appliance is also recorded in *DB*. Note that the location information can be regarded as an attachment to appliances. Figure 4 shows a usage database which consists of 17 usage intervals and 4 daily usage-interval sequences.

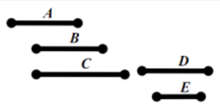
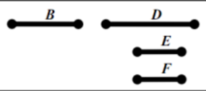
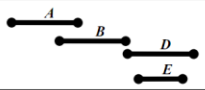
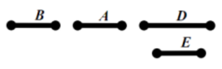
date	appliance symbol	turn-on time	turn-off time	interior location	pictorial example	usage representation (usage sequence, time sequence)
1	*A*	02:10	07:30	(1, 1, 1)		$\begin{pmatrix} A^+ & (B^+ & C^+) & A^- & B^- & C^- & D^+ & E^+ & E^- & D^- \\ 2 & 5 & 5 & 7 & 10 & 12 & 16 & 18 & 20 & 22 \end{pmatrix}$
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)		

Figure 4: An example of usage database

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 [5] 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 $Q = \langle (a_1, o_1, f_1), (a_2, o_2, f_2), \dots, (a_n, o_n, f_n) \rangle$, the set $TS_Q = \{o_1, f_1, o_2, f_2, \dots, o_n, f_n\}$ is the time set corresponding to Q. After we order all the elements of TS_Q in nondecreasing order, we can derive a sequence $T_Q = \langle t_1, t_2, \dots, t_{2n} \rangle$ where $t_i \in TS_Q$ and $t_i \le t_{i+1}$. T_Q is called a time sequence corresponding to Q.

Definition 3 (Usage-point and usage sequence)

Given a usage-interval sequence $Q = \langle (a_1, o_1, f_1), (a_2, o_2, f_2), \dots, (a_n, o_n, f_n) \rangle$ where $(a_i, o_i, f_i) \in I_A$ and corresponding time sequence $T_Q = \langle t_1, t_2, \dots, t_j, \dots, t_{2n} \rangle$, 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.

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

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^*, …, a_l^* (* can be ＋ or －) are collected in brackets and ordered the elements in parenthesis by (1) end point e_i^- (2) appliance symbol e and finally (3) start points e_i^+. We also attach occurrence number to usage-points in order to distinguish multiple occurrences of the same appliance in an usage-point sequence. A usage sequence S_Q of Q is denoted by $S_Q = \langle s_1, s_2, \dots, s_{2n} \rangle$ where s_i is a usage-point. For example, in Figure 4 the database collects 4 daily usage-interval sequences. The **usage sequence** of date 2 is $\langle B^+\ B^-\ D^+\ (E^+\ F^+)\ (E^-\ F^-)\ D^- \rangle$. Noted that $(E^+\ F^+)$ and $(E^-\ F^-)$ are represented with ordering within parenthesis because they occur at the same time respectively.

Definition 4 (Usage representation)

Given a usage-interval sequence $Q = \langle (a_1, o_1, f_1), \dots, (a_n, o_n, f_n) \rangle$ and corresponding time sequence $T_Q = \langle t_1, \dots, t_i, \dots, t_{2n} \rangle$, we can derive the usage sequence $S_Q = \langle s_1, \dots, s_i, \dots, s_{2n} \rangle$ by Definition 3. The **usage representation** of Q is defined as a pair. Noted that the time of usage point s_i in S_Q is t_i in T_Q.

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

By Definition 4, we can transform a usage-interval database into usage representation. Let us take the database in Figure 4 as an example. Without leading into ambiguity, we consider the turn-on and turn-off times by recording hours only. The usage representation of *DB* is shown in the last column in Figure 4. For the rest of this paper, we assume that the usage database has already been transformed into usage representation.

Definition 4.1 (Subsequence, support, and spatial similarity)

Let $S_1 = \langle x_1, \dots, x_i, \dots, x_n \rangle$ and $S_2 = \langle y_1, \dots, y_j, \dots, y_m \rangle$ be two usage sequences, where x_i, y_j are usage points and $n \le m$. S_1 is called a **subsequence** of S_2, denoted by $S_1 \sqsubseteq S_2$, if there exist integers $1 \le k_1 \le k_2 \le \cdots \le k_n \le m$ such that $x_1 = y_{k_1}$, $x_2 = y_{k_2}$, …, $x_n = y_{k_n}$. Given a usage-interval database *DB* in usage representation, the tuple $(sid_i, S_i, T_i) \in DB$ is said to contain a usage sequence V if $V \sqsubseteq S_i$. The **support** of a usage sequence V in *DB*, denoted as $suppport(V)$, is the number of tuples in the database containing V. More formally,

$$support(V) = |\{(sid_i, S_i, T_i) \in DB |\ V \sqsubseteq S_i\}| \tag{3}$$

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

$$similarity(a_1, a_2) = \begin{cases} 1 & if\ Loc(a_1) = Loc(a_2) \\ \dfrac{1}{|Loc(a_1) - Loc(a_2)|} & if\ Loc(a_1) \neq Loc(a_2) \end{cases} \tag{4.1}$$

$$, and\ |Loc(a_1) - Loc(a_2)| = |a_{1_x} - a_{2_x}| + |a_{1_y} - a_{2_y}| + |a_{1_z} - a_{2_z}| \tag{4.2}$$

For example, in Figure 4 the similarity of appliances B and C is $\frac{1}{|1-3|+|2-4|+|1-2|} = \frac{1}{5} = 0.2$. Obviously, the support count decides the significance of a usage sequence. We use a support threshold, *min_sup*, to filter out insignificant usage sequences. Furthermore, as mentioned above, the spatial distance may spoil the correlation dependency between two devices. When mining representative usage sequences, we use a similarity threshold, *min_sim*, to filter out non-promising appliances in a usage sequence. A usage sequence $S = \langle s_1, \ldots, s_n \rangle$ in *DB* is called a frequent sequence if $support(S) \geq min\_sup$ and $\forall s_i, s_j \in S, similarity(s_i, s_j) \geq min\_sim$.

Let us go into deeper abstraction for spatial similarity. We can regard the spatial similarity as the general similarity for items. That is, for the applications in library and information sciences, if we want to find the lending/borrowing relations among the same class of the books we can define each book is in an imaginary location in the pseudo-space of book space. Probably the location is related to the class of the book. For another financial domain, we can regard each stock is located in the stock space. And their location may be defined by their other feature like industry type, capital of company, the rate of return and etc. Of course the practical definition of similarity may vary from domain to domain. But here we just introduce the general idea of item similarity and we name it as spatial similarity. Also, the spatial similarity will have great impact on what kind of correlation patterns are mined, which is defined in Definition 5. Therefore, we can regard the spatial similarity as the constraints of the set of target patterns.

Definition 5 (Correlation pattern)

Given a usage-interval database *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*. The correlation pattern consists of two parts. One part is the frequent sequence $S = \langle s_1, \dots, s_n \rangle$. The other part is the probability function of each usage point happening in time of 24 hours and we use a set of functions $F(S) = \langle f_1, \dots, f_n \rangle$ to annotate the probability along with the frequent sequence S. The formal definition of correlation pattern *P* is defined as

$$P = \big(S, F(S)\big) = \begin{pmatrix} s_1 & \dots & s_i & \dots & s_n \\ f_1 & \dots & f_i & \dots & f_n \end{pmatrix},$$

where $S = \langle s_1, \dots, s_n \rangle \in FS$ and f_i is the probability function of s_i in *DB*. (5)

We also use the idea of Multivariate Kernel Density Estimation [21, 28] to estimate the probability function of each $s_i \in S$. Suppose that the time information of s_i in *DB* is $\{t_{i_1}, t_{i_2}, \dots, t_{i_m}\}$, the formal definition of probability function is defined as

$$f_i(x) = \langle K(x), h, \{t_{i_1}, \dots, t_{i_m}\} \rangle = \frac{1}{mh} \sum_{j=1}^{m} K\left(\frac{x - t_{i_j}}{h}\right), \text{ with } h = \frac{range(\{t_{i_1}, \dots, t_{i_m}\})}{\sqrt{m}}$$

$$\text{and where } K \text{ is Gaussian Normal Kernel}, K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \qquad (6)$$

We take the database in Figure 4 as an example again. Given *min_sup* = 2 and *min_sim* = 0.3, $\langle A^+ \, A^- \, D^+ \, D^- \rangle$ is a frequent sequence since it appears in date = 1, 3 and 4. Therefore $support(\langle A^+\ A^-\ D^+\ D^- \rangle) = 3 \geq min\_sup = 2$ and $similarity(A, D) = 0.5 \geq min\_sim = 0.3$. The correlation pattern with respective to $\langle A^+\ A^-\ D^+\ D^- \rangle$ is $\begin{pmatrix} A^+ & A^- & D^+ & D^- \\ f_{A^+} & f_{A^-} & f_{D^+} & f_{D^-} \end{pmatrix}$. Here we discuss $f_{A^+}(x)$ as an example. The time information of A^+ is $\{2,\ 6,\ 13\}$, hence $f_{A^+}(x) = \frac{1}{3h\sqrt{2\pi}}\left(e^{-\frac{1}{2}\left(\frac{x-2}{h}\right)^2} + e^{-\frac{1}{2}\left(\frac{x-6}{h}\right)^2} + e^{-\frac{1}{2}\left(\frac{x-13}{h}\right)^2}\right)$, with $h = \frac{range(\{2,6,13\})}{\sqrt{3}} = \frac{13-2}{\sqrt{3}} = 6.35$.

Obviously, with a turn-on time of appliance $a \in A$, we can derive probability functions $f_{a^+}(x)$ and $f_{a^-}(x)$ with respect to the usage of a. The practicability of correlation patterns will be discussed in chapter 4 and 6.

The reason that we define the correlation pattern combined with probability density functions (p.d.f.) makes the correlation pattern outperforms from related works [5, 6, 23, 25, 30]. Since we expect to see the p.d.f. of happening time of each interval not only simply interprets the relations. The gap of happening time also make the patterns contains different meanings. Consider that there are two patterns with the same interval relations. The first pattern says that next behavior pops up within 1 hour while another tells us the next behavior appears in 8 hours. This will tell us that although the relation might be same, but the happening time (or equivalently the p.d.f.) are different matters the semantics of the pattern. Also with the p.d.f. appended in correlation pattern, we can provide more information and application like anomaly detection and activity prediction by applying the calculations on p.d.f.

3.2 System Framework

In this section we will describe our framework of novel mining algorithm, CoPMiner, and also give a general view of each components or modules inside the CoPMiner. The general concepts or main idea of our algorithm or components will be described here. And the detailed content will be depicted in chapter 4 and 5.

Problem definition

Let $D = \{r_1, \dots, r_N\}$ be the usage database of appliances symbol set $A = \{a_1, \dots, a_k\}$. Inside the usage database D we can see that each record r_i collects the usage-interval sequence of certain day sid_i. Equivalently we can say that $r_i = \langle (a_{i_1}, o_{i_1}, f_{i_1}), \dots, (a_{i_n}, o_{i_n}, f_{i_n}) \rangle$ is the daily behavior that consist of several usage intervals $(a_{i_k}, o_{i_k}, f_{i_k}) \in A \times \mathbb{N} \times \mathbb{N}$. Each appliances $a \in A$ also has the interior information which is retrieved from the original floor plan and denoted as location function $Loc: A \to \mathbb{N} \times \mathbb{N} \times \mathbb{N}$. In order to explore the correlation patterns, the minimum support of database $min\_sup \in \mathbb{N}$ is used to verify the frequent sequences from the usage database. With respect to spatial constraint of the usage sequence of correlation pattern, the minimum spatial support of location function $min\_sim \in [0,1]$ is also used to ensure that each appliance in the sequence of explored correlations pattern are pairwise closed enough in the space.

Now given the usage database D and the minimum support thresholds $min\_sup \in \mathbb{N}$ and $min\_sim \in [0,1]$, our target is to explore the correlation pattern set L from the usage database D. Mathematically, if the usage sequence q is a frequent pattern for D (that is $support(q) \geq min\_sup$) and the appliances inside the sequence q are spatially closed enough (that is $\forall s_i, s_j \in q, similarity(s_i, s_j) \geq min\_sim$), then correlation pattern of q is collected in L, i.e. $(q, F(q)) \in L$.

The framework of CoPMiner

Now our framework is clearly shown in Figure 5. Obviously, our framework contains online and offline part. The main goal of the online part is to obtain the two parameters $min\_sup \in \mathbb{N}$ and $min\_sim \in [0,1]$ from user. After receiving the users' preference with database, we will give the two parameters to main algorithm, CoPMiner, to discover all the correlation patterns and then we show to users what the patterns are mined. The offline part consists of usage database, the location function, the database in usage representation and finally the CoPMiner algorithm.

Transforming the usage database in usage representation is the first step in the offline part and is shown as the blue cylinder in the Figure 5. We require following the

Definition 1 to 4 to transform every sequence in the original usage-interval database into the usage representation like Equation (2). We will further mining the patterns based on the usage representation and therefore this step is very important. This step will scan the usage database once and transform the usage-interval sequence into usage representation form like Equation (2).

The CoPMiner algorithm is the second step after converting the notations of usage sequences. The main mining concept of CoPMiner is similar to the previous framework of PrefixSpan [PrefixSpan]. CoPMiner will count the frequent usage points in the local projected database and construct the next projected database based on the current prefix. The recursive mining procedure is named UPrefixSpan in our algorithm. Owing to we are required to explore the correlation patterns, three strategies **pruneDB**, **pruneFI** and **pruneSup** are also used interactively inside the UPrefixSpan. The detailed concepts and implementation methods of pruning strategies are depicted in chapter 5.

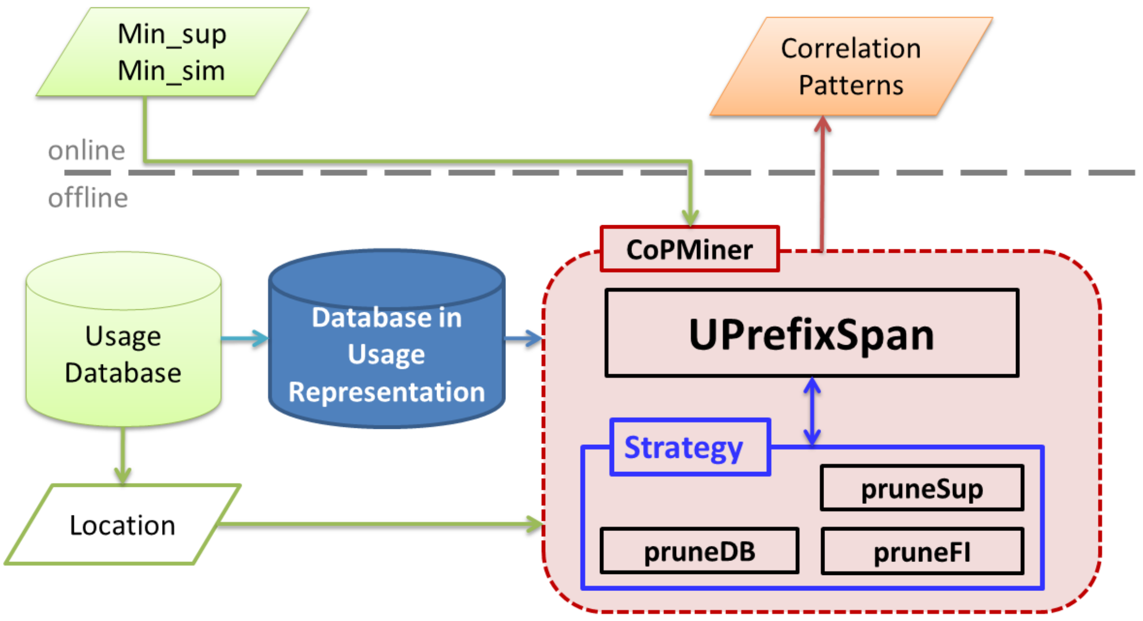


Figure 5: The framework of CoPMiner

Chapter 4

Correlation Pattern Discovery

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 three pruning strategies to effectively reduce the search space and speedup the mining process. In Section 4.1, we discuss some advantages of usage representation. In Section 4.2, we outline the practicability of proposed correlation pattern and describe how to use the correlation patterns to perform some applications like anomaly detection and behavior prediction. In Section 4.3, we proposed the novel idea of projected database used in CoPMiner. In a nutshell, we introduce the concept of affect the projected database of certain prefix by probability function. In Section 4.4, we give a complete and full view of detailed mining algorithm－CoPMiner.

4.1 Usage Representation

Practicability of Usage Representation

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 Figure 2. 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 [30] 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 Figure 2. Several merits of usage representation are discussed as follows,

- 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. Since each usage-interval has two usage-points, we only require 2k space for expressing a k-interval sequence and 2k space for describing turn-on/off time. The usage representation scales well even with plenty of intervals appearing in a sequence.
- Nonambiguity: According to [30], we can find that the usage representation has no ambiguous problem. First, by Definition 3 and 4, 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 Figure 2. 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.

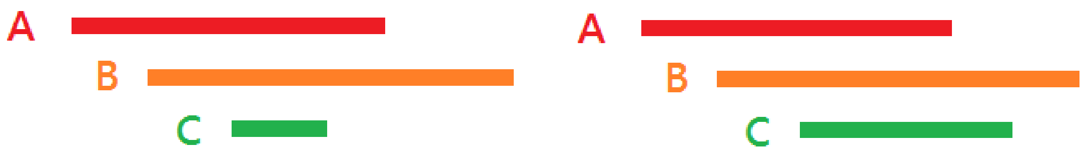
Advantages of Usage Representation

Here we list some examples to indicate the advantages of usage representation. Consider we have three appliances A, B and C and their pictorial temporal relations are depicted in Figure 6. Clearly, we can identify that the temporal relations of Figure 6.(a) and Figure 6.(b) are different. However, we may fall into ambiguity if simply use Allen's relation to represent Figure 6.(a) and Figure 6.(b). That is, using Allen's relation without parameters would cause ((A overlaps B) during C) to represent both Figure 6.(a) and Figure 6.(b). Traditional method could perform the disambiguity by providing the relation matrix like Table 2 and Table 3. Table 2 states that (A overlaps B), (A contains C) and (B contains C) while Table 3 states that (A overlaps B), (A overlaps C) and (B contains C). However, the relation matrix costs $\Theta(n^2)$ space for n intervals while our usage representation costs only $\Theta(n)$ space. Our usage representation for Figure 6 is clearer and can be constructed from only given the relation. The usage representation for Figure 6.(a) is $\langle A^+ \; B^+ \; C^+ \; C^- \; A^- \; B^- \rangle$ and Figure 6.(b) is $\langle A^+ \; B^+ \; C^+ \; A^- \; C^- \; B^- \rangle$.

Scalability and low cost of relation extension is another advantage of usage representation. We consider again in the Figure 7. If we have relation like Figure 7.(a), and we want to add new interval D into relation to be Figure 7.(b). Traditional relation matrix will add new column for interval D and fill in all the relations with other intervals. That is, Table 4 lists (A overlaps B), (A finished-by D), (A contains C), (B contains D), (B contains C) and (D contains C). In usage representation, we simply add two endpoints D^+ and D^- into $\langle A^+ \; B^+ \; C^+ \; C^- \; A^- \; B^- \rangle$ to be $\langle A^+ \; B^+ \; D^+ \; C^+ \; C^- \; (A^- \; D^-) \; B^- \rangle$. This scalability of usage representation is very useful when our mining algorithm use prefix-growth approach.

4.2 Correlation Pattern

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, users can know the distribution of usage time of appliances. With the turn-on/off time of an appliance, we can derive the usage probability of other appliances.



(a) Example 1 of A, B and C (b) Example 2 of A, B and C

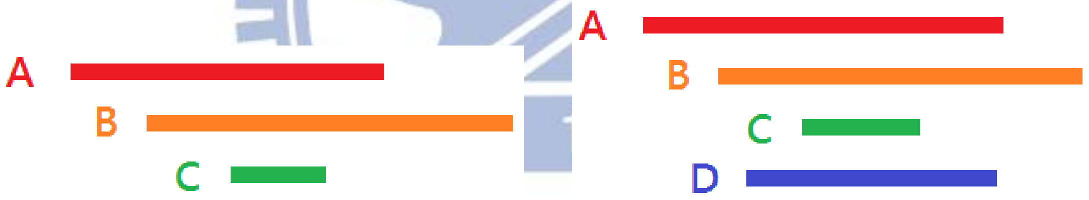
Figure 6: Two examples of different temporal Allen's relation

Table 3: Matrix of Figure 6.(a)

xRy	A	B	C
A	-	o	c
B		-	c
C			-

Table 2: Matrix of Figure 6.(b)

xRy	A	B	C
A	-	o	o
B		-	c
C			-



(a) Example 1 of A, B and C (b) Example 3 of A, B, C and D

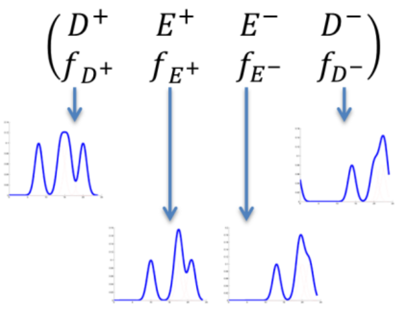
Figure 7: Two examples of extending temporal Allen's relation

Table 4: Matrix of Figure 7.(b)

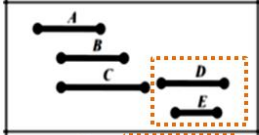
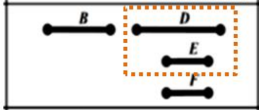
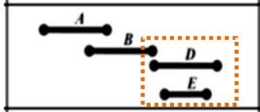
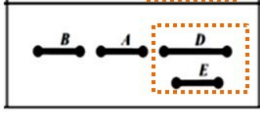
xRy	A	B	D	C
A	-	o	fi	c
B		-	c	c
D			-	c
C				-

Consider the correlation pattern in aforementioned example in Definition 5. Suppose the appliance A is the light and D is the coffee machine. Given the turn-on/off times of light and coffee machine, we can derive the usage probability for A and D respectively. That is to say, the probability functions for the light and coffee machine to be turned on/off at that time. This probability information is very useful for several applications, such as abnormal behavior detection and activity prediction. Suppose that a user forgets to turn off the light when she goes to supermarket. The home management system (HMS) detects that the light is still turn-on at a time when the turn-on probability 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 (appliance symbol D) is usually turned on after the light (appliance symbol A) is turned off. If we detect the light is turned off at a given time, the HMS may automatically turn on the coffee machine if the probability of happening time from the aforementioned correlation pattern is high enough.

Here we give another correlation example. Suppose the appliances D and E are spatially close enough in Figure 4. Given the minimum support $min\_sup = 2$ and $min\_sim = 0.3$. Then the sequence $\langle D^+\ E^+\ E^-\ D^- \rangle$ is frequent and therefore the correlation pattern is briefly shown in Figure 8.



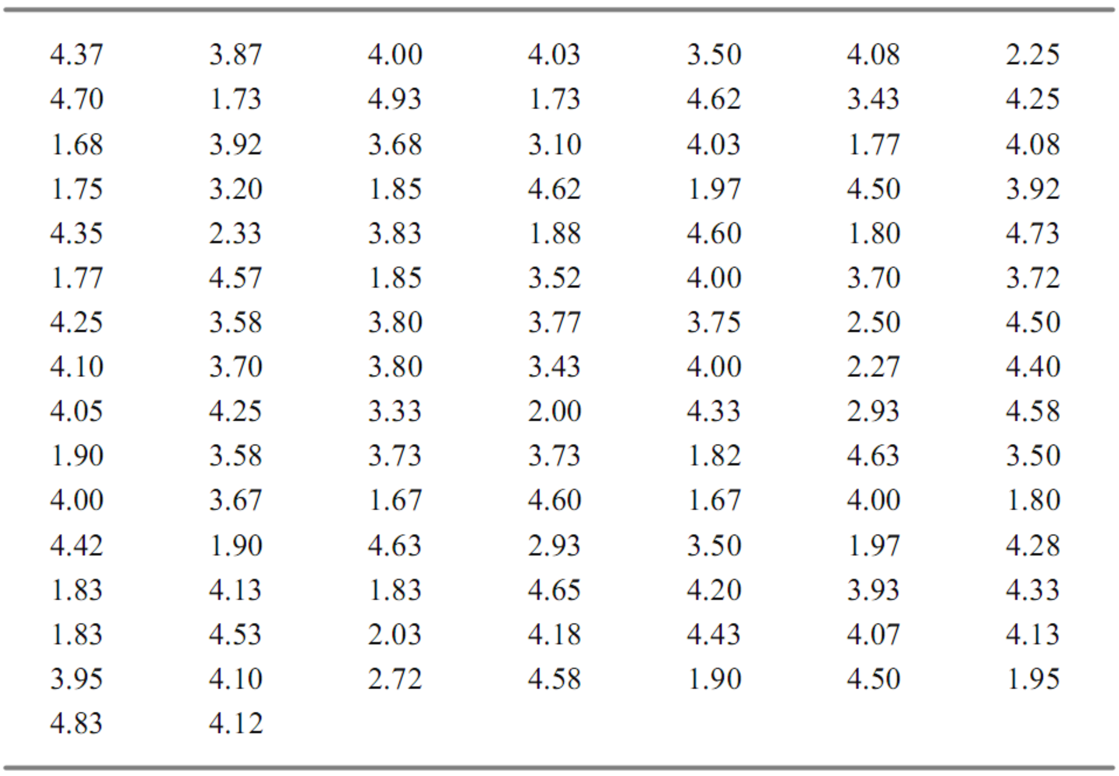
(a) Correlation pattern

pictorial example	usage representation (usage sequence, time sequence)
	$\begin{pmatrix} A^+ & (B^+ & C^+) & A^- & B^- & C^- & D^+ & E^+ & E^- & D^- \\ 2 & 5 & 5 & 7 & 10 & 12 & 16 & 18 & 20 & 22 \end{pmatrix}$
	$\begin{pmatrix} B^+ & B^- & D^+ & (E^+ & F^+) & (E^- & F^-) & D^- \\ 0 & 5 & 8 & 10 & 10 & 13 & 13 & 14 \end{pmatrix}$
	$\begin{pmatrix} A^+ & B^+ & A^- & (B^- & D^+) & E^+ & E^- & D^- \\ 6 & 7 & 12 & 14 & 14 & 17 & 19 & 20 \end{pmatrix}$
	$\begin{pmatrix} B^+ & B^- & A^+ & A^- & D^+ & E^+ & E^- & D^- \\ 8 & 10 & 13 & 16 & 20 & 21 & 22 & 23 \end{pmatrix}$

(b) Positions of Pattern in Database

Figure 8: Correlation Pattern Example

Now we will discuss more about the probability density functions of the correlation pattern. The union of sequential pattern and probability distributions makes the sow's ear into a silk purse. Probability distribution fitting (or simply distribution fitting) is the fitting of a probability distribution to a series of data concerning the repeated measurement of a variable phenomenon. Basically there are parametric methods, regression method and statistical methods to conquer the distribution fitting problem. We list the five major and present method as (1) Moments (2) Maximum Likelihood of Probability Density Function (3) Regression of the Cumulative Distribution Function (4) Histogram and (5) Multivariate Kernel Density Estimation (or simply KDE).



4.37	3.87	4.00	4.03	3.50	4.08	2.25
4.70	1.73	4.93	1.73	4.62	3.43	4.25
1.68	3.92	3.68	3.10	4.03	1.77	4.08
1.75	3.20	1.85	4.62	1.97	4.50	3.92
4.35	2.33	3.83	1.88	4.60	1.80	4.73
1.77	4.57	1.85	3.52	4.00	3.70	3.72
4.25	3.58	3.80	3.77	3.75	2.50	4.50
4.10	3.70	3.80	3.43	4.00	2.27	4.40
4.05	4.25	3.33	2.00	4.33	2.93	4.58
1.90	3.58	3.73	3.73	1.82	4.63	3.50
4.00	3.67	1.67	4.60	1.67	4.00	1.80
4.42	1.90	4.63	2.93	3.50	1.97	4.28
1.83	4.13	1.83	4.65	4.20	3.93	4.33
1.83	4.53	2.03	4.18	4.43	4.07	4.13
3.95	4.10	2.72	4.58	1.90	4.50	1.95
4.83	4.12					

Figure 9: Eruption lengths (in minute) of 107 eruptions of Old Faithful geyser

Given the set of observations of data, the methods of moments is to compute the several higher order moments of data. Let us take the Figure 9 (from [2]) as example. We can compute the mean $= \mathrm{E}[\mathrm{X}] = 3.46$, variance $\sigma^2 = \mathrm{E}[(\mathrm{X}-\mu)^2] = 1.0721$, skewness $\gamma_1 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu^3}{\sigma^3} = \frac{\mathrm{E}[(X-\mu)^3]}{(\mathrm{E}[(X-\mu)^2]^{3/2})} = -0.6238$ and kurtosis $\gamma_2 = \frac{\kappa_4}{\kappa_2^2} =$

$\frac{\mu^4}{\sigma^4} - 3 = \frac{\mathrm{E}[(X-\mu)^4]}{(\mathrm{E}[(X-\mu)^2])^2} - 3 = 1.1242$. However with only these moments, we cannot directly construct how the shape of probability distribution actually is.

Maximum likelihood method at first gives the assumption distribution of the data, and then applies the maximum-likelihood estimation (MLE) to estimating the parameters of the statistical model. And if we first make the assumption of normal distribution of the data in Figure 9, we obtained the $\hat{\mu} = 3.46,$ and $\widehat{\sigma^2} = 1.0721$. Therefore the Normal distribution with $\mu = 3.46$ and $\sigma^2 = 1.0721$ is the result of the maximum likelihood method.

The regression of cumulative distribution function uses a transformation of the cumulative function so that a linear relation is found between the cumulative probability and the values of data. For example, let $P = \Pr(x \leq X)$ and the regression equation $Y = -\ln(-\ln P) = aX + b$. We can explore that 90% of this dataset lies between $L_1\!: Y = -0.64X + 0.07$ and $L_2\!: Y = -0.18X + 0.05$.

Histogram is very common to estimate the distribution of data. However histogram will suffer from choosing the proper bin width and origin. The two factor pose a great impact on the histogram result. The left hand side of Figure 10 uses the bin width = 0.5 and origin = 1.3 while the right hand side of Figure 10 uses bin width = 0.5 and origin = 1.5. We can see the difference of the distribution estimated by Figure 10.

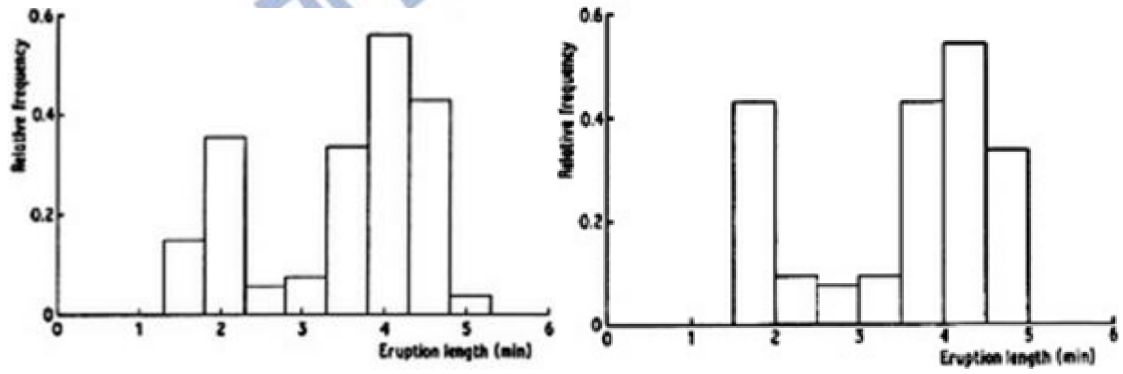


Figure 10: Histogram of Old Faithful geyser

Compared with aforementioned methods, multivariate kernel density estimation (or KDE) is the general version of histogram. In KDE we usually choose a proper symmetric kernel like standard normal distribution and a *bandwidth* parameter h to decide the shape when these observations combined in the distribution. Choosing the proper bandwidth parameter h is well studied in [21] and therefore choosing the parameters is reasonable. For example we choose h = 0.25 and the Figure 11 shows the distribution. Since KDE will accumulate the kernel of each observations on distribution, KDE have another advantage is what we desired. The distribution will reveal the shape of true distribution as the data observations getting larger. More observations for the KDE make the distribution more genuine to real distribution. Therefore these are the reasons that we choose KDE instead of the other five distribution fitting methods.

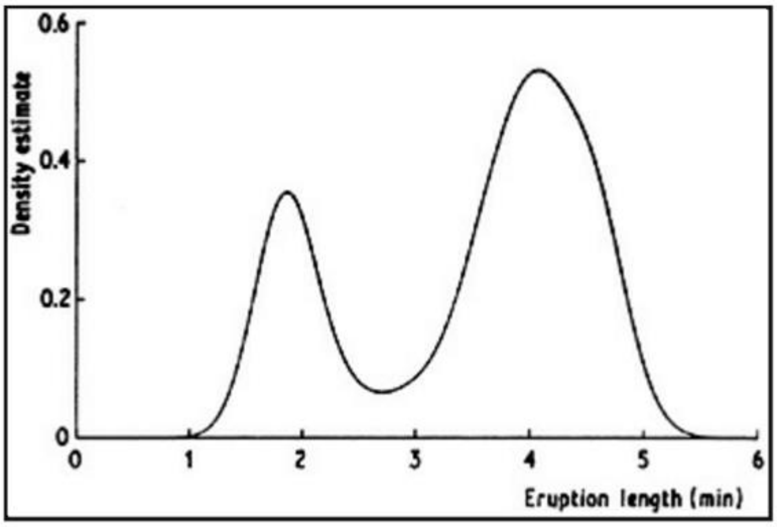
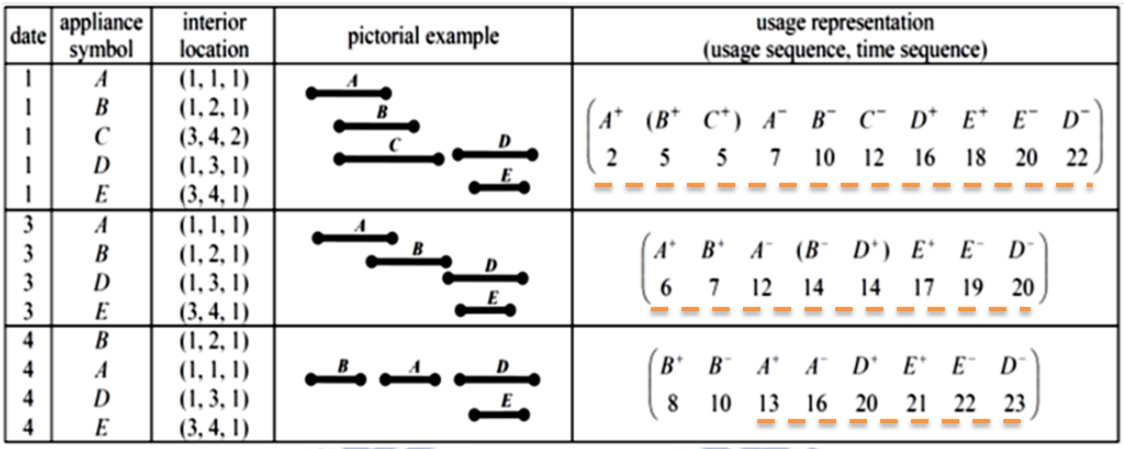


Figure 11: Kernel Density Estimation on h = 0.25

4.3 Probability-based Projected Database

Projected database is the set of subsequence in the original database with regarding to certain given prefix α. Therefore we usually call the local database as $\alpha-$projected database. In traditional method [26], the formal definition of the projected database is the collection of all the postfix of sequences with respect to prefix α. Let us take a concrete example for construct $\langle A^{+} \rangle -$projected database in the database of Figure 4. Now Figure 9 clearly indicate the $\langle A^{+} \rangle -$projected database by underlining the dashed orange line in the usage representation. For each sequence we will find the position of prefix. For example A^{+} is located at 1st position of sequence date = 1 and date = 3 while 3rd position of sequence date = 4. Therefore the $\langle A^{+} \rangle -$projected database, denoted as $D_{\langle A^{+}\rangle} = \{(1, \langle (B^{+}\ C^{+})\ A^{-}\ B^{-}\ C^{-}\ D^{+}\ E^{+}\ E^{-}\ D^{-} \rangle),$ $(3, \langle B^{+}\ A^{-}\ (B^{-}\ D^{+}) E^{+}\ E^{-}\ D^{-} \rangle),\ (4, \langle A^{-}\ D^{+}\ E^{+}\ E^{-}\ D^{-} \rangle)\}$.



date	appliance symbol	interior location	pictorial example	usage representation (usage sequence, time sequence)
1	A	(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	(1, 2, 1)		
1	C	(3, 4, 2)		
1	D	(1, 3, 1)		
1	E	(3, 4, 1)		
3	A	(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	(1, 2, 1)		
3	D	(1, 3, 1)		
3	E	(3, 4, 1)		
4	B	(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	(1, 1, 1)		
4	D	(1, 3, 1)		
4	E	(3, 4, 1)		

Figure 12: The example of projected database

However collecting all the postfix of every sequence may lead to outlier and noise sequences are also collected. In order to make our projected database more confidence on happening time information, we first observe that outlier sequence has the property of happening time of next endpoint of turned on of an appliance b^{+} is usually far away from the previous endpoint of turned on a^{+}. In order to ensure the time of each starting endpoint x^{+} are strongly time-connected, we evaluate the probability according to the probability function given in correlation pattern. We do not consider the finish endpoint y^{-} since the probability function should actually reflect the original time of starting endpoint y^{+}. Mathematically, the probability-based $\langle \alpha \diamond s \rangle$ −projected database is defined as

$$DP_{\langle \alpha \diamond s \rangle} = \{\langle sid_i, r \rangle \mid s \text{ is frequent starting endpoint in } D_{\langle \alpha \rangle},$$

$$D_{\langle \alpha \rangle} = \{\ \langle sid_i, v \rangle \in D, \alpha\ \sqsubseteq v\} \text{ and } \int_{t_\alpha}^{t_\alpha + \mu_S + \sigma_S} f_\alpha(x)dx > 0.34\ \} \qquad (7)$$

The reason why we choose 0.34 for the probability threshold is that Gaussian Normal Distribution from mean to variance is 0.34, i.e. $\int_0^1 \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx = 0.34$. Therefore, our projected database is different from original one and more fault tolerance and robustness to prevent containing the outlier sequences in database.

Let us see a concrete example for database of Figure 4. Consider the prefix to be $\langle D^+ \rangle$, we observed that E^+ is a frequent starting endpoints in $D_{\langle D^+ \rangle}$. Also the set $\{16, 8, 14, 20\}$ collects the time information of $\langle D^+ \rangle$ in $D_{\langle D^+ \rangle}$. The time information of E^+ is {18, 10, 17, 21}. We first evaluate the mean μ and variance σ for $t_{E^+} - t_{D^+}$. Therefore $\mu = \frac{(18-16)+(10-8)+(17-14)+(21-20)}{4} = 2$ and $\sigma = \sqrt{\frac{(18-16-\mu)^2+(10-8-\mu)^2+(17-14-\mu)^2+(21-20-\mu)^2}{4}} = 0.707$. Then the sequence will be considered in probability-based projected database $DP_{\langle D^+ E^+ \rangle}$ only if the $\int_{t_{D^+}}^{t_{D^+}+2+0.707} f_{D^+}(x)dx > 0.34$ for $t_{D^+} = 16, 8, 14$ and 20 . Therefore $DP_{\langle D^+ E^+ \rangle}$ collects the 4 sequences. The general concept for this example is depicted in Figure 13

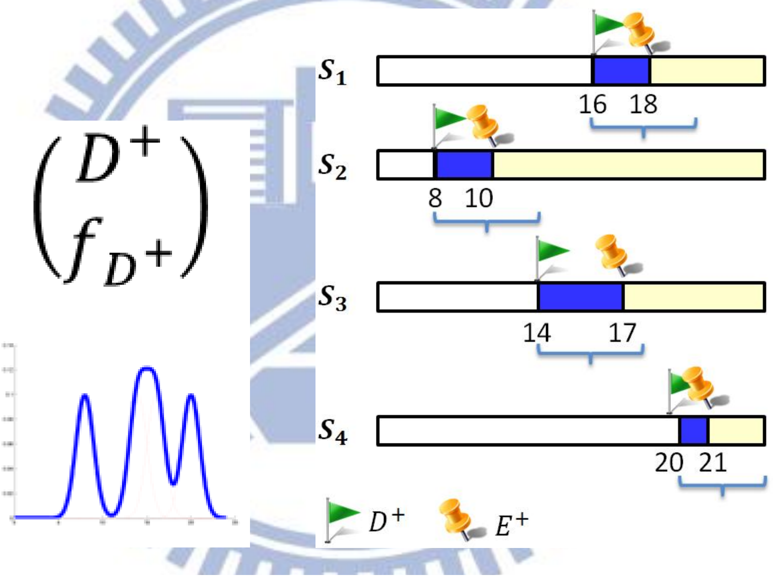


Figure 13: Example of probability-based projected database

4.4 CoPMiner Algorithm

To mine the correlation patterns, our algorithm CoPMiner is proposed. The main concepts of the CoPMiner are listed as follows: (1) **Pruning Database (pruneDB)**: Eliminate the faraway usage intervals in each of the usage sequence of the usage database by spatial similarity constraint (see Definition 4.1). (2) **Pruning Frequent Itemset of Endpoints (pruneFI)**: Eliminate the frequent endpoints that is not spatially related to our prefix α, and only collects those endpoints both frequent and related to prefix α. (3) **Pruning the Support Counting Space (pruneSup)**: Mark the positions of support counting bounds in $\alpha-$projected database according to prefix α. The detailed content of three strategies will be discussed in chapter 5.

Algorithm 1 illustrates the main framework of CoPMiner. Before we apply the **pruneDB** strategy, we need to collect those frequent but single usage interval out to form correlation patterns (line 1, algorithm 1). And then we apply the pruneDB strategy (line 2, algorithm 1) to reduce the numbers of intervals in database (line 3, algorithm 3). Then it transforms the usage database to usage representation and calculates the count of each usage point concurrently (line 3-4, algorithm 1). For each frequent starting usage point x, we find all its time information $T_x = \{t_{x_1}, t_{x_2}, \dots, t_{x_m}\}$ in DB and estimate the probability function $f_x(x)$ by Kernel Density Estimation in Definition 5 (lines 6-7, algorithm 1). Now we construct the probability-based $x-$projected database DB_x along with the **pruneSup** strategy is applied and attached in DB_x (line 8-9, algorithm 1). Then we can extend our prefix x to be a correlation pattern like form X (line 10, algorithm 1). Finally, we prepare the prefix X and its projected database DB_x to recursively explore the correlation patterns in further projected databases.

By borrowing the idea of the PrefixSpan [26], UPrefixSpan is developed with two search space pruning methods **pruneFI** and **pruneSup**. The pseudocode of the UPrefixSpan is shown in Algorithm 2. Since this is the recursive procedure, we need to first add prefix into correlation pattern set if the current prefix is ready to be (line 1-2, algorithm 2). For example, $\langle a^+ \rangle$ and $\langle a^+ \; b^+ \; a^- \rangle$ are not well-paired since there is at least one interval has only starting points (or respectively finish endpoints). Thus $\langle a^+ \; b^+ \; a^- \; b^- \rangle$ and $\langle a_1^+ \; (a_1^- \; b^+) \; (b^- \; a_2^+) \; a_2^- \rangle$ are well paired. After that for prefix Z,

Algorithm 1 **CoPMiner**(D, min_sup, min_sim)

```
Input: The usage database $D$
       The minimum support threshold $min\_sup$
       The minimum space similarity threshold $min\_sim$
Output: The set of all correlation patterns $L$
1: $L \leftarrow \{(\langle a^+a^- \rangle, \langle f_{a^+}, f_{a^-} \rangle) | a \in A, \langle a^+a^- \rangle$ is frequent pattern in $D\}$
2: $D \leftarrow$ pruneDB$(D, min\_sim)$
3: Let $DB$ be the usage representation of $D$                ▷ by Definition 4
4: $Freq \leftarrow \{x \in A, x$ is a frequent on-point in $DB$, i.e. $support(x) \geq min\_sup\}$
5: for all $x \in Freq$ do
6:     $T_x \leftarrow \{t_i | x$ happen at $t_i$ in $(sid_i, s_i) \in DB\ \}$     ▷ Collect time of $x$
7:     $f_x(x) \leftarrow \langle (K(x), h, T_x) \rangle$                          ▷ by Equation 6
8:     Construct the $x$-projected database, $DB_x$             ▷ by Equation 7
9:     $DB_x.Bounds \leftarrow$ pruneSup$(DB_x, x)$          ▷ Apply pruneSup strategy
10:    $X \leftarrow (\langle x \rangle, f_x)$                 ▷ Correlation pattern $X$ starts to grow
11:    $L \leftarrow$ UPrefixSpan$(DB_x, X, min\_sup, L)$
12: return $L$
```

Algorithm 2 **UPrefixSpan**(DB_z, Z, min_sup, L)

```
Input: The endpoint representation of $z$-projected database $DB_z$
       The prefix of correlation pattern $Z$
       The minimum support threshold $min\_sup$
       The set of previously found correlation patterns $L$
Output: The set of currently found correlation patterns $L$
1: if $Z$ is well-paired like balanced parentheses then
2:     $L \leftarrow L \cup \{Z\}$      ▷ output $Z$ if $Z$ is ready to be a correlation pattern
3: $Freq \leftarrow \{x \in A, x$ is a frequent endpoint in $DB_z\ \}$
4: $Freq \leftarrow$ pruneFI$(z, Freq, min\_sim)$
5: for all $x \in Freq$ do
6:     $q \leftarrow z \diamond x$                          ▷ Extends $x$ to the pattern $z$ to be $q$
7:     $T_x \leftarrow \{t_i | x$ happen at $t_i$ in $(sid_i, s_i) \in DB_z\ \}$     ▷ Collect time of $x$
8:     $f_x(x) \leftarrow \langle (K(x), h, T_x) \rangle$                          ▷ by Equation 6
9:     Construct the $q$-projected database, $DB_q$             ▷ by Equation 7
10:    $DB_q.Bounds \leftarrow$ pruneSup$(DB_q, q)$          ▷ Apply pruneSup strategy
11:    $Q \leftarrow (\langle z \diamond x \rangle, Z.F(x) \diamond f_x(x))$      ▷ Grows correlation pattern $Z$
12:    if $|DB_q| \geq min\_sup$ then
13:        $L \leftarrow$ UPrefixSpan$(DB_q, Q, min\_sup, L)$
14: return $L$
```

UPrefixSpan scans its projected database DB_Z once to discover all local frequent usage points as $Freq$ (line 3, algorithm 2) and remove some usage points unrelated to prefix z (line 4, algorithm 2). For frequent usage point x, we can append it to original prefix z to generate a new frequent sequence q with the length increased by 1 (line 6, algorithm 2). We also use again the time information of x in DB_x (line 7, algorithm 2) for the purpose of estimating the probability function $f_x(x)$ (line 8, algorithm 2) by Definition 5. And then we obtain the new prefix q and construct the $q-$projected database DB_q (line 9, algorithm 2) and update the set *Bounds* for pruneSup strategy (line 10, algorithm 2). As the method of pattern growth, the prefixes are appended and extended as Q (lines 11, algorithm 2). Finally, we can discover further correlation patterns by constructing the next projected database with at least minimum support size of database to keep extending prefix and recursively running until the prefix cannot be extended anymore or database is infrequent (lines 12-1, algorithm 2).

We take the database in Figure 4 with $min\_sup = 2$ and $min\_sim = 0.4$ as an example. There are 17 usage intervals which can be regarded as 4 usage interval sequences in the database. After transforming database into usage representation, we can find all frequent usage points. They are $\{(A^{+}:3),\ (A^{-}:3),\ (B^{+}:4),\ (B^{-}:4),\ (C^{+}:1),\ (C^{-}:1),\ (D^{+}:4),\ (D^{-}:4),\ (E^{+}:1),\ (E^{-}:1),\ (F^{+}:1),\ (F^{-}:1)\}$ where the notation (endpoint : count) represents the endpoint and its associated support in projected database. If now we extend the prefix to be $\langle A^{+}\rangle$, then the support count without pruneSup strategy is $\{(A^{-}:3),\ (B^{+}:2),\ (B^{-}:2),\ (C^{+}:1),\ (C^{-}:1),\ (D^{+}:3),\ (D^{-}:3),\ (E^{+}:1),\ (E^{-}:1)\}$ while with **pruneSup** strategy becomes $\{(A^{-}:3),\ (B^{+}:2),\ (C^{+}:1)\}$. From now on if we extend the usage point A^{-} to prefix to form $\langle A^{+}\ A^{-}\rangle$, then our correlation pattern for single interval A is completed. Therefore, $\begin{pmatrix} A^{+} & A^{-} \\ f_{A^{+}} & f_{A^{-}} \end{pmatrix}$ is the correlation pattern form and since the time information of A^{+}, $T_{A^{+}} = \{2, 6, 13\}$. The probability function $f_{A^{+}}(x) = \frac{1}{3\mathrm{h}\sqrt{2\pi}}\left(e^{-\frac{1}{2}\left(\frac{x-2}{h}\right)^{2}} + e^{-\frac{1}{2}\left(\frac{x-6}{h}\right)^{2}} + e^{-\frac{1}{2}\left(\frac{x-13}{h}\right)^{2}}\right)$ where $h = \frac{range(\{2,6,13\})}{\sqrt{3}} = \frac{13-2}{\sqrt{3}} = 6.35$. Likewise the probability function $f_{A^{-}}(x) = \frac{1}{3\mathrm{h}\sqrt{2\pi}}\left(e^{-\frac{1}{2}\left(\frac{x-7}{h}\right)^{2}} + e^{-\frac{1}{2}\left(\frac{x-12}{h}\right)^{2}} + e^{-\frac{1}{2}\left(\frac{x-16}{h}\right)^{2}}\right)$ where $h = 5.19$.

Algorithm 3 **pruneDB**(D, min_sim)

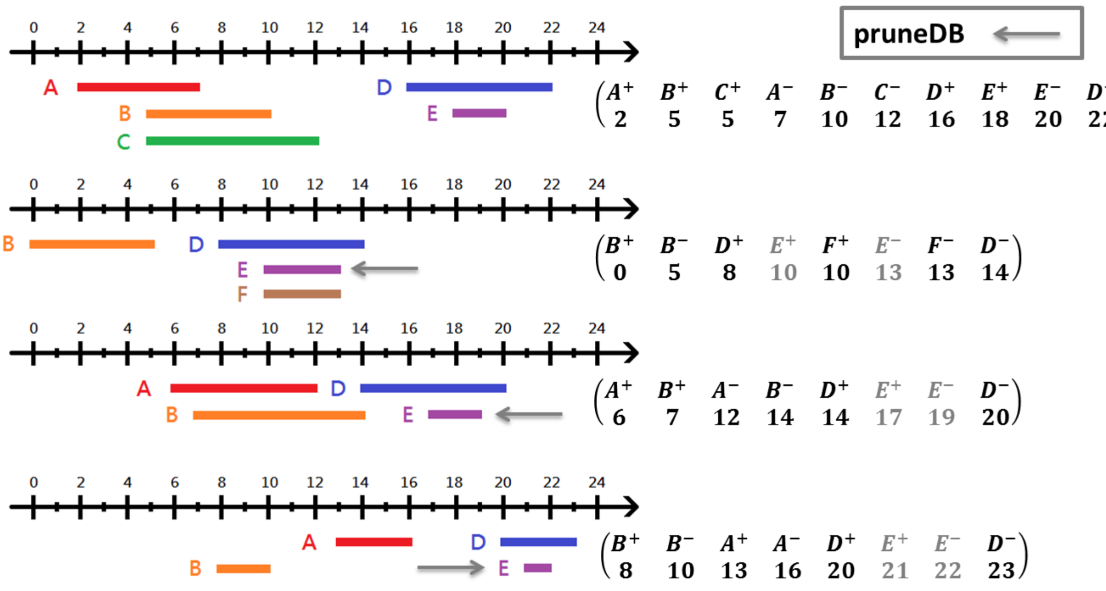
```
Input: The usage database $D$
       The minimum space similarity threshold $min\_sim$
Output: The usage database $DB$ after pruneDB strategy
1: $DB \leftarrow \emptyset$
2: for all $(sid_i, v_i) \in D$ do                    ▷ Pruning faraway intervals
3:     $u_i \leftarrow \{(e_j, s_j, f_j) \in v_i | \exists(e_k, s_k, f_k) \in v_i, e_k \neq e_j$, s.t. $similarity(e_j, e_k) \geq min\_sim\}$
4:     $DB \leftarrow DB \cup \{(sid_i, u_i)\}$.
5: return $DB$
```

Algorithm 4 **pruneFI**$(z, Freq, min\_sim)$

```
Input: The prefix $z$
       $Freq$ is the set of frequent endpoints of $DB_z$
       The minimum space similarity threshold $min\_sim$
Output: The frequent endpoints set $FI$ after pruneFI strategy.
1: $FI \leftarrow \emptyset$
2: for all $x \in Freq$ do
3:     $isAppend \leftarrow false$, $isSpace \leftarrow false$
4:     Let $x = e^+$ or $e^-$, $e \in E$.
5:     $h \leftarrow |\{e^+ \in z\}|$                  ▷ The start endpoint of $e$ in $z$
6:     $t \leftarrow |\{e^- \in z\}|$                  ▷ The finish endpoint of $e$ in $z$
7:     if $(x = e^+ \wedge h \geq t) \vee (x = e^- \wedge h > t)$ then
8:         $isAppend \leftarrow true$
9:     if $\forall y \in Eid_z$, s.t. $similarity(e, y) \geq min\_sim$ then
10:        $isSpace \leftarrow true$
11:    if $isAppend \wedge isSpace$ then
12:        $FI \leftarrow FI \cup \{x\}$
13: return $FI$
```

```
Algorithm 5 pruneSup($DB_z, z$)
Input: The $z$-projected database $D$ in usage representation
       The prefix $z$
Output: The set of support counting bounds, $Bounds$ after pruneDB strategy
1: $Bounds \leftarrow \{(sid_i, \{length\}) \mid length = |r_i|, (sid_i, r_i) \in DB\}$
2: for all $e^+ \in z$ do                                ▷ Pruning faraway intervals
3:     if $e^+$ does not have corresponding $e^-$ in $z$ then
4:         $m \leftarrow$ the position of corresponding $e^-$ in each $(sid_i, r_i) \in DB_z$
5:         $Bounds.sid_i \leftarrow Bounds.sid_i \cup \{m\}$
6: return $Bounds$
```

Now let us illustrate the mining algorithm by a quick example. Suppose that given usage database in Figure 4 with the minimum support $min\_sup = 2$ and $min\_sim = 0.3$. The further example clearly and down to earth illustrates how we perform the CoPMiner algorithm step by step. First, Figure 14 clearly shows the database with time axis attached (left part) and the usage representation (right part). This result is the snapshot of after implementing line 1-3 of algorithm 1 in CoPMiner. Also we show that what usage intervals is pruned by **pruneDB** in the arrow. The pruneDB strategy prunes the intervals E in date = 2, 3 and 4.



pruneDB
0 2 4 6 8 10 12 14 16 18 20 22 24
A B C D E F
(A+ 2, B+ 5, C+ 5, A− 7, B− 10, C− 12, D+ 16, E+ 18, E− 20, D− 22)
(B+ 0, B− 5, D+ 8, E+ 10, F+ 10, E− 13, F− 13, D− 14)
(A+ 6, B+ 7, A− 12, B− 14, D+ 14, E+ 17, E− 19, D− 20)
(B+ 8, B− 10, A+ 13, A− 16, D+ 20, E+ 21, E− 22, D− 23)

Figure 14: Mining Example 1/5

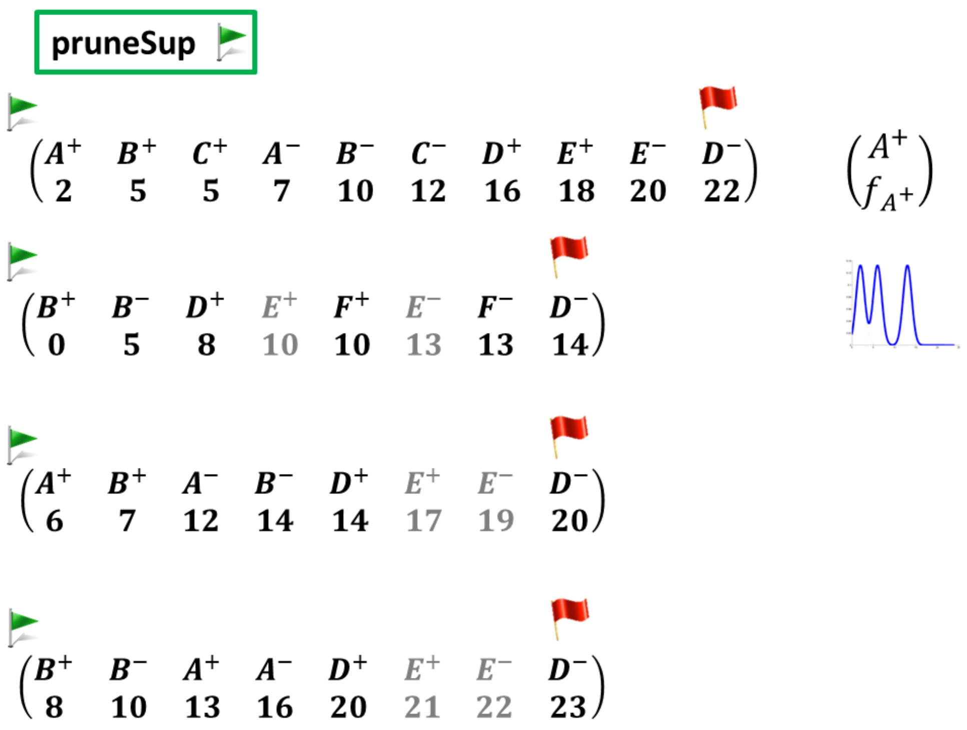


Figure 15: Mining Example 2/5

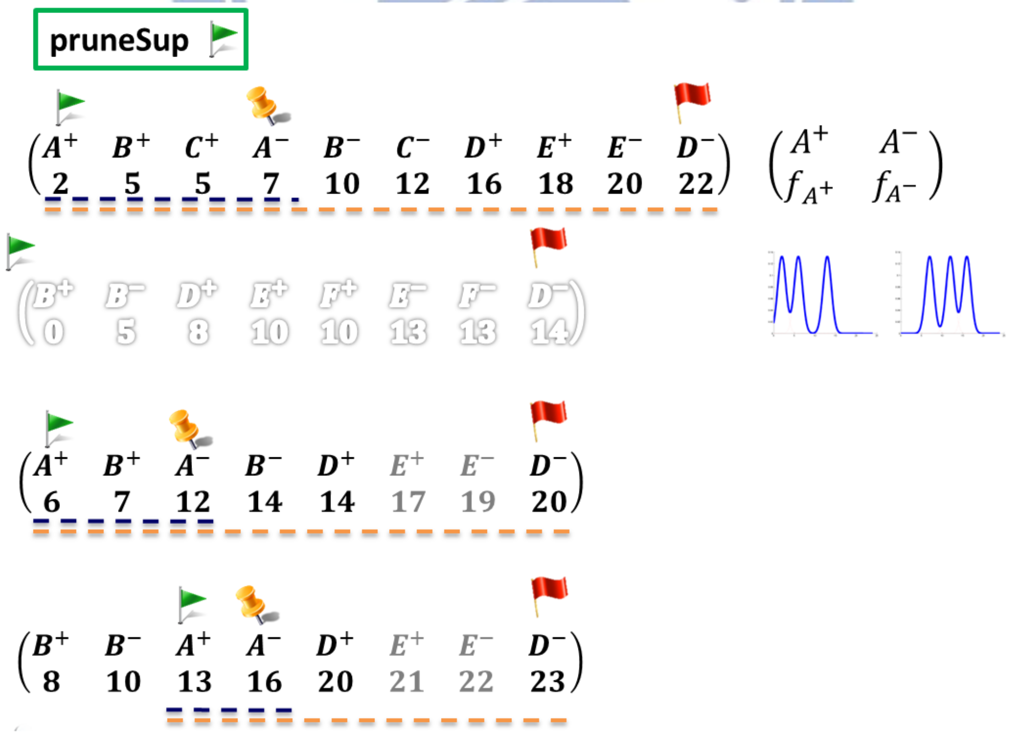


Figure 16: Mining Example 3/5

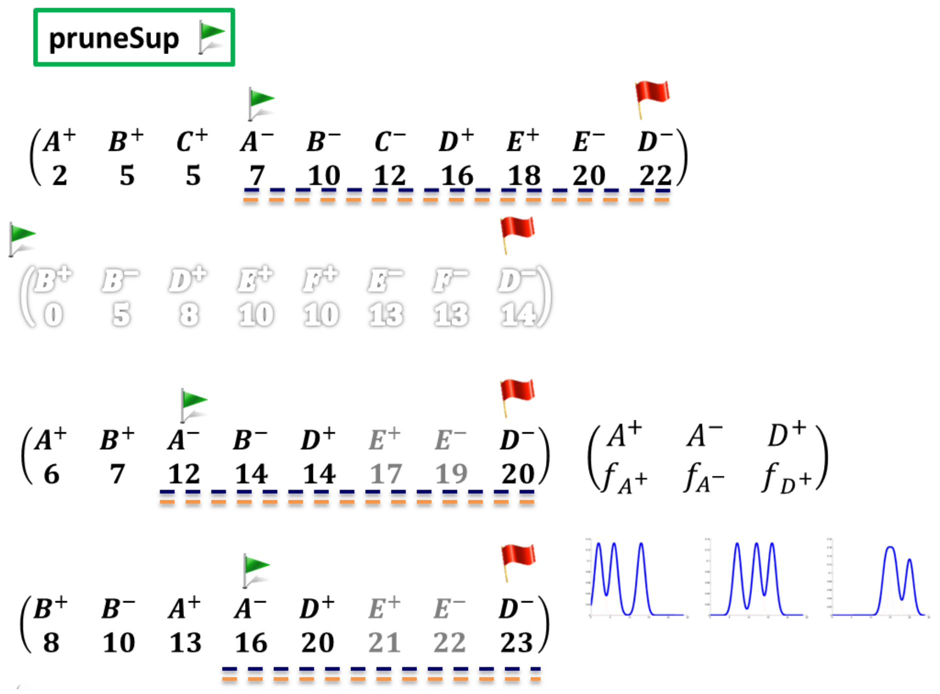


Figure 17: Mining Example 4/5

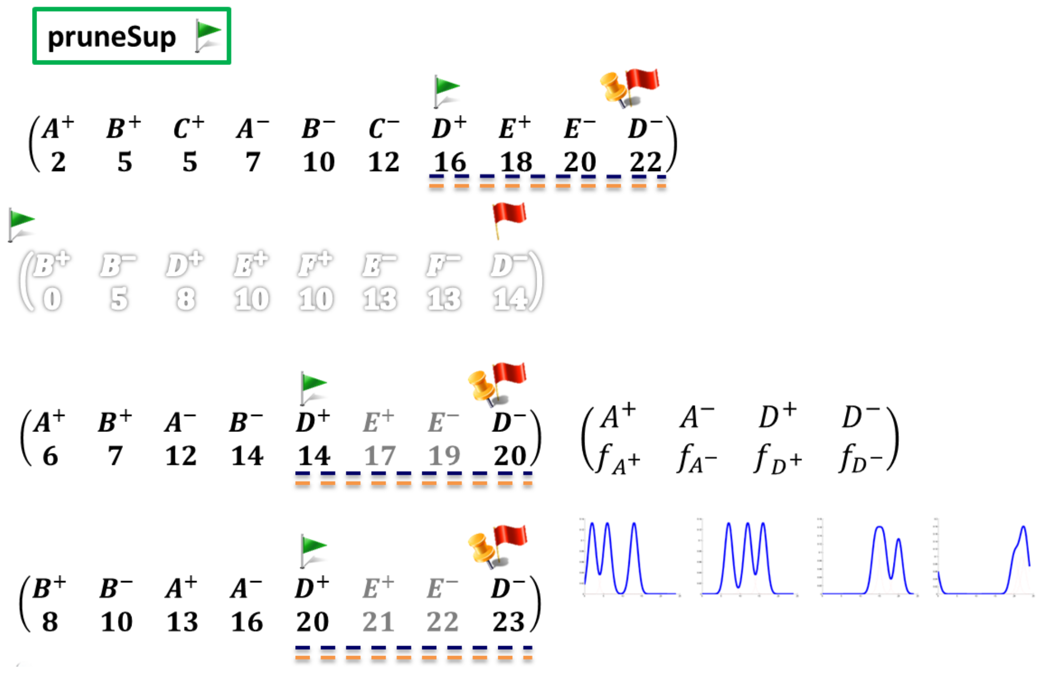


Figure 18: Mining Example 5/5

Now we obtain the support of Figure 14 as $\{(A^{+}:3), (A^{-}:3), (B^{+}:4), (B^{-}:4), (C^{+}:1), (C^{-}:1), (D^{+}:4), (D^{-}:4), (E^{+}:1), (E^{-}:1), (F^{+}:1), (F^{-}:1)\}$. And we will only construct the four projected databases respectively with prefix = $\{A^{+}, B^{+}, D^{+}, E^{+}\}$. Furthermore we extend the prefix A^{+} in Figure 14. Figure 15 shows the projected database of $\langle A^{+}\rangle$, $D_{\langle A^{+}\rangle}$. And the green triangle flags indicates the position of prefix $\langle A^{+}\rangle$ while the yellow thumbtack indicates the *Bounds* set. The red square flag indicates the *length* of sequence in database. The orange underlined part is the full $D_{\langle A^{+}\rangle}$ while the blue underlined part is the support counting space. Therefore, we obtain the support of Figure 15 as $\{(A^{-}:3), (B^{+}:2), (C^{+}:1)\}$. Now we choose to extend A^{-} and construct $D_{\langle A^{+}\,A^{-}\rangle}$. Meanwhile the correlation pattern for $\langle A^{+}\,A^{-}\rangle$ is added to output set. As $D_{\langle A^{+}\,A^{-}\rangle}$ is shown in Figure 16, we again obtain the support as $\{(B^{-}:2), (C^{-}:1), (D^{+}:3), (D^{-}:3), (E^{+}:1), (E^{-}:1)\}$. If we again choose to extend D^{+}, the $D_{\langle A^{+}\,A^{-}\,D^{+}\rangle}$ is shown in Figure 17. Now it has only the D^{-} as the frequent endpoints in the support $\{(D^{-}:3), (E^{+}:1), (E^{-}:1)\}$. As CoPMiner goes to $D_{\langle A^{+}\,A^{-}\,D^{+}\,D^{-}\rangle}$, it finds that there is no sequences so CoPMiner has to track back and explore other correlation patterns.

Chapter 5

Efficient Strategies for CoPMiner Algorithm

In this chapter, we focus on describing the detail and the efficient strategies used in our algorithm, CoPMiner. We will also give the conceptual and theoretical viewpoint along with the implementation method for each strategies. The **pruneDB** strategy is used before mining process to explore the correlation pattern satisfying spatial similarity constraint .The **pruneFI** strategy is applied in recursive mining process to explore the prefix with spatial similarity. The **pruneSup** strategy is the most significant, outstanding and only restricted to the usage representation. The pruneSup strategy can be said to be the most independent in mining algorithm. In Section 5.1, we discuss the pruneDB. In Section 5.2, we mention the pruneFI. In Section 5.3, we introduce the pruneSup strategy.

5.1 Pruning by Spatial Constraint – **pruneDB**

Now we clearly state how the strategy **pruneDB** works here. This strategy is to pruning the usage intervals previously by spatial constraint. The main concept of **pruneDB** is to eliminate the faraway intervals in the usage database.

In algorithm 3 **pruneDB**, for each usage sequence v_i of usage database, we only collect the usage interval (e_j, s_j, f_j) where there exists another event symbol e_k that is spatially close enough to e_j. This indicates that usage intervals e_j and e_k are possible to form the correlation pattern that we expected to see. That is, we collect those usage intervals that they are not relatively standing alone in the sequence v_i. This for loop in line 2 to 4 of algorithm 3 **pruneDB** discards the faraway intervals (e_w, s_w, f_w) that there is no other intervals (e_k, s_k, f_k) close to e_w in v_i. We know that e_w never appears as a frequent pattern with other intervals since the interval (e_w, s_w, f_w) is relatively stand alone in the event sequence v_i. Thus for each sequence v_i in *D*, we prune these faraway intervals (e_w, s_w, f_w) and only remain the intervals (e_j, s_j, f_j) that they are possible to form correlation patterns. Hence line 4 to 5 in algorithm 3 return the usage database *DB* after applying the pruneDB strategy.

In order to accomplish the concept of pruneDB, we add an attribute *Space_id* in our database and assign *Space_id* for each interval. Therefore we illustrate how pruneDB works in our example database *DB* in Figure 14. In the sequence S_2 of $sid = 2$, the usage interval $(E, 10{:}20, 13{:}10)$ is a faraway interval in S_2. That is, $\{\{B, D, F\}, \{E\}\}$ is the space-related set of $min\_sim = 0.6$. Therefore we assign *Space_id* = 2 in the other 3 intervals of S_2 and assign interval $(E, 10{:}20, 13{:}10)$ with *Space_id* = -1 for deprecated. The usage intervals in S_3 of $sid = 3$ are similar to $sid = 2$, (i.e. $\{\{A, B, D\}, \{E\}\}$). The gray intervals which is pointed by arrow is selected to be the faraway intervals and therefore is eliminated after pruneDB strategy. The Figure 14 shows the usage sequences of database after applying pruneDB strategy.

5.2 Pruning by Usage Points in Prefix – **pruneFI**

Different from pruneDB solving the problem by rearrangement the usage intervals in sequences of usage database, **pruneFI** strategy solves the problem from the viewpoint of frequent endpoints. The concept of **pruneFI** strategy is that we only extend the frequent points related to our current prefix. This strategy will also ensure the intervals in the prefix are always close enough in space.

Given the prefix z and $z-$ projected database D_z, the set of frequent endpoints may be large. We only select part of the frequent endpoints related to prefix z. Those frequent endpoints but not related to prefix z will be temporarily avoid to construct the projected database. This strategy decreases the number of projected databases potentially to be find. Therefore we avoid some projected database that the prefix is never been a correlation pattern. Hence we only construct the projected databases whose prefixes are possible to be correlation pattern.

The **pruneFI** stratgey is depicted in algorithm 4. In algorithm 4, we first check whether the frequent endpoint x is appendable to prefix z in line 4 to 8. The spactial similarity is examined in line 9 to 10. We collect those frequent endpoints that is both appendable and spatial related to prefix in line 11 to 12.

We indicate our concept of **pruneFI** with $D_{\langle A^+\rangle}$ in Figure 16. Given the minimum support $min\_sup = 2$ and $min\_sim = 0.6$, The traditional method PrefixSpan will find out 7 frequent endpoints, they are $\{A^-, B^+, B^-, D^+, D^-, E^+, E^-\}$. Then CTMiner will find out 4 frequent points $\{A^-, B^+, D^+, E^+\}$. Our proposed method will only consider two frequent endpoints $\{A^-, B^+\}$. If we construct the $D_{\langle A^+A^-\rangle}$, $\{D^+, D^-, E^+, E^-\}$ will be extended in PrefixSpan while our method will find out that no other endpoints need to be extended. Since **pruneFI** strategy is applied in recursive mining process, our strategy can effectively decrease the degree of projected databases need to create after each recursion. In previous mining example, $D_{\langle\,\rangle}$ in Figure 15 will extend $\{A^+, B^+, D^+\}$. Figure 16 of $D_{\langle A^+\rangle}$ extends $\{A^-, B^+\}$ and Figure 17 of $D_{\langle A^+A^-\rangle}$ seeks the empty set.

5.3 Pruning by Prefix in Projected Databases – **pruneSup**

In traditional PrefixSpan algorithm [PrefixSpan], the data has only pointwise information. However, our data now exist the idea of point time of starting and finish. Given the prefix z, counting the support of endpoints in $z-$ projected database D_z is a routine procedure. We can use the endpoints information to make great advance on the support counting procedure.

The main concepts of **pruneSup** strategy are listed as follows: (1) **Pseudo-projection**: First we number the position of usage points in sequence as natural numbers $\mathbb{N} = \{1, 2, \ldots\}$. And for each sequence in $z-$projected database D_z, we record the prefix z 's last position as an integer $p \in \mathbb{N}$ rather than construct the actual projected suffix sequence. (2) **Midway Termination**: For each sequence in

z −projected database D_z, we record the positions $Bounds$ (in pseudo-projection) of counting bounds according to the prefix z 's position in the sequence itself.

We depict the concepts in Figure 19 and the detail content is in algorithm 5. Suppose that there are k sequences, named $S_{n_1}, S_{n_2}, \ldots, S_{n_k}$, in p −projected database D_p. The green triangle flags indicate the last positions of prefix p in the sequence. The red square flags indicate the end of the sequence. The yellow thumbtacks indicate the counting bounds with respect to the prefix p. If we want to count the frequent endpoint for D_p, traditionally we need to visit the whole sequences. That is, for $S_{n_1}, S_{n_2}, \ldots, S_{n_k}$ in D_p, we must visit the endpoints between green triangle flag and red square flag (or equivalently the blue part and the light yellow part). But now we have the yellow thumbtacks (the information is obtained from prefix p). Therefore **pruneSup** strategy is that we visit the endpoints between green triangle flags to the yellow thumbtacks nearest to green flags (or equivalently the blue part) for each sequence in D_p.

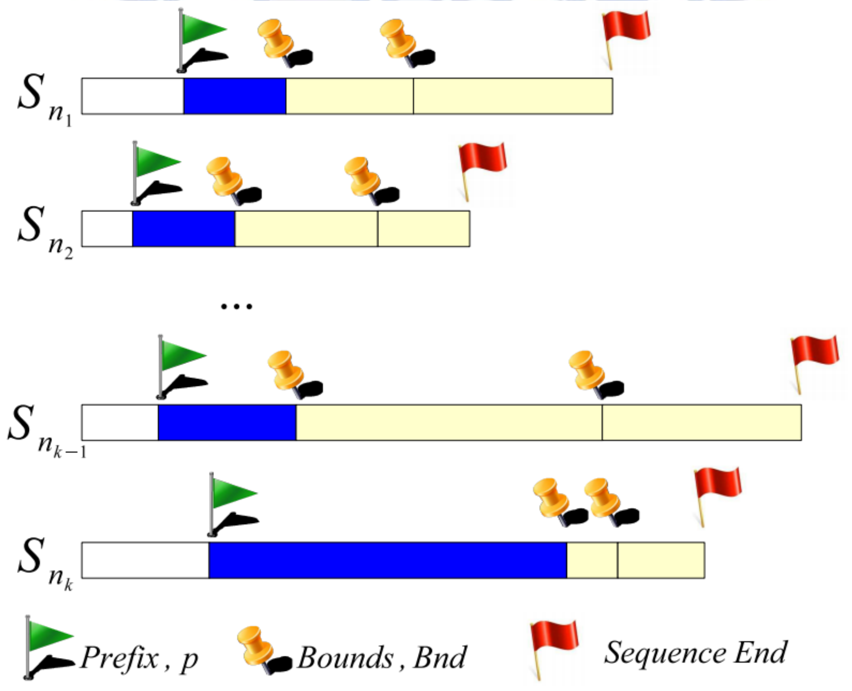


Figure 19: The concept of **pruneSup** strategy

The **pruneSup** strategy is effective since although frequent endpoints might be hidden for some prefixes, but the frequent endpoints definitely appear in certain shorter or longer prefixes. Also, this strategy is efficient since we visit only part of the sequence if the sequence has many usage intervals (like visiting little endpoints of $S_{n_{k-1}}$ in Figure 19).

Let's see a concrete example to explicitly show how this strategy works in the usage database D in Figure 4. And we give $min\_sup = 2$ and $min\_sim = 0.1$, the $\langle A^+ \rangle -$ projected database is $D_{\langle A^+ \rangle} = \{(1, (B^+ \; C^+) \; A^- \; B^- \; C^- \; D^+ \; E^+ \; E^- \; D^-),$ $(3, B^+ \; A^- \; (B^- \; D^+) \; E^+ \; E^- \; D^-), (4, A^- \; D^+ \; E^+ \; E^- \; D^-)\}$.

Without the **pruneSup** strategy, we count all the endpoints in $D_{\langle A^+ \rangle}$. So we obtain the support count of endpoints are $\{ (A^-: 3), (B^+: 2), (B^-: 2), (C^+: 1), (C^-: 1), (D^+: 3), (D^-: 3), (E^+: 3), (E^-: 3)\}$. Then we have four projected databases for further mining process and these prefixes are $\{\langle A^+ \; A^- \rangle, \langle A^+ \; B^+ \rangle, \langle A^+ \; D^+ \rangle, \langle A^+ \; E^+ \rangle\}$. However we can notice that mining $\langle A^+ \; D^+ \rangle -$ projected database can never extend the prefix to form a reasonable pattern since the endpoint A^- is always before D^+ in $D_{\langle A^+ \rangle}$.

We explain how to apply **pruneSup** strategy here. First, we mark the prefix positions and bounds positions for each sequence in $D_{\langle A^+ \rangle}$. Explicitly, the set $Bounds_{\langle A^+ \rangle} = \{ (1, \; 1, \; \{4, 10\}), \; (3, \; 1, \; \{3, 8\}), (4, \; 3, \; \{4, 8\}) \}$ records (sid, prefix's last position p, bounds positions $Bounds$) for each sequence in $D_{\langle A^+ \rangle}$. Now the **pruneSup** strategy only visits the endpoints from $p + 1$ to $\min(Bounds)$. So we visit $\{ B^+, C^+, A^-\}$ in $sid = 1$, $\{ B^+, \; A^-\}$ in $sid = 3$ and $\{ \; A^-\}$ in $sid = 4$. Therefore we obtain the support count of endpoints are $\{ (A^-: 3), (B^+: 2), (C^+: 1)\}$. Then we only consider the two prefixes $\{\langle A^+ \; A^- \rangle, \langle A^+ \; B^+ \rangle\}$.

Strategy **pruneSup** accelerates the mining process since it decreases the number of prefixes and projected databases. We explicitly list some bound sets for \$DB\$ as following. The $Bound_{\langle \rangle} = \{(1, 0, \{10\}), \; (2, 0, \{8\}), \; (3, 0, \{8\}), \; (4, 0, \{8\}) \}$, $Bound_{\langle A^+ A^- \rangle} = \{(1, \; 4, \{10\}), \; (3, \; 3, \{8\}), \; (4, \; 4, \{8\})\}$, $Bound_{\langle A^+ B^+ \rangle} = \{(1, \; 2, \{ 4, 5, 10\}), \; (3, \; 2, \{ 3, 4, 8\})\}$, $Bound_{\langle D^+ E^+ E^- \rangle} = \{(1, \; 9, \{10\}), \; (2, \; 6, \{8\}), \; (4, \; 7, \{8\})\}$.

Chapter 6

Applications – Anomaly Detection

In this chapter, we will briefly discuss the application of mining correlation pattern. As mentioned in chapter 1 Introduction, the residents can understand more about their behaviors of using appliances in the smart home. We are expected to see that our correlation patterns can explore where and how we use precious electricity for residents. Furthermore, with useful knowledge we can provide the anomaly behavior detection and activity prediction. We will briefly put our eyes on anomaly detection first. And we will also list some issue and propose a method to detect the anomaly.

6.1 Problem Definition of Anomaly Detection

Consider the regular home with four members, Dad, Mom, Son and Daughter. The family members have distinctive behaviors and we roughly state their own conditions. Dad is an engineer and usually goes to work at day but sometimes maybe to burn the midnight oil to finish some works and jobs. Therefore Dad can be regarded as a regular behavior at daylight but may have some occasional cases at night. Mom is a teacher in the senior high school and usually has the most regular behavior of daylight and night. Son is now an undergraduate student studying at other county and will come back home about once per month. Daughter is studying in a junior high school and need to prepare for her important exam this summer semester. Therefore, our system can perform some convenient intelligence for this home. For example, Dad usually tends to have a shower when he comes back to home at almost midnight. At that time, it is very likely that all other members have already taken shower and go to bed. So Dad turn on the electric heater for bathroom and usually he will also turn off after one hour. If a condition is that Dad forgets to turn off the electric heater and goes to bed since the entire day work, our system could detect the anomaly that the electric heater is still keep running with other appliances are turned off. And for this case, it seems that our system turns off the electric heater is a good way. Consider another scenario for Mom and daughter at day. After the family having breakfast together, they usually turned off the television and light tand then go to supermarket. If one day Mom and Daughter are not at home but the light or the television is turned on at morning. It should investigate that whether the home has been invaded or simply just the Son comes back. This could be achieved by providing the camera of the door sensors and send the pictures or messages to the home members to notify this kind of abnormal behaviors.

Therefore, how to detect the anomaly behavior is still not an easy task. In order to make this problem solvable, we formally define our anomaly detection problem as the following. We also give the abstract concept in Figure 20.

Given the usage database, the set of correlation patterns also have been discovered and input a user's behavior of certain time, determine the anomaly behavior of the turn on/off query behavior with user specified anomaly threshold min_pro.

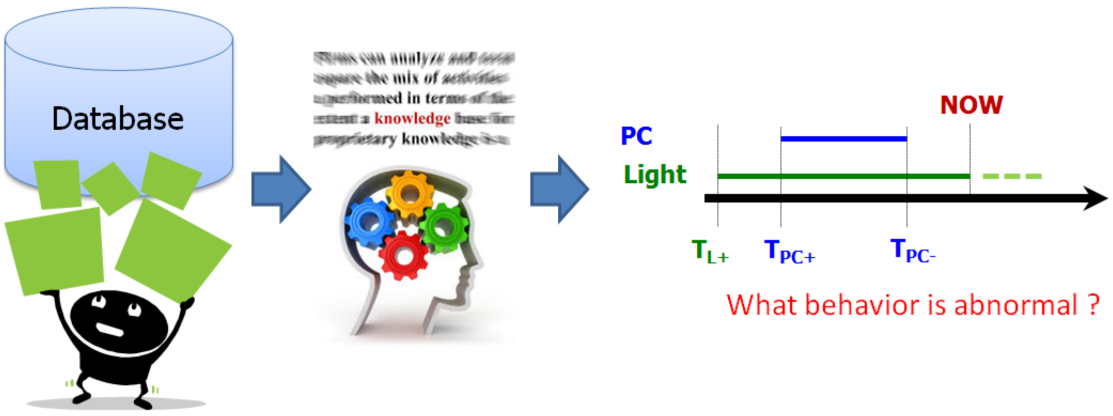


Figure 20: Abstraction of anomaly detection

6.2 Anomaly Threshold Setting

However there are some issues for anomaly detection. One is that the anomaly threshold setting. Due to the diversity of members' behaviors, the definition of anomaly behaviors may differ from people to people. For example, Dad usually comes home at 20:00 but sometimes comes home 23:30 if he has some projects to finish. But for Mom the three hour difference in day and night are strongly different. Therefore the anomaly setting issue is intuitively a user sensitive parameter problem.

We also give an example of input for anomaly detection in Figure 21. Given the query behavior like Figure 21, light A is used from 7:55 to 11:11 and light B is used from 10:10 to 15:00, we want to answer what turn on/off behaviors are normal or anomaly.

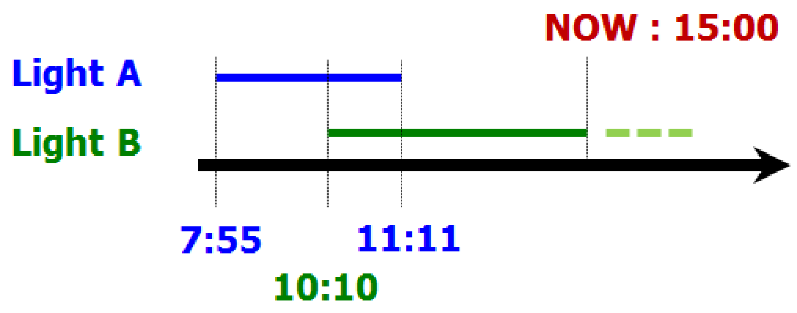


Figure 21: Example input of anomaly detection system

6.3 Fault Tolerance of Time

Another issue is that our system should have the fault tolerance of time and query. That is, similar queries should have similar outputs. This issue comes from the uncertainty of human beings. The clock will always ring at the certain fixed time but our behaviors will not happen so exactly as the clock. The fault tolerance of time is also a consideration for the detection system. The stability of the detection system will make the system more durable with different behaviors and more actually fit into the life of home members, just like we propose the probability concepts into the correlation patterns.

6.4 Anomaly Detection

Due to the two considerations, we can propose amethod for anomaly detection based on our correlation pattern. Since the user will input the time of the query behavior, we can evaluate the probability by the following equation.

$$Area = \frac{1}{|P|} \sum_{q \in P} \int_{T_S - \sigma_{P_S}}^{T_S + \sigma_{P_S}} f_S(x)dx \qquad (8)$$

where the P is the correlation pattern set, T_S is the input query time and σ_{P_S} is the variance of probability function $f_S(x)$.

We would test whether the averaged probability area is higher than the user given threshold min_pro to decide its anomaly behavior. Different correlation pattern would have its own $f_S(x)$ and σ_{P_S}. Also we considers the probability of happen within period of time, for example $10{:}10 \pm 1{:}00$ and $10{:}10 \pm 0{:}50$ would be the time for two correlation patterns. Therefore we say that our system has the fault tolerance of time. To decide the anomaly behavior, we could say that if $Area < min\_pro$ then the probability of happening the behavior at the certain time is not high enough and therefore the turn on/off behavior is regarded as anomaly behavior. Otherwise, we thought that we do not have sufficient information and confidence to say that the behavior is abnormal, then we regard it as a normal behavior.

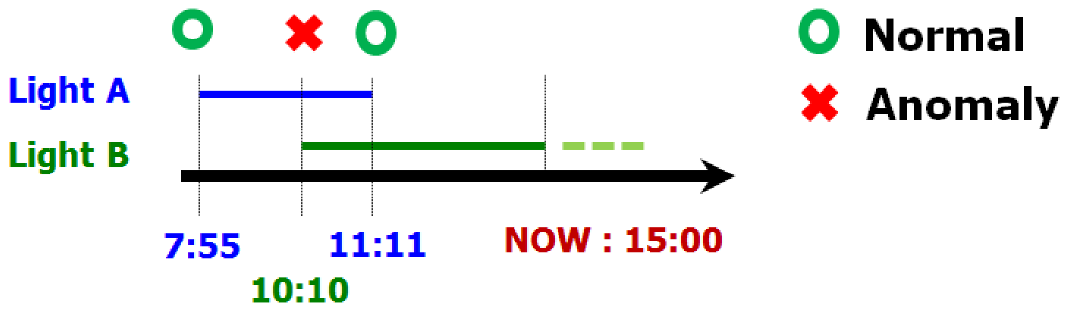


Figure 22: Example output of anomaly detection system

Chapter 7

Experiment Evaluations and 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 [5], IEMiner [25] and TPrefixSpan [30] 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.

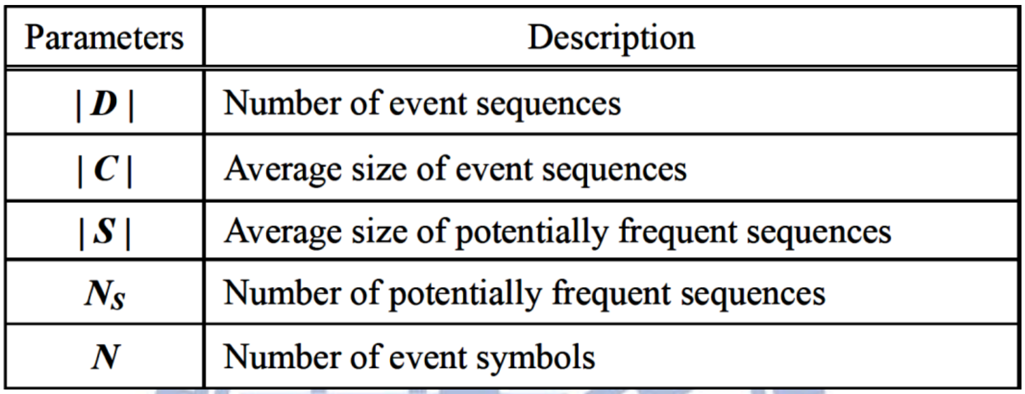
The performance study has been conducted on both synthetic and real world datasets. First, we compare the execution time using synthetic datasets at different minimum support. Second, we conduct an experiment to observe the memory usage and the scalability on execution time of CoPMiner. In addition to using synthetic datasets, we also have performed an experiment on real-world dataset to indicate the applicability of correlation pattern mining. Finally, CoPMiner is applied in real-world dataset to show the performance and the practicability of mining correlation patterns.

7.1 Synthetic Dataset

In order to evaluate the scalability of our algorithm, we use two kinds of synthetic dataset generator. The first synthetic generator is to evaluate the scalability of database size and minimum support threshold value. The second kind is to evaluate the scalability as the length of each record grows.

First Kind Synthetic Data Generator

The first kind synthetic datasets in the experiments are generated using synthetic generation program modified from [26]. 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 temporal data generator is shown in Figure 23.



Parameters	Description
$\|\boldsymbol{D}\|$	Number of event sequences
$\|\boldsymbol{C}\|$	Average size of event sequences
$\|\boldsymbol{S}\|$	Average size of potentially frequent sequences
$\boldsymbol{N_S}$	Number of potentially frequent sequences
$\boldsymbol{N}$	Number of event symbols

Figure 23: Parameters of type 1 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 [24], 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 $\{1, 2, \ldots, 100\}$. The off-time is the on-time plus the interval length. The location information attached to each appliance is uniformly chosen on $\{1, \ldots, 10\} \times \{1, \ldots, 10\} \times \{1, 2, 3\}$.

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. 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). Figure 24 shows the running time of the four algorithms with minimum supports varied from 1 % to 5 % on the dataset D10k–C20–N1k. 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.

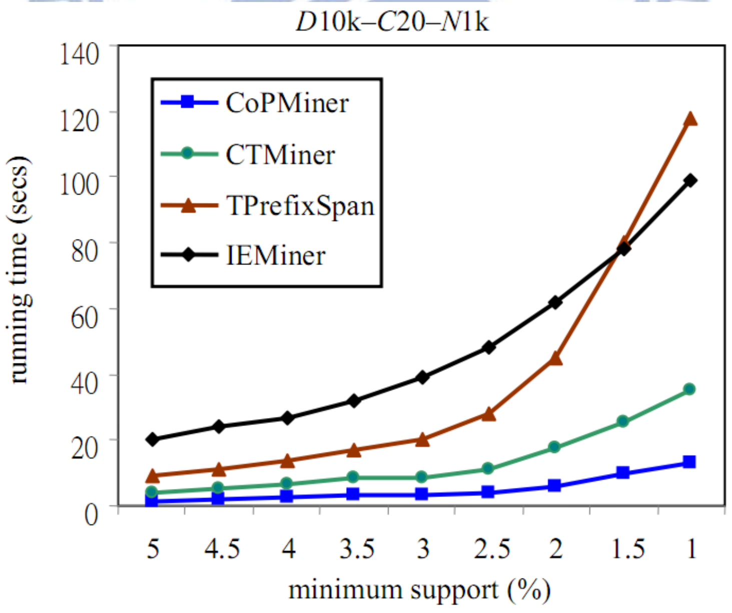


Figure 24: Runtime performance testing on D10k–C20–N1k dataset

Figure 25 shows the execution time of the four algorithms with minimum supports varied from 1 % to 5 % on the dataset D100k–C20–N10k, 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.

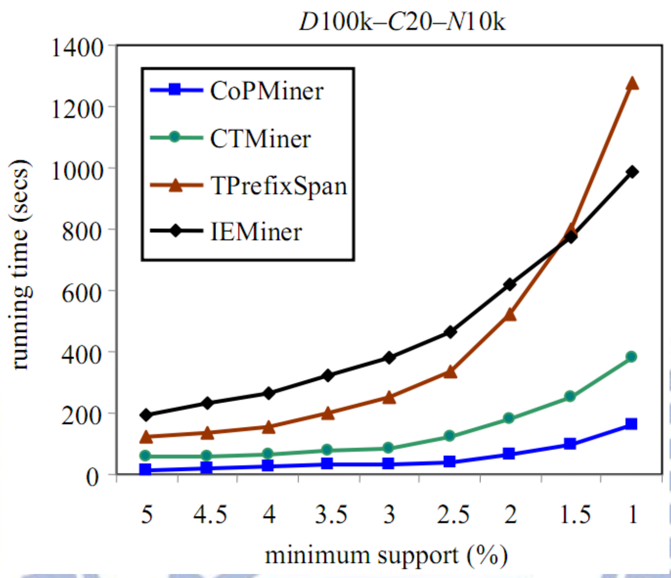


Figure 25: Runtime performance testing on D100k–C20–N10k dataset

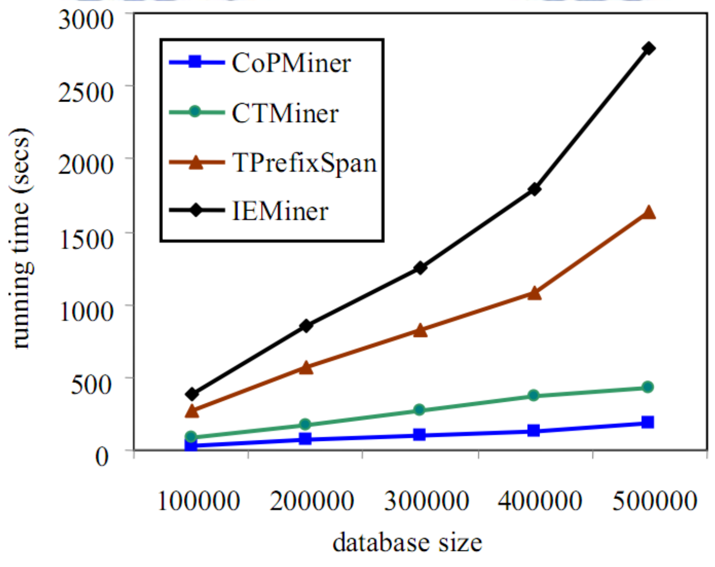


Figure 26: Scalability of comparing algorithms on different database size

Then, we study the scalability of CoPMiner. Here, we use the data set C = 20, N = 10k with varying different database size. Figure 26 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%. Figure 27 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 in Figure 27, 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.

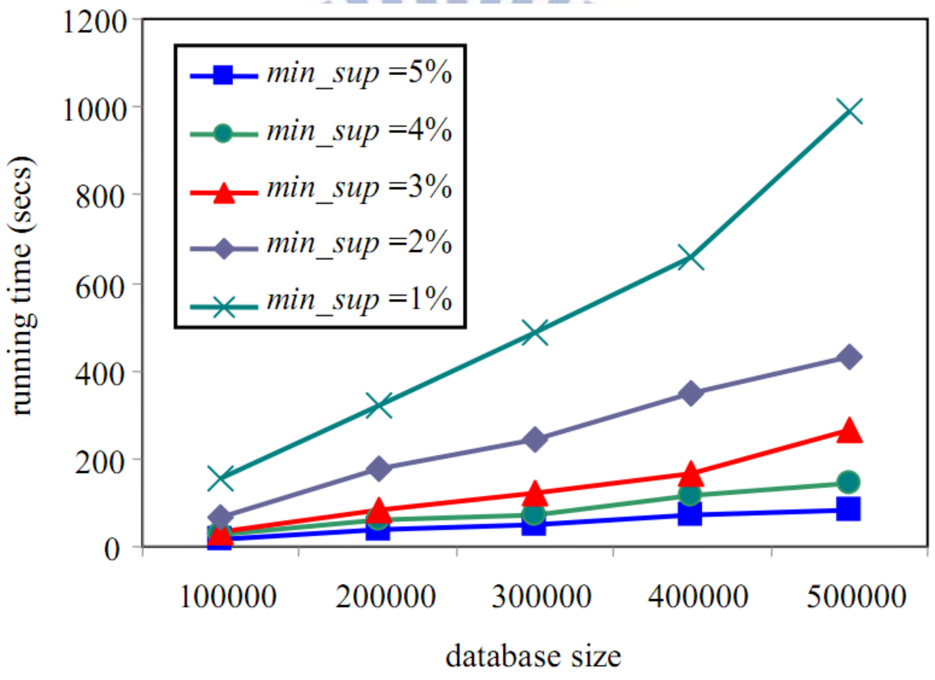


Figure 27: Scalability of different min_sup on different database size

Summarizing the results of type 1 synthetic data experiments, 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.

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 both pruning strategies), CoP_Point (only pruneSup strategy), CoP_Prefix (only pruneFI strategy), CoP_Spatial (only pruneDB strategy) and CoP_None (without any pruning strategy). The experiment is performed on the data set D100k–C20–N10k. Figure 28 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 pruneSup can improve about 25% performance of CoPMiner. The impact of the pruneFI also is presented. As can be seen from the graph, pruneFI can improve about 11% performance of CoPMiner. Figure 28 also depicts that pruneDB constantly ameliorate the performance of CoPMiner about 2.5%.

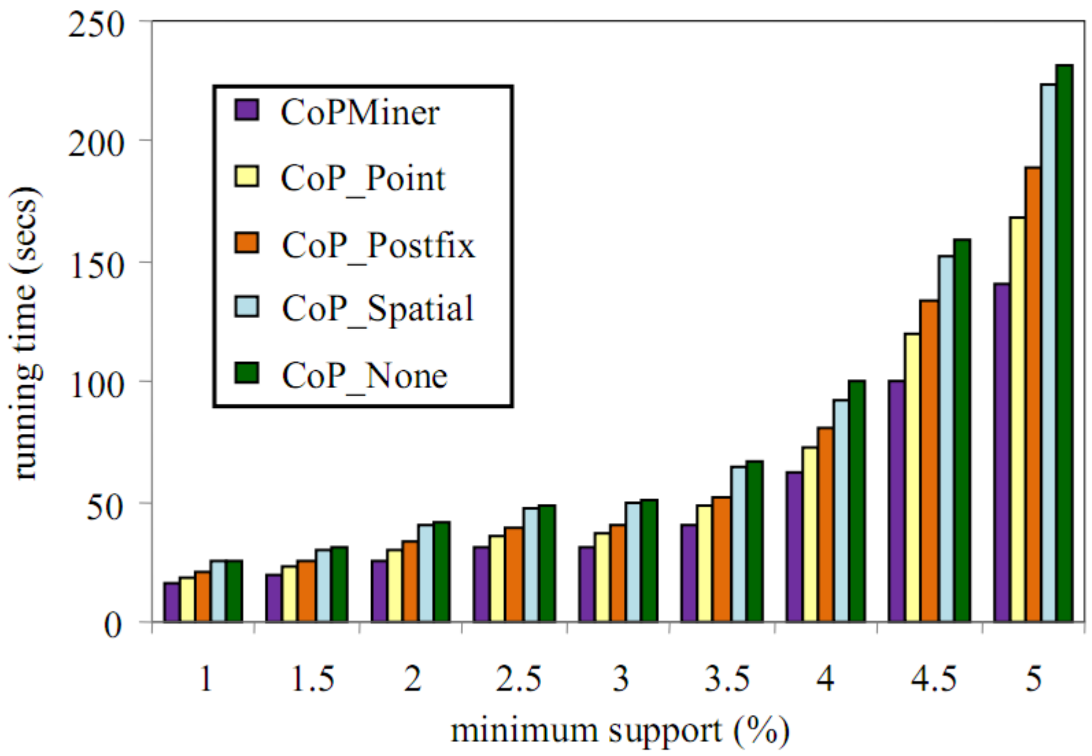


Figure 28: Influence of three pruning strategies on different min_sup

In summary, three pruning strategies constantly improve 35% runtime performance of CoPMiner. Consequently, the proposed pruning strategies not only effectively reduce the searching space but also efficiently improve the performance of CoPMiner.

Second Kind Synthetic Data Generator

Now let us focus on the second type synthetic data generators. We generate the synthetic database with two integer parameters N and L. The database consists of N sequences where sid is numbered from 1 to N. The set of event symbols is $E = \{e_1, e_2, \dots, e_{2L}\}$. Every sequence has L intervals and therefore $2L$ is the sequence length of usage representation. For each usage interval (e_i, o_i, f_i) in a sequence, e_i is chosen from discrete uniform distribution on event symbol set E while o_i and f_i are chosen from discrete uniform distribution on $\{1, 2, \dots, 100\}$ with $o_i < f_i$. We regard this synthetic database as uniformly-randomed. The locations of e_i is uniformly chosen on $\{1, \dots, 10\} \times \{1, \dots, 10\} \times \{1, 2\}$. We evaluate the average execution time by mining 60 different uniformly-randomed databases.

The Figure 29 shows that our fastest algorithm (CoPMiner) is significantly efficient when the intervals number L grow larger. Suppose that there are many appliances in a smart environment, like lights, television, microwaves, kitchen outlets, stoves and etc. The appliances could probably be turned on and off many times in one day. This indicates L will grow up to 100 in one daily log. As L varies from 10 to 80, CoPMiner is still durable in execution time while others (H-DFS[24], IEMiner[25], TPrefixSpan[30] and CTMiner[5]) grow much faster than CoPMiner.

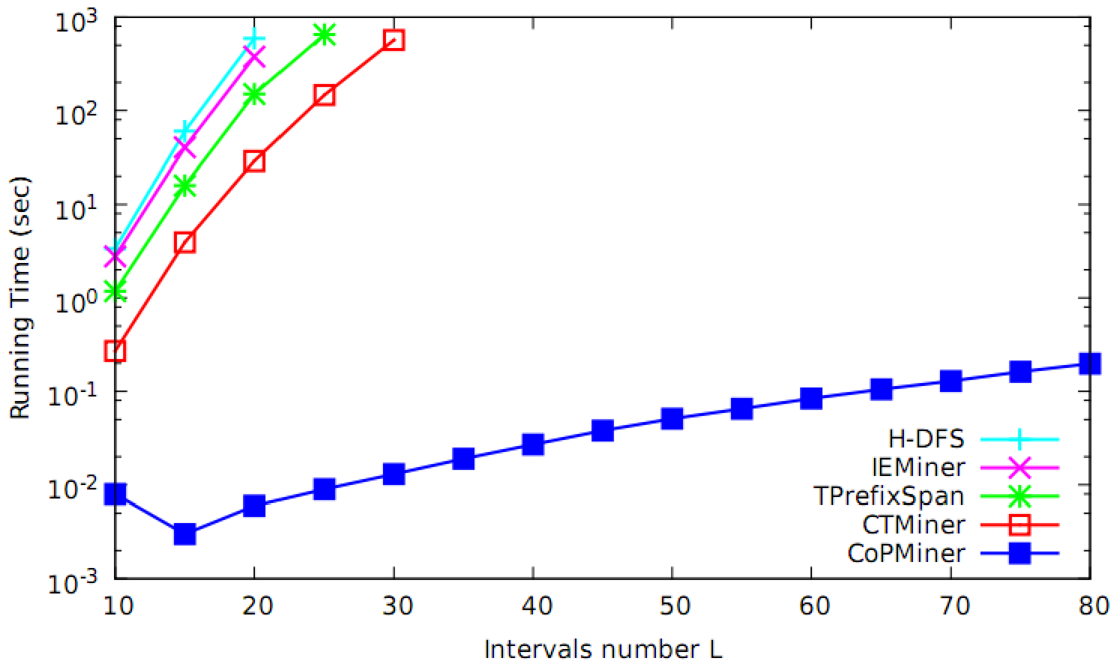


Figure 29: Execution time comparison on different algorithms

In order to compare CoPMiner algorithm with different strategies, **Late Checking (pruneFinal)** checks the spatial similarity requirement after a usage sequence q is found and add q into our frequent pattern set L if q is a correlation pattern.

Table 5: Strategies used by algorithms

Algorithm	pruneFinal	pruneDB	pruneFI	pruneSup
$CoPMiner_a$	Y			
$CoPMiner_A$	Y			Y
$CoPMiner_b$	Y	Y		
$CoPMiner_B$	Y	Y		Y
$CoPMiner_s$		Y	Y	
$CoPMiner_a$		Y	Y	Y

We first evaluate the performances of proposed three strategies (**pruneDB**, **pruneFI** and **pruneSup**) by Figure 29. We prepared six versions of CoPMiner and named as $CoPMiner_a$, $CoPMiner_A$, $CoPMiner_b$, $CoPMiner_B$, $CoPMiner_s$ and $CoPMiner$ as shown in Table 5. Our parameters are fixed as N = 100, min_sup = 0.3 and min_sim = 0.89. Although the frequent pattern sets mined by the five algorithms are completely identical, the execution time is significantly different. Figure 28 shows the average execution time of the six versions of CoPMiner by varying the sequence length L. In Figure 28 when the intervals number L = 20, $CoPMiner_a$ averagely runs in 29.022 second, $CoPMiner_A$ costs 0.085 second, $CoPMiner_b$ runs in 4.738 seconds, $CoPMiner_B$ costs 0.021 second, $CoPMiner_s$ runs in 0.066 and $CoPMiner$ costs 0.006 second.

The **pruneSup** strategy reduces about 30X to 1000X execution time in Figure 29. With the **pruneSup** strategy, we not only ensure that the set of correlation patterns is completely identical but the average execution time is within 0.2 seconds when L = 80 (for the fastest CoPMiner). Therefore we conclude that the **pruneSup** strategy is most significant efficient, the second is **pruneFI** and the third is **pruneDB** for CoPMiner.

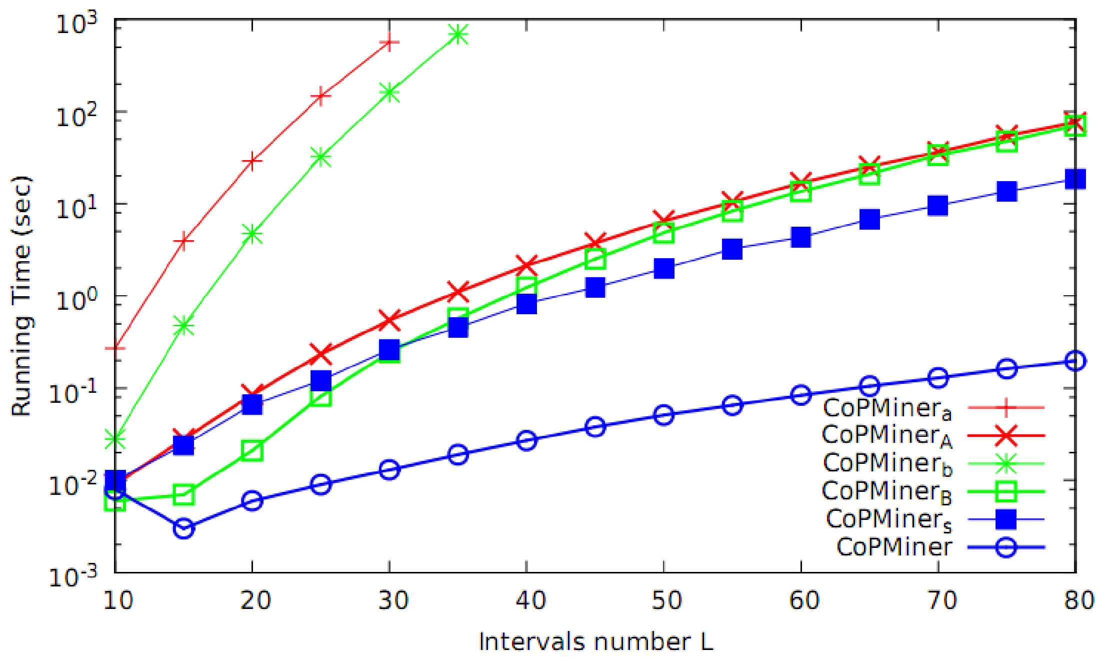


Figure 30: Execution time of algorithms with strategies

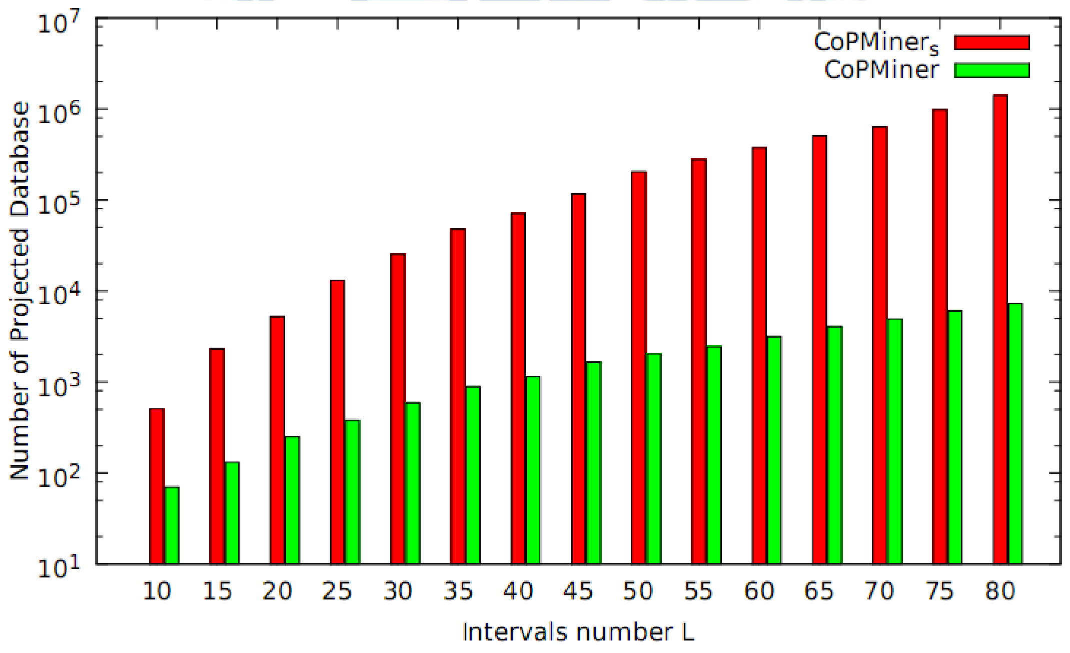


Figure 31: The number of constructed projected database

Figure 31 shows the number of projected databases constructed in mining process for the fastest two algorithms, $CoPMiner_s$ and $CoPMiner$. Although the memory usage depends on the detail of implementation, our algorithms uses about 20MB when

L = 10 and about 50MB when L = 80. In Figure 30 when the intervals number L = 20, $CoPMiner_S$ averagely visits 5265.033 projected database to discover the set of correlation patterns while CoPMiner visits 250.433 projected databases. We compare the growth orders in execution time (Figure 30) and the number of projected database visited (Figure 31). We believe that the efficiency of CoPMiner is highly related to the number of projected databases ever constructed. Therefore **pruneSup** strategy does significantly reduce the number of constructed projected databases.

7.2 Real-world Dataset – REDD

In this section, we describe our real-world dataset and show some correlation patterns that our algorithms discovered for each house. Although many smart home environment datasets are available, but little of them records the status along with space information for each appliance in smart home environment. Kolter et al. [18] collected the dataset REDD including detailed power readings of each appliance of six houses lasting for about five weeks. Therefore we can convert the raw data into suitable usage database. We use our sample house for the location information of each house. And we set the minimum space threshold fixed as 0.5 and mine the daily correlation patterns of each house.

Table 6: Description of six houses in REDD dataset

Home	Number of Appliances	Number of Daily Sequences	Number of Total 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

Precomputation

In the REDD dataset [18], the raw data of each appliance is real-valued with discrete time. For example, ... (1303133979,0.00) (1303133983,9.00) ... is recorded as

(UTC timestamp, power reading) for the stove of house 1. Therefore, we need to convert the raw data of REDD dataset into symbolic intervals to form the usage database.

Table 6 lists the descriptions of REDD dataset after our precomputation. We use 16 appliances of totally 20 (provided by [18]) in house 1. There are 36 days data collected and totally 1107 intervals generated after our precomputation steps. We briefly describe how to convert the real-valued raw data into time interval-based database by the following steps.

(1) Round each of the real-valued power reading p into an integer q by the equation $q = \lfloor p + 0.5 \rfloor$.

(2) Apply the four-level Otsu's method on the histogram of all the rounded values q in previous step by the multilevel thresholding algorithm proposed in [19]. Then we obtain the three thresholds t_1, t_2 and t_3 with $0 < t_1 < t_2 < t_3 < \infty$. Also we take the smallest t_1 and largest t_3 if the between-class variance values σ^2 are tied [19]. Thus we have partitioned the power reading into four states, *Definitely-Off* $[0, t_1)$, *Maybe-Off* $[t_1, t_2)$, *Maybe-On* $[t_2, t_3)$ and *Definitely-On* $[t_3, \infty)$.

(3) We scan the rounded power readings q to generate the intervals if the power reading jumps from *Definitely-Off* to *Definitely-On* and jumps back. More clearly, let q_t be the rounded power reading at timestamp t. We generate the interval (x, y) if (1) $q_x \in [0, t_1) \wedge q_{x+1} \in [t_3, , \infty)$ (2) $q_y \in [t_3, \infty) \wedge q_{y+1} \in [0, t_1)$ and (3) $\forall\, t \in \{x+1, x+2, \ldots, y-1\}, q_t \notin [t_3, \infty) \vee q_{t+1} \notin [0, t_1)$.

We list some representative patterns of each house in Figure 32. It is noted that the interval length in Figure 32 only states the relations between endpoints instead of the actual duration in the log of each home. By setting min_sup = 0.3 and min_sim = 0.5, we have discovered only home 1, 3, 4 and 6 has frequent daily patterns. Although home 2 and 5 have daily patterns in at most two days, we still regard home 2 and 5 have no outstanding daily patterns since their insufficiency of raw data. In home 1, the pattern 1 appears in 9 days and pattern 2 appears in 5 days. In home 3, pattern 1 and 2 appear in 5 days. Seven days of home 4 have pattern 1 while three days of home 6 exist pattern 1. The complete correlation pattern with probability function is listed in

7.3 Synthetic Dataset for Anomaly Detection

In this section, we describe our synthetic dataset for testing the performances of anomaly detection. Since original data does not label every day with normal behavior or anomaly. Therefore our target testing probability function is the probability distribution of turning on Light 3 in house 1 (The probability distribution is depicted in Figure 33 of second row and second column). We generate the synthetic query time of turning on Light 3 in house 1 uniformly distributed in [0, 24). For example, the query 8.25 indicates that testing the anomaly of turning on Light 3 at 8:15 in house 1. We uniformly pick 1 million queries in time. In other words, our testing set of query $Q = \{t_1, t_2, \ldots, t_{1000000} \mid t_i \in [0,24), \forall\, i \in \{1, 2, \ldots, 1000000\}\}$ and we also evaluate Equation 8 and obtain the distribution of the queries are shown in Figure 34.

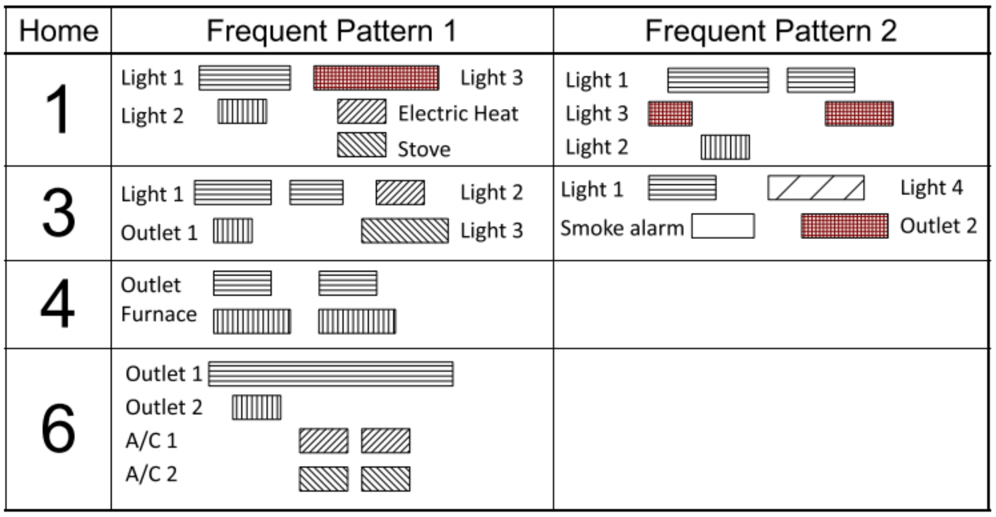


Figure 32: Frequent patterns' sequence of REDD dataset

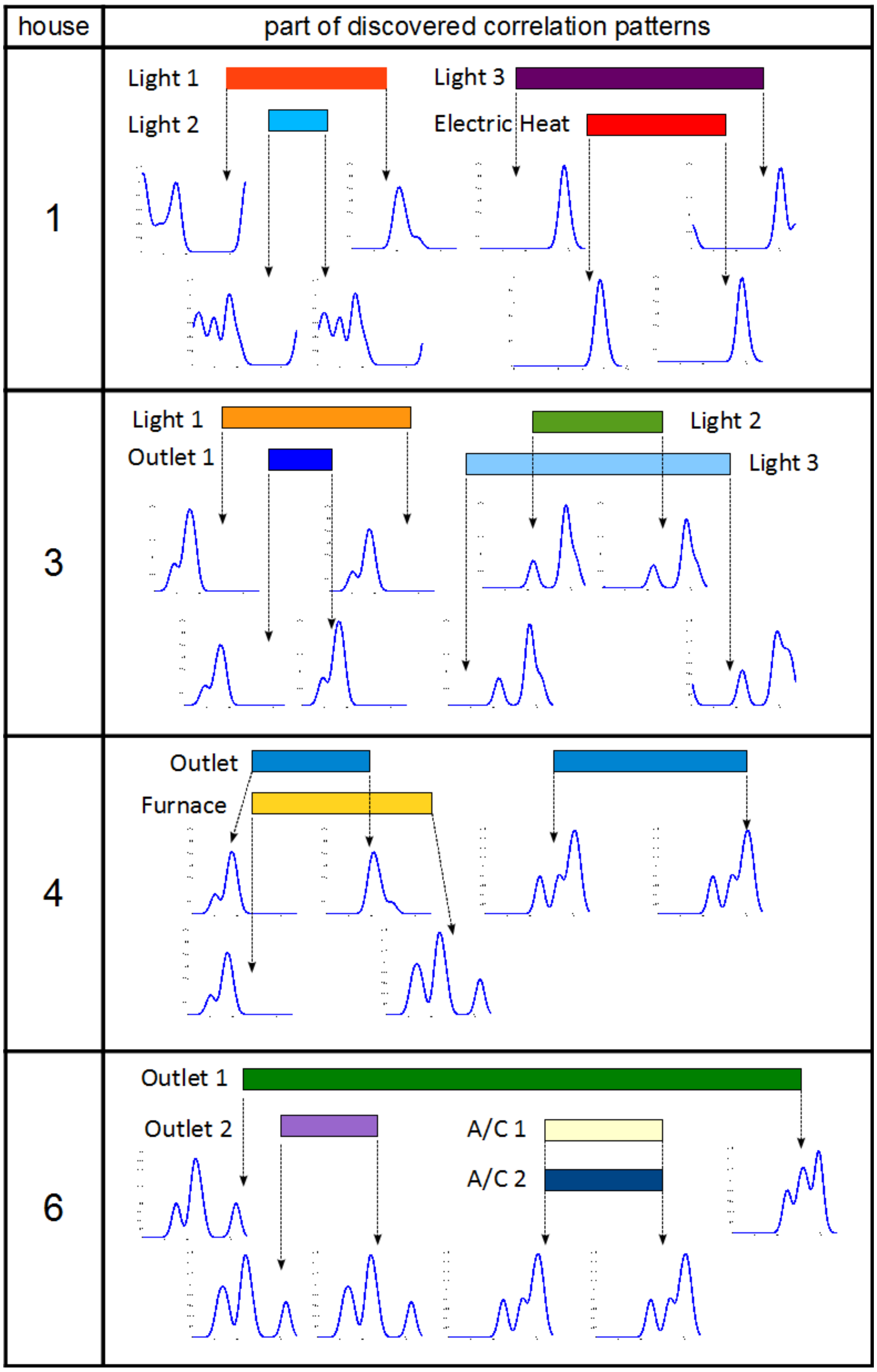


Figure 33: Part of detailed correlation pattern in REDD

We can see that there are two kinds of queries, normal time and anomaly time. Therefore we regard the area under 0.45 as anomaly queries (59.37%) and above 0.5 is normal query (40.63%). We evaluate the precision of anomaly queries by leveling up the parameter *min_pro* from 5% to 100% with step 5%. The figure 35 tells us that when *min_pro* = 0.05 we identify 91.36% anomaly queries out and correctly report to system while *min_pro* = 0.2 we only capture 61.69% anomaly queries. We conclude that the lower *min_pro* will raise more anomaly detection. However the actual value of *min_pro* is usually user sensitive and therefore the proper *min_pro* threshold should be found based on users.

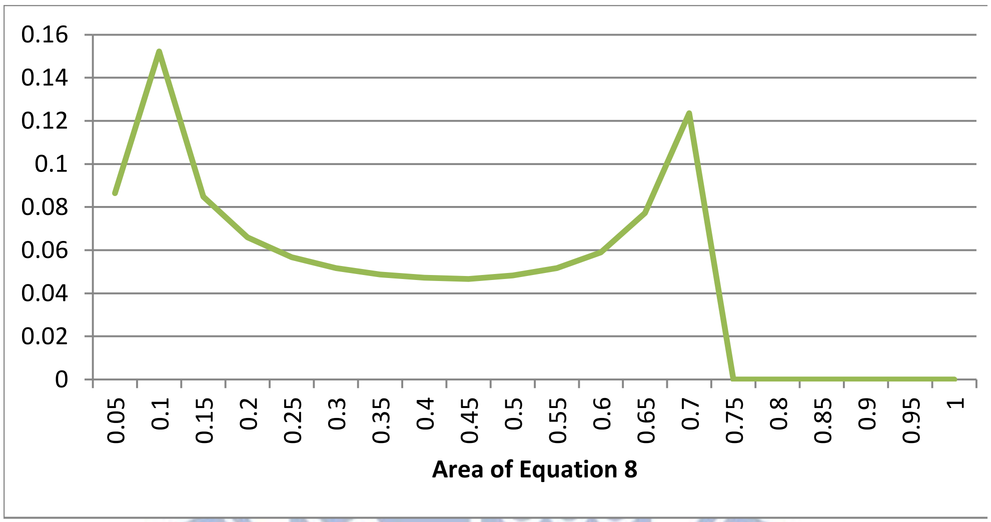


Figure 34: The distribution of query dataset *Q*

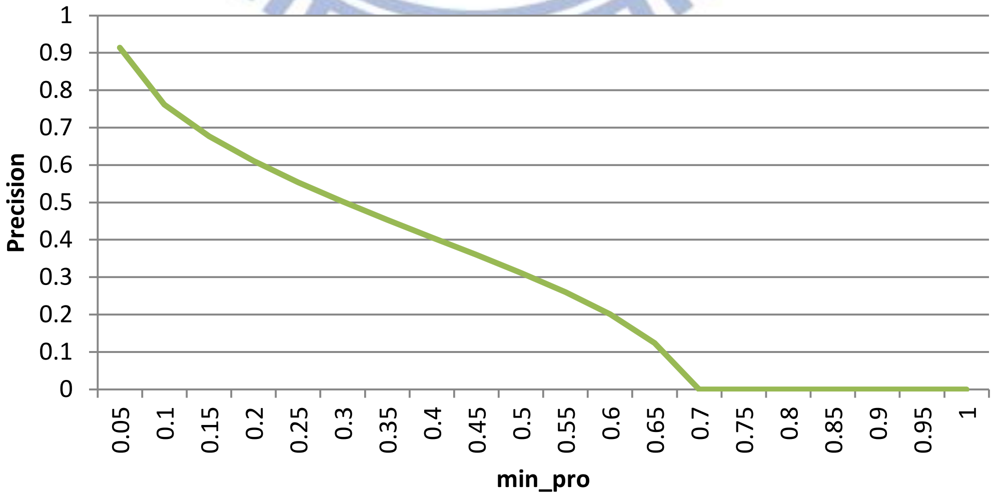


Figure 35: The precision of synthetic dataset

Chapter 8

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 usage patterns and correlations among appliances probabilistically. An efficient algorithm, CoPMiner is developed to discover patterns based on proposed usage representation. We also introduce three pruning strategies named pruneDB, pruneFI and pruneSup to improve the performance of the proposed algorithm. We also mention some applications of correlation pattern mining, like anomaly behavior detection and activity prediction. We also list some issues for anomaly detection and propose a method for anomaly detection system. The experimental studies on synthetic dataset indicate that CoPMiner is efficient and scalable. Furthermore, CoPMiner is applied on a real-world dataset to show the practicability of correlation pattern mining.

Bibliography

[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] S. Barker, A. Mishra, D. Irwin, E. Shenoy and J. Albrecht. Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes. *Proceedings of SustKDD Workshop on Data Mining Applications in Sustainability*, 2012.

[4] 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.

[5] 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.

[6] Y. Chen, W. Peng and S. Lee. CEMiner 一 An Efficient Algorithm for Mining Closed Patterns from Time Interval-based Data. *Proceedings of 11th IEEE International Conference on Data Mining (ICDM'11)*, pp. 121-130, 2011.

[7] Y. Chen, Y. Ko and W. Peng, An Intelligent System for Mining Usage Patterns from Appliance Data in Smart Home Environment. *Conference on Technologies and Applications of Artificial Intelligence (TAAI'12)*, pp. 319-322, 2012.

[8] 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.

[9] 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.

[10] 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.

[11] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M. Hsu. FreeSpan: frequent pattern-projected sequential pattern mining. *Proceedings of 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pp. 355-359, 2000.

[12] 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.

[13] V. Jakkula and D. Cook. Learning Temporal Relations in Smart Home Data. *Proceedings of the Second International Conference on Technology and Aging*, 2007.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery* 15.2 (2007) 181-215.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] I. Batal, D. Fradkin, J. Harrison, F. Moerchen and M. Hauskrecht. Mining Recent Temporal Patterns for Event Detection in Multivariate Time Seies Data. *Proceedings of 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12)*, pp. 280-288, 2012.

[32] Z. Liao, P. Lei, T. Shen, S. Li and W. Peng. AppNow: Predicting Usage of Mobile Applications on Smart Phones. *Conference on Technologies and Applications of Artificial Intelligence (TAAI'12)*, pp. 319-322, 2012.

[33] Z. Liao, P. Lei, T. Shen, S. Li and W. Peng. Mining Temporal Profiles of Mobile Applications for Usage Prediction. *IEEE 12th International Conference on Data Mining Workshops (ICDMW'12)*, 2012.