IPSJ Transactions on Mathematical Modeling and Its Applications Vol.9 No.1 33–42 (Feb. 2016)

Regular Paper

Periodic Pattern Mining with Periodical Co-occurrences of Symbols

Keisuke Otaki[1,2,a)] Akihiro Yamamoto[1]

Received: September 2, 2015, Revised: October 14, 2015,
Accepted: November 16, 2015

Abstract: Finding periodic regularity in sequential databases is an important topic in Knowledge Discovery and in pattern mining such regularity is modeled as periodic patterns. Although efficient enumeration algorithms have been studied, applying them to real databases is still challenging because they are noisy and most transactions are not extremely frequent in practice. They cause a (combinatorial) pattern explosion and the difficulty of tuning a threshold parameter. To overcome these issues we provide a novel pre-processing method called skeletonization, which was recently introduced for finding sequential patterns. It tries to find clusters of symbols in patterns, aiming at shrinking the space of all possible patterns in order to avoid the combinatorial explosion by considering co-occurrences of symbols. Although the original method cannot allow for periods, we generalize it by using the periodicity. We give experimental results using both synthetic and real datasets to show the effectiveness of our approach, and compare results of mining with and without the skeletonization to see that our method is helpful for mining comprehensive patterns.

Keywords: periodic pattern mining, comprehensive patterns, similarity graph, spectral clustering

1. Introduction

Finding patterns frequently appearing in databases is one of the important problems in data mining. Transactions in databases naturally have timestamps as their auxiliary attributes. They are often ordered with timestamps and a typical sorting is the chronological order. For example sequences of transactions, describing products bought by a user in a EC site, are sorted in the chronological order from old to new. If such orders are important to a database, such a database is called a sequential database. As the order is directly related to time, typical periods related to clocks or calendars (e.g., hour, day, etc.) may contribute to hidden regularity in it. Therefore assuming that such periodic behaviors may appear in various sequential databases (e.g., trajectory, life-log) is natural in data mining. To get valuable but hidden insights from databases by capturing periodic regularity, *periodic pattern mining* have been studied [4], [5], [14] and applied to various problems [6], [7].

We have several variations on the definition of periodic patterns. The fundamental ones are *full periodic patterns* and *partial periodic patterns* [4]. Note that we now assume that patterns are sequences of symbols drawn from an alphabet Σ for the sake of simplicity. For example, let $\Sigma = \{\mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{shops}\}$. We consider a sequence s_{user}:

$$s_{\mathrm{user}} = (\mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{sns}, \mathrm{shop}, \mathrm{blog})$$

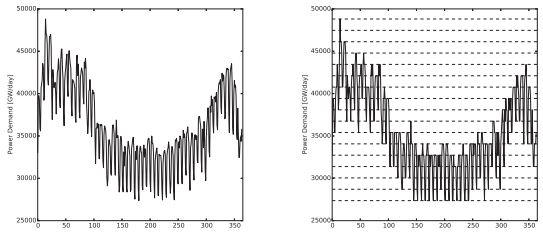
representing a log of categories of Web sites visited by a user. A pattern $(\mathrm{sns}, \mathrm{news}, \mathrm{blog})$ in s_{user} appears twice, and this is called *full periodic pattern* of period length 3. Full periodic patterns require that all symbols should be fully specified. In some cases, such requirement is so strong and it is difficult to handle various periodic behaviors. As more flexible patterns, *partial periodic patterns* have been studied [5]. For example, a partial periodic pattern $(\mathrm{sns}, \star, \mathrm{blog})$ appears 3 times, where $\star$ is the wildcard symbol of length 1 representing any symbol in Σ. As partial periodic patterns can contain the symbol $\star$, they are more flexible than full periodic patterns to capture periodic behaviors in databases. In mining these periodic patterns, we assume that a given sequence s is divided into $\lceil \frac{|s|}{P} \rceil$ fragments, where P is a period of users' interest, and the fragments are used to evaluate patterns: In the example above, the pattern $(\mathrm{sns}, \star, \mathrm{blog})$ appears 3 times in fragments $(\mathrm{sns}, \mathrm{news}, \mathrm{blog})$, $(\mathrm{sns}, \mathrm{news}, \mathrm{blog})$, and $(\mathrm{sns}, \mathrm{shop}, \mathrm{blog})$ of s_{user}.

Although many efficient mining algorithms have been developed [4], [14], it is still challenging to use them in practice because the number of enumerated patterns highly depends on the number $|\Sigma|$ of symbols we use. When databases get large, $|\Sigma|$ increases as well. This fact consequently makes evaluating patterns by their support counts (i.e., the number of occurrences) difficult because the support counts of most patterns are similar and relatively small. That is, the space of (frequent) patterns on Σ get *sparse* with respect to the space of all possible patterns[*1].

Motivating Examples: For both numerical (e.g., price, temperature) and symbolic (e.g., item, product) sequences, preparing

[1] Kyoto University, Kyoto 606–8501, Japan
[2] Research Fellow of the Japan Society for the Promotion of Science, Chiyoda, Tokyo 102–0083, Japan
[a)] ootaki@iip.ist.i.kyoto-u.ac.jp

[*1] Consider to find all partially periodic patterns up to the length k on Σ. Let $\Sigma_\star = \Sigma \cup \{\star\}$. All possible combinations are in $\Sigma_\star \cup \Sigma_\star^2 \cup \cdots \cup \Sigma_\star^k$, which can become much larger than that of all patterns appearing in databases in practice.



(a) Original (b) Discretized ($|\Sigma| = 16$)

Fig. 1 A numeric sequence of electric power demand in UK, 2013 in Fig. 1 (a), and its discretization with 16 symbols (corresponding to dashed lines) in Fig. 1 (b).

a large set Σ of symbols is essential to achieve the high resolution of describing phenomena. For example, **Fig. 1** (a) shows a sequence of electric power demand per day in UK, 2013. We discretize the sequence with dividing values into $|\Sigma|$ bins uniformly[*2] as seen in Fig. 1 (b) with $|\Sigma| = 16$. Clearly, we can represent a sequence as a symbolic sequence with a smaller loss with a larger set Σ. In Fig. 1 (b), however, only a few combinations of Σ appear consecutively. It is difficult to tune the set Σ while taking a balance among the expressiveness and the sparseness. Now a typical periodic behavior is that *the demand gets higher every weekend*, which could be obtained by frequent patterns, where symbols corresponding to low values are followed by those doing to high values. We believe that such high-level patterns are more informative and useful to analyzing databases.

As another example we consider those stored from Last.fm (`http://last.fm/`), which are sequences of songs logged by users. We take some logs of users from an open dataset (See Section 3 of [2]), where a sequence $s = \langle S_1, S_2, \dots \rangle$ is a log of a user and each S_i is the set of songs heard in the index i in which the index i corresponds to a 1 hour interval of the log (e.g., the set S_4 shows the listened songs during 0 a.m. to 1 a.m.). For example, the sequence for user ID 808 is length 16,913 log, where the user listened to 24,310 songs and 1,340 out of 16,913 intervals are not empty (i.e., in other intervals the user did not listen to any songs). Then if we would like to analyze some daily behaviors (i.e., $P = 24$) including the empty situation, $24{,}310^{24}$ is the upper bound of the size of all possible combinations. This is intractable and data we have are obviously sparse.

Approaches: In pattern mining, therefore, Liu et al. [8] and others insisted that users carefully need to tune the set Σ and proposed the temporal skeletonization for symbolic sequential patterns. Their idea is to construct clusters of symbols and assign each cluster a label. Then a sequence can be translated into a high-level and potentially comprehensive sequences of the labels. By grouping symbols into clusters, we can reduce the size $|\Sigma|$. We develop such method for periodic analyses by generalizing the idea [8], and discuss frequently occurring high-level patterns with the periodicity. In the existing method [8], to represent comprehensive sequential patterns, *left-to-right* HMMs are *implicitly* assumed as their models. Based on the study we take into account the periodicity of the given period length P by considering a bit different models of sequences, i.e., *cyclic HMMs*, which can be applicable to describe periodical behaviors of sequences comprehensively.

In addition to that, we would like to emphasis on the fact that many existing studies dealt with DNA sequences, which requires only 4 symbols (i.e., $\Sigma = \{\mathrm{T}, \mathrm{C}, \mathrm{A}, \mathrm{G}\}$). In such a situation, the combinatorial explosion is only depending on the length l of patterns we try to mine. However, this situation is a bit restricted as many sequences require more symbols in general. Therefore developing a new approach with the periodicity is an important remained problem. We thus focus on the point by following the existing method [8]. Since the temporal skeletonization cannot be applied to periodic settings, we generalize it by using the idea of periodic extensions of functions.

The rest of this paper is organized as follows. We give preliminaries in Section 2. Our method is formally described in Section 3. We provide our experimental results and discuss them in Sections 4 and 5, and conclude our study in Section 6.

2. Preliminary

Let Σ be the alphabet. The set $\Sigma^\star$ denotes the Kleene closure of Σ. We use $\Sigma^+ \equiv \Sigma^\star \setminus \{\epsilon\}$, where ϵ is the empty string. For a sequence $s \in \Sigma^+$, $|s|$ denotes the length of s. We let $|\epsilon| = 0$. In addition, s_i and $s_{i,j}$ represent i-th element and the continuous subsequence from i to j of s ($i < j$), respectively. Let P be a fixed integer representing the period of users' interest.

2.1 Frequent Partially Periodic Pattern Mining

Periodic behaviors of databases can be modeled as *partially periodic patterns*. An important concept used is *periodic segments* of sequences.

Definition 1 (Event Sequence and Periodic Segment) For an *event sequence* $s \in \Sigma^+$ and a period P, s can be divided into m ($= \lceil \frac{|s|}{P} \rceil$) *mutually disjoint segments*. We denote it by $s = \langle ps_1, ps_2, \dots, ps_m \rangle$, where for $1 \le i \le m$, $ps_i = s_{(i-1)m, im-1}$.

For example with a sequence $s = abcabdabb$ of symbols $\{a, b, c, d\}$ and $P = 3$, the sequence s is divided into three parts; $ps_1 = abc$, $ps_2 = abd$, and $ps_3 = abb$.

Definition 2 (Partial Pattern) A sequence from $\Sigma \cup \{\star\}$ is a (*partial*) *pattern*, where the special character $\star \notin \Sigma$ represents any event of length 1.

In this paper we focus on the partial patterns appearing in periodic segments frequently. Then, for a sequence s and a pattern p, it is necessary to evaluate whether or not p has an interesting occurrence in s. The traditional measure for the purpose is to adopt the *support count* of patterns.

Definition 3 (Support Count) The *support count* of a pattern p, denoted by $\mathrm{Sup}_P(p)$, is defined as $\mathrm{Sup}_P(p, s) = |\{ps_i \mid s = \langle ps_1, \dots, ps_m \rangle, ps_i \preceq p\}|$, where $ps_i \preceq p$ means that for all $1 \le i \le |p|$ it satisfies either $p_i = \star$ or $p_i = s_i$. We say that a pattern p is *frequent* if $\mathrm{Sup}_P(p) \ge \theta$ for a user-specific threshold

[*2] If the range of values $[0, 10)$ and $|\Sigma| = 4$, values in $[0, 10]$ would be categorized into either $[0, 2.5)$, $[2.5, 5.0)$, $[5.0, 7.5)$, or $[7.5, 10)$, and symbolic alphabets are assigned to encode the sequence into a symbolic sequence.

θ. We call such a p a partially periodic pattern (PPP) when the period P is given and important.

For $s_2 = abcabdabb = \langle abc, abd, abb\rangle$ and the a constant (period) $P = 3$, $\mathrm{Sup}_P(ab\star, s_2) = \frac{3}{3} = 1.0$.

Now the mining problem is formally described below.

Problem 1 (The FPPPM problem) Let θ be a user-specific threshold and P a given period length. For a sequence s, the FPPPM problem is defined as to list all partially periodic patterns p from s satisfying $\mathrm{Sup}_P(p) \geq \theta$.

Several efficient algorithms have been developed: For example, Han et al. showed a fundamental algorithm using max sub-pattern trees [5]. This method is used frequently in several application problems based on periodic patterns [6], [7]. Yang et al. proposed to use tuple representations for periodic patterns and a depth-first search algorithm based on the PrefixSpan [11] used in sequential pattern mining [14].

2.2 Temporal Skeletonization

We review the original definition of *temporal graphs* to explain the idea of the temporal skeletonization in [8], which tries to build a *similarity graph*[*3] from a given database DB for capturing the similarity within symbols in Σ.

Definition 4 (Temporal Graph [8]) Let $G = (V, E)$ be a weighted undirected graph, where V corresponds to Σ. Let $\mathrm{DB} = \{s^{(1)}, \ldots, s^{(N)}\}$ be the set of sequences of symbols from Σ. For two symbols $x, y \in \Sigma$, the weight $W_{x,y}$ of the edge corresponding to $\{x, y\}$ is defined as

$$W_{x,y} = \frac{1}{N}\sum_{n=1}^{N} \sum_{\substack{1\leq i\leq j\leq |s^{(n)}|,\\ |i-j|\leq r}} \mathbf{1}_{s_i^{(n)}=x \wedge s_j^{(n)}=y} \qquad (1)$$

where N is the number of sequences, r be the *window width*, $\mathbf{1}_f$ is the indicator function that returns 1 if and only if the predicate f is true; 0 otherwise. We count the number of *co-occurrences* of symbols x and y in a window.

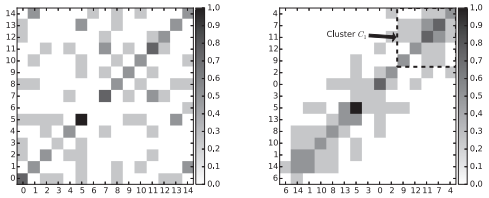
The right-hand side of Eq. (1) can be computed by checking the given database DB, where the indicator function can be replaced with other similarity measures. The authors in [8] used the exponential function $\exp(-k|i-j|)$ with a parameter k. In constructing a temporal graph G, all indices of sequences in DB are taken into account several times. In each time we focus on some index i, we check all neighbors within width r. That is, for indices $1 \leq i \leq j \leq |s|$, a simple implementation of Eq. (1) is to increment the weight W_{s_i,s_j} if $|i-j| \leq r$ for $s \in \mathrm{DB}$.

After constructing G, users try to find clusters of symbols by applying clustering methods to G (e.g., *spectral clustering* [9], [12]). The problem of finding clusters can be formulated as a standard graph-based optimization problem with some constraints as explained in [8], where an important step is to compute

[*3] A *similarity graph* is a weighted graph in which vertices represent data points and edges represent the similarity between two points with their weights.

$$\Sigma = \{0, 1, 2, \ldots, 15\}.$$
$$\mathrm{DB} = \{s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)}\},$$
$$s^{(1)} = (12, 7, 9, 5, 3, 0, 8, 10, 1)$$
$$s^{(2)} = (9, 11, 12, 0, 13, 5, 1, 14, 6)$$
$$s^{(3)} = (4, 7, 11, 2, 5, 7, 9, 1, 14)$$
$$s^{(4)} = (7, 11, 4, 2, 0, 7, 10, 14, 8)$$

(a) Input $\mathrm{DB} = \{s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)}\}$



(b) Computed $W_{s,t}$ (c) Re-ordered

Fig. 2 A toy example in [8] and two heatmaps: Fig. 2 (b) shows the original W computed from DB, and Fig. 2 (c) is the re-ordered one, in which a cluster C_1 is drawn.

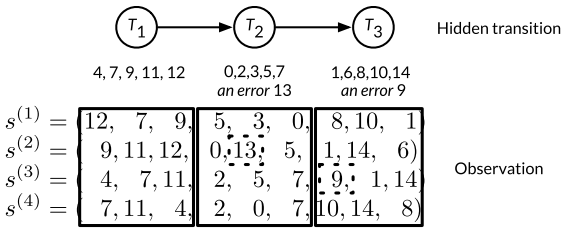


Fig. 3 Temporal clusters and their transition.

eigenvalues and eigenvectors from G. Now a computed matrix W of weights can be represented as a heatmap as shown in **Fig. 2**.

Example 1 The input is shown in Fig. 2 (a). We compute the weights from $\{s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)}\}$ as seen in Fig. 2 (b) and represent them by a heatmap, where both the x-axis and y-axis correspond to the order of the alphabet Σ. That is, on some (i, j), the thickness in the heatmap corresponds the value W_{Σ_i,Σ_j}. After applying the spectral clustering, we can re-order indices of W as shown in Fig. 2 (c). For example, we can find a cluster of symbols such as $C_1 = \{4, 7, 9, 11, 12\}$, which is the upper right area in Fig. 2 (c). Note that C_1 appears in prefixes of sequences in DB. Then we can now conjecture that all sequences in DB are in the form $\langle C_1, C_1, C_1, \ldots\rangle$. This prefix consisting of cluster labels can be regarded as a *high-level* pattern of the sequences.

Models and Assumptions: This method is based on an implicit assumption; *symbols x, y appearing closely and temporally may belong to some meaningful cluster*. We can assume several generative models behind sequences. In this scenario, a typical (left-to-right) HMM is adopted (as seen in **Fig. 3**), and based on the strong assumption above, the method tries to find such clusters by using a moving window of width r and graph clustering methods.

3. Periodical Skeletonization

The key idea for taking into account periodic information is simple: Extending functions representing areas that we check in computing weights to some periodic functions of the periodicity P of our interest. In order to analyze some monthly behaviors, for

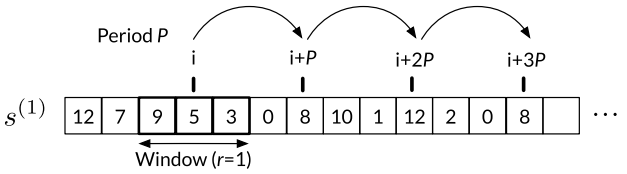


Fig. 4 Periodic extensions and co-occurrences.

example, we set $P = 31$.

The *sliding window* of width r used in the temporal skeletonization can be modeled by a *rectangular function* with width r and the origin i[*4]. By modifying this function in a periodic manner, we can deal with the periodicity of occurrences of symbols. We can easily imagine such techniques on the analogy of Fourier series and Fourier transforms. Recall the toy examples in Section 2.1. For an input sequence $s = abcabdabb$ and $P = 3$, a frequent partially periodic pattern $ab\star$ appears 3 times in every segment abc, abd, and abb. This means that not only neighbors according to the sliding window $\mathrm{Rect}_{i,r}(\cdot)$, but also periodic information from i, that is, $i + P, i + 2P, i + 3P, \ldots$ could be used to search for similar intervals as shown in **Fig. 4**.

The above observation inspired our method, *periodical skeletonization*, in which a similarity graph named a *periodic graph* is computed by observing the *periodic co-occurrences* of symbols from an input sequence s.

Definition 5 (Periodic Graph) Let $G = (V, E)$ be a similarity graph. The weights of G from an input sequence s and a period P for two symbols x, y are computed as:

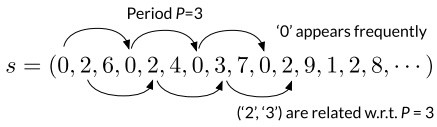
$$W_{x,y} = \sum_{\substack{1\le i,j\le |s| \\ \textbf{if } s_i=x \wedge s_j=y}} \mathbf{1}_{|i-j|\le r} + \mathbf{1}_{i\equiv j \pmod P} \qquad (2)$$

The second term is newly introduced based on the periodic information. We can also replace the right-hand side of Eq. (2) with similarity functions by adapting them in a similar fashion based on the temporal skeletonization.

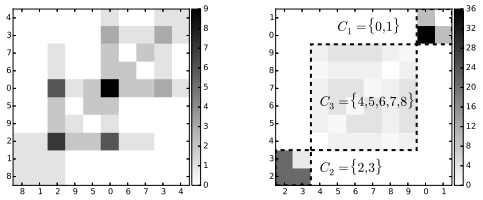
Example 2 **Figure 5** illustrates examples of computing Eq. (2) from $s = (0, 2, 6, 0, 2, 4, \ldots)$ with $\Sigma = \mathbb{N}$. Figure 5 (b) is computed by the temporal skeletonization, while Fig. 5 (c) adopts the periodic term only in Eq. (2). We can see 3 clusters as rectangles: $C_1 = \{0, 1\}$, $C_2 = \{2, 3\}$ and $C_3 = \{4, 5, 6, 7, 8\}$ in Fig. 5 (c), and they are clear than those in Fig. 5 (b) (it is hard to find clusters from it).

Models and Assumptions: Our basic observation is that we can have periodic sequences by *cyclic HMMs*. That is, from the definition of patterns, we assume that symbols appearing frequently *and periodically* in s should belong a meaningful cluster in periodic data analyses with period P. Our computation is then a bit generalized from the existing study, where the *periodical co-occurrences of symbols* in s are newly taken into account. An example is given in **Fig. 6**. For example, to simulate a partially periodic pattern $02\star$, in T_1 and T_2, $\mathcal{H}$ outputs 0 and 2, respectively with high probability $100 \times (1 - u)\%$ and outputs 1 and 3

[*4] It is defined as $\mathrm{Rect}_{i,r}(t) = 0$ if $|t - i| > r$, 1 otherwise.



(a) Input and observations with $P = 3$



(b) By temporal method (c) By periodic method

Fig. 5 An example of *periodic co-occurrences* is shown in Fig. 5 (a). Figure 5 (b) is a result only using the temporal information and Fig. 5 (c) is that adopting the periodic information only, where rectangles are the discovered clusters.

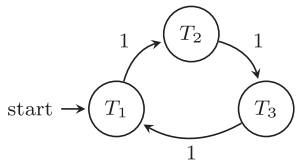


Fig. 6 Settings of HMMs used for Example 2. The output symbols from states T_1, T_2, and T_3 are set to be $o(T_1) = \{0, 1\}$, $o(T_2) = \{2, 3\}$, and $o(T_3) = \{4, 5, 6, 7, 8, 9\}$. The two symbols 0 and 2 are generative with a high probability $1 - u$ and 1 and 3 are done with a low probability u ($0 < u \le 0.25$). Symbols from T_3 is generated uniformly.

with low probability $100 \times u\%$. On the other hand in T_3, $\mathcal{H}$ generates $\{4, 5, 6, 7, 8, 9\}$ uniformly. By generating sequences of length N from $\mathcal{H}$, we can obtain a sequence s containing the partially periodic pattern $02\star$ very frequently. Compared with the result in Fig. 5 (b), we can confirm that C_1, C_2, C_3 correspond to T_1, T_2, T_3 are found by the skeletonization more clearly as blocks in the matrix in Fig. 5 (c).

3.1 Set Sequences

Let us reminder the computation of W_{s_i,s_j} when indices i and j are considered in $s \in \mathrm{DB}$. The discussion above and the existing study [8] only consider the case when s is a sequence of symbols from Σ. However, some applications of sequential and periodic pattern mining assume that s is a sequence of subsets of Σ: $s \in (2^\Sigma)^+$. To deal with *set sequences* we consider two naïve methods below:

(sum-up) for two sets S_i and S_j of indices i and j, compute straightforwardly weights and sum up them. That is, compute all $W_{S_{i,k},S_{j,l}}$ for $S_{i,k} \in S_i$ and $S_{j,l} \in S_l$ $S_{j,l} \in S_j$ in a pair-wise manner, and use them.

(average) for two sets S_i and S_j, compute the weights in a pair-wise manner as well, and divide the weights $W_{S_{i,k},S_{j,l}}$ with $|S_i| + |S_j|$, to take an average contribution of each symbols in S_i and S_j.

By these small generalizations, we note that the skeletonization techniques can be generalized to set sequences. If not otherwise stated, we adopt the sum-up idea below.

Table 1 Summary of datasets used in experiments.

Name	Length	$\|\Sigma\|$	P	Note
HMM-600-u	600	10	3	with $u = 0.25$
PD-32	365	32	7	Discretized with level 32
PD-128F	100	128	7	Subset with level 128
Kyoto	43,833	359	365	The resolution 0.1 celsius

4. Experiments

We report experiments with synthetic and real datasets which should have simple periodic behaviors to observe the effect of our proposal. We also discuss the difference between two skeletonization methods.

4.1 Datasets

Now in this part, we use both synthetic and real datasets. The summary of these datasets is shown in **Table 1**. A synthetic dataset is generated by using the HMM shown in Fig. 6. A real dataset, named PowerDemand, is a set of sequences of electric power demand in 2013, extracted from the GridWatch system[*5], which were previously used in Fig. 1. The original sequence records power demand in UK 12 times per hour, that is, roughly 300 times per day. We take the simple average of them to construct a hourly sequence of power demand in 2013, named PD-32. Because an yearly record may contain many periodic behaviors (e.g., daily, weekly, monthly, etc.), we extract a small subset, named PD-128F, of PD-32 and make the resolution of Σ more clear by increasing the size Σ from 32 to 128 and taking a part roughly from summer to autumn. For PD-128F, we expect that the sequence have the period $P = 7$. As another real dataset, we use Kyoto, a sequence of the daily temperatures from 1880 to 2014 with $P = 365$ and $|\Sigma| = 359$.

4.2 Preparations

We implemented the periodic skeletonization part in C++[*6], and apply the spectral clustering algorithm (and k-means algorithm in it) by using a built-in implementation by scikit-learn [10] based on Python 2.7.8. All experiments are run on a machine of Mac OS X 10.10 with 2×2.26 GHz Quad-Core Intel Xeon processors and 64 GB memory.

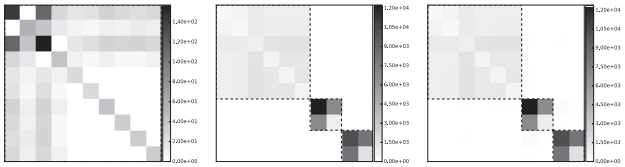
We would like to show computed graphs and the discovered clusters by the spectral clustering algorithm via temporal/periodic graphs. We set k by using the heuristic of the spectral clustering (Please see [13]), or by a small number (2 or 3, for example). In experiments we basically use only the original definition, i.e., we only use the delta function by the indicator function $\mathbf{1}_f$. In the following, we prepare the following labels to represent methods: 1) DT means the temporal skeletonization, 2) DP users the periodic information only, and 3) DTP adopts the both of them.

4.3 Results

Out of several parameter settings we tried, we took a part of results to compare our periodic skeletonization with the temporal one. We showed results of synthetic data in **Fig. 7**, and those of real datasets in **Figs. 8** and **9** with varying methods of computing weights, where the labels {DT, DP, DTP} represent the methods used.

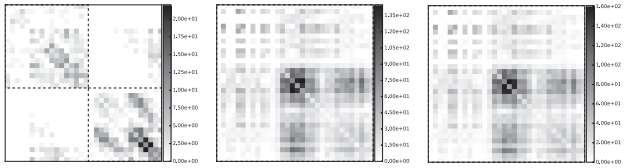
[*5] http://www.gridwatch.templar.co.uk/

[*6] gcc 4.7 with -std=c++11 without parallelization.

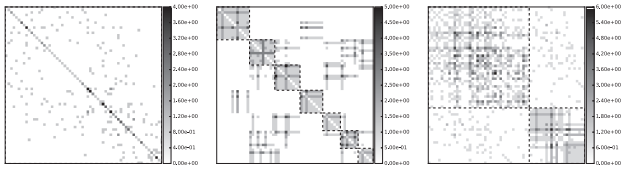


(a) DT (b) DP (c) DTP

Fig. 7 Heatmaps representing similarity matrices of graphs from the synthetic sequence with $P = 3$ and $r = 3$. Figures 7 (b) and 7 (c) successfully show clusters as *rectangles*.

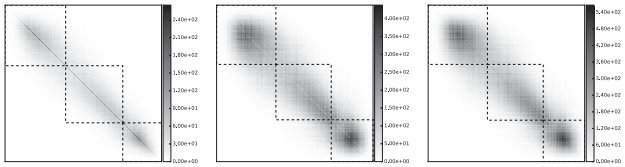


(a) DT (b) DP (c) DTP



(d) DT (e) DP (f) DTP

Fig. 8 Heatmaps from PD-32 (top row) and PD-128F (bottom row) with DT, DP, and DTP.



(a) DT (b) DP (c) DTP

Fig. 9 Heatmaps from Kyoto with DT, DP, and DTP.

Synthetic Datasets: From results using synthetic data, we can conjecture that periodic information of temporal graphs are helpful to find clusters of symbols compared with Fig. 7 (a) and Fig. 7 (b) and Fig. 7 (c), where we would like to extract periodic clusters, that is, clusters representing $\{0, 1\}$ and $\{2, 3\}$, which corresponds to T_1 and T_2 in the HMM in Fig. 6, respectively. From the result using only temporal information in Fig. 7 (a), however, we *cannot* find them. On the another hand, results using periodic information seen in Fig. 7 (b) and both of them in Fig. 7 (c) show two clusters $\{0, 1\}$ and $\{2, 3\}$ much clearly.

Real Datasets: Results from real datasets should be affected by properties of sequences and the periodicity of them. In two cases with PD-32 and PD-128F, for example, results were symmetric with respect to methods: If we use the periodic information in Fig. 8 (b) and Fig. 8 (c), we cannot find any clusters but in Fig. 8 (e) and Fig. 8 (f), we can find a few clusters of symbols,

which are similar to results of synthetic data. We guessed that the difference between PD-32 and PD-128F is whether or not there exists many periodic behaviors in sequences. Because we selected a subsequence from PD-32 as PD-128F to remove multiple periodic information, the periodic skeletonization with a fixed period parameter $P = 7$ seemed to work well.

In results from Kyoto, we can see that there exist roughly 3 clusters in all outputs in Fig. 9. If we adopt the periodic information, those clusters are also emphasized on visualized temporal graphs. For example, by comparing results in Fig. 9 (a) and Fig. 9 (c), we confirm that two dense clusters (top left and bottom right) in Fig. 9 (c) are much more clear than in those Fig. 9 (a). Note that these clusters are related to winter and summer, respectively. We conjecture that these visualized results are helpful to analyze given sequential databases and enumerated patterns, particularly when we need to run methods many times to tune parameters.

Conclusions: We conclude experiments using sequences containing clear periodic behaviors. Originally, results of clustering symbols are sensitive to the definition of similarities. The previous study reported in [8] that results of the skeletonization seemed to be stable. As far as we investigated in experiments, with respect to the parameter r, which control a kind of smoothing of sequences, the results could be stable as well. We also see that our method could be helpful to *highlight periodic behaviors of sequences*. We guess that this result is also affected from the multiple periodicity, and conclude that the periodic skeletonization help us to find underlying structures. Although the method sometimes (as seen in PD-32) disturbs results, it seems to work as we expected particularly when the periodicity is clear.

4.4 Unstable Case

As we show in Fig. 6, the model behind our skeletonization procedure is a cyclic HMM. This assumption suits the FPPPM problem with (fixed) periodic segments. However, in general, the period of a sequence s is unstable because of noisy symbols or outliers. In the pattern mining part, mining algorithms allowing for a small gap (i.e., a kind of wildcard) between segments have been studied [17]. Therefore we would like to discuss the same point on the skeletonization techniques. Because temporal clusters correspond to a kind of moving average on symbols, we conjecture that adopting both the temporal skeletonization and the periodic one should be useful to deal with such a situation.

To investigate this issue, we generated synthetic sequences from a cyclic HMM of P states, and see whether or not we can discover P clusters from the data. The results by DT (only temporal), DP (only periodic), and DTP (both) of the transition failure percentage $\eta = 20\%$ in **Fig. 10** were summarized in **Table 2**. We can confirm that, at least for synthetic data, periodic information used in the skeletonization is helpful to estimate hidden periodic clusters of sequences. From experiments, we guess that using both information could be helpful when real sequences are processed.

4.5 Case Studies

We provide results using (more) real datasets. One is from

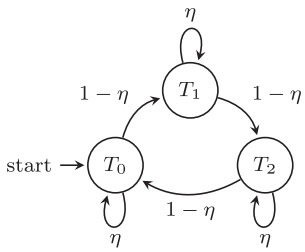


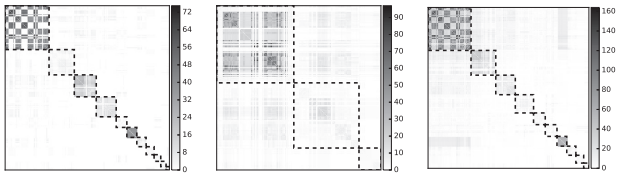
Fig. 10 A cyclic HMM including unstable periods with transition failures with probability η of the case $P = 3$.

Table 2 From P states cyclic HMM, for each state we generate 5 symbols uniformly. The transition failure $\eta = 0.2$ (i.e., 20%). Values show accuracy of recovering hidden clusters.

P	DT	DP	DTP
3	0.33	0.80	0.80
5	0.24	0.92	0.92

Table 3 Statistics of user logs from the Last.fm dataset.

User ID	$\lvert L_{\text{all}}\rvert$	$\lvert L_{\text{ne}}\rvert$ (non-empty)	$\lvert\Sigma\rvert$	$\lVert\mathcal{S}\rVert$
User 672	384	147	247	2,329
User 808	529	147	578	2,108



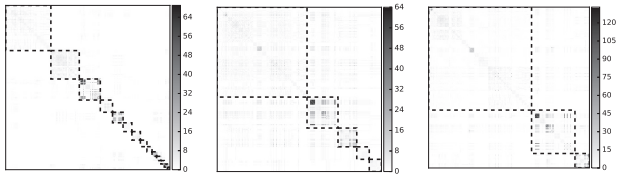
(a) DT (b) DP (c) DTP

Fig. 11 Heatmaps of User 672 from Last.fm datasets with varying DT, DP, and DTP where $P = 24$ and $w = 2$.

Last.fm data that we have used in Section 1. In addition to that, we here adopt 2-dimensional sequences (i.e., X and Y) representing trajectories of movements, and encode them as (1-dim) symbolic sequences for experiments.

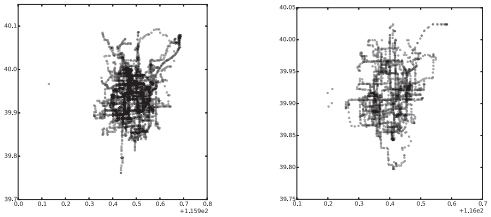
Last.fm Datasets: Because properties of data vary according to users, we would like to investigate how results get for real datasets. Datasets are obtained from [2] by gathering and ordering the logs of songs listened by users based on focusing the granularity "hour". One database corresponds to one sequence of sets of symbols (i.e., songs) by one user. For experiments, we take randomly users from the whole dataset, obtain sequences of sets of symbols, and use small parts of such sequences. We provide statistics of selected parts of sequences chosen by our method in **Table 3**. For a sequence $s = \langle S_1, S_2, \ldots, S_M \rangle$ of $S_i \subseteq \Sigma$, L_{all} means $|s| = M$, L_{ne} shows $|\{S_i \mid S_i \neq \emptyset, S_i \text{ is in } s\}|$, Σ means the size of the set $|S_1 \cup \cdots \cup S_M|$, and $\|\mathcal{S}\| = \sum_i |S_i|$, respectively. We set $P = 24$ to analyze hourly behaviors.

We show results in **Figs. 11** and **12**. Here we do not want to say which clustering results are good (or bad). From experiments by periodic information in the skeletonization we can confirm two kind of results: A type increases the number of clusters compared with the ordinal temporal skeletonization (e.g., from Fig. 11 (a) to Fig. 11 (c)). Another type, in contrast, decreases the number of clusters (e.g., from Fig. 11 (a) to Fig. 11 (b), Fig. 12 (a) to Fig. 12 (b) and 12 (c)). As the periodic information help us to



(a) DT (b) DP (c) DTP

Fig. 12 Heatmaps of User 808 from Last.fm datasets with varying DT, DP, and DTP where $P = 24$ and $w = 2$.



(a) ID 1277 (b) ID 6275

Fig. 13 Examples of trajectory plots.

Table 4 Statistics of discretized trajectories.

ID	|DB|	# of non-empty (|Σ|)
1277	8187	3410
6275	3960	2450

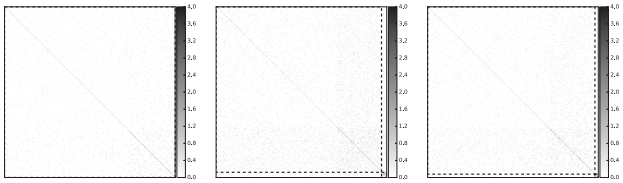
consider periodic co-occurrences of symbols, if there exist some periodic behaviors of sequences, then applying the periodicity skeletonization should be helpful. We can only confirm that in some cases the clustering work for our purpose. We conjecture that for some databases our method does not work as they contain no periodic regularity.

Trajectory Datasets: As an another example of skeletonization and mining, we adopt some trajectory time-stamped databases used in Refs. [15], [16][*7]. A trajectory here is an ordered sequence of pairs, i.e., (X, Y) corresponding to longitude and latitude of an entity. Its movements is regularly recorded and stored as a sequential database. An example of trajectory can be plotted in 2-dimensional space as shown in **Fig. 13**. Note that a sequential database DB is now encoded as a single sequence s by discretizing X and Y, and putting some integer $n \in \mathbb{N}$ on each grid. **Table 4** shows some statistics of trajectories used. All trajectories are discretized with level $d = 256$, and many (X, Y) slots on the grid could be empty. Thus the table also shows the number of non-empty slots in discretized sequences. As they include some obvious outliers or noise, such data are cleaned by checking the mean and standard decreases of X and Y.

In contrast to the datasets used above, for trajectories, we have no idea on what are its period. The following experiments, therefore, we give a kind of exploratory data analyzing processes using skeletonization by observing the similarity graphs constructed. In addition, we only try the case $k = 2$, i.e., try to divide all symbols Σ into two groups as the basis of recursively decomposition of Σ.

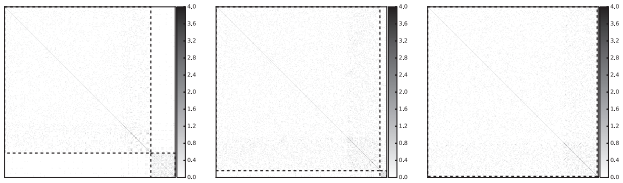
Figures 14 and **15** are results of the skeletonization with $w = 45$ and $P = 60$ and 120. First of all, compared with synthetic cases or some real datasets used above, trajectory data are much more sparse. That is, most symbols in Σ appear only once in our discretized sequential databases. In results, similar results are obtained compared with Last.fm datasets, but it was difficult to find clusters automatically without tuning algorithms. In fact, only small clusters (illustrated in the right-bottom part) can be found: From Fig. 14 (a) to Fig. 14 (b) and Fig. 14 (c), a small cluster containing a few symbols was found. By contrast, from Fig. 15 (a) to Fig. 15 (b), Fig. 15 (c), a (relatively) large cluster disappeared, and a small cluster was found again.

[*7] See also http://research.microsoft.com/en-us/projects/tdrive/



(a) DT (b) DTP (60) (c) DTP (120)

Fig. 14 Heatmaps of ID 1277 with DT and DTP ($w = 45$, $P = 60$ and 120), trying with the number $k = 2$ of clusters.



(a) DT (b) DTP (60) (c) DTP (120)

Fig. 15 Heatmaps of ID 6275 with DT and DTP ($w = 45$, $P = 60$ and 120), trying with the number $k = 2$ of clusters.

4.6 Discussions

Discussing the quality of clusters is fundamentally impossible as we do not have any labels. Conceptually, the skeletonization does *not* use any semantic information of symbols, and results only depend on the co-occurrences of symbols. In our method, we intend that adding more computations by the periodicity have increased information we can use in the pre-processing step. Introducing additional resources for computing the similarities such as background knowledge or taxonomy is one of interesting future work. However, such knowledge resources are in general *high cost* compared with the skeletonization. Therefore, we guess that combining both methods is much effective for solving the sparseness problem. As another direction on the viewpoint of the increase of syntactic information when constructing similarity graphs, taking into account the *order* of indices or the *distance* of them should be an interesting future problem. These are also related to the application step of graph clustering methods. In addition, we also expect that introducing sophisticated clustering algorithms is important: For example, hierarchical spectral clustering [1], Non-negative Matrix Factorization (NMF) (e.g., [3]) should be helpful (e.g., considering multiple periods with hierarchy).

5. Mining Meets Skeletonization

Recall our start point of the discussion on pattern mining in real situations. We now try to apply pattern mining algorithms based

Algorithm 1 Incremental Mining with Skeletonization

1: **procedure** Inc-MS(s (input), θ (threshold))
2: Construct a similarity graph G from s
3: Estimate the number k of clusters (with heuristics)
4: Compute assignments of symbols ▷ $\Sigma \to \{1, 2, \ldots, k\}$
5: Sort clusters $C_1, \cdots C_k$ with their cardinality ($|C_1| > |C_2| > \ldots |C_k|$, descending)
6: **for** $j = 1$ to k **do** ▷ (Or, any j ($1 \le j \le k$) if you want)
7: Replace symbols in s using the cluster C_j and get the encoded sequence $s^{(\ge j)}$ ▷ Re-encoding
8: Apply a mining method to $s^{(\ge j)}$ with θ

on the clusters discovered by the skeletonization techniques.

In the following, we consider two cases: 1) We have some assumptions on P, and 2) we have no idea on P. For the case 1), we solve the FPPPM problem with a fixed P we have in mind. For the case 2), we allow for patterns contain a gap among symbols in the following form based on [17]: Let $p = c_1 c_2$, where $c_1, c_2 \in \Sigma$ and $[\alpha, \beta]$ be an closed interval of integers. Then, we interpret the pattern p as follows: p has at least α and at most β wildcard characters between c_1 and c_2. For example $s_1 = abc$ and $s_2 = abbbc$, a pattern ac with $[\alpha, \beta] = [0, 2]$ *matches* with s_1, but does not match with s_2 because three b occurs between the first a and c.

Experiments: We now apply the periodic skeletonization with mining problems (e.g., the FPPPM problem). Our purpose here is to obtain readable and high-level patterns in mining by reducing the size $|\Sigma|$. We use clustering results obtained by the above experiments.

For enumeration of patterns in the case 1), we use the algorithm proposed by Yang et al. [14], and call it PPPMiner. For the purpose, we re-implemented PPPMiner in Python 2.7.8. Experimental settings are the same to those in Section 4.2. To examine how the periodic skeletonization affects enumerating patterns by the PPPMiner, we use Kyoto and Last.fm datasets. On the other hand, for the case 2) we adopt the algorithm by Zhang et al. [17], which is also re-implemented, named by GAPMiner, in Python 2.7.8.

Based on the number k of clusters found (or estimated, fixed, etc.), we propose an *incremental method*, where users replace symbols with cluster labels incrementally. The overview of this process is shown in Algorithm 1. We first sort clusters by the size (in Line 5). In descending order of the size, we incrementally re-encode an original sequence s with cluster labels. That is, we first replace symbols in s belonging to the largest cluster label C_1 (this new sequence is denoted by $s^{(\ge 1)}$ in Algorithm 1). We then do the same with the second largest cluster C_2, and continue this replacement (corresponding to Line 6 and Line 7). For each step in Line 8, we apply a mining method.

5.1 Mining with the PPPMiner

In the following experiments, we adopt both (temporal and periodical) information in the clustering step. For the Kyoto dataset, using $k = 3$ clusters, we prepared four cases: Kyoto (original), Kyoto$^{(\ge 1)}$, Kyoto$^{(\ge 2)}$, and Kyoto$^{(\ge 3)}$ to apply the PPPMiner. We show the number of enumerated patterns with $P = 365$ and with varying θ in **Table 5** (a). For the User 672 dataset from the Last.fm dataset, we use the number $k = 10$ of clusters to preprocess. Out of $k = 10$ clusters illustrated in Fig. 11 (c), for the integer j in Line 6, we use the largest cluster C_1 and get the re-encoded sequence User 672$^{(\ge 1)}$ corresponding to $j = 1$. In the same manner, we adopt the top three largest clusters C_1, C_2, and C_3 (i.e., $j = 3$) and get the sequence User 672$^{(\ge 3)}$. We finally use all clusters ($j = 10$) and label the obtained sequence as User 672$^{(\ge 10)}$. We show in Table 5 (b) the numbers of enumerated patterns.

Table 5 Numbers of enumerated patterns with (i.e., re-encoding labeled as $^{(\ge j)}$) and without the skeletonization with the PPPMiner.

(a) For the Kyoto dataset ($P = 365$)

Datasets	$\theta = 0.9$	0.7	0.5	$\lvert\Sigma\rvert$
Kyoto	0	0	0	359
Kyoto$^{(\ge 1)}$	9,065	57,596	133,027	224
Kyoto$^{(\ge 2)}$	28,134	210,806	523,021	97
Kyoto$^{(\ge 3)}$	54,354	349,648	917,403	3

(b) For User 672 dataset ($P = 24$)

Datasets	$\theta = 0.3$	0.2	0.1	$\lvert\Sigma\rvert$
User 672	0	0	0	247
User 672$^{(\ge 1)}$	128	318	51,304	177
User 672$^{(\ge 3)}$	128	319	22,540	144
User 672$^{(\ge 10)}$	127	260	5,718	10

Discussions: In both cases we cannot find any frequent patterns *without* the periodic skeletonization. That is, without any pre-processing, databases are sparse and we cannot evaluate the support count well to get insights from datasets in the form of partially periodic patterns. However, with thanks to the periodic skeletonization, we can find many frequent patterns in other cases.

Because the periodic skeletonization help us to find rough, characteristic patterns by clustering, we can find abstract but readable and high-level frequent patterns. For example in the Kyoto$^{(\ge 1)}$ setting, we can find 9,065 patterns which characterize 90% of segments in the given sequence. In addition, in the settings of User 672$^{(\ge 1)}$ and User 672$^{(\ge 3)}$, we *successfully* find roughly 300 frequent patterns that characterize 20% of segments, and this number of patterns is relatively small and easy to analyze.

We confirmed that clustering using the skeletonization as a preprocessing of pattern mining work well to get more frequent patterns than those obtained from raw sequences.

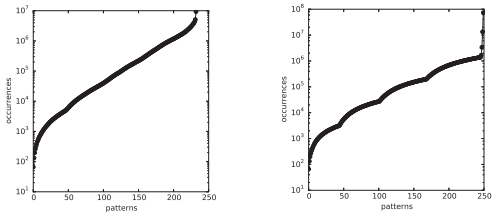
5.2 Mining with the GAPMiner

We adopt the trajectory sequence of ID 1277 used the above. In this case, we use the number $k = 2$ of clusters because we have no idea on the numbers of discretized trajectories. Then the largest cluster C_1 is replaced and a new encoded dataset, denoted by ID 1277$^{(\ge 1)}$, is obtained. We let $[\alpha, \beta]$ – the gap admitted among characters – to be $\alpha = 0$ and $\beta = 5$, and mine patterns with gaps up to the length 6 as an example. As shown in **Table 6**, we could find almost similar numbers of patterns of length from 3 to 6.

Discussions: To investigate the differences by using the support count of patterns we give logarithmic plots of them in **Fig. 16**. The figure shows the difference of patterns enumerated by GAPMiner clearly. Without any pre-processing, in Fig. 16 (a),

Table 6 Numbers of enumerated patterns for ID 1277 trajectory, with threshold 0.00022 of the gap $[\alpha,\beta] = [0,5]$ by using the GAPMiner.

Datasets	Length 3	4	5	6
ID $1277^{(\geq 1)}$	46	58	68	78
ID 1277	72	59	51	51



(a) From ID 1277 (b) From ID $1277^{(\geq 1)}$

Fig. 16 Two logarithmic plots of the support count of patterns enumerated by the GAPMiner from two sequences.

the frequency varies in exponential. On the other hand, by the skeletonization in Fig. 16 (b), patterns can have similar frequency, as results of removing rare symbols from Σ by replacing the symbols in the large cluster discovered with a cluster label. This result appeared as an waved curve on the logarithmic plot in Fig. 16 (b).

Through experiments, we confirmed that the skeletonization affected the distribution of the frequency of patterns. We have many choices on how to replace symbols originally in Σ with cluster labels, and in these experiments we use cluster labels in an descending order.

5.3 Future Directions

It is the always problem in pattern mining how to deal with a kind of redundancy among patterns. In our experiments, many patterns constructed by shifting symbols are mined. For example, we often have the case a frequent pattern p occurring at 8 a.m. again becomes frequent patterns at 9 a.m., 10 a.m., and so on. That is, the primitive definition of partially periodic patterns have some redundancy. It is our important future work to overcome the redundancy problem by considering well-studied conceptions (e.g., *closed* or *maximal* patterns) and combining them with our pre-processing method. Tuning hyper-parameters including the number k of clusters and the width r of sliding windows is also our future work to enrich partially periodic pattern mining with pre-processing using the skeletonization.

Last, we show a future problem should be taken into account the pre-processing of pattern mining with Machine Learning methods by using the support count. **Figure 17** is a logarithmic chart (of the y-axis) representing the frequency of each symbol[*8] in Σ for the ID 1277 sequence. If given databases are not merely noise, we can have an assumption that the symbol frequency is not uniform. This fact encourages more developments on applications of Machine Learning methods for the sparseness of several pattern mining problems. For example considering the TF-IDF model in pattern mining, utility-based pattern mining, and unsupervised metric learning on symbols area related topics on the viewpoint of pre-processing using Machine Learning.

[*8] It is equal to the support count of each symbol, i.e., the total number of occurrences of symbols in a database.

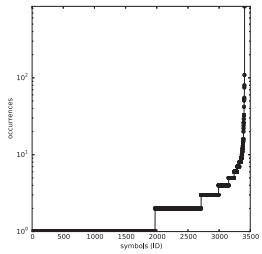


Fig. 17 The logarithmic plot of the frequency of symbols.

6. Concluding Remarks

In this paper we provide a new skeletonization method for dealing with partially periodic patterns based on the temporal skeletonization and periodic information. Our experiments with synthetic and real datasets show that our method could help us to obtain clusters of symbols even for periodic settings, particularly for a case where sequences have only one fixed period. Pattern mining results with the skeletonization indicate that our method is helpful to obtain readable results with a relatively small computational cost as Σ get small. Even we use a large threshold, we can find frequent patterns which cannot be listed without the skeletonization. Using more real datasets, we test that our method give some insights on relation of symbols used for describing databases, and their analyses might be important and helpful for Knowledge Discovery.

In future work, we would like to develop algorithms to reduce the redundancy of patterns more, based on well-studied concepts (e.g., closed patterns) together with the skeletonization. Further discussion using other pre-processing methods, particularly comparing methods using semantic information (i.e., hierarchy, background knowledge) are also our important future work.

Acknowledgments This work was partially supported by Grant-in-Aid for JSPS Fellows (26-4555) and JSPS KAKENHI Grant Number 26280085.

References

[1] Alzate, C. and Suykens, J.A.: Hierarchical kernel spectral clustering, *Neural Networks*, Vol.35, pp.21–30 (2012).
[2] Celma, O.: *Music Recommendation and Discovery in the Long Tail*, Springer (2010).
[3] Cichocki, A., Zdunek, R. and Amari, S.-I.: Nonnegative matrix and tensor factorization [lecture notes], *Signal Processing Magazine, IEEE*, Vol.25, No.1, pp.142–145 (2008).
[4] Han, J., Dong, G. and Yin, Y.: Efficient mining of partial periodic patterns in time series database, *Proc. 15th ICDE*, pp.106–115 (1999).
[5] Han, J., Gong, W. and Yin, Y.: Mining segment-wise periodic patterns in time-related databases, *Proc. 4th KDD*, pp.214–218 (1998).
[6] Huang, P., Liu, C.-J., Xiao, L. and Chen, J.: Wireless spectrum occupancy prediction based on partial periodic pattern mining, *Proc. 20th MASCOTS*, pp.51–58 (2012).
[7] Le, A. and Gertz, M.: Mining periodic event patterns from RDF datasets, Catania, B., Guerrini, G. and Pokorný, J., (Eds.), *Advances in Databases and Information Systems*, Lecture Notes in Computer Science, Vol.8133, Springer Berlin Heidelberg, pp.162–175 (2013).
[8] Liu, C., Zhang, K., Xiong, H., Jiang, G. and Yang, Q.: Temporal skeletonization on sequential data: patterns, categorization, and visualization, *Proc. 20th KDD*, pp.1336–1345 (2014).
[9] Ng, A.Y., Jordan, M.I. and Weiss, Y.: On spectral clustering: analysis and an algorithm, *Advances in Neural Information Processing Systems 13*, pp.849–856 (2001).
[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E.: Scikit-learn: machine learning in Python, *Journal*

of Machine Learning Research, Vol.12, pp.2825–2830 (2011).
[11] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U. and Hsu, M.-C.: Mining sequential patterns by pattern-growth: the PrefixSpan approach, *IEEE Transactions on Knowledge and Data Engineering*, Vol.16, No.11, pp.1424–1440 (2004).
[12] Shi, J. and Malik, J.: Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22, pp.888–905 (1997).
[13] Von Luxburg, U.: A tutorial on spectral clustering, *Statistics and Computing*, Vol.17, No.4, pp.395–416 (2007).
[14] Yang, K.-J., Hong, T.-P., Chen, Y.-M. and Lan, G.-C.: Projection-based partial periodic pattern mining for event sequences, *Expert Systems with Applications*, Vol.40, No.10, pp.4232–4240 (2013).
[15] Yuan, J., Zheng, Y., Xie, X. and Sun, G.: Driving with knowledge from the physical world, *Proc. 17th KDD*, ACM, pp.316–324 (2011).
[16] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G. and Huang, Y.: T-drive: driving directions based on taxi trajectories, *Proc. 18th SIGSPATIAL GIS*, pp.99–108 (2010).
[17] Zhang, M., Kao, B., Cheung, D.W. and Yip, K.Y.: Mining periodic patterns with gap requirement from sequences, *ACM Trans. Knowledge Discovery from Data*, Vol.1, No.2 (2007).

Keisuke Otaki received his B.E. degree and the M.S. degree in Informatics from Kyoto University in 2011 and 2012, respectively. He is currently a Ph.D. student at Graduate School of Informatics, Kyoto University. His research interests include pattern mining from structured data and Information Theory-based mining algorithms.

Akihiro Yamamoto received his B.S. degree from Kyoto University in 1985, and Dr.Sci. degree from Kyushu University in 1990. Currently, he is a Professor of the Department of Intelligence Science and Technology, Graduate School of Informatics at Kyoto University. He has made research contributions to foundations of intelligence science, with a particular focus on application of mathematical logic to machine learning. His recent research interest includes developing machine learning theory with discrete mathematics, in particular formal concept analysis. He is joining Information Processing Society Japan, Japan Society for Software Science and Technology, and the Japanese Society for Artificial Intelligence.