Vol.2015-MPS-105 No.7
2015/9/29

Periodic Pattern Mining with Periodical Co-occurrences of Symbols

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

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 explosion of patterns 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 important problems in data mining. Transactions in databases naturally have timestamps and are often ordered with their timestamps in the chronological order, from old to new. If such an order is important to a database, it 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 the data. 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 based on the periodicity, *periodic pattern mining* have been studied [3], [4], [10].

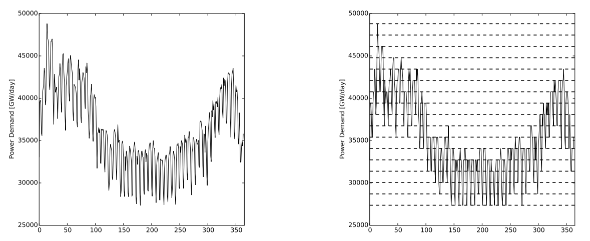
We have several variations on the definition of periodic patterns in the literature. The fundamental ones are *full periodic patterns* and *partial periodic patterns* [3]. For example, let $\Sigma = \{\mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{shops}\}$. We consider a sequence $s = (\mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{sns}, \mathrm{news}, \mathrm{blog}, \mathrm{sns}, \mathrm{shop}, \mathrm{blog})$ representing categories of Web sites visited by a user. A pattern (sns, news, blog) in s appears twice, and this is called *full periodic pattern* of period length 3. Full periodic patterns require that all symbols be fully specified. In some cases, such requirement is not flexible and it is difficult to handle various periodic behaviors. As more flexible patterns, *partial periodic patterns* have been studied [4]. 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 (sns, news, blog), (sns, news, blog), and (sns, shop, blog) of s.

Although many efficient algorithms have been developed [3], [10], 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 supports difficult because most patterns have similar and relatively small supports. That is, the space of (frequent) patterns on Σ get *sparse* with respect to the space of all possible patterns.

Motivating Examples: For both numerical (e.g., price, temperature) and symbolic (e.g., item, product) sequences, preparing 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[*1] as seen in Fig. 1(b) with $|\Sigma| = 16$ bins. 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

[1] Kyoto University
[2] Research Fellow of the Japan Society for the Promotion of Science
[a] ootaki@iip.ist.i.kyoto-u.ac.jp

[*1] 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.



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

Fig. 1: A numeric sequence in Fig. 1(a), and its discretization with 16 symbols (corresponding to dashed lines) in Fig. 1(b).

informative and useful to analyzing databases.

As another example we consider those stored from Last.fm (`http://last.fm/`), which generate sequences of songs logged by users, where similar tendencies can be seen in symbolic data as well. We take some logs of users from an open dataset (See Section 3 of [2]), where a sequence $s = (S_1, S_2, \dots)$ is a log of a user and each S_i is the set of songs heard in the index i, where 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 interval, $24,310^{24}$ is the upper bound of all combinations. Again, this is intractable and sparse.

Approaches: In pattern mining, therefore, Liu et al. [5] 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 cluster labels, which roughly characterize the given sequence. By grouping symbols into clusters, we reduce the size of Σ. We develop such method for periodic analyses by generalizing the idea [5], and discuss frequently occurring high-level periodic patterns.

We would like to emphasis on the fact that many existing studies dealt with DNA sequences, which requires only 4 symbols (i.e., $\Sigma = \{T, C, A, G\}$). In such a situation, the combinatorial explosion is only depending on the length l of patterns we try to mine. This situation, however, 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 [5]. 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 an 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 is *periodic segments*.

Definition 1 (Event Sequence and 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 $1 \le i \le m$, $ps_i = s_{im,(i+1)m-1}$.

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

Periodic patterns we want to find are those appearing in periodic segments frequently. For a sequence s and a pattern p, the traditional measure for evaluating the interestingness of patterns is to adopt *support* of patterns defined below.

Definition 3 (Support) For a sequence s and a pattern p of the same length, *s is covered by* the pattern p if and only if $p_i = \star$ or $p_i = s_i$ for all $1 \le i \le |s|$, denoted by $s \preceq p$. The *support* of p, denoted by $\mathrm{Supp_P}(p)$, is defined as $\mathrm{Supp_P}(p, s) = |\{ps_i \mid s = \langle ps_1, \dots, ps_m \rangle, ps_i \preceq p\}|$. If a pattern p satisfies $\mathrm{Supp_P}(p) \ge \theta$ with θ and P, it is called a *frequent partially periodic pattern* (PPP).

Problem 1 (The FPPPM problem) For a sequence s, list all partially periodic patterns p from s satisfying $\mathrm{Supp_P}(p) \ge \theta$.

Several efficient algorithms have been developed for the FPPPM problem. For examples, Han *et al.* showed a fundamental algorithm using max sub-pattern trees [4] and Yang *et al.* proposed a depth-first search algorithm based on projections.

2.2 Temporal Skeletonization

We refer to the original definition of *temporal graphs* to explain the idea of the temporal skeletonization in [5], which tries to build a *similarity graph*[*2] from a given database DB.

Definition 4 (Temporal Graph [5]) Let $G = (V, E)$ be an undirected graph, where V corresponds to Σ. Let $\mathrm{DB} = \{s^{(1)}, \dots, s^{(N)}\}$ be the set of sequences. For $x, y \in \Sigma$, the weight $W_{x,y}$ of the edge corresponding to $\{x, y\}$ is defined as

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

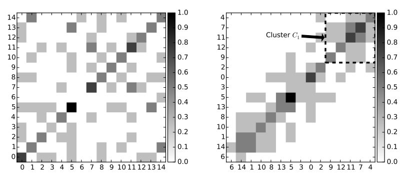
where N is the number of sequences, r be the *window width*, $\mathbf{1}_f$ is the indicator function that returns 1 if and only if f is true.

The right-hand side of Equation 1 can be computed by checking the given database DB, where the indicator function can be replaced with other similarity measures. The authors in [5] used

[*2] A *similarity graph* is a weighted graph in which vertices represent data 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 DB $= \{s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)}\}$



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

Fig. 2: A toy example in [5] 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.

the exponential function $\exp(-k|i-j|)$ with a parameter k. In constructing G, for indices $1 \le i \le j \le |s|$, a simple implementation is to increment the weight W_{s_i,s_j} if $|i-j| \le r$ for $s \in \mathrm{DB}$. After constructing G, users try to find clusters of symbols by applying clustering methods to G (such as *spectral clustering* [6], [8]). The problem of finding clusters can be formulated as a standard graph-based optimization problem with some constraints as shown in [5], where an important step of clustering is to compute eigenvalues and eigenvectors from G. Now a matrix W of weights from G can be represented as a heatmap as shown in Fig. 2.

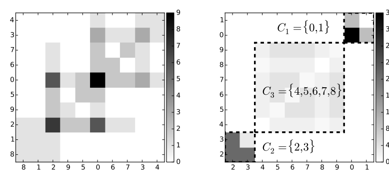
Example 1 The input is shown in Fig. 2(a). We compute the weights from $\{s^{(1)}, s^{(2)}, s^{(3)}, s^{(4)}\}$ as seen in Fig. 2(b) and represent them by a heatmap, where both the x-axis and y-axis correspond to some order of the alphabet Σ. After applying the spectral clustering, we can re-order indices 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 are in the form $(C_1, C_1, C_1, \ldots)$.

3. Periodic 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. 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[*3]. 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. Please recall the toy examples used 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. This observation inspired our modification for *periodic skeletonization*.

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

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



(a) Temporal (b) Periodic

Fig. 3: Fig. 3(a) is a result only using the temporal information and Fig. 3(b) is that adopting the periodic information only, where rectangles are the discovered clusters.

$$\begin{tikzcd}
 & T_1 \arrow[dr, bend left, "1"] & \\
\text{start} \arrow[r] & T_0 \arrow[u, bend left, "1"] & T_3 \arrow[l, bend left, "1"]
\end{tikzcd}$$

Fig. 4: An HMM for Example 2.

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

The second term is newly introduced with the period P.

Example 2 Fig. 3 shows examples of computing Equation 2 from $s = (0, 2, 6, 0, 2, 4, \ldots)$ with $\Sigma = \mathbb{N}$. Fig. 3(a) is computed by the temporal skeletonization, while Fig. 3(b) adopts the periodic term only in Equation 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. 3(b), and they are clear than those in Fig. 3(a).

Our basic observation is that we can have periodic sequences by *cyclic HMMs*. An example is given in Fig. 4. 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 with low probability $100 \times u\%$. On the other hand in T_3, $\mathcal{H}$ generates $\{4, 5, 6, 7, 8, 9\}$ uniformly. A sequence generated from this HMM includes $02\star$ frequently. Compared with the result in Fig. 3(a), we can see that C_1, C_2, C_3 correspond to T_1, T_2, T_3 more clearly as blocks in the similarity matrix in Fig. 3(b). Our periodic skeletonization is a generalized method using the *periodic co-occurrences* of symbols.

4. Experiments

We report experiments with synthetic and real datasets which should have simple periodic behaviors to observe the effect of our proposal. 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. 4. A real dataset, named PowerDemand, is a set of sequences of electric power demand in 2013, extracted from the GridWatch system[*4], which were previously used in Fig. 1. With the discretization level $d = 32$, we make a hourly sequence 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

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

Table 1: Summary of datasets having some clear period.

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°C

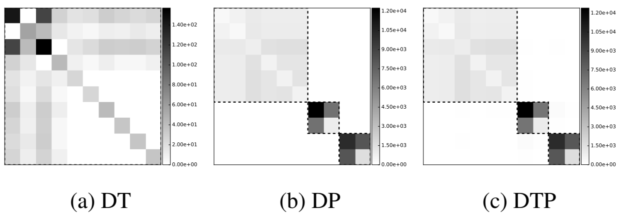

(a) DT (b) DP (c) DTP

Fig. 5: Heatmaps representing similarity matrices of graphs from synthetic sequences (HMM-600-u) with $P = 3$ and $k = 3$. Figures 5(b) and 5(c) successfully show clear clusters as *rectangles*.

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 dataset, we use Kyoto, a sequence of the daily temperatures from 1880 to 2014 with $P = 365$ and $|\Sigma| = 359$.

We implemented the periodic skeletonization part in C++ [*5], and apply spectral clustering (and k-means algorithm in it) by using the SCIKIT-LEARN [7] package 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 64GB memory.

We would like to show computed graphs and the discovered clusters. We set k by using the heuristic of the spectral clustering (Please see [9]), or to be 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. Out of several parameter settings we tried, we took a part of results. We showed results of synthetic data in Fig. 5, and those of real datasets in Fig. 6 with varying methods.

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. 5(a) and Figures 5(b) and 5(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. 4, respectively. From the result using only temporal information in Fig. 5(a), however, we *cannot* find them. On the another hand, results using periodic information seen in Fig. 5(b) and both of them in Fig. 5(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 Figures 6(b) and 6(c), we cannot find any clusters but in Figures 6(e) and 6(f), we can find a few clusters of symbols, which

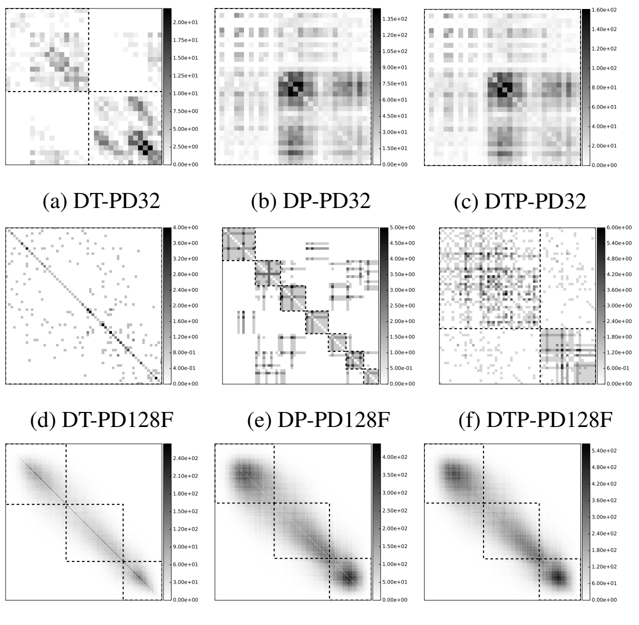

(a) DT-PD32 (b) DP-PD32 (c) DTP-PD32
(d) DT-PD128F (e) DP-PD128F (f) DTP-PD128F
(g) DT-Kyoto (h) DP-Kyoto (i) DTP-Kyoto

Fig. 6: Heatmaps from PD-32 (top row), PD-128F (middle row), and Kyoto (bottom row), with DT, DP, and DTP.

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 periods, 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. If we adopt the periodic information, those clusters seem to get much clearer. For example, in Fig. 6(g) and Fig. 6(i), we confirm that two dense clusters (top left and bottom right) in Fig. 6(i) are much clearer than in those Fig. 6(g). 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.

Conclusions: We conclude experiments using sequences containing clear periodic behaviors. Originally, results of clustering symbols are sensitive to definitions of similarities. The previous study reported in [5] that results of the skeletonization seemed to be stable. As far as we investigated in experiments, with respect to the parameters r, which control a kind of smoothing of symbols 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.

4.1 Case Studies

We provide results using (more) real datasets. One is from Last.fm data that we have used in Section 1. We also adopt 2-dimensional sequences representing trajectories of movements, and encode them as (1-dim) symbolic sequences.

Last.fm Datasets: Because properties of data vary according

[*5] GCC 4.7 with -STD=c++11 without parallelization

Table 2: Statistics of user logs from the Last.fm dataset.

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

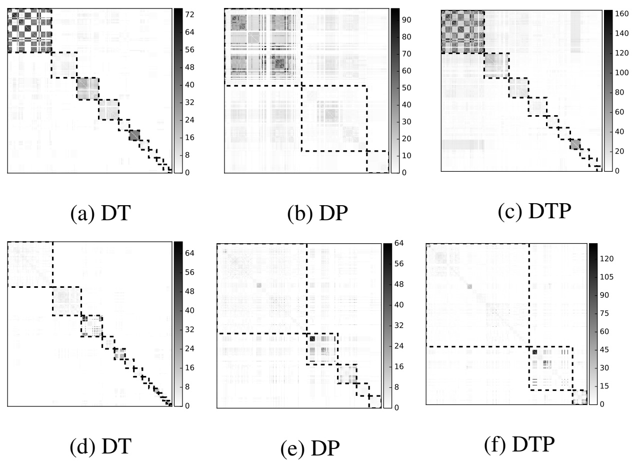

(a) DT (b) DP (c) DTP
(d) DT (e) DP (f) DTP

Fig. 7: Heatmaps of Users 672 and 808 from Last.fm datasets with varying DT, DP, and DTP where $P = 24$ and $w = 2$.

Table 3: Statistics of discretized trajectories.

ID	$\lvert$DB$\rvert$	# of non-empty ($\lvert\Sigma\rvert$)
1277	8187	3410
6275	3960	2450

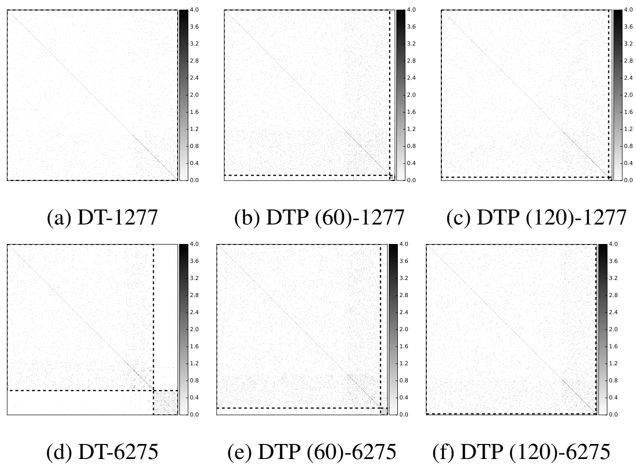

(a) DT-1277 (b) DTP (60)-1277 (c) DTP (120)-1277
(d) DT-6275 (e) DTP (60)-6275 (f) DTP (120)-6275

Fig. 8: Heatmaps of IDs 1277 and 6275 with DT and DTP ($w = 45$, $P = 60$ and 120), trying with $k = 2$ (# of clusters).

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 sampled users from the whole dataset, obtained sequences of sets of symbols, and used small parts of them. We provided statistics of selected data in Table 2. For a sequence $s = S_1, S_2, \ldots, S_M$ of $S_i \subseteq \Sigma$, $L_{\rm all}$ means $|s| = M$, $L_{\rm ne}$ 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 Fig. 7. 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. 7(a) to Fig. 7(c)). Another type, in contrast, decreases the number of clusters (e.g., from Fig. 7(a) to Fig. 7(b), Fig. 7(d) to Figures 7(e) and 7(f)). As the periodic information help us to consider periodic co-occurrences of symbols, if there exist some periodic behaviors of sequences, then applying our method should be helpful.

Trajectory Datasets: As an another example of skeletonization and mining, we adopt some trajectory time-stamped databases used in [11], [12]. A trajectory here is an ordered sequence of pairs, i.e., (X, Y) corresponding to longitude and latitude of an entity. A sequential database DB is now encoded as a single sequence s by discretizing X and Y, and putting some integer $n \in \mathbb{N}$. Table 3 shows some statistics of trajectories used, in which data 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 grids used.

Figure 8 are results of the skeletonization with $w = 45$ and $P = 60$ and 120. Compared with synthetic cases or some real datasets used above, trajectory data are 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 (in the right-bottom part) can be found: From Fig. 8(a) to Fig. 8(b) and Fig. 8(c), a small cluster containing a few symbols was found. By contrast, from Fig. 8(d) to Fig. 8(e) Fig. 8(f), a (relatively) large cluster disappeared, and a small cluster was found again.

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 preprocessing 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. In addition, we also expect that introducing sophisticated clustering algorithms is important (e.g., hierarchical spectral clustering [1]).

5. Mining Meets Skeletonization

We try to apply pattern mining algorithms based on the clusters discovered by the skeletonization techniques. We consider the 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 based on [13]. In experiments, 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.* [10], and call it PPPMiner. To examine how

Table 4: Numbers of enumerated patterns with and without the skeletonization together with the PPPMiner.

Table 5: For the Kyoto dataset (P = 365)

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

Table 6: For User 672 dataset (P = 24)

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

our method affects patterns enumerated 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 [13], named GAPMiner. All algorithms are re-implemented in Python 2.7.8.

Mining with the PPPMiner: For the Kyoto dataset, using $k = 3$, we prepared four cases: Kyoto (original), Kyoto$^{(\geq 1)}$, Kyoto$^{(\geq 2)}$, and Kyoto$^{(\geq 3)}$ to apply the PPPMiner. We show the number of enumerated patterns with $P = 365$ and with varying θ in Table 5. For the User 672 dataset from the Last.fm dataset, we first apply both skeletonization methods as shown in Fig. 7(c) as well. We use the number $k = 10$ of clusters to pre-process. Out of $k = 10$ clusters illustrated in Fig. 7(c), for the integer j in Line 6, we use the largest cluster C_1 and get the re-encoded sequence User 672$^{(\geq 1)}$ 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$^{(\geq 3)}$. We finally use all clusters ($j = 10$) and label the obtained sequence as User 672$^{(\geq 10)}$. We show in Table 6 the numbers of enumerated patterns.

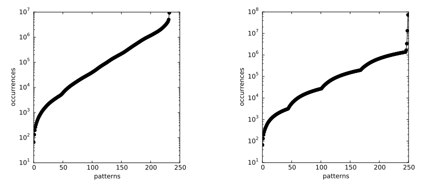
In both cases we cannot find any frequent patterns *without* the periodic skeletonization. That is, without any pre-processing, databases are sparse and we cannot evaluate the support count well. However, with thanks to the periodic skeletonization, we can find many frequent patterns in other cases. Because the method helps us to find rough, characteristic patterns by clustering, we can find readable and high-level frequent patterns. For example in the Kyoto$^{(\geq 1)}$ setting, we found 9,065 patterns which characterize 90% of segments in the given sequence. In addition, in the settings of User 672$^{(\geq 1)}$ and User 672$^{(\geq 3)}$, we found roughly 300 frequent patterns that characterize 20% of segments, and this number is *relatively small* and easy to analyze.

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. Then the largest cluster C_1 is replaced and a new encoded data ID 1277$^{(\geq 1)}$ is obtained. We used $[\alpha, \beta]$, the minimum and maximum gap admitted among characters, to be $\alpha = 0$ and $\beta = 5$, and mine patterns up to the length $l = 6$. As shown in Table 7, we listed almost similar numbers of patterns (from $l = 3$ to 6). However, we can confirm a large difference between two patterns enumerated by checking the support of each pattern enumerated in the logarithmic scale as shown in Fig. 9.

Although we cannot discuss the quality of clusters now,

Table 7: ID 1277 with $\theta = 0.00022$ and $[\alpha, \beta] = [0, 5]$.

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. 9: The logarithmic frequency of patterns by the GAPMiner.

through experiments, we confirmed that the skeletonization affected the distribution of the frequency of patterns. Compared with the result without the skeletonization in Fig. 9(a), Fig. 9(b) shows our method made many patterns have similar support counts. We conjectured that such effects are important to provide a way for the exploratory data analysis by pattern mining.

6. Conclusion

We provide a new skeletonization method for dealing with the periodicity of sequential patterns. Our experiments 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. 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).

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] Han, J., Dong, G. and Yin, Y.: Efficient mining of partial periodic patterns in time series database, *Proc. of 15th ICDE*, pp. 106–115 (1999).

[4] Han, J., Gong, W. and Yin, Y.: Mining segment-wise periodic patterns in time-related databases, *Proc of. 4th KDD*, pp. 214–218 (1998).

[5] Liu, C., Zhang, K., Xiong, H., Jiang, G. and Yang, Q.: Temporal skeletonization on sequential data: patterns, categorization, and visualization, *Proc. of the 20th KDD*, pp. 1336–1345 (2014).

[6] 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).

[7] 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).

[8] Shi, J. and Malik, J.: Normalized cuts and image segmentation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 888–905 (1997).

[9] Von Luxburg, U.: A tutorial on spectral clustering, *Statistics and Computing*, Vol. 17, No. 4, pp. 395–416 (2007).

[10] 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).

[11] Yuan, J., Zheng, Y., Xie, X. and Sun, G.: Driving with knowledge from the physical world, *Proc. of the 17th KDD*, ACM, pp. 316–324 (2011).

[12] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G. and Huang, Y.: T-drive: driving directions based on taxi trajectories, *Proc. of the 18th SIGSPATIAL GIS*, pp. 99–108 (2010).

[13] Zhang, M., Kao, B., Cheung, D. W. and Yip, K. Y.: Mining periodic patterns with gap requirement from sequences, *ACM Transaction on Knowledge Discovery from Data*, Vol. 1, No. 2 (2007).