Discovering Periodic Patterns Common to Multiple Sequences

Philippe Fournier-Viger[1(✉)], Zhitian Li[2], Jerry Chun-Wei Lin[3], Rage Uday Kiran[4], and Hamido Fujita[5]

[1] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen, China
philfv8@yahoo.com
[2] School of Computer Sciences and Technology, Harbin Institute of Technology, Shenzhen, China
tymonlee1@163.com
[3] Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway
jerrylin@ieee.org
[4] Institute of Industrial Science University of Tokyo, Tokyo, Japan
uday.rage@gmail.com
[5] Faculty of Software and Information Science, Iwate Prefectural University, Takizawa, Iwate, Japan
hfujita-799@acm.org

Abstract. Discovering periodic itemsets in transaction databases is an emerging data mining task. However, current algorithms are designed to discover periodic itemsets in a single sequence. But in real-life, it is desirable to find periodic patterns that are common to multiple sequences. For example, a retail store manager can benefit from finding that many customers buy the same products every week in a retail store, to adapt its marketing and sale strategies. To address this drawback of previous work, this paper defines the problem of mining periodic patterns common to multiple sequences and proposes an efficient algorithm named MPFPS, which relies on a novel PFPS-list structure and two novel periodicity measures to assess periodicity with more flexibility. Experiments on several synthetic and real-life databases show that MPFPS is efficient and can filter many non-periodic itemsets to reveal the desired patterns.

Keywords: Frequent pattern · Periodic pattern · Sequence database

1 Introduction

Frequent pattern mining consists of discovering patterns that appear frequently in a database. It plays an important role in data mining and has numerous applications [3,10]. Several work in frequent pattern mining have focused on discovering patterns such as frequent itemsets and association rules that do not

C. Ordonez and L. Bellatreche (Eds.): DaWaK 2018, LNCS 11031, pp. 231–246, 2018.
https://doi.org/10.1007/978-3-319-98539-8_18

consider the sequential ordering of data. But for several applications, considering the sequential ordering is meaningful and can reveal interesting information. For example, when analyzing customer behavior, it can be discovered that some actions are repeated by several customers over time. Discovering such information can be used to design effective marketing and sales strategies.

To discover patterns in data while considering the sequential ordering, itemset mining has been generalized as the problem of sequential pattern mining. It consists of discovering frequent subsequences in a set of sequences [1,14]. Although sequential pattern mining has numerous applications and several algorithms have been designed, an important limitation is that they are inappropriate to discover recurring patterns. But recurring patterns are found in many domains [6,15]. For example, one may detect recurring customer behavior such that some customers buy some products every day, week, or month.

To discover recurring patterns, also called periodic patterns, several algorithms have been proposed in recent years [6,8,9,11–13]. However, most algorithms for discovering periodic patterns are designed to find patterns in a single sequence. But periodic patterns also commonly appear in sequence databases (a set of multiple sequences). For example, it is desirable to discover periodic behavior of not just one but common to several customers. To our best knowledge, only one algorithm, named PHUSPM, was proposed to mine periodic patterns in a sequence database [16]. However, this algorithm simply applies the same periodicity measures as algorithms for discovering patterns in a single sequence. As a result, PHUSPM can find patterns that regularly appear in a sequence database, but it does not consider whether these patterns are periodic in each sequence. But finding patterns that are periodic in many sequences is useful. For example, consider sequences of customer transactions in a retail store. The PHUSPM algorithm could find that bread and milk are periodically sold by the store (appear periodically in the database) but would fail to find that many customers periodically buy bread and milk (bread and milk are periodic in multiple sequences, each corresponding to a customer). Finding such information is useful for designing effective sale and marketing strategies.

To address the above limitations of previous work, this paper proposes the task of mining Periodic Frequent Patterns common to multiple Sequences (PFPS), which considers the periodicity of patterns in each sequence and their frequency in the overall database. Moreover, an efficient algorithm is presented to mine PFPS. The contributions of this paper are as follows:

- The problem of mining periodic frequent patterns common to multiple sequences is defined, and its properties are studied.
- To evaluate the periodicity of patterns in each sequence, a new measure is defined, which is the standard deviation of periods, to find patterns that occur with regularity. Moreover, a novel periodicity measure named *Sequence Periodic Ratio* (SPR) is defined to find patterns that are periodic in multiple sequences. To effectively reduce the search space, an upper-bound on the SPR called *boundRa* is developed and a novel pruning property is proposed.

- An algorithm named MPFPS is presented to efficiently find all periodic frequent patterns common to multiple sequences, which relies on a novel PFPS-list structure to avoid repeatedly scanning the database.
- An experimental evaluation on several real and synthetic datasets reveal that the proposed algorithm is efficient and can filter many non periodic patterns.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem of periodic frequent pattern mining. Section 4 presents the proposed algorithm. Section 5 describes the experimental evaluation. Finally, Sect. 6 draws the conclusion and discusses future work.

2 Related Work

Several Frequent Itemset Mining (FIM) algorithms have been proposed to discover frequent itemsets, i.e. sets of items that appear frequently in a customer transaction database [2]. One of the most influential, named Apriori explores the search space of itemsets using a breadth-first search. It combines pairs of itemsets to generate larger candidate itemsets, and then scans the database to calculate the support of itemsets and eliminate infrequent itemsets. However, repeated database scans result in poor performance for large databases. The Eclat [4] algorithm addresses this issue by creating a structure called *id-list* for each considered itemset, which can be built from other id-lists to avoid performing database scans. Eclat performs a depth-first search and divides the search space into equivalence classes [4]. To consider the sequential ordering between transactions, FIM algorithms have been extended for the task of Sequential Pattern Mining (SPM), which consists of discover subsequences appearing in many sequences. Some representative SPM algorithms are AprioriAll [1], and PrefixSpan [5]. The former extends the Apriori algorithm.

Recently, FIM has been extended to discover periodic patterns in a single sequence. A frequent itemset is said to be periodic in a sequence if it appears multiple times and the time between each occurrence is less than a user-defined maximum periodic threshold. More efficient algorithms have been designed, and variation of the problem of mining periodic patterns in a single sequences have been proposed [11]. For example, a study [11] addressed the "rare item problem" by proposing to define a minimum support threshold for each item rather than using the same threshold for all items. In another study, an algorithm named $MTKPP$ was proposed to discover the k periodic patterns that are the most frequent in a sequence [13]. However, a drawback of all these studies is that the maximum periodicity constraint is very strict. If a pattern appears regularly in a database but appear a single time with a time interval larger than the maximum periodicity threshold, it is discarded. Thus more flexible measures are needed [9]. Although much work has been done to find periodic patterns in a sequence, to the best of our best knowledge, only one algorithm named PHUSPM [16] considers discovering patterns in multiple sequences. But to do that, it considers multiple sequences as a sequence, and then simply applies the same periodicity measures designed for a single sequence. As a result, the algorithm does not consider

whether a pattern is periodic in each sequence. Thus, PHUSPM is unable to find patterns such that several customers periodically buy some products every week. This paper addresses this issue by proposing a more general model and algorithm for mining periodic patterns that are periodic in multiple sequences.

3 Definitions and Problem Statement

This section is divided into two parts. It first presents the problem of discovering periodic patterns in a single sequence, and proposes a novel measure called the periodic standard deviation to filter non interesting periodic patterns. Then, the problem of periodic pattern mining is generalized to mine patterns in a sequence database (multiple sequences).

Table 1. An example sequence database

Sequence_id	Sequence
1	$\langle(a,b,e),(a,b,e),(a,d),(a,e),(a,b,c)\rangle$
2	$\langle(c),(a,b,c,e),(c,d),(a,b,c,e),(a,b,d)\rangle$
3	$\langle(b,c),(a,b),(a,c,d),(a,c),(a,b)\rangle$
4	$\langle(a,b,d,e),(a,b,e),(a,b,c),(a,b,d,e),(a,b)\rangle$

Let there be a set of items I representing all the symbols in a database. An itemset X is a subset of I, that is $X \subseteq I$. An itemset containing k items is said to be a k-itemset. A sequence s is an ordered list of itemsets $s = \langle T_1, T_2, \ldots T_m\rangle$, where $T_j \subseteq I$ $(1 \leq j \leq m)$, j is the transaction identifier of T_j, and T_j is said to be a transaction. A sequence database D is an ordered set of n sequences, denoted as $D = \langle s_1, s_2, ..., s_n\rangle$. The sequence s_i in D is said to be the i-th sequence of D, and its sequence identifier is said to be i. A sequence $s_a = \langle A_1, A_2, ..., A_k\rangle$ is said to be a subsequence of a sequence $s_b = \langle B_1, B_2, ..., B_l\rangle$ iff there exist integers $1 \leq i1 < i2 < ... < ik \leq m$ such that $A_1 \subseteq B_{i1}, A_2 \subseteq B_{i2}, ..., A_k \subseteq B_{il}$ (denoted as $s_a \sqsubseteq s_b$). For example, consider the database of Table 1, containing four sequences, which will be used as running example. The first sequence contains five itemsets. The first itemset contains three items (a, b and c). It is thus a 3-itemset. The sequence $\langle(a,b),(a)\rangle$ is a subsequence of s_1. In previous work, to find periodic patterns in a single sequence, the concept of periods was introduced, which is defined as follows [9].

Definition 1 ***(periods of an itemset in a sequence).*** *Consider a sequence s_i of a database D and an itemset X. Let $TR(X, s_i) = \langle T_{g_1}, T_{g_2}, ..., T_{g_k}\rangle \sqsubseteq s_i$ be the ordered set of transactions in which itemset X occurs in sequence s_i. Two transactions T_x and T_y in s_i are said to be consecutive with respect to X if there does not exist a transaction $T_z \in s_i$ such that $x < z < y$ and $X \subseteq T_z$. The period of two consecutive transactions T_x and T_y for a pattern X is $per(T_x, T_y) = y - x$.*

The periods of X in a sequence s_i are $pr(X, s_i) = \{per_1, per_2, ..., per_{k+1}\}$ where $per_1 = g_1 - g_0$, $per_2 = g_2 - g_1$, $\ldots per_{k+1} = g_{k+1} - g_k$, where g_1, g_2, ..., g_k are the numbers of the transactions itemset X occurs and g_0 and g_{k+1} are defined as $g_0 = 0$ and $g_{k+1} = n$, respectively.

For example, the itemset $\{a, b\}$ occurs in transactions T_1, T_2 and T_5 of sequence s_1. Thus $TR(\{a, b\}, s_1) = \{T_1, T_2, T_5\}$ and the periods of pattern $\{a, b\}$ are $pr(\{a, b\}, s_1) = \{1, 1, 3, 0\}$.

The most widely used measure to assess the periodicity of a pattern in a single sequence is the maximum periodicity [9]:

Definition 2 ***(maximum periodicity).*** *The maximum periodicity of an itemset X in a sequence s is defined as $maxPr(X, s) = argmax(pr(X, s))$.*

In previous work, a pattern is deemed periodic if its maximum period is smaller than a user-defined $maxPr$ threshold, and its support is no less than a threshold $minSup$. The support of an itemset X in a sequence s is the number of transactions containing X in s, that is $sup(X, s) = |TR(X, s)|$. For example, consider $maxPr = 3$ and $minSup = 3$. The itemset $\{a, b\}$ is periodic in sequence s_1 since its periods in that sequence are $pr(\{a, b\}, s_1) = \{1, 1, 3, 0\}$, its maximum period is $maxPr(\{a, b\}, s_1) = max\{1, 1, 3, 0\} = 3 \leq maxPr$ and $sup(\{a, b\}, s_1) = 3 \geq minSup$. However, a problem of $maxPr$ measure is that if it is set to a small value, patterns may be discarded if they only have a few periods greater than $maxPr$, while if $maxPr$ is set to a large value, patterns having periods that vary greatly may be considered as periodic. To address this problem, this paper proposes to consider the standard deviation of periods.

Definition 3 ***(standard deviation of periods).*** *The standard deviation of the periods of an itemset X in a sequence s is denoted as $stanDev(X, s)$.*

For example, the itemset $\{a, b\}$ has four periods in sequence s_1, that is $pr(X) = \{1, 1, 3, 0\}$. The average period of $\{a, b\}$is: $avgPr(\{a, b\}, s_1) = (1 + 1 + 3 + 0)/4 = 1.25$. The standard deviation is calculated as: $stanDev(\{a, b\}, s_1) = \sqrt{[(1 - 1.25)^2 + (1 - 1.25)^2 + (3 - 1.25)^2 + (0 - 1.25)^2]/4}$.

Based on this definition, we consider than an itemset X is periodic in a sequence s if it meets the following conditions.

Definition 4 ***(periodic pattern in a sequence).*** *Let there be three user-specified thresholds $maxPer$, $minSup$ and $maxStd$. An itemset X is periodic in a sequence s if $maxPr(X, s) \leq maxPr$, $sup(X, s) \geq minSup$ and $stanDev(X, s) \leq maxStd$.*

For example, the itemset $\{a, b\}$ is periodic in sequence s_1 for $maxStd = 2.0$, since $stanDev(\{a, b\}, s_1) = 1.09$. The above definition is proposed to identify periodic patterns in a single sequence. The following paragraphs explain how it is extended to discover periodic frequent patterns common to multiple sequences using a novel measure called the sequence periodic ratio.

Definition 5 ***(sequence periodic ratio).*** *The number of sequences where an itemset X is periodic in a sequence database D is denoted as $numSeq(X)$. The sequence periodic ratio of X in D is defined as $ra(X) = numSeq(X)/|D|$, where $|D|$ is the number of sequences in D.*

For instance, the number of sequences where $\{a, b\}$ is periodic is $numSeq(\{a, b\}) = 3$ (it is periodic in s_1, s_2, and s_4). The total number of sequences is $|D| = 4$. Thus, the sequence periodic ratio of $\{a, b\}$ is $ra(\{a, b\}) = 3/4 = 0.75$.

Problem Statement. Let there be a sequence database D, and four user-defined thresholds, namely the minimum support threshold $minSup$, maximum periodicity threshold $maxPr$, maximum standard deviation threshold $maxStd$, and minimum sequence periodic ratio threshold $minRa$. An itemset X is a Periodic Frequent Pattern (PFPS) in D if $ra(X) \geq minRa$. The problem of mining periodic patterns common to multiple sequences is to find all PFPS.

For example, Table 2 shows the PFPS found for different thresholds values. The first line uses $minSup = 2$, $maxPr = 3$, $maxStd = 5.0$ and $minRa = 0.3$, while the following lines change one parameter with respect to the first line (in bold). It can be seen that each parameter is useful to reduce the number of patterns, and thus provide a lot of flexibility to the user to select the desired patterns.

Also, we introduce a new measure called *boundRa* that is an upper-bound on the *ra* measure to be able to reduce the search space.

Table 2. Patterns found for different threshold values

No.	$minSup$	$maxPr$	$maxStd$	$minRa$	Patterns found
1	2	3	1.0	0.6	$\{a\}, \{e\}, \{a, e\}$
2	**3**	3	1.0	0.6	$\{a\}$
3	2	**1**	1.0	0.6	$\{a\}$
4	2	3	**1.5**	0.6	$\{a\}, \{b\}, \{e\}, \{a, b\}, \{a, e\},$
5	2	3	1.0	**0.4**	$\{a\}, \{b\}, \{c\}, \{e\}, \{a, b\}, \{a, e\}, \{b, e\}, \{a, b, e\}$

Definition 6 ***(boundRa).*** *Given two user-specified thresholds $maxPer$ and $minSup$, an itemset X is a candidate in a sequence s if $maxPr(X, s) \leq maxPr$ and $sup(X, s) \geq minSup$. The number of sequences where an itemset X is a candidate in a sequence database D is denoted as $numCand(X)$. The boundRa of X in D is defined as $boundRa(X) = numCand(X)/|D|$.*

Property 1 **(pruning property).** For two itemsets $X \subseteq X'$, (1) $boundRa(X) \geq ra(X)$ and (2) $boundRa(X) \geq boundRa(X')$.

Proof. It can be easily seen that the relationship (1) holds since the definition of $ra(X)$ and $boundRa(X)$ are the same except that the former adds one more constraint. The relationship (2) is proven as follows. In a sequence s, it was shown

that $maxPr(X, s) \leq maxPr(X', s)$ [9], and that $sup(X, s) \leq sup(X', s)$ [2]. Thus, the number of sequences where X' is a candidate cannot be greater than the number of sequences where X is a candidate, i.e. $numCand(X') \leq numCand(X)$. Hence, $boundRa(X) \geq boundRa(X')$. □

4 The MPFPS Algorithm

This section introduces an efficient algorithm to mine PFPS, named MPFPS (Mining Periodic Frequent Pattern common to Sequences). MPFPS explores the search space of itemsets using a depth-first search. The search space consists of $2^{|I|}$ itemsets. MPFPS starts from single items, and recursively appends an item to each itemset to generate a larger itemset, following the $\succ$ order. MPFPS reduces the search space by exploiting the fact that *boundRa* is an upper-bound on the *ra* measure and satisfies the *downward closure property* (Property 1). To calculate all the measures to evaluate patterns without having to repeatedly scan the database, MPFPS utilizes a novel data structure called PFPS-list. A PFPS-list is created for each itemset that is visited in the search space. The PFPS-list of an itemset X contains three fields. The *i-set* field stores X. The *tidlist-list* field stores the list of identifiers of transactions containing X. The *sid-list* field contains the list of identifiers of sequences containing X. For example, the PFPS-list of the itemset $\{a\}$ is: *i-set*:{a}, *tidlist-list*: [{0,1,2,3,4}, {1,3,4},{1,2,3,4},{0,1,2,3,4}], *sid-list*:{0,1,2,3}. The PFPS-list of an itemset allows to quickly find the transactions and sequences where it appears, and thus to determine if the itemset is periodic without scanning the database. Moreover, as it will be explained, the PFPS-list of any itemset containing more than one item can be obtained by performing an intersection operation on the PFPS-lists of two of its subsets (without scanning the database).

The proposed MPFPS (Algorithm 1) takes as input a database with multiple sequences and the *maxStd*, *minRa*, *maxPr*, and *minSup* thresholds. MPFPS outputs all the PFPS. It first scan the database once to calculate $sup(\{i\}, s)$, $pr(\{i\}, s)$, $maxpr(\{i\}, s)$ and $stanDev(\{i\}, s)$ for each item i and sequence s (line 1). Then, MPFPS checks if each item i is periodic in each sequence of the database (line 2 to 4). For an item i appearing in a sequence s, if $sup(\{i\}, s) \geq minSup$, $maxpr(\{i\}, s) \leq maxPr$ and $stanDev(\{i\}, s) < maxStd$, then i is said to be periodic in that sequence. Then, the algorithm calculates the sequence periodic ratio of item i by dividing the number of sequences where i is periodic by the total number of sequences. If this value is not less than *minRa*, i is a PFPS (line 4). Also, *boundRa* of $\{i\}$ is calculated to prune the search space (line 5 and 6) and the PFPS-list of each single item i such that $boundRa(\{i\}) \geq minRa$ is stored in a set *boundPFPSSingle* (line 8), which is sorted according to the $\succ$ order of ascending support. Then, the depth-first search of PFPS starts by calling the recursive procedure *Search* with *boundPFPSSingle*, *minSup*, *maxPr*, *maxStd*, *minRa* and D.

The *Search* procedure (Algorithm 2) takes as input PFPS-lists of extensions of an itemset P, the *minSup*, *maxPr*, *maxStd*, *minRa* thresholds and

Algorithm 1. The MPFPS algorithm

input : D: a database with multiple sequences, $maxStd, minRa, maxPr, minSup$: the thresholds.
output: the set of periodic frequent patterns (PFPS).

1 Scan each sequence $s \in D$ to calculate $sup(\{i\}, s)$, $pr(\{i\}, s)$, $maxpr(\{i\}, s)$ and $stanDev(\{i\}, s)$ for each item $i \in I$;
2 **foreach** *item* $i \in I$ **do**
3 $\quad numSeq(\{i\}) \leftarrow |\{s | maxpr(\{i\}, s) \leq maxPr \wedge stanDev(\{i\}, s) \leq maxStd \wedge sup(\{i\}, s) \geq minSup \wedge s \in D\}|$;
4 $\quad ra(\{i\}) \leftarrow numSeq(\{i\})/|D|$;
5 $\quad numCand(\{i\}) \leftarrow |\{s | maxpr(\{i\}, s) \leq maxPr \wedge sup(\{i\}, s) \geq minSup \wedge s \in D\}|$;
6 $\quad boundRa(\{i\}) \leftarrow numCand(\{i\})/|D|$;
7 **end**
8 $boundPFPSSingle \leftarrow$ {PFPS-list of item $i | i \in I \wedge boundRa(\{i\}) \geq minRa$};
9 Sort $boundPFPSSingle$ by the order $\succ$ of ascending support values;
10 `Search` ($boundPFPSSingle$, $minSup$, $maxPr$, $maxStd$, $minRa$, D);

the database. An *extension* of an itemset P is an itemset that is obtained by appending an item z to P, and is denoted as Pz. When the procedure is first called, P is the empty set, and extensions of P are single items. The *Search* procedure performs a loop for each extension Px of P. The procedure first calculates $numCand(Px)$ and $boundRa(Px)$ using the PFPS-list of Px, which is denoted as LPx (line 2 to 3). Then, if $boundRa(Px) \geq minRa$, $ra(Px)$ is calculated to check if Px is a PFPS, and if yes, Px is output (line 5 to 7). Then, a loop is performed to combine Px with each extension Py of P such that $y \succ x$, to generate an extension Pxy containing $|Px|+1$ items (line 8 to 11). The PFPS-list of each such extension Pxy, denoted as $LPxy$ is created without scanning the database by applying the *Intersect* procedure (Algorithm 3) on the PFPS-lists of Px and Py. Then, the *Search* procedure is recursively called with the PFPS-lists of PFPS that extend Px to explore their transitive extensions (line 13).

Algorithm 2. The *Search* procedure

input : $ExtensionsOfP$: a set of PFPS-lists of extensions of an itemset P, $minSup, maxPr, maxStd, minRa$: the thresholds, D: the database.
output: the set of periodic frequent patterns that extend P.

1 **foreach** *PFPS-list* $LPx \in ExtensionsOfP$ *and* $Px = LPx.i\text{-}set$ **do**
2 $\quad numCand(Px) \leftarrow |\{s | maxpr(Px, s) \leq maxPr \wedge sup(Px, s) \geq minSup \wedge s \in D\}|$;
3 $\quad boundRa(Px) \leftarrow numCand(Px)/|D|$;
4 $\quad$ **if** $boundRa(Px) \geq minRa$ **then**
5 $\quad\quad numSeq(Px) \leftarrow |\{s | maxpr(Px, s) \leq maxPr \wedge stanDev(Px, s) \leq maxStd \wedge sup(Px, s) \geq minSup \wedge s \in LPx.sid\text{-}list\}|$;
6 $\quad\quad ra(Px) \leftarrow numSeq(Px)/|D|$;
7 $\quad\quad$ **if** $ra(Px) \geq minRa$ **then** output Px ;
8 $\quad\quad$ **foreach** *PFPS-list* $LPy \in ExtensionsOfP$ *and* $Py = LPy.i\text{-}set$ *such that* $y \succ x$ **do**
9 $\quad\quad\quad LPxy \leftarrow$ `Intersect` (LPx, LPy);
10 $\quad\quad\quad ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{LPxy\}$;
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 $\quad$ `Search` ($ExtensionsOfPx$, $minSup$, $maxPr$, $maxStd$, $minRa$, D);
14 **end**

The *Intersect* procedure takes as input the PFPS-lists of two itemsets Px and Py, denoted as LPx and LPy, and outputs the PFPS-list of the itemset

Pxy. The algorithm first initialize an empty PFPS-list $LPxy$ for Pxy (line 1). Then, the algorithm performs a loop to consider each sequence that appears in both the PFPS-lists of Px and Py. For each such sequence, let si be the sequence identifier representing that sequence. That sequence identifier is used to retrieve the lists of identifiers of transactions containing Px and Py in that sequence, denoted as $tidListSiPx$ and $tidListSiPy$, respectively (line 3 and 4). These two lists are then intersected to obtain the list of identifiers of transactions containing Pxy in that sequence, named $tidListSiPxy$ (line 5). If that latter list is not empty, si is added to the PFPS-list of Pxy as well as the list of transactions $tidListSiPxy$. The procedure returns the PFPS-list of Pxy, which is obtained without scanning the database.

Algorithm 3. The Intersect procedure

input : LPx and LPy: the PFPS-lists of two extensions Px and Py of an itemset
output: the PFPS-list $LPxy$ of itemset Pxy

1 $LPxy.i\text{-}set \leftarrow Px \cup \{y\}$; $LPxy.tidlist\text{-}list \leftarrow \emptyset$; $LPxy.sid\text{-}list \leftarrow \emptyset$;
2 **foreach** *sequence identifier* $si \in LPx.sid\text{-}list$ *such that* $si \in LPy.sid\text{-}list$ **do**
3 $tidListSiPx \leftarrow$ the tid list of si in $LPx.tidlist\text{-}list$;
4 $tidListSiPy \leftarrow$ the tid list of si in $LPy.tidlist\text{-}list$;
5 $tidListSiPxy \leftarrow tidListSiPx \cap tidListSiPy$;
6 **if** $tidListSiPxy \neq \emptyset$ **then**
7 $LPxy.sid\text{-}list.append(si)$; $LPxy.tidlist\text{-}list.append(tidListSiPxy)$;
8 **end**
9 **end**
10 **return** $LPxy$;

4.1 A Detailed Example

Consider the example database of Table 1 and that $minSup = 2, maxPr = 3$, $maxStd = 1.0$ and $minRa = 0.6$. The main procedure of MPFPS (Algorithm 1) first processes single items. Consider the item a. By scanning the database, it finds that $pr(\{a\}, s_1)$ = [1,1,1,1,1,0], $pr(\{a\}, s_2)$ = [2,2,1,0], $pr(\{a\}, s_3)$ = [2,1,1,1,0], $pr(\{a\}, s_4)$ = [1,1,1,1,1,0], $maxpr(\{a\}, s_1) = 1$, $maxpr(\{a\}, s_2) = 2$, $maxpr(\{a\}, s_3) = 2$, $maxpr(\{a\}, s_4) = 1$, $stanDev(\{a\}, s_1) = 0.152$ $stanDev(\{a\}, s_2) = 0.256$, $stanDev(\{a\}, s_3) = 0.632$, and $stanDev(\{a\}, s_4) = 0.152$. As $maxpr(\{a\}, s_1) < maxPr$ and $stanDev(\{a\}, s_1) < maxStd$, item $\{a\}$ is periodic in sequence s_1. Similarly, it is also periodic in sequences s_2, s_3 and s_4, and (a) is periodic in $numSeq(\{a\}) = 4$ sequences and the ratio $ra(\{a\})$=1$\geq minRa$. Thus, {a} is a PFPS. Then, the same process is repeated for the other items and it is found that the items $\{a\}$ and $\{e\}$ are PFPS. The PFPS-lists of these items are built, and the *Search* procedure is called to find larger PFPS (Algorithm 2). Since $\{a\}$ is a PFPS and $boundRa(\{a\}) \geq minRa$, the *Search* procedure outputs a. Then, the loop is continued with the PFPS-lists of extensions of $\emptyset$, that is the PFPS-list of $\{e\}$. The Intersect procedure is applied on the PFPS-lists of $\{a\}$ and $\{e\}$ to generate the PFPS-list of $\{a, e\}$. Then, the sequences in which $\{a\}$ and $\{e\}$ are both periodic and candidates are found. The *sid-list* of $\{a\}$ is $\{0, 1, 2, 3\}$ and that of $\{e\}$ is *sid-list* is $\{0, 1, 3\}$. Thus, they are both periodic in the first, second and fourth sequences. The *sid-list* of

$\{a, e\}$ is $\{0, 1, 3\}$. For the first sequence, $\{a\}$'s tidlist is $\{0, 1, 2, 3, 4\}$, $\{e\}$'s tidlist is $\{0, 1, 3\}$, and their intersection is $\{0, 1, 3\}$. Hence, itemset $\{a, e\}$ is periodic in this sequence. The sequence identifier 0 is added to the *sid-list* of $\{a, e\}$ and the intersection $\{0, 1, 3\}$ to the *tidlist-list* of $\{a, e\}$. The same process is repeated for the two other sequences, and the PFPS-list of $\{a, e\}$ is returned to the *Search* procedure, which adds it to the PFPS-list extensions of $\{a\}$ and it recursively calls itself until all PFPS are found.

5 Experimental Evaluation

In prior work, a single algorithm named PHUSPM was proposed to mine periodic patterns in a sequence database. However, it finds patterns that regularly appear in a database whereas the proposed MPFPS algorithm finds patterns that are periodic in many sequences. Thus, the performance of these algorithms cannot be compared. For this reason, this experimental evaluation compares the performance of MPFPS for several parameter values with a baseline version of MPFPS designed to find all frequent patterns. MPFPS is implemented in Java, and was run on a Windows 10 computer equipped with a 3.60 GHz Xeon E3 CPU with 64 GB of memory. The experiment is carried out on two real databases (*FIFA*, *Bible*) obtained from the website of the SPMF library [7], and on a synthetic sequence database created using the SPMF generator. These databases were chosen because they have different characteristics. *FIFA* contains 2,990 distinct items and 20,450 sequences, with an average of 34.7 items per sequence. *Bible* contains 13,905 distinct items and 36,369 sequences, with an average of 21.6 items per sequence. In *FIFA* and *Bible*, each transaction contains one item per transaction. For this reason, the synthetic $T15I1KD300K$ database was also used, where each transaction has at most 10 items per transactions. This large sparse sequence database contains 300,000 transactions and 1,000 distinct items, and 10 items on average per sequence. The source code of MPFPS and datasets can be obtained from http://www.philippe-fournier-viger.com/spmf/.

MPFPS has four parameters: *maxStd*, *minRa*, *maxPr* and *minSup*. The latter is used to find frequent patterns in each sequence and has been found to have little influence on the number of periodic patterns. Hence, *minSup* is set to a fixed value in the experiment ($minsup = 2$). Moreover, when *maxStd* and *maxPr* are set to large values and $minRa = 0$, the algorithm finds all frequent patterns. Thus, in the following, the MPFPS algorithm with $maxStd = 1000$, $minRa = 0$ and $maxPr = 1000$ will be used as baseline, and will be denoted as $MPFPS\text{-}baseline(0, 1000)$. The baseline will be compared with other parameter settings to assess the influence of the *maxStd*, *maxPr* and *minRa* parameters on the performance of the algorithm and number of patterns found. In the following, $MPFPS(x, y)$ denotes MPFPS with $minRa = x$, $maxPr = y$ and $minSup = 2$.

5.1 Influence of *minRa* and *maxStd*

The *minRa* and *maxStd* parameters were first varied to evaluate their joint influence, while the *maxPr* parameter was set to a fixed value ($maxPr = 10$

for $FIFA$ and $Bible$, and $maxPr = 20$ for $T15I1KD300K$). Figure 1 shows the runtimes and number of PFPS found for various values of the $maxStd$ and $minRa$ parameters.

In general, it can be observed that mining PFPS in the two real databases $FIFA$ and $Bible$ can be up to 20% faster than mining all frequent patterns using the baseline algorithm, whereas for $T15I1KD300K$, it can be 30 times faster. The reason for this performance difference among databases is that there is only one item per transaction in $FIFA$ and $Bible$, and thus the search space of itemsets is small, whereas transactions of $T15I1KD300K$ contains many items per transactions, which results in a very large search space. Because a large number of frequent patterns in $T15I1KD300K$ are non periodic, pruning non periodic patterns leads to a massive performance improvement.

For the $maxStd$ parameter, it is observed that the number of periodic patterns can become very small compared to the baseline algorithm when this parameter is varied. For example, in $T15I1KD300K$ there are 89,081 frequent patterns but only 20,986 PFPS when $maxStd$ is decreased from 1000 to 10, and only 1003 PFPS when $maxStd$ is decreased from 10 to 5.

For the $minRa$ parameter, it is observed that it also greatly influences the number of PFPS. For $Bible$ and $maxStd = 1000$, when $minRa$ is varied from 0 to 0.001 (which means there should be at least 0.1% sequences in which an itemset is periodic), the number of PFPS decreases from 1126 to 63. This shows that most patterns are periodic in few sequences, and that the proposed algorithm can filter many non periodic patterns.

5.2 Influence of $maxPr$

The $maxPr$ parameter was also varied to evaluate its influence. Since the average number of transaction per sequence in the datasets is not greater than 35, this parameter was set to small values, and also to $maxPr = 1000$ as a baseline, indicating that the parameter is deactivated. Figure 2 compares the runtimes and number of PFPS found for various values of $maxPr$, when $maxStd$ is varied as in the previous subsection.

It is first observed that the $maxPr$ parameter can be used to greatly reduce the number of patterns found. For example, on the $FIFA$ dataset for

Table 3. Memory used by MPFPS for different parameter values on $T15I1KD300K$

Memory(MB) $maxStd$ / MPFPS(x,y)	3	4	5	10	15	20	1000
MPFPS-baseline(0,1000)	1837	1995	2035	2452	2668	2613	3309
MPFPS(0,20)	1837	1995	2095	2272	2549	2604	2773
MPFPS(0.0001,20)	1797	1876	2035	2233	2550	2550	2589
MPFPS(0.001,20)	1797	1797	1797	2233	2589	2589	2589
MPFPS(0,10)	1797	1837	1837	1837	1837	1837	1837
MPFPS(0.0001,10)	1797	1797	1797	1797	1797	1797	1797
MPFPS(0.001,10)	1797	1797	1797	1797	1797	1797	1797

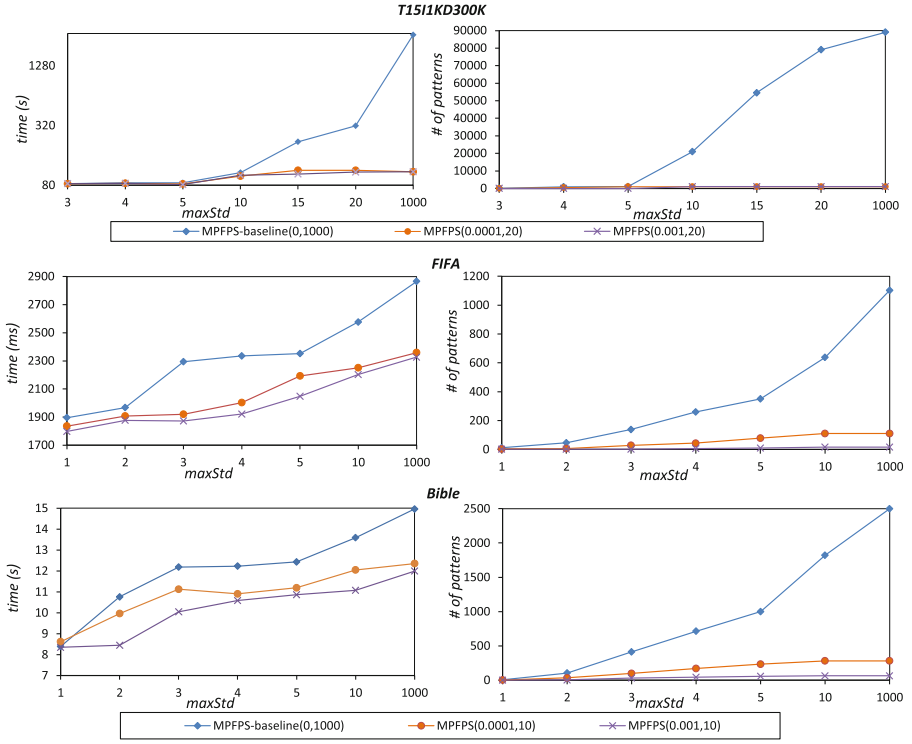

Fig. 1. Runtimes and number of patterns for different $minRa$ and $maxStd$ values

$maxStd = 1000$, $minRa = 0$ and $maxPr = 5$, 134 patterns are found, while 1000 patterns are found for $maxPr = 1000$. This is reasonable since the condition that the periods of a pattern must all be less than $maxPr$ is very strict.

In terms of runtime, decreasing the $maxPr$ parameter improves performance but not by a large amount on the real $FIFA$ and $Bible$ datasets, while it provides a considerable improvement on the synthetic dataset. The reason is that the search space is larger in that dataset and pruning properties are more effective as few periodic patterns are expected to appear in this database since the data is synthetic.

5.3 Memory Used for Different Parameter Values

Tables 3, 4 and 5 compares the memory used for various values of $minRa$ and $maxPr$, when $maxStd$ is varied as in the previous subsection for the three databases. Due to the space limitation, for the $FIFA$ and $Bible$ databases, only three lines of results are shown. The ommitted lines are almost the same (around 320 MB).

It can firstly be observed that, when $maxStd$ is increased, MPFPS needs more space to store the PFPS-lists of the itemsets. For example, when $maxStd$

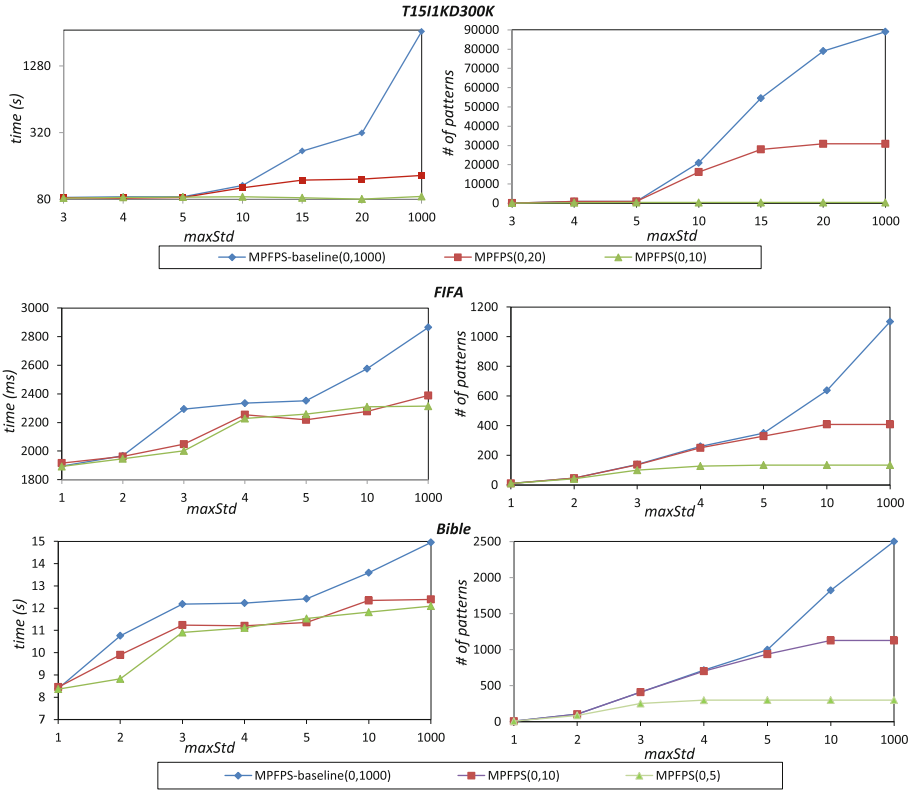

Fig. 2. Runtimes and number of patterns for different $maxPr$ and $maxStd$ values

is increased from 3 to 15 for MPFPS(0.001, 20) in Table 3, memory usage also increases from 1797 MB to 2589 MB. This shows that $maxStd$ can be used to greatly reduce the number of patterns stored in memory.

For the $minRa$ parameter, it is observed that this parameter also influences memory usage. For example, for $T15I1KD300K$ when $minRa$ is increased from 0 to 0.001, $maxStd = 4$ and $maxPr = 20$, the memory decreases from 1995 MB to 1797 MB. This shows that $minRa$ is also helpful for reducing the number of stored patterns.

The $maxPr$ threshold can also reduce memory usage, as observed in the tables. For example, for $T15I1KD300K$, when $maxPr$ is decreased from 1000 to 10, $maxStd = 10$ and $minRa = 0$, the memory decreases from 2452 MB to 1837 MB.

5.4 Discussion

On overall, it has been found that the $maxPr$, $minRa$ and $maxPr$ parameters can be used to filter a large number of non periodic patterns. Thus the proposed

Table 4. Memory used by MPFPS for different parameter values on $FIFA$

Memory(MB) *maxStd* / MPFPS(x,y)	1	2	3	4	5	10	1000
MPFPS-baseline(0,1000)	280	320	320	320	320	400	525
MPFPS(0,10)	280	320	320	320	360	360	360
MPFPS(0.0001,10)	320	320	320	320	320	320	320

Table 5. Memory used by MPFPS for different parameter values on *Bible*

Memory(MB) *maxStd* / MPFPS(x,y)	1	2	3	4	5	10	1000
MPFPS-baseline(0,1000)	280	320	380	400	520	847	1386
MPFPS(0,10)	280	320	360	400	480	560	560
MPFPS(0.0001,10)	280	320	320	320	320	320	320

algorithm can be used to find a small set of periodic patterns. It was also shown that parameters can also reduce the runtime. How to set the parameters is dataset dependent as different databases may contain patterns with shorter or longer periods, or periods that vary more or less greatly. It has been found that the *minSup* parameter has a very small influence on the algorithm's output and performance (results were not shown due to space limitation). Thus, it is recommended to set the *minSup* parameter to a small value. It is recommended to set the *maxPr* parameter to a large value as the pruning condition for this parameter is very strict (as explained in the related work section). It should thus be used to filter patterns that have very large periods. Thus, the two most important parameters are the *maxStd* and *maxRa* parameters. The former allows to specify the maximum variation in terms of periodicity over time for a periodic patterns. The latter is the ratio of sequences where a pattern must be periodic, and is proposed to find periodic patterns common to multiple sequences.

6 Conclusion

Previous work on periodic pattern mining have mainly focused on analyzing patterns in a single sequence. This paper has thus proposed a novel problem of mining periodic frequent patterns common to multiple sequences, which consists of discovering all patterns respecting user-defined minimum support, maximum periodicity, maximum standard deviation and minimum sequence periodic ratio thresholds. The problem was formally defined and properties of the problem have been studied. Moreover, an efficient algorithm named MPFPS was proposed to discover these patterns, based on a novel PFPS-list structure and *boundRa* upper-bound to reduce the search space. Experiments on several databases have

shown that the proposed algorithm is effective and can greatly reduce the number of patterns found by filtering non periodic patterns.

There are several opportunities for future work. In future work, we will consider adapting the proposed model for various distributions. Furthermore, we will consider extending the model to discover more complex types of periodic patterns such as partial orders, rules and sequential patterns from sequences.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: 11th International Conference on Data Engineering, pp. 3–14. IEEE Press, Taipei (1995)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, Santiago de Chile (1994)
3. Aggarwal, C.C., Han, J.: Frequent Pattern Mining. Springer, Switzerland (2014). https://doi.org/10.1007/978-3-319-07821-2
4. Zaki, M.J., Gouda, K.: Fast vertical mining using diffsets. In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 326–335. ACM, Washington (2003)
5. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 355–359. ACM, Boston (2000)
6. Surana, A., Kiran, R.U., Reddy, P.K.: An efficient approach to mine periodic-frequent patterns in transactional databases. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 254–266. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28320-8_22
7. Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C., Tseng, V.S.: SPMF: a Java open-source pattern mining library. J. Mach. Learn. Res. **15**(1), 3389–3393 (2014)
8. Fournier-Viger, P., Lin, J.C.-W., Duong, Q.-H., Dam, T.-L.: PHM: mining periodic high-utility itemsets. In: Perner, P. (ed.) ICDM 2016. LNCS (LNAI), vol. 9728, pp. 64–79. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41561-1_6
9. Fournier-Viger, P., Lin, C.-W., Duong, Q.-H., Dam, T.-L., Sevcic, L., Uhrin, D., Voznak, M.: PFPM: discovering Periodic Frequent Patterns with Novel Periodicity Measures. In: 2nd Czech-China Scientific Conference, pp. 1–13 (2017)
10. Fournier-Viger, P., Lin, J.C.-W., Vo, B, Chi, T.T., Zhang, J., Le, H.B.: A survey of itemset mining. J. WIREs Data Mining Knowl. Discov. **7**(4) (2017)
11. Kiran, R.U., Reddy, P.K.: Mining rare periodic-frequent patterns using multiple minimum supports. In: 15th International Conference on Management of Data, pp. 7–8. IEEE, Mysore (2009)
12. Kiran, R.U., Kitsuregawa, M., Reddy, P.K.: Efficient discovery of periodic-frequent patterns in very large databases. J. Syst. Softw. **112**, 110–121 (2016)
13. Amphawan, K., Lenca, P., Surarerks, A.: Mining Top-K periodic-frequent pattern from transactional databases without support threshold. In: Papasratorn, B., Chutimaskul, W., Porkaew, K., Vanijja, V. (eds.) IAIT 2009. CCIS, vol. 55, pp. 18–29. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10392-6_3
14. Fan, Y., Ye, Y., Chen, L.: Malicious sequential pattern mining for automatic malware detection. Expert Syst. Appl. **52**, 16–25 (2016)

15. Halder, S., Samiullah, M., Lee, Y.-K.: Supergraph based periodic pattern mining in dynamic social networks. Expert Syst. Appl. **72**, 430–442 (2017)
16. Dinh, T., Huynh, V.-N., Le, B.: Mining periodic high utility sequential patterns. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) ACIIDS 2017. LNCS (LNAI), vol. 10191, pp. 545–555. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54472-4_51