PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures

Philippe Fournier-Viger[a,*], Chun-Wei Lin[b], Quang-Huy Duong[d], Thu-Lan Dam[d], Lukáš Ševčík[e], Dominik Uhrin[e], Miroslav Voznak[e]

[a]*School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China*
[b]*School of Computer Science and Technology Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China*
[c]*College of Computer Science and Electronic Engineering, Hunan University, China*
[d]*Faculty of Information Technology, Hanoi University of Industry, Vietnam*
[e]*Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, 17. listopadu 15, 708 00 Ostrava-Poruba, Czech Republic*

Abstract

Periodic pattern mining is the task of discovering sets of items that periodically appear in transactions. Several algorithms have been proposed for mining periodic patterns. Most of them discover periodic patterns based on a measure called the maximum periodicity, which measures the largest amount of time or number of transactions between two occurrences of a pattern. Typically, periodic pattern mining algorithms will discard a pattern as being non periodic if it has a single period greater than a maximal periodicity threshold, defined by the user. A major drawback of this approach is that it is not flexible, as a pattern can be discarded based on only one of its periods. In this paper, we present a solution to this issue by proposing to discover periodic patterns using three measures: the minimum periodicity, the maximum periodicity, and the average periodicity. The combination of these measures has the advantage of being more flexible. Properties of these measures are studied. Moreover, an efficient algorithm named PFPM (Periodic Frequent Pattern Miner) is proposed to discover all frequent periodic patterns using these measures. An experimental evaluation on real datasets shows that the proposed PFPM algorithm is efficient, and can filter a huge number of non periodic patterns to reveal only the desired periodic patterns.

Keywords: frequent pattern mining, periodic patterns, periodicity measures

[*]Corresponding author
Email addresses: `philfv8@yahoo.com` (Philippe Fournier-Viger), `jerrylin@ieee.org` (Chun-Wei Lin), `huydqyb@gmail.com` (Quang-Huy Duong), `lanfict@gmail.com` (Thu-Lan Dam), `lukas.sevcik@vsb.cz` (Lukáš Ševčík), `dominikuhrin@gmail.com` (Dominik Uhrin), `miroslav.voznak@vsb.cz` (Miroslav Voznak)

1. Introduction

In the field of data mining, frequent itemset mining (FIM) [1, 2, 3, 4, 5] is widely-viewed as a fundamental task for discovering knowledge in databases. Given a transaction database, it consists of discovering sets of items frequently purchased by customers. Besides market basket analysis, FIM has many applications in other fields [5]. Although numerous algorithms have been proposed for FIM [1, 2, 3, 4, 5], an inherent limitation of traditional FIM algorithms is that they are not designed to discover patterns that periodically appear in a database. Discovering periodic patterns has many applications such as to discover recurring customer purchase behavior. For example, customers of retails stores may buy some products on a periodical basis. Analyzing this kind of periodic purchase patterns is desirable to design appropriate marketing strategies.

Several algorithms have been proposed to discover periodic frequent patterns (PFP) [6, 7, 8, 9, 10, 11] in a transaction database (a sequence of transactions). Most of these algorithms discover periodic patterns based on a measure called the maximum periodicity, which measures the largest amount of time or number of transactions between two occurrences of a pattern. Typically, periodic pattern mining algorithms will discard a pattern as being non periodic if it has a single period greater than a maximal periodicity threshold, defined by the user. A major drawback of this approach is that it is not flexible, as a pattern can be discarded based on only one of its periods. In this paper, we propose a solution to this problem by discovering periodic patterns using three measures: the minimum periodicity, the maximum periodicity, and the average periodicity.

This paper has three main contributions. First, novel measures named *average periodicity* and *minimum periodicity* are proposed to assess the periodicity of patterns. Second, a fast algorithm named PFPM (Periodic Frequent Pattern Miner) is proposed to efficiently discover frequent periodic patterns using these measures. Third, we have conducted several experiments on real-life datasets to evaluate the efficiency of PFPM, and the usage of the novel periodicity measures. Experimental results show that the PFPM algorithm is efficient, and can filter a huge number of non periodic patterns to reveal only the desired periodic itemsets.

The rest of this paper is organized as follows. Section 2, 3, 4, 5, 6 and 7 respectively present preliminaries related to FIM, related work, the novel periodicity measures, the PFPM algorithm, the experimental evaluation, and the conclusion.

2. Related work

The problem of frequent itemset mining is defined as follows. Let I be a set of items (symbols). A *transaction database* is a set of transactions $D = \{T_1, T_2, ..., T_n\}$ such that for each transaction T_c, $T_c \in I$, and T_c has a unique identifier c called its Tid. For example, consider the database of Table 1, which will be used as running example. This database contains seven transactions $(T_1, T_2, \ldots, T_7)$. Transaction T_3 indicates that items a, b, c, d, and e appear in this transaction. The support of an itemset X in a database D is denoted as $s(X)$ and defined as $|\{t | t \in D \wedge X \subseteq t\}|$. In other words, $s(X) = |g(X)|$, where $g(X)$ is defined as the set of transactions containing X.

Table 1: A transaction database

TID	Transaction
T_1	$\{a, c\}$
T_2	$\{e\}$
T_3	$\{a, b, c, d, e\}$
T_4	$\{b, c, d, e\}$
T_5	$\{a, c, d\}$
T_6	$\{a, c, e\}$
T_7	$\{b, c, e\}$

Let there be any total order $\succ$ on items in I. The *extensions* of an itemset X are the itemsets that can be obtained by appending an item y to X such that $y \succ i$, $\forall i \in X$.

Definition 1 (frequent itemset mining). The *problem of frequent itemset mining* consists of discovering the frequent itemsets [1]. An itemset X is a *frequent itemset* if its support $s(X)$ is no less than a user-specified minimum support threshold *minsup* given by the user.

For example, consider that $minsup = 4$ transactions. The set of frequent itemsets is $\{a\}$, $\{a, c\}$, $\{e\}$, $\{c, e\}$, $\{c\}$, having respectively a support of 4, 4, 5, 4, and 6.

To discover frequent itemsets, various algorithms have been proposed such as Apriori [1], FP-Growth [12], LCM [3], and Eclat [4]. However, these algorithms are not designed to discover periodic patterns.

Inspired by the work on FIM, researchers have designed several algorithms to discover periodic frequent patterns (PFP) in transaction databases [6, 7, 8, 9, 10, 11]. Several applications of mining periodic frequent patterns have been reported in previous work [11]. A periodic frequent pattern is defined as follows [11].

Definition 2 (Periods of an itemset). Let there be a database $D = \{T_1, T_2, ..., T_n\}$ containing n transactions, and an itemset X. The set of transactions containing X is denoted as $g(X) = \{T_{g_1}, T_{g_2} ..., T_{g_k}\}$, where $1 \leq g_1 < g_2 < ... < g_k \leq n$. Two transactions $T_x \supset X$ and $T_y \supset X$ are said to be *consecutive with respect to* X if there does not exist a transaction $T_w \in g(X)$ such that $x < w < y$. The *period* of two consecutive transactions T_x and T_y in $g(X)$ is defined as $pe(T_x, T_y) = (y - x)$, that is the number of transactions between T_x and T_y. The *periods of an itemset* X is a list of periods defined as $ps(X) = \{g_1 - g_0, g_2 - g_1, g_3 - g_2, ...g_k - g_{k-1}, g_{k+1} - g_k\}$, where g_0 and $g_k + 1$ are constants defined as $g_0 = 0$ and $g_k + 1 = n$. Thus, $ps(X) = \bigcup_{1 \leq z \leq k+1} (g_z - g_{z-1})$.

For example, consider the itemset $\{a, c\}$. This itemset appears in transactions T_1, T_3, T_5, and T_6, and thus $g(\{a, c\}) = \{T_1, T_3, T_5, T_6\}$. The periods of this itemset are $ps(\{a, c\}) = \{1, 2, 2, 1, 1\}$.

Definition 3 (Periodic Frequent Pattern). The maximum periodicity of an itemset X is defined as $maxper(X) = max(ps(X))$ [11]. An itemset X is a periodic frequent pattern

(PFP) if $|g(X)| \geq minsup$ and $maxper(X) < maxPer$, where *minsup* and *maxPer* are user-defined thresholds [11].

The PFP-tree algorithm is the first algorithm that has been proposed for mining PFPs [11]. It utilizes a tree-based and pattern-growth approach for discovering PFPs, inspired by the FP-Growth algorithm [12]. Thereafter, an algorithm called MTKPP [6] was designed. It relies on a depth-first search and a vertical database representation. To use this algorithm, a user needs to set a parameter k. The algorithm then outputs the k most frequent PFPs in a database. Approximate algorithms for mining periodic patterns have also been developped. For example, the ITL-tree algorithm [7] mines PFPs by considering an approximation of the periodicities of patterns. Another approximate algorithm for PFP mining was recently proposed [10]. Other extensions of the PF-Tree algorithm named MIS-PF-tree [8] and MaxCPF [9] were respectively proposed to mine PFPs using multiple *minsup* thresholds, and multiple *minsup* and *minper* thresholds. A drawback of the maximum periodicity measure used by most PFP algorithms is that an itemset is automatically discarded if it has a single period of length greater than the *maxPer* threshold. Thus, this measure may be viewed as too strict.

3. Novel Periodicity Measures

To address the above limitation of traditional PFP mining algorithms, and provide a more flexible way of evaluating the periodicity of patterns, the concept of *average periodicity* is proposed, which is defined as follows.

Definition 4 (Average periodicity of an itemset). Let there be an itemset X'. The average periodicity of X is denoted and defined as $avgper(X) = \sum_{g \in ps(X)} /|ps(X)|$.

For example, consider the itemsets $\{a, c\}$ and $\{e\}$. Their periods are $ps(\{a, c\}) = \{1, 2, 2, 1, 1\}$ and $ps(\{e\}) = \{2, 1, 1, 2, 1, 0\}$, and their average periodicities are $avgper(\{a, c\}) = 1.4$ and $avgper(\{e\}) = 1.16$.

Lemma 1 (Relationship between average periodicity and support). *Let there be an itemset X that appears in the database D. An alternative and equivalent way of calculating the average periodicity of X is $avgper(X) = |D|/(|g(X)| + 1)$.*

Proof 1. *Let $g(X) = \{T_{g_1}, T_{g_2}, \ldots, T_{g_k}\}$ be the set of transactions containing X, such that $g_1 < g_2 < \ldots < g_k$. By definition, $avgper(X) = \sum_{g \in ps(X)} /|ps(X)|$. To prove that the lemma holds, we need to show that $\sum_{g \in ps(X)} /|ps(X)| = |D|/(|g(X)| + 1)$.*
(1) We first show that $\sum_{g \in ps(X)} = |D|$, as follows:
$\sum_{g \in ps(X)} = (g_1 - g_0) + (g_2 - g_1) + \ldots (g_k - g_{k-1}) + (g_{k+1} - g_k)\}$
$= \sum_{g \in ps(X)} = g_0 + (g_1 - g_1) + (g_2 - g_2) + \ldots (g_k - g_k) + (g_{k+1})$
$= g_{k+1} - g_0 = |D|$.
(2) We then show that $|ps(X)| = |g(X)| + 1$, as follows:

By definition, $ps(X) = \bigcup_{1 \leq z \leq k+1} (g_z - g_{z-1})$. *Thus, the set* $ps(X)$ *contains* $k + 1$ *elements. Since* X *appears in* k *transactions,* $sup(X) = k$*, and thus* $|ps(X)| = |g(X)| + 1$.
Since (1) and (2) holds, the lemma holds. □

The above lemma is important as it provides an efficient way of calculating the average periodicity of itemsets in a database D. The term $|D|$ can be calculated once, and thereafter the average periodicity of any itemset X can be obtained by only calculating $|g(X)| + 1$, and then dividing $|D|$ by the result. This is more efficient than calculating the average periodicity using Definition 4. Besides, this lemma is important as it shows that there is a relationship between the support used in FIM and the average periodicity of a pattern.

Although the average periodicity is useful as it measures what is the typical period length of an itemset, it should not be used as the sole measure for evaluating the periodicity of a pattern because it does not consider whether an itemset has periods that vary widely or not. For example, the itemset $\{b, d\}$ has an average periodicity of 2.33. However, this is misleading since this itemset only appears in transaction T_3 and T_4, and its periods $ps(\{T_3, T_4\}) = \{3, 1, 4\}$ vary widely. Intuitively, this pattern should not be a periodic pattern. To avoid finding patterns having periods that vary widely, our solution is to combine the average periodicity measure with other periodicity measure(s). The following measures are combined with the average periodicity to achieve this goal.

First, we define the *minimum periodicity* of an itemset as $minper(X) = min(ps(X))$ to avoid discovering itemsets having some very short periods. But this measure is not reliable since the first and last period of an itemset are respectively equal to 1 or 0 if the itemset respectively appears in the first or the last transaction of the database. For example, the last period of itemset {e} is 0, because it appears in the last transaction (T_7), and thus its minimum periodicity is 0. Our solution to this issue is to exclude the first and last periods of an itemset from the calculation of the minimum periodicity. Moreover, if the set of periods is empty as a result of this exclusion, the minimum periodicity is defined as ∞. In the rest of this paper, we consider this definition.

Second, we consider the *maximum periodicity* of an itemset $maxper(X)$ as defined in the previous section. The rationale for using this measure in combination with the average periodicity is that it can avoid discovering periodical patterns that do not occur for long periods of time. In terms of calculation costs, a reason for choosing the minimum periodicity, maximum periodicity and average periodicity as measure is that they can be calculated very efficiently for an itemset X by scanning the list of transactions $g(X)$ only once. That is, calculating these measures do not require to store the set of periods $ps(X)$ in memory.

Thus, we define the concept of periodic frequent itemsets by considering the minimum periodicity, maximum periodicity, and average periodicity measures as follows.

Definition 5 (Periodic Frequent Itemsets with novel measures). Let $minAvg$, $maxAvg$, $minPer$, and $maxPer$ be positive numbers, provided by the user. An itemset X is a *periodic frequent itemset* if and only if $minAvg \leq avgper(X) \leq maxAvg$, $minper(X) \geq minPer$, and $maxper(X) \leq maxPer$.

Table 2: The set of PFPs for the running example

Itemset	support s(X)	minper(X)	maxper(X)	avgper(X)
$\{b\}$	3	1	3	1.75
$\{b, e\}$	3	1	3	1.75
$\{b, c, e\}$	3	1	3	1.75
$\{b, c\}$	3	1	3	1.75
$\{d\}$	3	1	3	1.75
$\{c, d\}$	3	1	3	1.75
$\{a\}$	4	1	2	1.4
$\{a, c\}$	4	1	2	1.4
$\{e\}$	5	1	2	1.17
$\{c, e\}$	4	1	3	1.4
$\{c\}$	6	1	2	1.0

For example, if $minPer = 1$, $maxPer = 3$, $minAvg = 1$, and $maxAvg = 2$, the 11 PFPs are found (shown in table 2).

To develop an efficient algorithm for mining PFPs, it is important to design efficient pruning strategies. To use the periodicity measures for pruning the search space, the following theorems are presented.

Lemma 2 (Monotonicity of the average periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $avgper(Y) \geq avgper(X)$.

Proof 2. *The average periodicities of X and Y are respectively $avgper(X) = |D|/(|g(X)| + 1)$ and $avgper(Y) = |D|/(|g(Y)| + 1)$. Because $X \subset Y$, it follows that $g(Y) \subseteq g(X)$. Hence, $avgper(Y) \geq avgper(X)$.* □

Lemma 3 (Monotonicity of the minimum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $minper(Y) \geq minper(X)$.

Proof 3. *Since $X \subset Y$, $g(Y) \subseteq g(X)$. If $g(Y) = g(X)$, then X and Y have the same periods, and thus $minper(Y) = minper(X)$. If $g(Y) \subset g(X)$, then for each transaction $T_x \in g(X) \setminus g(Y)$, the corresponding periods in $ps(X)$ will be replaced by a larger period in $ps(Y)$. Thus, any period in $ps(Y)$ cannot be smaller than a period in $ps(X)$. Hence, $minper(Y) \geq minper(X)$.* □

Lemma 4 (Monotonicity of the maximum periodicity). Let X and Y be itemsets such that $X \subset Y$. It follows that $maxper(Y) \geq maxper(X)$ [11].

Theorem 1 (Maximum periodicity pruning). Let X be an itemset appearing in a database D. X and its supersets are not PFPs if $maxper(X) > maxPer$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded.

Proof 4. *By definition, if $maxper(X) > maxPer$, X is not a PFP. By Lemma 4, supersets of X are also not PFPs.*

Theorem 2 (Average periodicity pruning). Let X be an itemset appearing in a database D. X is not a PFP as well as all of its supersets if $avgper(X) > maxAvg$, or equivalently if $|g(X)| < (|D|/maxAvg) - 1$. Thus, if this condition is met, the search space consisting of X and all its supersets can be discarded.

Proof 5. *By definition, if $avgper(X) > maxAvg$, X is not a PFP. By Lemma 2, supersets of X are also not PFPs. The pruning condition $avgper(X) > maxAvg$ is rewritten as: $|D|/(|g(X)|+1) > maxAvg$. Thus, $1/(|g(X)|+1) > maxAvg/|D|$, which can be further rewritten as $|g(X)|+1 < |D|/maxAvg$, and as $|g(X)| < (|D|/maxAvg) - 1$.* $\square$

4. The PFPM algorithm

Based on the novel periodicity measures introduced in the previous sections, an efficient algorithm named PFPM (Periodic Frequent Pattern Miner) is proposed to efficiently discover periodic patterns using these measures.

The proposed PFPM algorithm is a tid-list based algorithm, inspired by the Eclat algorithm [4]. The *tid-list* of an itemset X in a database D is defined as the set of transactions $g(X)$ that contains the itemset X. In the proposed algorithm, the tid-list of an itemset X is further annotated with two values: $minper(X)$ and $maxper(X)$.

The PFPM (Algorithm 1) takes as input a transaction database, and the $minAvg$, $maxAvg$, $minPer$ and $maxPer$ thresholds. The algorithm first scans the database to calculate $minper(\{i\})$, $maxper(\{i\})$, and $s(\{i\})$ for each item $i \in I$. Then, the algorithm calculates the value $\gamma = (|D|/maxAvg) - 1$ to be later used for pruning itemsets using Theorem 2. Then, the algorithm identifies the set I^* of all items having a periodicity no greater than $maxPer$, and appearing in no less than γ transactions (other items are ignored since they cannot be part of a PFP by Theorem 1 and 2. Items in I^* are then sorted according to the order $\succ$ of ascending support values, as suggested in [4]. A database scan is then performed. During this database scan, items in transactions are reordered according to the total order $\succ$, and the tid-list of each item $i \in I^*$ is built. Then, the depth-first search exploration of itemsets starts by calling the recursive procedure *Search* with the set of single items I^*, γ, $minutil$, $minAvg$, $minPer$, $maxPer$, the EUCS structure, and $|D|$.

The *PFPMSearch* procedure (Algorithm 2) takes as input extensions of an itemset P (initially assumed that $P = \emptyset$) having the form Pz meaning that Pz was previously obtained by appending an item z to P, γ, $minAvg$, $minPer$, $maxPer$, and $|D|$. The search procedure performs a loop on each extension Px of P. In this loop, the average periodicity of Px is calculated by dividing $|D|$ by the number of elements in the tid-list of Px plus one (by Lemma 1). Then, if the average periodicity of Px is in the $[minAvg, maxAvg]$ interval, the minimum/maximum periodicity of Px is no less/not greater than $minPer/maxPer$ according to the values stored in its tid-list, then Px is a PFP and it is output. Then, if the number of elements in the tid-list of Px is no less than γ, and $maxper(Px)$ is no greater than

Algorithm 1: The PFPM algorithm

```
input  : $D$: a transaction database,
         $minAvg$, $maxAvg$, $minPer$ and $maxPer$: the thresholds
output: the set of periodic high-utility itemsets
1 Scan $D$ once to calculate $minper(\{i\})$, $maxper(\{i\})$, and $s(\{i\})$ for each item $i \in I$;
2 $\gamma \leftarrow (|D|/maxAvg) - 1$;
3 $I^* \leftarrow$ each item $i$ such that $s(\{i\}) \geq \gamma$ and $maxper(\{i\}) \leq maxPer$;
4 Let $\succ$ be the total order of support ascending values on $I^*$;
5 Scan $D$ to build the tid-list of each item $i \in I^*$;
6 PFPMSearch ( $I^*$, $\gamma$, $minAvg$, $minPer$, $maxPer$, $|D|$);
```

$maxPer$, it means that extensions of Px should be explored (by Theorem 1 and 2). This is performed by merging Px with all extensions Py of P such that $y \succ x$ to form extensions of the form Pxy containing $|Px| + 1$ items. The tid-list of Pxy is then constructed by calling the *Intersect* procedure (cf. Algorithm 3), to join the tid-lists of P, Px and Py. This latter procedure is similar to the join of utility-list described in the Eclat algorithm [4], with the exception that periods are calculated during tid-list intersection to obtain $maxPer(Pxy)$ and $minPer(Pxy)$ (not shown). Then, a recursive call to the $PFPMSearch$ procedure with Pxy to explore its extension(s). The $PFPMSearch$ procedure starts from single items, recursively explores the search space of itemsets by appending single items, and only prunes the search space using Theorem 1 and 2. Thus, it can be easily seen that this procedure is correct and complete to discover all PFPs.

Furthermore, in the implementation of PFPM, two additional optimizations are included, which are briefly described next.

Optimization 1. Estimated Average Periodicity Pruning (EAPP). This optimization consists of creating a structure called ESCS (Estimated Support Co-occurrence Structure) to store the support of all pairs of items occurring in the database. The ESCS is defined as a set of triples of the form $(a, b, c) \in I^* \times I^* \times \mathbb{R}$. A triple (a,b,c) indicates that $s(\{a, b\}) = c$. The ESCS can be implemented as a triangular matrix (as shown in Fig. 1 for the running example), or as a hash map of hash maps where only tuples of the form (a, b, c) such that $c \neq 0$ are kept. The strategy EAPP is a novel strategy that prune itemsets using the average periodicity. During the second database scan, the ESCS is created to store $s(\{x, y\})$ for each pair of items {x,y} (as shown in Figure 1). Then, Line 7 of the search procedure is modified to prune itemset Pxy if $s(\{x, y\})$ is less than γ by Theorem 2.

Optimization 2. Abandoning List Construction early (ALC). Another strategy introduced in PFPM is to stop constructing the utility-list of an itemset if a specific condition is met, indicating that the itemset cannot be a PFP. By Theorem 2, an itemset Pxy cannot be a PFP, if it appears in less than $\gamma = (|D|/maxAvg) - 1$ transactions. The strategy ALC consists of modifying the Intersect procedure (Algorithm 3) as follows. The first modification is to initialize a variable max with the value γ in Line 1. The second modification is to the following lines, where the tid-list of Pxy is constructed by checking if each tuple in the

Algorithm 2: The $PFPMSearch$ procedure

input : *ExtensionsOfP*: a set of extensions of an itemset P, γ, $minAvg$, $minPer$, $maxPer$, $|D|$

output: the set of periodic frequent itemsets

```
1  foreach  itemset $Px \in ExtensionsOfP$ do
2      $avgperPx \leftarrow |D|/(|Px.tidlist| + 1)$;
3      if  $minAvg \leq avgperPx \leq maxAvg \wedge Px.tidlist.minp \geq minPer \wedge Px.tidlist.maxp \leq maxPer \wedge$ then  output $Px$;
4      if  $avgperPx \geq \gamma$ and $Px.tidlist.maxp \leq maxPer$ then
5          $ExtensionsOfPx \leftarrow \emptyset$;
6          foreach  itemset $Py \in ExtensionsOfP$ such that $y \succ x$ do
7              $Pxy \leftarrow Px \cup Py$;
8              $Pxy.tidlist \leftarrow$ Intersect $(Px, Py)$;
9              $ExtensionsOfPx \leftarrow ExtensionsOfPx \cup \{Pxy\}$;
10         end
11         PFPMSearch ($ExtensionsOfPx$, $\gamma$, $minAvg$, $minPer$, $maxPer$, $|D|$);
12     end
13 end
```

Item	a	b	c	d
b	1	▨	▨	▨
c	4	3	▨	▨
d	2	2	3	▨
e	2	3	4	2

Figure 1: The ESCS

tid-lists of Px appears in the tid-list of Py (Line 2). For each tuple not appearing in Py, the variable max is decremented by 1. If max is smaller than γ, the construction of the tid-list of Pxy can be stopped because $|g(Pxy)|$ will not be higher than γ. Thus Pxy is not a PFP by Theorem 2, and its extensions can also be ignored.

5. Experimental Study

To evaluate the performance of the proposed PFPM algorithm, we compared its performance with Eclat, a state-of-the-art algorithm for frequent itemset mining. The Eclat algorithm was chosen for comparison as the PFPM algorithm is based on Eclat. The PFPM and Eclat algorithms are implemented in Java. The experiment was carried out on a sixth generation 64 bit Core i5 processor running Windows 10, and equipped with 12 GB of free RAM. Four benchmark datasets were utilized in the experiment, which are commonly used in the FIM litterature. The *retail*, *chainstore*, and *foodmart* datasets contain anonymized

Algorithm 3: The Intersect procedure

```
input  : $Px$: the extension of $P$ with an item $x$,
         $Py$: the extension of $P$ with an item $y$
output: the tid-list of $Pxy$
1 $TidListOfPxy \leftarrow \emptyset$;
2 foreach  $Tid\ v \in Px.tidlist\ such\ that\ v \in Py.tidlist$ do
3     $period_v \leftarrow calculatePeriod(v,\text{TidListOfPxy})$;
4     $UpdateMinPerMaxPer(TidListOfPxy, period_v)$;
5     $TidListOfPxy \leftarrow TidListOfPxy \cup \{v\}$;
6 end
7 return $TidListOfPxy$;
```

customer transactions from retail stores, while the *mushroom* dataset is a dense benchmark dataset. The datasets have been chosen because they represents the main types of data encountered in real-life scenarios (dense, sparse and long transactions). Let $|I|$, $|D|$ and A represents the number of transactions, distinct items and average transaction length of a dataset. Characteristics of the four datasets are presented in Table 3

Table 3: Dataset characteristics

Dataset	$\lvert I\rvert$	$\lvert D\rvert$	A	Type
retail	88,162	16,470	10.30	sparse, many items
mushroom	8,124	119	23.0	dense, long transactions
chainstore	1,112,949	46,086	7.26	sparse, many transactions
foodmart	4,141	1,559	4.4	sparse, short transactions

The experiment consisted of running the PFPM algorithm on each dataset with fixed *minPer* and *minAvg* values, while varying the *maxAvg* and *maxPer* parameters. To be able to compare PFPM with Eclat, Eclat was run with the γ value calculated by PPFM. Execution times, memory consumption, and number of patterns found were measured for each algorithm. All memory measurements were done using the Java API.

For each dataset, values for the periodicity thresholds have been found empirically for each dataset (as they are dataset specific), and were chosen to show the trade-off between the number of periodic patterns found and the execution time. Note that results for varying the *minPer* and *minAvg* values are not shown because these parameters have less influence on the number of patterns found than the other parameters. Thereafter, the notation *PFPM V-W-X* represents the PFPM algorithm with $minPer = V$, $maxPer = W$, and $minAvg = X$. Fig. 2 compares the execution times of PFPM for various parameter values and Eclat. Fig. 3, compares the number of PFPs found by PFPM for various parameter values, and the number of frequent itemsets found by Eclat.

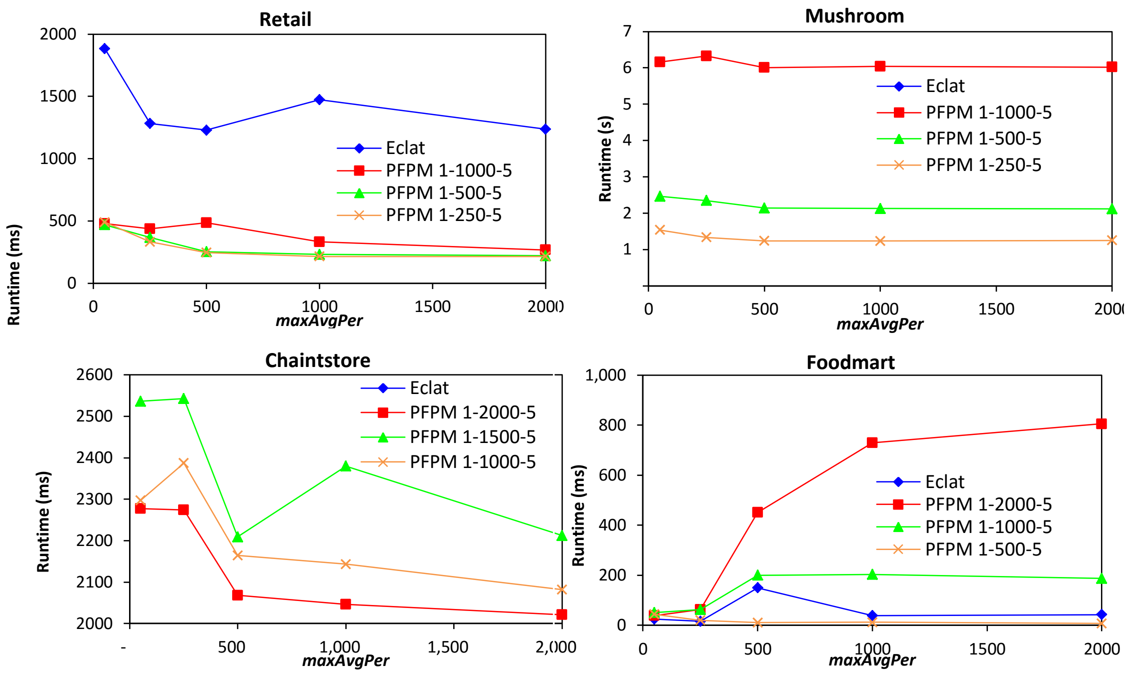

Figure 2: Execution times

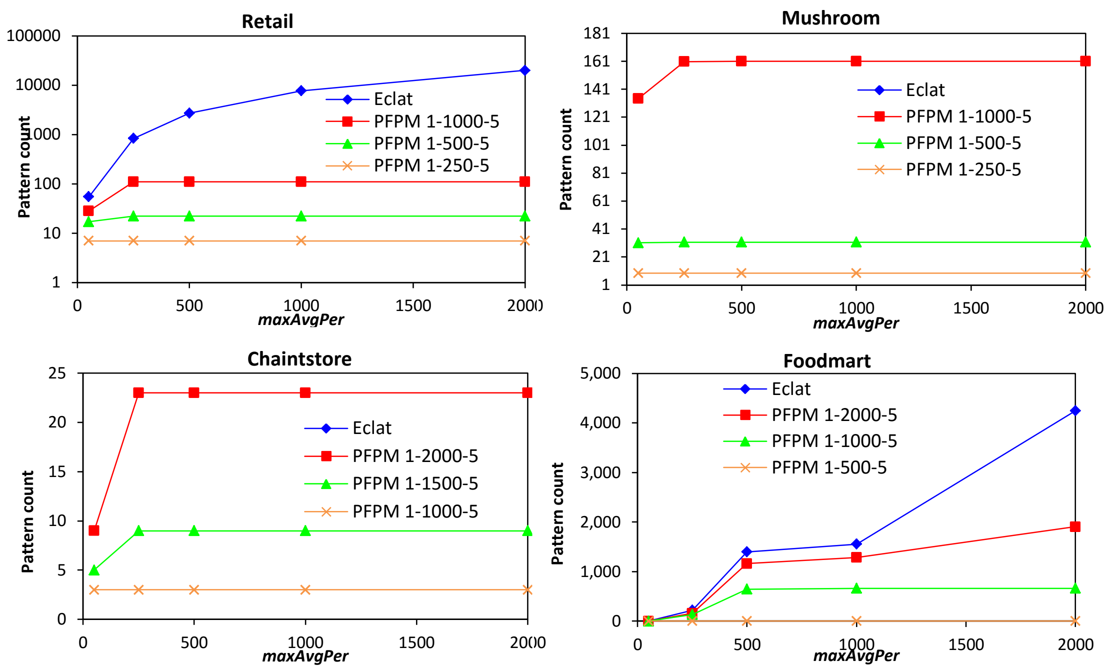

Figure 3: Number of patterns found

It can first be observed that mining PFPs using PFPM is generally much faster than mining frequent itemsets. On the *retail* dataset, PFPM is up to four times faster than Eclat.

On the *mushroom* and *chainstore* datasets, no results are shown for Eclat because it cannot terminate whithin 1,000 seconds or ran out of memory, while PFPM terminates in less than 10 seconds. The reason is that the search space is huge for these datasets when the minimum support is set to γ. The PFPM algorithm still terminates on these datasets because it only searches for periodic patterns, and thus prunes a large part of the search space containing non periodic patterns. On the *foodmart* dataset, PFPM can be up to five times faster than Eclat depending on the parameters. But it can also be slightly slower in some cases. The reason is that *foodmart* is a sparse dataset and thus the gain in terms of pruning the search space does not always offset the cost of calculating the periodicity measures. In general, the more the periodicity thresholds are restrictive, the more the gap between the runtime of Eclat and PFPM increases.

A second observation is that the number of PFPs can be much less than the number of frequent itemsets (see Fig. 3). For example, on *retail*, 19,836 frequent itemsets are found for $maxAvg = 2,000$. But only 110 frequent itemsets are PFPs for PFPM 1-1000-5, and only 7 for PFPM 1-250-5. Some of the patterns found are quite interesting as they contain several items. For example, it is found that items with product ids 32, 48 and 39 are periodically bought with an average periodicity of 16.32, a minimum periodicity of 1, and a maximum periodicity of 170. A similar reduction in terms of number of patterns is also observed on the three other datasets. This demonstrates that the PFPM algorithm is effective at filtering non periodic patterns, and that a huge amount of non periodic patterns are found in real-life datasets. Memory consumption was also compared, although detailed results are not shown. It was observed that PFPM use up to four and five times less memory than Eclat on the *retail* and *foodmart* datasets, depending on parameter values. For example, on *retail* and $maxAvg = 2,000$, Eclat and PFPM 1-5000-5-500 respectively consumes 900 MB and 189 MB of memory.

6. Conclusion

In this paper, we have proposed the use of three periodic measures for the discovery of periodic frequent patterns: the minimum periodicity, the maximum periodicity, and the average periodicity. Using three measures provides the advantages of giving more flexibility to the users. Properties of the novel minimum utility and average utility measures have been studied. Moreover, an efficient algorithm named PFPM (Periodic Frequent Pattern Miner) was proposed to efficiently discover all frequent periodic patterns using these measures. An experimental evaluation on real datasets shows that the proposed PFPM algorithm is efficient, and can filter a huge number of non periodic patterns to reveal only the desired periodic patterns. The source code of the PFPM algorithm and datasets will be made available as part of the SPMF open source data mining library [13] `http://www.philippe-fournier-viger.com/spmf/`. For future work, we will consider designing alternative algorithms to discover more complex types of periodic patterns.

Acknowledgements. This publication has been created within the project support of VŠB-TUO activities with China with financial support from the Moravian-Silesian Region and partially was supported by the grant SGS reg. no. No. SP2016/170 conducted at VSB-Technical University of Ostrava, Czech Republic.

References

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, SIGMOD Record 22 (2) (1993) 207–16.

[2] P. Jian, H. Jiawei, L. Hongjun, S. Nishio, T. Shiwei, Y. Dongqing, H-mine: fast and space-preserving frequent pattern mining in large databases, IIE Transactions 39 (6) (2007) 593–605.

[3] S.-i. Minato, T. Uno, H. Arimura, Lcm over zbdds: Fast generation of very large-scale frequent itemsets using a compact graph-based representation, Advances in Knowledge Discovery and Data Mining (2008) 234–246.

[4] M. J. Zaki, K. Gouda, Fast vertical mining using diffsets, Proceedings of the 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003.

[5] C. C. Aggarwal, J. Han, Frequent pattern mining, Springer, 2014.

[6] K. Amphawan, P. Lenca, A. Surarerks, Mining top-k periodic-frequent pattern from transactional databases without support threshold, in: Advances in Information Technology, Springer, 2009, pp. 18–29.

[7] K. Amphawan, A. Surarerks, P. Lenca, Mining periodic-frequent itemsets with approximate periodicity using interval transaction-ids list tree, in: Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on, IEEE, 2010, pp. 245–248.

[8] R. U. Kiran, P. K. Reddy, Mining rare periodic-frequent patterns using multiple minimum supports, work 5 (6) (2009) 7–8.

[9] A. Surana, R. U. Kiran, P. K. Reddy, An efficient approach to mine periodic-frequent patterns in transactional databases, in: New Frontiers in Applied Data Mining, Springer, 2011, pp. 254–266.

[10] R. U. Kiran, M. Kitsuregawa, P. K. Reddy, Efficient discovery of periodic-frequent patterns in very large databases, Journal of Systems and Software 112 (2016) 110–121.

[11] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, Y.-K. Lee, Discovering periodic-frequent patterns in transactional databases, in: Advances in Knowledge Discovery and Data Mining, Springer, 2009, pp. 242–253.

[12] J. W. Han, J. Pei, Y. W. Yin, Mining frequent patterns without candidate generation, Sigmod Record 29 (2) (2000) 1–12.

[13] P. Fournier-Viger, A. G. Penalver, T. Gueniche, A. Soltani, C.-W. Wu, V. S. Tseng, Spmf: a java open-source pattern mining library, Journal of Machine Learning Research (JMLR) (2014) 15:3389–3393.