Discovering Periodic Itemsets Using Novel Periodicity Measures

Philippe FOURNIER-VIGER[1], *Peng YANG*[2], *Jerry Chun-Wei LIN*[3], *Quang-Huy DUONG*[4], *Thu-Lan DAM*[4], *Jaroslav FRNDA*[5], *Lukas SEVCIK*[6], *Miroslav VOZNAK*[6]

[1]School of Humanities and Social Sciences, Harbin Institute of Technology (Shenzhen), University Town of Shenzhen, Shenzhen, China
[2]School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), University Town of Shenzhen, Shenzhen, China
[3]Department of Computing, Mathematics and Physics, Faculty of Engineering and Science, Western Norway University of Applied Sciences (HVL), Inndalsveien 28, 5063 Bergen, Norway
[4]Department of Computer and Information Science, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Hogskoleringen 1, 7491 Trondheim, Norway
[5]Department of Quantitative Methods and Economic Informatics, Faculty of Operation and Economics of Transport and Communications, University of Zilina, Univerzitna 8215/1, 010 26 Zilina, Slovak Republic
[6]Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB–Technical University of Ostrava, 17. listopadu 15, 708 00 Ostrava, Czech Republic

philfv8@yahoo.com, pengyeung@163.com, jerrylin@ieee.org, huydqyb@gmail.com, lanfict@gmail.com, jfrnda@gmail.com, lukas.sevcik@vsb.cz, miroslav.voznak@vsb.cz

DOI: 10.15598/aeee.v17i1.3185

Abstract. *Discovering periodic patterns in a customer transaction database is the task of identifying itemsets (sets of items or values) that periodically appear in a sequence of transactions. Numerous methods can identify patterns exhibiting a periodic behavior. Nonetheless, a problem of these traditional approaches is that the concept of periodic behavior is defined very strictly. Indeed, a pattern is considered to be periodic if the amount of time or number of transactions between all pairs of its consecutive occurrences is less than a fixed maxPer (maximum periodicity) threshold. As a result, a pattern can be eliminated by a traditional algorithm for mining periodic patterns even if all of its periods but one respect the maxPer constraint. Consequently, many patterns that are almost always periodic are not presented to the user. But these patterns could be considered as interesting as they generally appear periodically. To address this issue, this paper suggests to use three measures to identify periodic patterns. These measures are named average, maximum and minimum periodicity, respectively. They are each designed to evaluate a different aspect of the periodic behavior of patterns. By using them together in a novel algorithm called Periodic Frequent Pattern Miner, more flexibility is given to users to select patterns meeting specific periodic requirements. The designed algorithm has been evaluated on several datasets. Results show that the proposed solution is scalable, efficient, and can identify a small sets of patterns compared to the Eclat algorithm for mining all frequent patterns in a database.*

Keywords

Itemset mining, periodic pattern, periodicity, average periodicity.

1. Introduction

Analyzing symbolic data to discover frequently co-occurring symbols is a problem called Frequent Itemset Mining (FIM) [1], [2], [3], [4], [5] and [7]. FIM is considered by many as an important data science task with various applications in different fields [7]. The input of FIM is a database consisting of a set of records described by binary attributes. A typical example of such database is a customer transaction database, where each record is a transaction and each attribute indicates whether a given item was purchased or not in each transaction. To discover frequent itemsets, a user must set a minimum frequency threshold

called *minsup*. Then, a FIM algorithm outputs all sets of items (itemsets) that co-occur in more than *minsup* transactions.

Many studies have proposed efficient techniques to enumerate all frequent itemsets from a binary database, and numerous applications of these techniques have been presented [1], [2], [3], [4], [5] and [7]. Nonetheless, these techniques are inappropriate for identifying patterns that have a periodic behavior. Analyzing the periodic behavior of patterns is desirable for some applications such as the analysis of customer shopping behavior. For instance, one could analyze a transaction database and discover that a person typically purchases some items such as wine and cheese every weekend. Finding such periodic patterns is useful for the purpose of marketing. For example, a mobile application could provide discounts on wine and cheese to that customer before every weekend, anticipating that the customer will repeat that purchase. In a shopping context, periodic patterns for multiple customers could be also discovered. For example, if a retail store identifies that some type of bread is generally sold every hour, this information can be used to improve inventory management of this product. Besides analyzing shopping data, periodic pattern mining can be used in many other applications.

To identify recurring patterns in symbolic data, the concept of Periodic Frequent Patterns (PFP) was defined, and many algorithms were designed to enumerate these patterns [8], [9], [10], [11], [12], [13] and [14]. The input of a traditional PFP mining algorithm is a transaction database where transactions are ordered by time, and a parameter called *maxper* (maximum periodicity) is provided by the user. Then, the output is all periodic patterns, where an itemset is said to be periodic if the number of transactions between any of its consecutive occurrences (the length of its *periods*) is no greater or equal to *maxPer*. Mining such patterns was shown to be useful in previous studies. But a drawback of this definition is that it is a very strict definition of what is a periodic pattern. In fact, algorithms for mining these periodic patterns will eliminate a pattern from the result set even if all but one of its periods satisfy the *maxPer* constraint. Thus, many patterns that are generally periodic but not always periodic will not be presented to the user. However, such patterns still carry some interesting information. One may think that a solution to find such missing patterns is to increase the *maxPer* threshold. However, this can result in finding a very large number of patterns, and it is typically difficult and time-consuming for users to analyze a large pattern set.

To address this issue, a novel problem is defined in this paper, which is to identify periodic patterns using three measures. Those are the average, maximum and minimum periodicity, respectively. They are each designed to each evaluate a different aspect of the periodic behavior of patterns. By using them together in a novel algorithm called PFPM (Periodic Frequent Pattern Miner), more flexibility is given to users to select patterns meeting specific periodic requirements.

The contributions of this paper are as follows:

- Two measures called *minimum* and *average* periodicity are defined to be used jointly with the *maximum* periodicity measure. These measures let the user more precisely specify requirements about the periodic behavior of patterns to be discovered. Using the average periodicity measure, patterns that are generally but not always periodic can be discovered.
- To enumerate all periodic patterns satisfying constraints on the above measures, an efficient algorithm is proposed, called Periodic Frequent Pattern Miner (PFPM) - Note that an early version of this paper was published in a conference proceedings [18].
- The performance in terms of scalability, runtime and memory has been evaluated on four real datasets. It has been observed that the newly designed algorithm is scalable, efficient and can eliminate numerous patterns that are not periodic to show a small set of patterns to the user.

The following sections describe relevant related work and some important preliminaries about frequent itemset mining (Sec. 2.), present the designed minimum periodicity and average periodicity measures (Sec. 3.), describe the proposed algorithm (Sec. 4.), report results from the experimental evaluation (Sec. 5.) and draw a conclusion and discusses opportunities for future work (Sec. 6.), respectively.

2. Related Work

FIM consists of analyzing a binary database (also called transaction database) to identify sets of symbols that appear together in many records. FIM was defined by Agrawal and Srikant [1] as follows.

A binary database DB is a set of records (called transactions), described using several binary attributes (items). Let the set of all database attributes (items) be called A. Each record (transaction) R is defined by the set of attributes $R \subseteq A$ for which it has positive values. A transaction database DB containing n records is denoted as $DB = \{R_1, R_2, ..., R_n\}$. The indice c of a record R_c in a database is also called its identifier, and it is unique. It is assumed that records in a database are ordered from the oldest one to the newest one.

Tab. 1: A binary database containing seven records.

Records	Binary attributes				
	a	b	c	d	e
R_1	x		x		
R_2					x
R_3	x	x	x	x	x
R_4		x	x	x	x
R_5	x		x	x	
R_6	x		x		x
R_7		x	x		x

A small binary database is depicted in Tab. 1. It contains seven records having the identifiers $1, 2, 3, \ldots, 7$, respectively. Each record is described using five binary attributes denoted as a, $b, \ldots, e$. Consider the first record R_1. It indicates that the attributes a and c have positive values. If this database represents a shopping database, it could indicate that some products a and c were bought. The rest of this paper uses the database of Tab. 1 as running example.

A pattern (also called itemset) X is a set of positive attribute values, and is defined as $X \subseteq A$. In traditional FIM, interesting patterns are discovered on the basis of their support (occurrence frequency). For a database DB, let $s(X)$ denote the support of X defined as $s(X) = |\{R | R \in DB \wedge X \subseteq R\}|$.

An alternative definition of the support is the following. For an itemset X, let the notation $g(X)$ refer to the set of records containing X, where records are ordered by time. For a database DB containing n records, $g(X)$ is formally defined as $g(X) = \{R_{g_1}, R_{g_2}..., R_{g_k}\}$, where $1 \leq g_1 < g_2 < ... < g_k \leq n$. Hence, the support of X can be also defined as $s(X) = |g(X)|$.

In FIM, the goal is to enumerate all frequent itemsets, given a minimum support threshold *minsup*, set by the user [1]. An itemset X is said to be *frequent* if and only if $s(x) \geq minsup$. For example, consider the database of Tab. 1 and that a user sets $minsup = 4$. In that cases, five frequent itemsets are found, which are $\{a\} : 4$, $\{a, c\} : 4$, $\{e\} : 5$, $\{c, e\} : 4$, $\{c\} : 6$, where the notation $X : s(X)$ is used.

Many algorithms have been developed to enumerate all frequent patterns in a database. Some popular algorithms are for example, FP-Growth [15], Apriori [1], LCM [3], and Eclat [4]. These algorithms all produce the same result but utilize different data structure, search strategies and optimizations. The Apriori algorithm utilizes a breadth-first search and repeatedly scan the database, which is very costly, and can generate many candidate patterns. The FP-Growth compresses the database in a compact database structure, and performs database projections to avoid generating candidates. The Eclat algorithm utilizes a vertical database representation to calculate the support of patterns to avoid repeatedly scanning the database. Although these algorithms are efficient, they are designed for finding frequent itemsets rather than identifying itemsets having a periodic behavior.

To discover patterns exhibiting periodic behaviors, traditional FIM algorithms have been adapted and extended. Many algorithms were defined to identify Periodic Frequent Patterns (PFPs) in a binary attribute database [8], [9], [10], [11], [12], [13] and [14]. They generally extend FIM algorithms with the ability of calculating periodicity measures, and rely on appropriate data structures and optimizations to perform these calculations efficiently. Periodic frequent pattern mining techniques are useful for various applications [13]. Formally, the concept of PFP is defined as follows, based on a concept of periods [13].

Definition 1 (Consecutive records)**.** *Consider an itemset X appearing in a set of records $g(X) = \{R_{g_1}, R_{g_2}, \ldots, R_{g_k}\}$ of a binary database $DB = \{R_1, R_2, \ldots, R_n\}$. Two records R_x and R_y are consecutive w.r.t. X in DB if there does not exist an identifier w of a record R_w such that $x < w$ and $w < y$.*

Definition 2 (Period of consecutive records of an itemset)**.** *Let there be two consecutive records R_x and R_y in $g(X)$ for an itemset X. The period of R_x and R_y is denoted as $pe(R_x, R_y)$, and defined as the number of records between R_x and R_y, that is $pe(R_x, R_y) = y - x$.*

Definition 3 (Periods of an itemset)**.** *The periods of an itemset X appearing in a set of records $g(X) = \{R_{g_1}, R_{g_2}, \ldots, R_{g_k}\}$ is defined formally as $ps(X) = \{g_1 - g_0, g_2 - g_1, g_3 - g_2, \ldots g_k - g_{k-1}, g_{k+1} - g_k\}$, where $g_0 = 0$ and $g_{k+1} = n$.*

For instance, the database records containing $\{a, c\}$ are R_1, R_3, R_5, and R_6 in the running example. Hence, the periods of $\{a, c\}$ are calculated as $ps(\{a, c\}) = \{1 - 0, 3 - 1, 5 - 3, 6 - 5, 7 - 6\} = \{1, 2, 2, 1, 1\}$.

Definition 4 (Maximum periodicity)**.** *Let there be an itemset X. Its maximum periodicity is defined as $maxper(X) = max(ps(X))$ [13].*

Definition 5 (Periodic Frequent Pattern)**.** *Let there be an itemset X and $maxPer$ be a threshold set by the user. X is said to be a periodic frequent pattern if $maxper(X) < maxPer$ and $s(X) \geq minsup$ [13].*

For instance, assume that $maxPer = 2$ and $minsup = 4$. A traditional PFP mining algorithm will enumerate all periodic frequent patterns, i.e. (4, 2), {a,c} : (4, 2), {e} : (5, 2), {c} : (6, 2), where the notation $X : (s(X), maxper(X))$ is used.

Many algorithms have been developed to identify PFPs from a transaction database. The first algorithm, named PFP-tree [13], adopts a pattern-growth

approach extending FP-Growth [15] to mine PFPs. Then, the MTKPP [8] algorithm was presented to discover the k most frequent PFPs in a database, where k and $maxPer$ are parameters that must be set by the user. MTKPP is inspired by Eclat [4]. It relies on the same vertical database representation and explores the search space in the same depth-first way. The ITL-tree algorithm [9] was then proposed. It is an approximate algorithm, which uses a tree based approach to find PFPs. Unlike previous algorithms, ITL-tree does an approximate calculation of the periodicities of itemsets. PFP mining is another algorithm for the approximate discovery of PFPs [12]. In another work, the MCPF-tree algorithm [10] was proposed by extending PF-Tree to mine PFPs when considering many *minsup* thresholds. Another extension of the PF-Tree algorithm is MaxCPF [11], which considers multiple *minper* thresholds. Recently, to find PFPs common to multiple sequences of a sequence database, an algorithm named MPFPS was proposed [14]. But this algorithm is defined for a problem that is quite different from the proposed addressed in this paper as it consider multiple sequences. In summary, many papers have defined a periodic pattern as an itemset having no period greater than some $maxPer$ threshold set by the user. A problem of this definition is that it is very strict. For example, an itemset will be discarded even if it has a single period greater than $maxPer$. In practice, it would be desirable to discover such patterns that are almost always periodic.

3. Two Novel Measures

To provide more flexibility when searching for periodic patterns, this paper proposes to consider two novel periodicity measures, called the *average* and *minimum periodicity*, respectively. The following paragraphs present these measures and discuss their properties.

Definition 6 (Average periodicity)**.** *The average periodicity of an itemset X is the average of its periods. It is formally defined as:* $avgper(X) = \frac{\sum_{p \in ps(X)} p}{|ps(X)|}$.

For instance, for the itemset $\{a, c\}$, we have $ps(\{a, c\}) = \{1, 2, 2, 1, 1\}$ and $avgper(\{a, c\}) = 1.4$. For the itemset $\{e\}$, we have $ps(\{e\}) = \{2, 1, 1, 2, 1, 0\}$ and $avgper(\{e\}) = 1.16$.

It is interesting that the average periodicity measure can be calculated using the support measure, and vice-versa.

Lemma 1 (Correspondence between support and average periodicity)**.** *Consider a database DB. The average periodicity $avgper(X)$ of an itemset X can be calculated as* $avgper(X) = |DB|/(s(X) + 1) = |DB|/(|g(X)| + 1)$.

Proof. The ordered list of records containing X in DB is $g(X) = \{R_{g_1}, R_{g_2}, \ldots, R_{g_k}\}$, where $g_1 < g_2 < \ldots < g_k$. By definition, $avgper(X) = \frac{\sum_{p \in ps(X)} p}{|ps(X)|}$. Thus, we only need to demonstrate that $\frac{\sum_{p \in ps(X)} p}{|ps(X)|} = \frac{|DB|}{|g(X)|+1}$ to prove the lemma.

(1) We first show that $\sum_{p \in ps(X)} p = |DB|$, as follows:

$\sum_{p \in ps(X)} p = (g_1 - g_0) + (g_2 - g_1) + \ldots + (g_{k+1} - g_k)$
$= g_0 + (g_1 - g_1) + (g_2 - g_2) + \ldots + (g_k - g_k) + g_{k+1}$
$= g_{k+1} - g_0 = |DB|$.

(2) To demonstrate that $|ps(X)| = |g(X)| + 1$, we proceed as follows:
An equivalent definition of $ps(X)$ is $ps(X) = \bigcup_{1 \leq z \leq k+1} (g_z - g_{z-1})$. Hence, $k + 1$ elements are in $ps(X)$. Because X is contained in k records, $sup(X) = k$. Thus, it is found that $|ps(X)| = |g(X)| + 1$.

Because (1) and (2) have been proven, the lemma holds.

Lemma 1 explains the relationship between the average periodicity and support of each itemset. From a practical perspective, this lemma is very useful as it allows to efficiently compute $avgper(X)$ for any itemset X of a database DB. The reason why this calculation is efficient is that the number of database records $|DB|$ is known and can be precalculated once. Then, to calculate $avgper(X)$, only $|g(X)| + 1$ remains to be computed, and then divided by the number of database records. Doing this calculation is easier than computing $avgper$ using Def. 6, as this later requires to sum all periods of X and then divide by the number of periods.

The *average periodicity* can be viewed as an interesting measure because it allows finding patterns that are on average periodic. However, this measure can be easily influenced by outliers. This is illustrated with an example. The periods of itemset $\{c, d, e\}$ are $ps(\{R_3, R_4\}) = \{3, 1, 4\}$ because $\{c, d, e\}$ appears in records R_3 and R_4. Hence, $avgper(\{R_3, R_4\}) = 2.33$. But it can be argued that this pattern should not be viewed as periodic since there are very large differences between its periods. To address this drawback of the average periodicity, this paper proposes to combine this measure with the maximum periodicity measure and a novel minimum periodicity measure. Using this combination ensures that the average periodicity of an itemset is not influenced by outliers.

Definition 7 (Minimum periodicity)**.** *Let there be an itemset X. Its minimum periodicity is defined as $minper(X) = min(ps(X))$ (where the first and last periods of $ps(X)$ are ignored). In the case where $ps(X) = \emptyset$, $minper(X)$ is defined as ∞.*

Setting a constraint on the minimum periodicity of itemsets can be useful to eliminate itemsets that have very short periods. But if we simply defines the minimum periodicity as $minper(X) = min(ps(X))$, some problems occur because the first and last periods of an itemset are special cases that can greatly influence the overall minimum periodicity. For instance, consider the itemset $\{e\}$. Because $\{e\}$ is contained in the last record, the last period of $\{e\}$ is 0, as well as its minimum periodicity. To avoid such situations, the first and last periods are excluded from the calculation of the minimum periodicity. A second problem is that if these periods are excluded, the set of periods of an itemset may then become empty. To handle this case, the minimum periodicity is then defined as ∞.

By combining the minimum, maximum and average periodicity measures to find periodic patterns, it is possible to finely evaluate the periodic behavior of patterns. Another interesting aspect of these measures is that they are easy to calculate if the set of records $g(X)$ is known for an itemset X. This is especially useful for extending the Eclat frequent itemset mining to find periodic patterns, because Eclat already calculates $g(X)$ for each itemset X encountered during search space exploration. Besides, another reason why calculating these measures is efficient is that it is not necessary to calculate and store $ps(X)$ in memory to calculate these measures. In fact, all measures can be calculated while scanning $g(X)$ once.

Based on the above discussion and novel measures, this paper defines a novel problem of mining periodic frequent itemsets with novel measures as follows.

Definition 8 (Problem definition)**.** *Let there be four user-specified thresholds* $minAvg \geq 0$, $maxAvg \geq 0$, $minPer \geq 0$, *and* $maxPer \geq 0$. *An itemset* X *is a periodic frequent itemset if and only if three conditions are met: (1)* $minper(X) \geq minPer$, *(2)* $maxper(X) \leq maxPer$, *and (3)* $minAvg \leq avgper(X) \leq maxAvg$.

For instance, consider the database of the running example and that the thresholds are set to $minPer = minAvg = 1$, $maxPer = 3$ and $maxAvg = 2$. Table 2 shows the eleven PFPs.

To efficiently enumerate all periodic frequent itemsets from a binary database, it is necessary to avoid considering the whole search space of itemsets. For this purpose, the following paragraphs presents strategies that can be used to reduce the search space using the periodicity measures considered in this paper.

Lemma 2 (Monotonicity of the average periodicity)**.** For a database DB and any itemsets $X \subset Y$, the relationship $avgper(Y) \geq avgper(X)$ holds.

Proof. By the definition of the average periodicity measure, we have $avgper(X) = \frac{|DB|}{|g(X)|+1}$ and $avgper(Y) = \frac{|DB|}{|g(Y)|+1}$. Since X is a subset of Y, it is clear that $g(Y) \subseteq g(X)$ and thus that $avgper(Y) \geq avgper(X)$.

Lemma 3 (Monotonicity of the minimum periodicity)**.** For a database DB and any itemsets $X \subset Y$, the relationship $minper(Y) \geq minper(X)$ holds.

Proof. Because Y is a superset of X, it follows that $g(Y) \subseteq g(X)$. Two cases are considered. In the first case ($g(Y) \subset g(X)$). Then, for each record $R_x \in g(X) \setminus g(Y)$, the corresponding periods in $ps(X)$ will be replaced by a larger period in $ps(Y)$. Hence, each period of $ps(Y)$ must be greater than the corresponding period(s) in $ps(X)$. Hence, $minper(Y) \geq minper(X)$. In the second case, if $g(Y) = g(X)$, then $ps(X) = ps(Y)$ and hence $minper(Y) = minper(X)$.

Lemma 4 (Monotonicity of the maximum periodicity)**.** For a database DB and any itemsets $X \subset Y$, the relationship $maxper(Y) \geq maxper(X)$ holds [13].

Based on the above lemmas, two search space pruning theorems are defined. The first one was used in previous work [13], while the second one is novel.

Theorem 1 (Search space pruning using the maximum periodicity)**.** *Consider a database* DB *and an itemset* X. *If it is found that* $maxper(X) > maxPer$, *then all supersets of* X *are not periodic frequent itemsets and can be ignored during further exploration of the search space [13].*

Theorem 2 (Search space pruning using the average periodicity)**.** *Consider a database* DB *and an itemset* X. *If it is found that* $avgper(X) > maxAvg$, *then all supersets of* X *are not periodic frequent itemsets and can be ignored during further exploration of the search space [13]. Note that the condition* $|g(X)| < \frac{|D|}{maxAvg} - 1$ *can be equivalently used for pruning.*

Proof. Because $avgper(X) > maxAvg$, the itemset X is not a periodic frequent itemset. Moreover, by Lem. 2, it is known that $avgper(Y) \geq avgper(X)$ for any superset Y of X. The pruning condition $avgper(X) > maxAvg$ is rewritten as: $\frac{|DB|}{|g(X)|+1} > maxAvg$. Thus, $\frac{1}{|g(X)|+1} > \frac{maxAvg}{|DB|}$, which can be further rewritten as $|g(X)| + 1 < \frac{|DB|}{maxAvg}$, and as $|g(X)| < \frac{|DB|}{maxAvg} - 1$.

4. The PFPM Algorithm

The previous section has defined the problem of mining periodic itemsets using novel measures. This section defines an efficient algorithm named Periodic Frequent

Tab. 2: The periodic frequent itemsets for $minPer = minAvg = 1$, $maxPer = 3$ and $maxAvg = 2$.

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

Pattern Miner (PFPM), which extends the Eclat [4] algorithm to efficiently enumerate periodic patterns using the novel measures. As the Eclat algorithm, PFPM utilizes a structure called *tid-list* to annonate each potential periodic itemset X with the list of records $g(X)$ where it appears. This structure is suitable for the proposed problem as it allows to quickly obtain $|g(X)|$ to calculate the periodicity measures of each itemset X. Moreover, the $minper(X)$ and $maxper(X)$ values are used to annotate each itemset X.

The pseudocode of the proposed PFPM algorithm is presented in Alg. 1. PFPM explores the search space of itemsets in a recursivey way by extending each itemset one item at a time. PFPM processes items following an order $\succ$ on items, defined as the ascending order of support values. In the following, the *extensions* of an itemset X are the itemsets that can be obtained by appending an item y to X such that $y \succ a$, $\forall a \in X$. The input of PFPM is four thresholds ($minPer$, $maxPer$ $minAvg$, $maxAvg$) and a binary database DB, while the output is the set of all periodic itemsets. PFPM initially reads the database to compute for each item $a \in A$, $s(\{a\})$, $minper(\{a\})$ and $maxper(\{a\})$. Thereafter, a constant called γ is calculated. This latter is used for reducing the search space based on Thm. 2. PHPM then selects items having a support no less than γ and having periods no greater than $maxPer$. These items, called A^*, are the only items that can appear in periodic frequent itemsets according to the search space pruning Thm. 1 and Thm. 2. This set is sorted according to the $\succ$ order, as suggested in [4]. Thereafter, PFPM reads the database again to construct the tid-list of each item in A^*. Finally, the *Search* procedure is called with A^*, γ, $minAvg$, $minPer$, $maxPer$, and $|DB|$ to recursively explore the search space in a depth-first way.

Algorithm 2 provides the pseudo-code of the *Search* procedure. An itemset P is taken as input, as well as a set of extensions of P of the form Pz where z is an item. When the *Search* procedure is called for the first time, P is the empty set and the set of extensions contains single items. *Search* also takes as parameters γ,

Algorithm 1: Periodic Frequent Pattern Miner

input : $minPer$, $maxPer$, $minAvg$, $maxAvg$, a binary database DB
output: the set of periodic itemsets

1 Read the database once to compute for each item $a \in A$: $s(\{a\})$, $minper(\{a\})$ and $maxper(\{a\})$;
2 Calculate $\gamma = (|DB|/maxAvg) - 1$;
3 $A^* \leftarrow \{a | a \in A \wedge s(\{a\}) \geq \gamma \wedge maxper(\{a\}) \leq maxPer\}$;
4 Create the tid-list of each item $a \in A^*$ by reading the database DB;
5 `Search` (A^*, γ, $minAvg$, $minPer$, $maxPer$, $|DB|$);

and the $minAvg$, $minPer$, $maxPer$, and $|DB|$ tresholds. The procedure then performs a loop to consider each extension Px of P from the set received as parameter. The procedure first calculates $avgper(Px)$ as $|DB|$ divided by the number of elements in the tid-list of Px plus one (according to Lem. 1). Then, if the conditions (1) $minAvg \leq avgper(Px) \leq maxAvg$, (2) $minper(Px) \leq minPer$ and (3) and $maxper(Px) \leq maxPer$ are satisfied according to the tid-list of Px, Px is output as a periodic frequent itemset. Thereafter, if the third condition is satisfied and the tid-list of Px contains at least γ records, the algorithm will consider extensions of Px. Otherwise, those extensions are ignored according to Thm. 1 and Thm. 2. Exploring larger extensions is done by combining two itemsets Px and Py where $y \succ x$ to get an itemset Pxy. After obtaining an itemset Pxy, the $BuildTIDList$ procedure (Alg. 3) is called to construct the tid-list of Pxy. This procedure takes as input the tid-lists of Px and Py and returns the tid-list of Pxy. The $BuildTIDList$ procedure is almost identical to the tid-list join operation of Eclat. But a key difference is that periods of Pxy, $maxPer(Pxy)$ and $minPer(Pxy)$ are calculated during list construction (not shown). Finally, the *Search* procedure is recursively called with Pxy to explore its extension(s). Because the PFPM algorithm starts from single items and recursively explores the search space

of patterns by appending items and only prunes the search space using Thm. 1 and Thm. 2, it can be seen that this procedure is correct and complete to discover all PFPs.

Algorithm 2: Search

input : *ExtensionsOfP*: a set of extensions of an itemset P, γ, $minAvg$, $minPer$, $maxPer$, $|DB|$
output: the set of periodic frequent itemsets

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

To illustrate how the algorithm works, an example is given. Consider the database of Tab. 3 and that $minAvg = 1$, and $maxAvg = 2$, $minPer = 1$, and $maxPer = 3$. PFPM (Alg. 1) first processes single items. Consider the item $\{a\}$. By scanning the database, PFPM finds that $s(\{a\}) = 4$, $minper(\{a\}) = 1$ and $maxper(\{a\}) = 2$. PFPM calculates that $\gamma = (|DB|/maxAvg) - 1 = (7/2) - 1 = 2.5$. Since $s(\{a\}) \geq \gamma$ and $maxper(\{a\})$ is less than the maximum periodicity threshold, the item $\{a\}$ will be considered during further processing (each item that does not meet the requirements is pruned). Then, the same process is repeated for the other items and a set of PFP candidates is found. Iitems are then sorted by ascending support values ($\{b\}, \{d\}, \{a\}, \{e\}, \{c\}$). The *tid-lists* of these candidates are built (e.g. $\{b\}$'s *tid-list* $[T_3,T_4,T_7]$ is built), and the Search procedure is called to find all PFPs (Alg. 2). The search procedure performs a loop on each extension Px of P. Consider $Px = \{b\}$. By using $\{b\}$'s *tid-list*, PFPM finds that $avgper(\{b\}) = \frac{7}{3+1} = 1.75$, $maxper(\{b\}) = 3$ and $minper(\{b\}) = 1$. As $minAvg \leq avgper(\{b\}) \leq maxAvg$, $minper(\{b\}) \geq minPer$ and $maxper(\{b\}) \leq maxPer$, item {b} is

Algorithm 3: The BuildTIDList procedure

input : two extensions Px and Px of an itemset P
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 $period_v \leftarrow calculatePeriod(v, TidListOfPxy)$;
4 $Update(TidListOfPxy, period_v)$;
5 $TidListOfPxy \leftarrow TidListOfPxy \cup \{v\}$;
6 **end**
7 **return** $TidListOfPxy$;

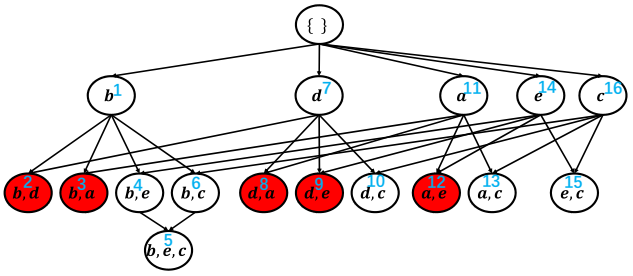

Fig. 1: The search space of the example.

a PFP and it is output. The extensions of $\{b\}$ are: $\{b,d\}$,$\{b,a\}$,$\{b,e\}$ and $\{b,c\}$, which will be explored because $s(\{b\}) \geq \gamma$ and $maxper(\{b\}) \leq maxPer$. The tid-list of $\{d\}$ is $[T_3,T_4,T_5]$, thus the tid-list of $\{b,d\}$ is $[T_3,T_4]$, obtained by joining the tid-list of $\{b\}$ and $\{d\}$. Thereafter, the Search procedure is recursivelly invoked with these extensions to find larger PFPs. The search space of the algorithm is shown in Fig. 1. In that figure, itemsets that do not meet one of the constraint are colored in red and their extensions (descendant nodes) are not explored. The complete set of PFPs that is found is shown in Tab. 2.

The above paragraphs have described the main idea of the PFPM algorithm. The next paragraphs presents two optimizations to enhance PFPM's performance.

Optimization 1. Estimated Average Periodicity Pruning (EAPP). During the second database scan, a structure named ESCS (Estimated Support Cooccurrence Structure) is built, which stores the support $s(\{a,b\}) = c$ of each pair of item a and b as a triple of the form (a,b,c). From a practical perspective, the ESCS can be implemented using vairous structure such as a hashmap of hashmaps or a triangular matrix (as shown in Fig. 2). An advantage of using a hash map is that each tuple (a,b,c) where $c \neq 0$ may be ommitted to reduce memory usage. The EAPP optimizations consists of modifying Line 7 of the search procedure to prune an itemset Pxy if $s(\{x,y\})$ is less than γ (based on Thm. 2).

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

Fig. 2: The Estimated Support Co-occurrence Structure.

Optimization 2. Abandoning List Construction early (ALC). The ALC optimization stops the construction of an itemset's tid-list if current calculations of its periodicity measures show that it cannot be a periodic frequent itemset. According to Thm. 2, an itemset Pxy cannot be a periodic frequent itemset if less than $\gamma = (|DB|/maxAvg) - 1$ records contains Pxy. The ALC strategy is obtained by first changing Line 1 of Alg. 3 to initialize a variable max as γ in Line 1. Modifications are also made in the loop of Line 2. For each tuple not appearing in Py, the variable max is decremented by 1. If max becomes less than γ, ALC stops the construction of Pxy's tid list. This can be done without compromising the algorithm's completeness because $|g(Pxy)|$ will not be higher than γ, and hence Pxy is not a PFP by Thm. 2, and its extensions can also be ignored.

5. Experimental Study

This section presents an experimental study to assess the performance of the designed PFPM algorithm in terms of runtime, memory consumption, number of patterns found and scalability, when parameters are varied. The experimental environment is a Windows 10 computer equipped with 12 GB of RAM and a 6th gen 64 bit Core i5 processor. The designed algorithm is programmed in Java. In the experiments, the performance of PFPM is compared with that of the Eclat algorithm because it is one of the most efficient algorithm for FIM and PFPM extends the Eclat algorithm. Thus, a comparison with Eclat allows to see the cost or benefits of using PFPM to find only the periodic patterns rather than finding all frequent itemsets. Performance of the algorithms was evaluated on four datasets often used for comparing itemset mining algorithms, having varied characteristics (short or long transactions, or dense or sparse data). *chainstore*, *retail*, and *foodmart* are customer transaction databases. *mushroom* is a dataset about mushrooms. Characteristics of the four datasets are presented in Tab. 3, where $|DB|$, $|A|$ and A denote the number of records, distinct items and average record length.

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 $minAvg = X$. Figure 3 compares the execution times of PFPM for various parameter values and Eclat. Figure 4, compares the number of PFPs found by PFPM for various parameter values, and the number of frequent itemsets found by Eclat.

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. 4). 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.

Tab. 3: Dataset characteristics.

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

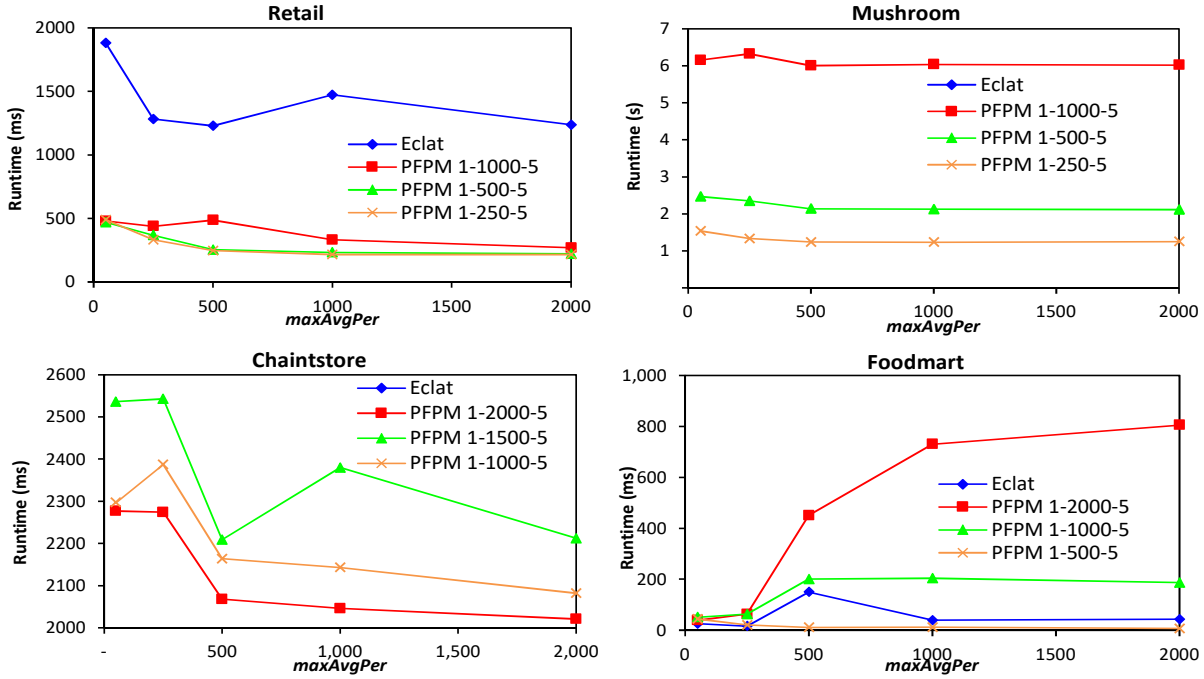

Fig. 3: Execution times.

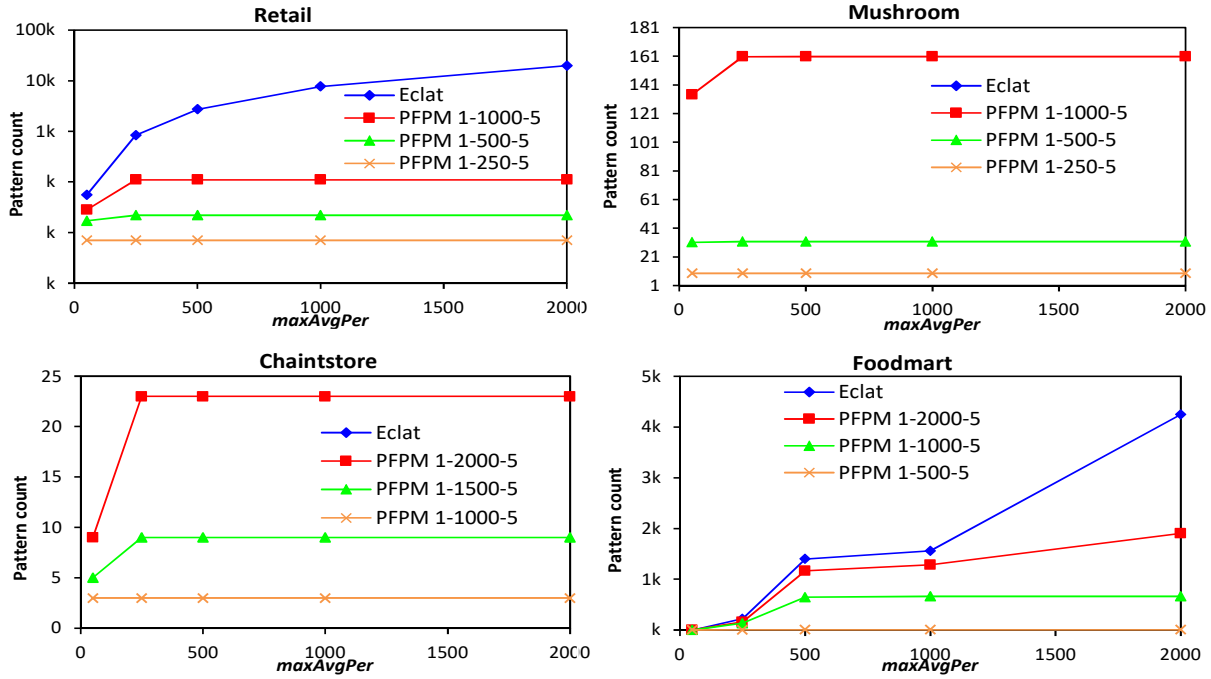

Fig. 4: Number of patterns found.

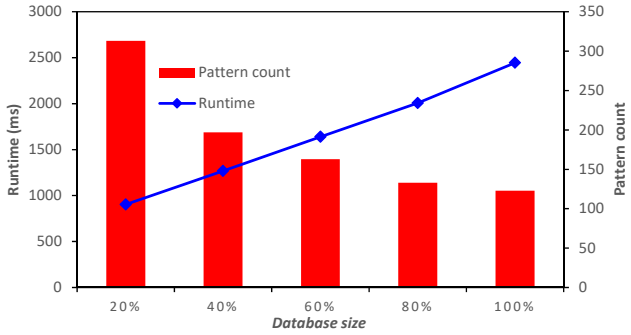

Fig. 5: The scalability.

The scalability of the algorithm with respect to database size was also assessed in another experiment. The execution time and number of patterns found by PFPM was measured for various number of records. In this experiment, the real *chainstore* dataset was used, since it is a huge sparse dataset with a large number of distinct items and records (its characteristics are shown in Fig. 3). We divided the dataset into five parts. Then, we ran PFPM on 20 %, 40 %, 60 %, 80 % and 100 % of the data. Figure 5 shows the experimental result for $minPer = 1$, $maxPer = 5000$, $minAvg = 5$ and $maxAvg = 5000$ for each database size. The black line indicates the execution time of the algorithm, and bars indicate the number of patterns found. It can be observed that the runtime increases and the number of patterns decreases when the database size is increased, respectively. This is reasonable because it takes more time to process a larger database, and fewer patterns may meet the constraints of periodic patterns for larger databases.

Performance in terms of memory usage was also evaluated. The detailed results are not presented. But it was generally observed that the designed algorithm can consumes up to five times less memory than Eclat on the *foodmart* and *retail* databases 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.

Overall, from these experiments, it is found that the performance of PFPM can be considered as satisfying in terms of runtime, memory and scalability. Moreover, as shown in the experiments, using PFPM can greatly reduce the number of patterns presented to the user compared to using a traditional frequent pattern mining algorithm such as Eclat.

In this experiments, the parameter have been set empirically. The number of parameters may appear to be large. However, it allows the user to precisely specify the type of patterns to be discovered. Designing an approach to automatically find a good set of parameters based on the type of data can be an opportunity for future work.

Note that, preprocessing can also be used before applying the proposed algorithm to find different types of patterns. For example, consider the task of finding patterns appearing more or less every week-ends in the transaction database of a customer. To find such patterns, one can first eliminate all transactions from weekdays, and merge the transactions of each week-end into a single transaction. Then, the algorithm can be applied to find periodic patterns with an average periodicity of more or less 1 transaction (e.g. $minAvg = 0.5$ and $maxAvg = 1.5$). Then the algorithm will output patterns that appear more or less every week-end. If one wants to be stricter, he can set these parameters to values closer to 1. Similarly, preprocessing can be applied to find patterns appearing more or less every month or on a specific day (e.g. Monday).

6. Conclusion

This paper has proposed a novel problem of mining periodic itemsets using a combination of three measures to avoid drawbacks of traditional periodic itemset mining algorithms, which rely on a single measure. The minimum periodicity and the average periodicity measures have been introduced and their properties have been studied. To efficiently enumerate all periodic itemsets using these measures, an algorithm called Periodic Frequent Pattern Miner was developed. Results from an experimental evaluation on real databases have shown that the designed algorithm is efficient and can find a small set of periodic patterns while filtering many non periodic patterns. The designed algorithm's Java implementation is available under the GPL open-source license in the SPMF open source data mining library [16] http://www.philippe-fournier-viger.com/spmf/.

The research presented in this paper set forward several possibilities for future work. First, one can design more efficient algorithms to discover the desired patterns. Second, one can design alternative algorithms to discover more complex types of periodic patterns [17] or patterns from complex types of data. Third, alternative measures could be investigated for measuring the periodicity of patterns, and quantitative transactions could be considered. We also plan to investigate possible applications related to sensor network data analysis [19], sequential pattern mining [20] and high-utility itemset mining [21].

Acknowledgment

The research of Prof. Fournier-Viger is funded by the National Science Fundation of China. Moreover, au-

thors would like to thank the project support of VSB–TUO for activities with China under the financial support of the Moravian-Silesian Region.

References

[1] AGRAWAL, R., T. IMIELINSKI and A. SWAMI. Mining association rules between sets of items in large databases. In: *International Conference on Management of Data*. Washington: ACM, 1993, pp. 207–216. ISBN 0-89791-592-5. DOI: 10.1145/170035.170072.

[2] PEI, J., J. HAN, H. LU, S. NISHIO, S. TANG and D. YANG. H-mine: fast and space-preserving frequent pattern mining in large databases. *IIE Transactions*. 2007, vol. 39, iss. 6, pp. 593–605. ISSN 0740-817X. DOI: 10.1080/07408170600897460.

[3] MINATO, S., T. UNO and H. ARIMURA. LCM over ZBDDs: Fast generation of very large-scale frequent itemsets using a compact graph-based representation. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Osaka: Springer, 2008, pp. 234–246. ISBN 978-3-540-68124-3. DOI: 10.1007/978-3-540-68125-0_22.

[4] ZAKI, M. J. and K. GOUDA. Fast vertical mining using diffsets. In: *International Conference on Knowledge Discovery and Data Mining*. Washington: ACM, 2003, pp. 326–335. ISBN 1-58113-737-0. DOI: 10.1145/956750.956788.

[5] FOURNIER-VIGER, P., J. C.-W. LIN, B. VO, T. C. TRUONG, J. ZHANG and H. B. LE. A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*. 2017, vol. 7, iss. 4, pp. 1–18. ISSN 1942-4795. DOI: 10.1002/widm.1207.

[6] FONG, A. C. M., B. ZHOU, S. C. HUIM, G. Y. HONG and T. DO. Web content recommender system based on consumer behavior modeling. *IEEE Transactions on Consumer Electronics*. 2011, vol. 57, iss. 2, pp. 962–969. ISSN 0098-3063. DOI: 10.1109/TCE.2011.5955246.

[7] AGGARWAL, C. C. and J. HAN. *Frequent pattern mining*. New York: Springer, 2014. ISBN 978-3-319-07820-5.

[8] AMPHAWAN, K., P. LENCA and A. SURARERKS. Mining top-k periodic-frequent pattern from transactional databases without support threshold. In: *International Conference on Advances in Information Technology*. Bangkok: Springer, 2009, pp. 18–29. ISBN 978-3-642-10391-9. DOI: 10.1007/978-3-642-10392-6_3.

[9] AMPHAWAN, K., A. SURARERKS and P. LENCA. Mining periodic-frequent itemsets with approximate periodicity using interval transaction-ids list tree. In: *International Conference on Knowledge Discovery and Data Mining, WKDD*. Phuket: IEEE, 2010, pp. 245–248. ISBN 978-1-4244-5398-6. DOI: 10.1109/WKDD.2010.126.

[10] KIRAN, R. U. and P. K. REDDY. Towards efficient mining of periodic-frequent patterns in transactional databases. In: *International Conference on Database and Expert Systems Applications*. Bilbao: Springer, 2010, pp. 194–208. ISBN 978-3-642-15250-4. DOI: 10.1007/978-3-642-15251-1_16.

[11] SURANA, A., R. U. KIRAN and P. K. REDDY. An efficient approach to mine periodic-frequent patterns in transactional databases. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Shenzhen: Springer, 2011, pp. 254–266. ISBN 978-3-642-28319-2. DOI: 10.1007/978-3-642-28320-8_22.

[12] KIRAN, R. U., M. KITSUREGAWA and P. K. REDDY. Efficient discovery of periodic-frequent patterns in very large databases. *Journal of Systems and Software*. 2016, vol. 112, iss. 1, pp. 110–121. ISSN 0164-1212. DOI: 10.1080/07408170600897460.

[13] TANBEER, S. K., C. F. AHMED, B.-S. JEONG and Y.-K. LEE. Discovering periodic-frequent patterns in transactional databases. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Bangkok: Springer, 2009, pp. 242–253. ISBN 978-3-642-01306-5. DOI: 10.1007/978-3-642-01307-2_24.

[14] FOURNIER-VIGER, P., Z. LI, J. C.-W. LIN, R. U. KIRAN and H. FUJITA. Discovering Periodic Patterns Common to Multiple Sequences. In: *International Conference on Big Data Analytics and Knowledge Discovery*. Regensburg: Springer, 2018, pp. 231–246. ISBN 978-3-319-98538-1. DOI: 10.1007/978-3-319-98539-8_18.

[15] HAN, J., J. PEI and Y. YIN. Mining Frequent Patterns without Candidate Generation. In: *International Conference on Management of Data*. Dallas: ACM, 2000, pp. 1–12. ISBN 1-58113-217-4. DOI: 10.1145/342009.335372.

[16] FOURNIER-VIGER, P., A. GOMARIZ, T. GUENICHE, A. SOLTANI, C.-W. WU and V. S. TSENG. SPMF: a Java open-source pattern mining library. *Journal of Machine Learning Research*. 2014, vol. 15, iss. 1, pp. 3389–3393. ISSN 1532-4435.

[17] LIN, J. C.-W., W. GAN, P. FOURNIER-VIGER, L. YANG, Q. LIU, J. FRNDA, L. SEVCIK and M. VOZNAK. High utility-itemset mining and privacy-preserving utility mining. *Perspectives in Science*. 2016, vol. 7, iss. 1, pp. 74–80. ISSN 2213-0209. DOI: 10.1016/j.pisc.2015.11.013.

[18] FOURNIER-VIGER, P., J. C.-W. LIN, Q.-H. DUONG, T.-L. DAM., L. SEVCIK, D. UHRIN and M. VOZNAK. PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures. In: *Proceedings of the 2nd Czech-China Scientific Conference 2016*. London: IntechOpen, 2016, pp. 64–79. ISBN 978-953-51-2858-8. DOI: 10.5772/66780.

[19] FAJKUS, M., J. NEDOMA, R. MARTINEK, V. VASINEK, H. NAZERAN and P. SISKA. A Non-Invasive Multichannel Hybrid Fiber-Optic Sensor System for Vital Sign Monitoring. *Sensors*. 2017, vol. 17, iss. 1, pp. 1–17. ISSN 1424-8220. DOI: 10.3390/s17010111.

[20] FOURNIER-VIGER, P., J. C.-W. LIN, R. U. KIRAN, Y. S. KOH and R. THOMAS. A Survey of Sequential Pattern Mining. *Data Science and Pattern Recognition*. 2017, vol. 1, iss. 1, pp. 54–77. ISSN 2520-4165.

[21] FOURNIER-VIGER, P., J. C.-W. LIN, T. TRUONG and R. NKAMBOU. A survey of high utility itemset mining. In: *Studies in Big Data*. Berlin: Springer, 2019, pp. 1–46. ISBN 978-3-030-04920-1.

About Authors

Philippe FOURNIER-VIGER is full professor at the Harbin Institute of Technology (Shenzhen). He has a Ph.D. in Computer Science from the University of Quebec in Montreal. He published more than 200 articles in international conference proceedings and journals. He is editor-in-chief of the Data Mining and Pattern Recognition journal, and director of the Center of Innovative Industrial Design. He is the founder of SPMF, an open-source data mining software, specialized in pattern mining (http://www.philippe-fournier-viger.com/spmf/). SPMF has been cited in 680 research papers since 2010.

Peng YANG is pursuing a master degree in Computer Science at the School of Computer Science and Technology of the Harbin Institute of Technology (Shenzhen), China. His research interest is data mining.

Jerry Chun-Wei LIN is currently working as an Associate Professor at Department of Computing, Mathematics and Physics, Western Norway University of Applied Sciences (HVL), Bergen, Norway. He has published more than 250 research papers in referred journals and international conferences. He is also a project leader of SPMF: An Open-Source Data Mining Library, and also serves as the Editor-in-Chief of the international journal of Data Science and Pattern Recognition.

Quang-Huy DUONG is a Ph.D. student at the Computer Science at the Norwegian University of Science and Technology, Norway. His research interests include data mining, artificial intelligence, and machine learning. He received an M.Sc. in Computer Science from Hunan University, China.

Thu-Lan DAM is a postdoctoral fellow at the Norwegian University of Science and Technology, Norway. Her research interests include data mining, artificial intelligence, and machine learning. She obtained her Ph.D. in Computer Science at Hunan University, China in 2017. She received her Bachelor degree and Master degree in computer science at Hanoi University of Science and Technology, Vietnam in 2004 and 2009, respectively.

Jaroslav FRNDA was born in 1989 in Martin, Slovakia. He received his M.Sc. and Ph.D. from the VSB–Technical University of Ostrava, Department of Telecommunications, in 2013 and 2018 respectively. Now he works as an assistant professor at University of Zilina in Slovakia. His research interests include Quality of Triple play services and IP networks and articial intelligence.

Lukas SEVCIK was born in 1989 in Cadca, Slovakia. He received his M.Sc. in Informatics from the Faculty of Management Science and Informatics, University of Zilina, in 2013 and Ph.D. at the Department of Telecommunications, VSB–Technical University of Ostrava in 2018. His research interests include Quality of Triple play services and IP networks.

Miroslav VOZNAK is a full professor at the Department of Telecommunications, VSB–Technical University of Ostrava. He received his Ph.D. in telecommunications, and his dissertation thesis was entitled "Voice traffic optimization with regard to speech quality in networks with VoIP technology" in 2002. Topics of his research interests are NextGeneration Networks, IP telephony, speech quality and network security.