PFPM: Discovering Periodic Frequent Patterns with Novel Periodicity Measures

Philippe Fournier-Viger, Chun-Wei Lin,
Quang-Huy Duong, Thu-Lan Dam, Lukáš Ševčík,
Dominik Uhrin and Miroslav Voznak

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66780

Abstract

Periodic pattern mining is the task of discovering patterns that periodically appear in transactions. Typically, periodic pattern mining algorithms will discard a pattern as being nonperiodic 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 chapter, 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 data sets shows that the proposed PFPM algorithm is efficient and can filter a huge number of nonperiodic patterns to reveal only the desired periodic patterns.

Keywords: frequent pattern mining, periodic patterns, periodicity measures

1. Introduction

In the field of data mining, frequent itemset mining (FIM) [1–3] 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. Although numerous algorithms have been proposed for FIM [1–3], 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.

Several algorithms have been proposed to discover periodic frequent patterns (PFP) [4–9] in a transaction database (a sequence of transactions). Typically, periodic pattern mining algorithms will discard a pattern as being nonperiodic 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 chapter, 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 chapter 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 data sets 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 nonperiodic patterns to reveal only the desired periodic itemsets. The rest of this chapter is organized as follows. Sections 2, 3, 4, 5, and 6, 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. 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 $y \succ i$, $\forall i \in X$. 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 $\{a\}$, $\{a,c\}$, $\{e\}$, $\{c,e\}$, and $\{c\}$, having respectively a support of 4, 4, 5, 4, and 6.

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

Table 1. A transaction database.

To discover frequent itemsets, various algorithms have been proposed such as Apriori [1], FP-Growth [10], LCM [2], and Eclat [3]. 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 [4–9]. Several applications of mining periodic frequent patterns have been reported in previous work [9].

A periodic frequent pattern is defined as follows [9]. 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 (including T_x). The *periods of an itemset* X is a list of periods defined as $ps(X) = \{g_1 - g_0, g_2 - g_1, g_3 - g_2, \cdots g_k - g_{k-1}, g_{k+1} - g_k\}$, where g_0 and g_{k+1} are constants defined as $g_0 = 0$ and $g_{k+1} = n$. Thus, $ps(X) = \cup_{1 \leq z \leq k+1}(g_z - g_{z-1})$. 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 $ps(\{a,c\}) = \{1, 2, 2, 1, 1\}$. The maximum periodicity of an itemset X is defined as $maxper(X) = max(ps(X))$ [9]. An itemset X is a periodic frequent pattern (PFP) if $|g(X)| \geq minsup$ and $maxper(X) \leq maxPer$, where *minsup* and *maxPer* are user-defined thresholds [9].

The PFP-tree algorithm is the first algorithm that has been proposed for mining PFPs [9]. It utilizes a tree-based and pattern-growth approach for discovering PFPs, inspired by the FP-Growth algorithm [10]. Thereafter, an algorithm called MTKPP [4] 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 developed. Ref. [5] mines PFPs by considering an approximation of the periodicities of patterns. Another approximate algorithm for PFP mining was recently proposed [8]. Other extensions of the PF-Tree algorithm named MIS-PF-tree [6] and MaxCPF [7] 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, this chapter proposes the concept of *average periodicity.*

Definition 1 (Average periodicity of an itemset): *The average periodicity of an itemset X is defined as* $avgper(X) = \sum_{g \in ps(X)} / |ps(X)|$.

For instance, the periods of the itemsets $\{a, c\}$ and $\{e\}$ are $ps(\{a,c\}) = \{1, 2, 2, 1, 1\}$ and $ps(\{e\}) = \{2, 1, 1, 2, 1, 0\}$. Thus, their average periodicities are respectively $avgper(\{a,c\}) = 1.4$ and $avgper(\{e\}) = 1.17$.

Lemma 1 (An alternative definition of the average periodicity): *Consider an itemset X. The average periodicity can also be calculated as* $avgper(X) = |D|/(|g(X)| + 1)$.

The proof for that proposed lemma is omitted due to space limitations. This lemma is interesting because it shows that there is a relationship between the average periodicity and the support measure. Moreover, this lemma gives a second method for calculating the average periodicities of itemsets. An interesting observation is that if the term $|D|$ is precalculated, calculating the average periodicity of any itemset only requires obtaining $|g(X)| + 1$, to then divide it by $|D|$. Using this method, average periodicities are calculated more efficiently than using Definition 1.

The average periodicity is a very interesting measure since it measures the average period length of an itemset. However, this measure should always be considered with other measure. The reason is that the average periodicity does not consider if the period lengths vary widely. We illustrate this with an example. In the running example, the average periodicity of $\{b, d\}$ is 2.33. This itemset may seem like a periodic pattern. However, this is not the case. $\{b, d\}$ actually only appears in T_3 and T_4, and its periods $ps(\{T_3, T_4\}) = \{3, 1, 4\}$ vary widely. To ensure that such patterns having periods that vary too much are discovered, we propose to use the average periodicity in combination with other periodicity measure(s), and in particular the following two measures.

The first measure is the *minimum periodicity* and it is used to filter out itemsets having short periods. Let there be an itemset X. The minimum periodicity is defined as $minper(X) = min(ps(X))$. However, a problem with this measure is that the first and last periods of an itemset are respectively equal to 1 or 0 if the itemset respectively appears in the first or the last transactions of the database. For example, because itemset {e} occurs in T_7, its last period is 0, and thus $minper(\{e\}) = 0$. To avoid this problem, we have decided to change the definition of the minimum periodicity of an itemset so that it is calculated by excluding the first and last periods of the itemset. By ignoring the first and last periods, the set of periods may however become empty. In this case, we define the minimum periodicity as ∞.

The second measure that we consider is the *maximum periodicity,* to avoid finding period patterns that do not occur for very long periods of time. This measure is defined as in Section 2 and is denoted as $maxper(X)$ for an itemset X.

In the following, we consider the minimum periodicity, maximum periodicity, and average periodicity measures to discover frequent periodic patterns as they let the user finely specify the type of periodic patterns to be found. But another reason for choosing these measures is that an algorithm can calculate them very efficiently. Consider an itemset X. The three measures can be calculated by browsing the list of transactions $g(X)$ only once, without storing the periods $ps(X)$ in memory. This chapter defines the problem of periodic frequent itemsets using the three measures as follows.

Definition 2 (Periodic frequent itemsets with novel measures): *An itemset X is said to be a periodic frequent itemset if and only if* $minAvg \leq avgper(X) \leq maxAvg$, $minper(X) \geq minPer$, and

$maxper(X) \leq maxPer$, where $minAvg$, $maxAvg$, $minPer$, and $maxPer$ are thresholds (positive numbers), set by the user.

As an example, assume that $minPer = 1$, $maxPer = 3$, $minAvg = 1$, and $maxAvg = 2$. For these parameters, 11 PFPs are discovered (illustrated in **Table 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

Table 2. The set of PFPs for the running example.

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. Proofs are omitted due to space limitations.

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

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

Lemma 4 (Monotonicity of the maximum periodicity): *Let X and Y be itemsets such that* $X \subset Y$. It follows that $maxper(Y) \geq maxper(X)$ [9].

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. (This follows from Lemma 4.)

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.

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 [3]. 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 Theorems 1 and 2. Items in I^* are then sorted according to the order $\succ$ of ascending support values, as suggested in [3]. 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*, and $|D|$.

```
Algorithm 1: The PFPM algorithm
   input  : a transaction database $D$, $minAvg$, $maxAvg$, $minPer$ and $maxPer$
   output: the set of periodic 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 ( $I^*$, $\gamma$, $minAvg$, $minPer$, $maxPer$, $|D|$);
```

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 *maxPer*, it means that extensions of Px should be explored (by Theorems 1 and 2). This is performed by merging Px with all extensions Py of P

such that *y*>*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 tid-list described in the Eclat algorithm [3], 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 Theorems 1 and 2. Thus, it can be easily seen that this procedure is correct and complete to discover all PFPs.

Algorithm 2: The *PFPMSearch* procedure

```
input  : $ExtensionsOfP$: a set of extensions of an itemset $P$, $\gamma$, $minAvg$, $minPer$, $maxPer$, $|D|$
output: the set of periodic frequent itemsets
1  foreach itemset $Px \in ExtensionsOfP$ do
2      $avgperPx \leftarrow |D|/(|Px.tidlist| + 1)$;
3      if $minAvg \leq avgperPx \leq maxAvg \wedge Px.tidlist.minp \geq minPer \wedge Px.tidlist.maxp \leq maxPer$
        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
```

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      $period_v \leftarrow calculatePeriod(v, \mathrm{TidListOfPxy})$;
4      $UpdateMinPerMaxPer(TidListOfPxy, period_v)$;
5      $TidListOfPxy \leftarrow TidListOfPxy \cup \{v\}$;
6  end
7  return $TidListOfPxy$;
```

In the implementation of PFPM, two additional optimizations are included. The first optimization is to calculate the support of all pairs of items {*x*, *y*} occurring in the database during the first database scan, and store it in memory. Then, Line 7 of the search procedure is modified to prune itemset *Pxy* if $s(\{x, y\})$ is less than γ by Theorem 2. The second optimization is in the *Intersection* procedure. During the loop, if it is found that *Pxy* cannot have no more than $\gamma = (|D|/maxAvg)-1$ transactions, the loop can be stopped.

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 data sets were utilized in the experiment, which are commonly used in the FIM literature. The *retail, chainstore*, and *foodmart* data sets contain anonymized customer transactions from retail stores, while the *mushroom* data set is a dense benchmark data set. The data sets have been chosen because they represent the main types of data encountered in real-life scenarios (dense, sparse, and long transactions). Let $|I|$, $|D|$, and A represent the number of transactions, distinct items, and average transaction length of a data set. Characteristics of the four data sets are *retail* ($|I| = 88,162, |D| = 16,470, A = 10.30$), *mushroom* ($|I| = 8,124, |D| = 119, A = 23.0$), *chainstore* ($|I| = 1,112,949, |D| = 46,086, A = 7.26$), and *foodmart* ($|I| = 4,141$, $|D| = 1,559, A = 4.4$). The experiment consisted of running the PFPM algorithm on each data set 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 data set, values for the periodicity thresholds have been found empirically for each data set (as they are data set 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 1** compares the execution times of PFPM for various parameter values and Eclat. **Figure 2** 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* data set, PFPM is up to four times faster than Eclat. On the *mushroom* and *chainstore* data sets, no results are shown for Eclat because it cannot terminate within 1000 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 data sets when the minimum support is set to γ. The PFPM algorithm still terminates on these data sets because it only searches for periodic patterns, and thus prunes a large part of the search space containing nonperiodic patterns. On the *foodmart* data set, 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 data set 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 **Figure 2**). This demonstrates that a huge amount of nonperiodic patterns are found in real-life data sets. 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. Memory consumption was also compared, although detailed results are not shown. It was observed that

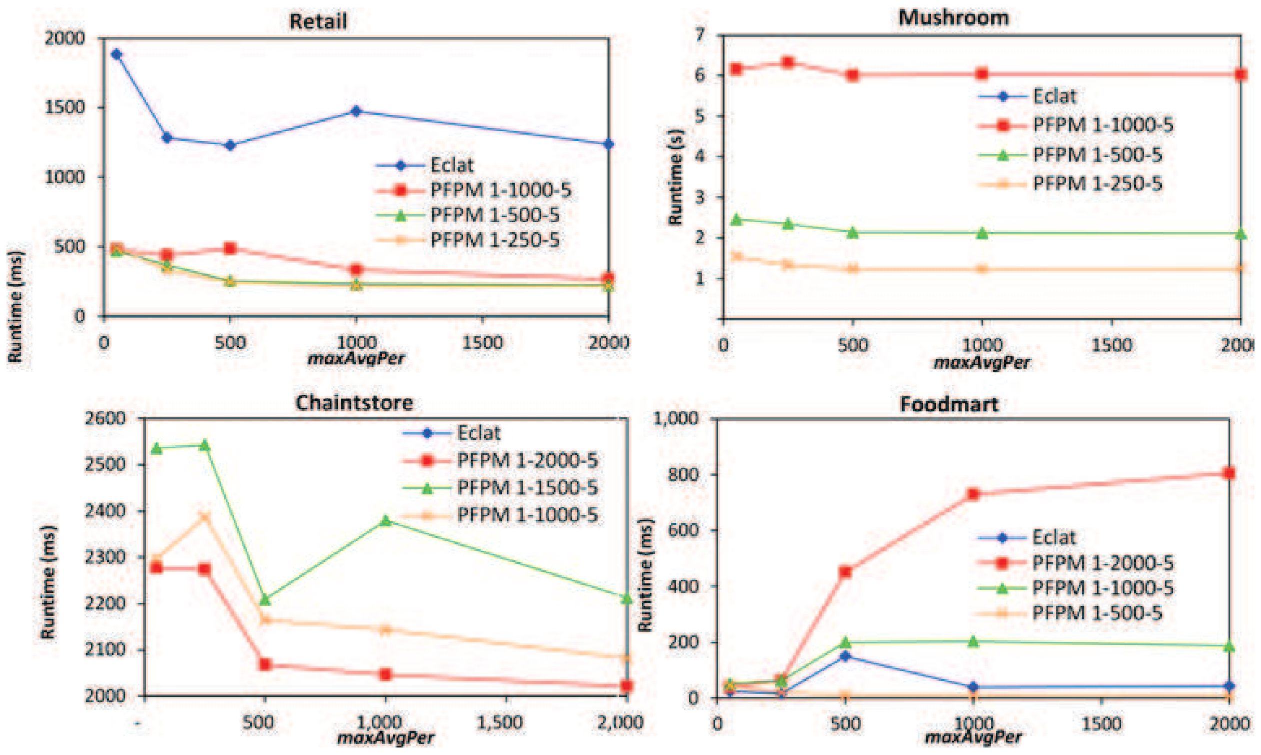

Figure 1. Execution times.

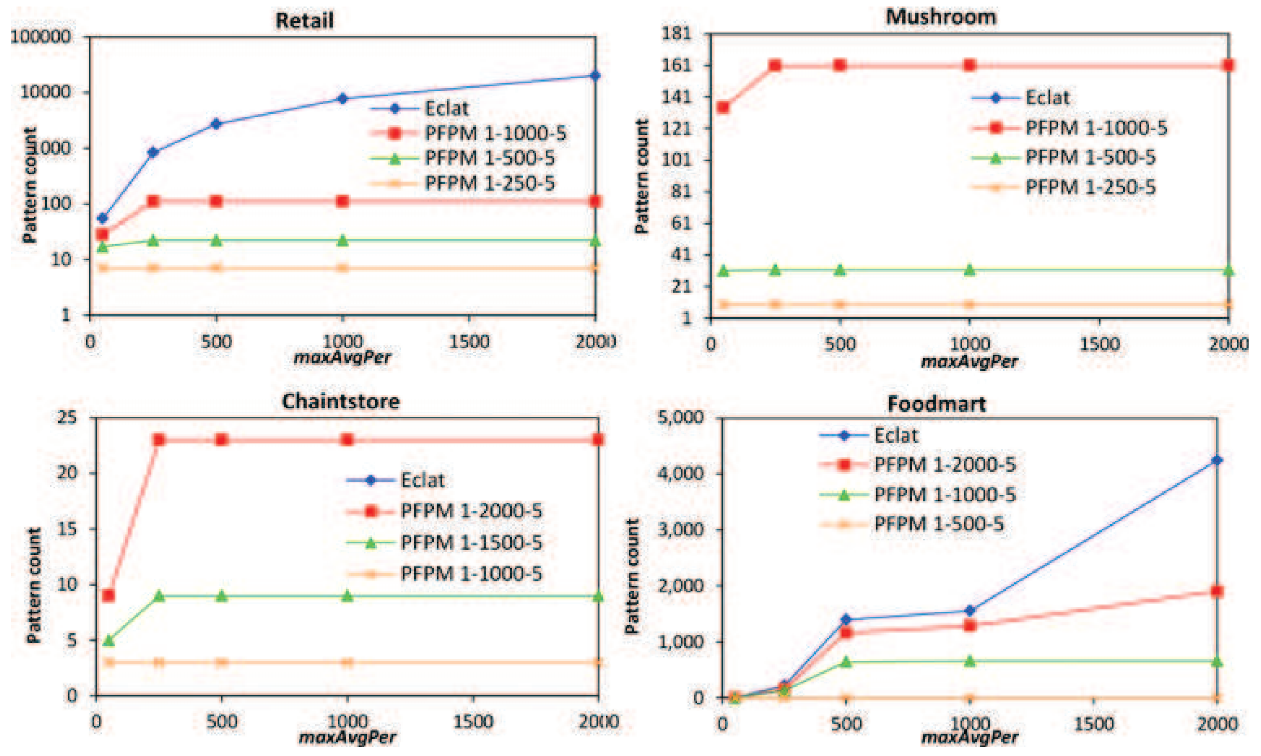

Figure 2. Number of patterns found.

PFPM use up to four and five times less memory than Eclat on the *retail* and *foodmart* data sets, depending on parameter values. For example, on *retail* and $maxAvg = 2,000$, Eclat and PFPM 1-5000-5-500 respectively consumes 900 and 189 MB of memory.

6. Conclusion

In this chapter, an efficient algorithm named PFPM (Periodic Frequent Pattern Miner) was proposed to efficiently discover all frequent periodic patterns using three periodicity measures (the minimum, average, and maximum periodicity). An experimental evaluation on real data sets shows that the proposed PFPM algorithm is efficient and can filter a huge number of nonperiodic patterns to reveal only the desired periodic patterns. Source code and data sets will be made available as part of the SPMF open source data mining library [11] http://www.philippe-fournier-viger.com/spmf/. For future work, we are working on mining correlated periodic patterns [12], and periodic high-utility itemsets [13].

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.

Author details

Philippe Fournier-Viger[1]*, Chun-Wei Lin[2], Quang-Huy Duong[3], Thu-Lan Dam[3,4], Lukáš Ševčík[5], Dominik Uhrin[5] and Miroslav Voznak[5]

*Address all correspondence to: philfv8@yahoo.com

1 School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China

2 School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China

3 College of Computer Science and Electronic Engineering, Hunan University, China

4 Faculty of Information Technology, Hanoi University of Industry, Hanoi, Vietnam

5 Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava-Poruba, 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) 207–216.

[2] S.-i. Minato, T. Uno, H. Arimura, LCM over ZBDDS: Fast generation of very large-scale frequent item-sets using a compact graph-based representation, in: Proc. Twelfth Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Osaka, Japan, May 20-23, Springer, 2008, pp. 234–246.

[3] M. J. Zaki, K. Gouda, Fast vertical mining using diffsets, in: Proc. ninth ACM SIGKDD Intern. Conf. Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, ACM, 2003, pp. 326–335.

[4] K. Amphawan, P. Lenca, A. Surarerks, Mining top-k periodic-frequent pattern from transactional databases without support threshold, in: Proc. Third Intern. Conf. Advances in Information Technology, Bangkok, Thailand, December 1-5, Springer, 2009, pp. 18–29.

[5] 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.

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

[7] A. Surana, R. U. Kiran, P. K. Reddy, An efficient approach to mine periodic-frequent patterns in transactional databases, in: Proc. 15th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Shenzhen, China, May 24-27, Springer, 2011, pp. 254–266.

[8] 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.

[9] S. K. Tanbeer, C. F. Ahmed, B.-S. Jeong, Y.-K. Lee, Discovering periodic-frequent patterns in transactional databases, in: Proc. Eleventh Pacific-Asia Conf. on Knowledge Discovery and Data Mining, Bangkok, Thailand, April 27-30, Springer, 2009, pp. 242–253.

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

[11] 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) 15 (2014) 3389–3393.

[12] P. Fournier-Viger, C.-W. Lin, T. Dinh, H. B. Le, Mining correlated high-utility itemsets using the bond measure, in: Proc. 11th Intern. Conf. Hybrid Artificial Intelligent Systems, Seville, Spain, April 18-20, 2016, pp. 53–65.

[13] P. Fournier-Viger, C.-W. Lin, Q.-H. Duong, T.-L. Dam, Phm: Mining periodic high-utility itemsets, in: Proc. 16th Industrial Conf. on Data Mining, Newark, USA, July 13–17, 2016.