Discovering Periodic-Frequent Patterns in Transactional Databases Using All-Confidence and Periodic-All-Confidence

J.N. Venkatesh[1(✉)], R. Uday Kiran[2], P. Krishna Reddy[1], and Masaru Kitsuregawa[2,3]

[1] Kohli Center on Intelligent Systems (KCIS), International Institute of Information Technology Hyderabad, Hyderabad, India
jn.venkatesh@research.iiit.ac.in,pkreddy@iiit.ac.in
[2] Institute of Industrial Science, The University of Tokyo, Tokyo, Japan
{uday_rage,kitsure}@tkl.iis.u-tokyo.ac.jp
[3] National Institute of Informatics, Tokyo, Japan

Abstract. Periodic-frequent pattern mining involves finding all frequent patterns that have occurred at regular intervals in a transactional database. The basic model considers a pattern as periodic-frequent, if it satisfies the user-specified minimum support ($minSup$) and maximum periodicity ($maxPer$) constraints. The usage of a single $minSup$ and $maxPer$ for an entire database leads to the *rare-item problem*. When confronted with this problem in real-world applications, researchers have tried to address it using the item-specific $minSup$ and $maxPer$ constraints. It was observed that this extended model still generates a significant number of uninteresting patterns, and moreover, suffers from the issue of specifying item-specific $minSup$ and $maxPer$ constraints. This paper proposes a novel model to address the rare-item problem in periodic-frequent pattern mining. The proposed model considers a pattern as interesting if its *support* and *periodicity* are close to that of its individual items. The *all-confidence* is used as an interestingness measure to filter out uninteresting patterns in *support* dimension. In addition, a new interestingness measure, called *periodic-all-confidence*, is being proposed to filter out uninteresting patterns in *periodicity* dimension. We have proposed a model by combining both measures and proposed a pattern-growth approach to resolve the rare-item problem and extract interesting periodic-frequent patterns. Experimental results show that the proposed model is efficient.

Keywords: Data mining · Rare-item problem · Periodic patterns

1 Introduction

Periodic-frequent pattern mining is an important model in data mining. It involves finding all frequent patterns that are occurring at regular intervals in a transactional database. The periodic-frequent patterns provide useful

S. Hartmann and H. Ma (Eds.): DEXA 2016, Part I, LNCS 9827, pp. 55–70, 2016.
DOI: 10.1007/978-3-319-44403-1_4

Table 1. Running example: a transactional database

tid	Items	tid	Items
1	$a,\ b$	7	$a,\ b,\ c,\ e$
2	$a,\ b,\ d$	8	$c,\ d$
3	$c,\ d,\ g$	9	$c,\ d$
4	$c,\ e,\ f$	10	$a,\ b,\ e,\ f$
5	$a,\ b$	11	$c,\ d,\ g$
6	h	12	$a,\ e,\ f$

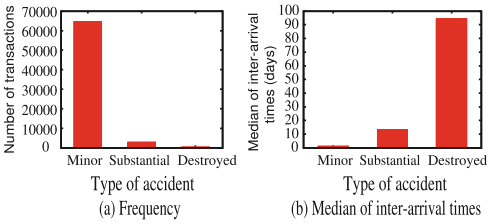

Fig. 1. Statistics on different damage types in FAA data set.

information in many real-world applications. Examples include finding regularities in body sensor networks [14], intrusion detection in computer networks [8], and finding minor events in twitter data [7].

The basic model of periodic-frequent patterns [13] is as follows. Let I be the set of items, and $X \subseteq I$ be a **pattern** (or an itemset). A pattern containing β number of items is called a β**-pattern**. A **transaction**, $t_k = (tid, Y)$ is a tuple, where $tid \in \mathbb{R}$ represents the transaction-id (or timestamp) at which the pattern Y has occurred. A **transactional database** TDB over I is a set of transactions, $TDB = \{t_1, \cdots, t_m\}$, $m = |TDB|$, where $|TDB|$ can be defined as the number of transactions in TDB. For a transaction $t_k = (tid, Y)$, $k \geq 1$, such that $X \subseteq Y$, it is said that X occurs in t_k and such transaction-id is denoted as tid^X. Let $TID^X = \{tid_j^X, \cdots, tid_k^X\}$, $j, k \in [1, m]$ and $j \leq k$, be an **ordered set of transaction-ids** where X has occurred in TDB. In this paper, we call this list of transaction-ids of X as **tid-list** of X. The number of transactions containing X in TDB is defined as the **support** of X and denoted as $sup(X)$. That is, $sup(X) = |TID^X|$. Let tid_q^X and tid_r^X, $j \leq q < r \leq k$, be the two consecutive transaction-ids in TID^X. The time difference (or an inter-arrival time) between tid_r^X and tid_q^X is defined as a **period** of X, say p_a^X. That is, $p_a^X = tid_r^X - tid_q^X$. Let $P^X = (p_1^X, p_2^X, \cdots, p_r^X)$ be the set of all *periods* for a pattern X. The **periodicity** of X denoted as $per(X) = max(p_1^X, p_2^X, \cdots, p_r^X)$. The pattern X is a **frequent pattern** if $sup(X) \geq minSup$, where $minSup$ refers to the user-specified *minimum support* constraint. The frequent pattern X is said to be **periodic-frequent** if $per(X) \leq maxPer$, where $maxPer$ refers to the user-specified *maximum periodicity* constraint. The **problem definition** of periodic-frequent pattern mining involves discovering all patterns in TDB that satisfy the user-specified $minSup$ and $maxPer$ constraints.

Example 1. Table 1 shows the transactional database with the set of items $I = \{a, b, c, d, e, f, g, h\}$. The set of items '$a$' and '$b$,' i.e., '$ab$' is a pattern. This pattern contains only two items. Therefore, this is a 2-pattern. In the first transaction, $t_1 = \{1 : ab\}$, 1 denotes the transaction-id at which the pattern 'ab' has appeared in the database. In the entire database, this pattern appears at the transaction-ids of 1, 2, 5, 7 and 10. Therefore, $TID^{ab} = \{1, 2, 5, 7, 10\}$. The *support* of '$ab$,' i.e., $sup(ab) = |TID^{ab}| = |1, 2, 5, 7, 10| = 5$. If the user-specified $minSup = 5$, then

'ab' is a frequent pattern as $sup(ab) \geq minSup$. The periods for this pattern are: $p_1^{ab} = 1\ (= 1 - tid_{ini})$, $p_2^{ab} = 1\ (= 2 - 1)$, $p_3^{ab} = 3\ (= 5 - 2)$, $p_4^{ab} = 2\ (= 7 - 5)$, $p_5^{ab} = 3\ (10 - 7)$ and $p_6^{ab} = 2\ (= tid_{fin} - 10)$, where $tid_{ini} = 0$ represents the transaction-id of initial transaction and $tid_{fin} = 12$ represents the transaction-id of final transaction in the database. Therefore, $P^{ab} = (1, 1, 3, 2, 3, 2)$. The *periodicity* of '$ab$,' i.e., $per(ab) = max(1, 1, 3, 2, 3, 2) = 3$. If the user-defined $maxPer = 3$, then the frequent pattern 'ab' is a periodic-frequent pattern because $per(ab) \leq maxPer$.

The *support* and *periodicity* are two dimensions to determine the interestingness of a periodic-frequent pattern. The constraints, $minSup$ and $maxPer$, determine the interestingness of a pattern with respect to these two dimensions. Since only a single $minSup$ and $maxPer$ is used for the whole database, the model works efficiently in the databases in which all the items have uniform *support* and similar *periodicity*. However, this is often not the case as items are non-uniformly distributed in many real-world databases. That is, some items appear very frequently in the data, while others rarely appear. Moreover, rare items typically have longer *periods* (or inter-arrival times) as compared with the *periods* of the frequent items.

Example 2. Consider a case of accident database in which reports related to the completely *destroyed* vehicles do not appear as frequently as the reports related to the *minor* damages to a vehicle. As a result, former type of accidents generally have less frequency and longer inter-arrival times as compared against the latter type of accidents. The same can be observed from Figs. 1(a) and (b), which respectively show the $frequency$ and $median$ of inter-arrival times of three different damage types reported in the Federal Aviation Administration (FAA) database [1]. For a domain expert, any information pertaining to *destroyed* aircrafts may be found useful as it includes materialistic and/or human loss.

Henceforth, finding periodic-frequent patterns with a single $minSup$ and $maxPer$ leads to the following problems:

- If the $minSup$ is set too high and/or the $maxPer$ is set too short, we will miss the interesting periodic-frequent patterns involving rare items.
- In order to find the interesting periodic-frequent patterns involving rare items, we have to set a low $minSup$ and a long $maxPer$. However, this may result in combinatorial explosion producing too many patterns, because frequent items can combine with one another in all possible ways and many of them will be meaningless.

This dilemma is known as the *rare-item problem* [15] (refer Example 3). In this paper, we make an effort to address this problem in periodic-frequent pattern mining.

Example 3. Consider the rare items 'e' and 'f' in Table 1. If we set a high $minSup$ and a short $maxPer$, say $minSup = 5$ and $maxPer = 3$, we will miss the periodic-frequent patterns containing these rare items. In order to discover

the periodic-frequent patterns containing these rare items, we have to set a low *minSup* and a long *maxPer*, say $minSup = 2$ and $maxPer = 6$. All periodic-frequent patterns discovered at these threshold values are shown in the column titled **I** in Table 2. It can be observed from this table that setting a low *minSup* and a long *maxPer* has not only resulted in finding 'ef' as a periodic-frequent pattern, but also resulted in generating the uninteresting patterns '*ce*' and '*cd*' as periodic-frequent patterns. The pattern '*ce*' is uninteresting (with respect to *support* dimension), because the rare item '*e*' is randomly occurring with a frequent item '*c*' in very few transactions. The pattern '*cd*' is uninteresting (with respect to *periodicity* dimension), because it contains the frequent items '*c*' and '*d*' appearing together at very long inter-arrival times (or *periodicity*).

Table 2. Periodic-frequent patterns discovered from Table 1. The terms *Pat, s, allConf, p* and *perAllConf* refer to *pattern*, *support*, *all-confidence*, *periodicity* and *periodic-all-confidence*, respectively. The columns titled **I**, **II** and **III** represent the periodic-frequent patterns generated using basic model, extending *all-confidence* to the basic model and the proposed model, respectively.

Pat	s	allConf	p	perAllConf	I	II	III
a	6	1	3	1	✓	✓	✓
b	5	1	3	1	✓	✓	✓
c	6	1	3	1	✓	✓	✓
d	5	1	5	1	✓	✓	✓
e	4	1	4	1	✓	✓	✓

Pat	s	allConf	p	perAllConf	I	II	III
f	3	1	6	1	✓	✓	✓
ab	5	0.833	3	1	✓	✓	✓
ef	3	0.75	6	1.5	✓	✓	✓
ce	2	0.4	5	1.67	✓	×	×
cd	4	0.8	5	1.67	✓	✓	×

In this paper, we propose a novel model to extract the interesting periodic-frequent patterns by addressing the *rare-item problem*. We consider a pattern as interesting if it satisfies the following two conditions: (*i*) if the *support* of a pattern is close to the *support* of its individual items, and (*ii*) if the *periodicity* of a pattern is close to the *periodicity* of its individual items. For this, we employ two measures. To filter out patterns based on *support* and resolve rare-item problem in *support* dimension, we employ *all-confidence* [9]. Similarly, to filter out patterns based on *periodicity* and resolve rare-item problem in *periodicity* dimension, we propose a new measure called *periodic-all-confidence*. These two measures facilitate us to achieve the objective of generating interesting periodic-frequent patterns involving rare items without causing the generation of too many uninteresting patterns. A pattern-growth algorithm, Extended Periodic-Frequent pattern-growth (EPF-growth), has also been proposed to extract all interesting periodic-frequent patterns. Experimental results demonstrate that the proposed model can discover useful information and is efficient as compared to the existing approaches.

The rest of the paper is organized as follows. Section 2 describes the related work of finding periodic-frequent patterns in a transactional database. Section 3 introduces the extended model of periodic-frequent patterns. Section 4 describes the EPF-growth algorithm. Section 5 reports on the experimental results. Finally, Sect. 6 concludes the paper with future research directions.

2 Related Work

The problem of finding periodic patterns has been widely studied in time series data [3,17]. These studies consider time series data as a symbolic sequence, and therefore, do not take into account the actual temporal information of the items within the data. Ozden et al. [10] have enhanced the transactional database by a time attribute that describes the time when a transaction has appeared and investigated the periodic behavior of the patterns to discover cyclic association rules. In this study, a database needs to be fragmented into non-overlapping subsets with respect to time. Henceforth, the drawback of this study is that patterns (or association rules) that span multiple windows cannot be discovered.

Tanbeer et al. [13] have proposed a simplified periodic-frequent model, which does not require data fragmentation. This model implicitly assumes all items occur uniformly in the data, and henceforth, suffer from *rare-item problem*.

Kiran et al. [6] have tried to address the *rare-item problem* by finding periodic-frequent patterns using multiple $minSup$ and $maxPer$ values. In that model, every item in the database is specified with the *minimum item support* ($minIS$) and the *maximum item periodicity* ($maxIP$). Next, the $minSup$ and $maxPer$ of a pattern X are specified as follows:

$$minSup(X) = min(minIS(i_j)|\forall i_j \in X) \tag{1}$$

$$maxPer(X) = max(maxIP(i_j)|\forall i_j \in X) \tag{2}$$

where, $minIS(i_j)$ and $maxIP(i_j)$ represent the *minimum item support* and *maximum item periodicity* of an item $i_j \in X$. A pattern-growth algorithm, MCPF-growth has been discussed to find the patterns. The periodic-frequent patterns discovered by that model do not satisfy the *anti-monotonic property*. That is, all non-empty subsets of a periodic-frequent pattern may not be periodic-frequent patterns. Henceforth, MCPF-growth is computationally expensive or impracticable in very large real-world databases.

Surana et al. [11] have proposed an alternative model to address the rare-item problem. In that model, the $minSup$ and $maxPer$ of a pattern are specified as:

$$minSup(X) = max(minIS(i_j)|\forall i_j \in X) \tag{3}$$

$$maxPer(X) = min(maxIP(i_j)|\forall i_j \in X) \tag{4}$$

A pattern-growth algorithm, MaxCPF-growth has been discussed to find the patterns. The periodic-frequent patterns discovered by that model satisfy the *anti-monotonic property*. Therefore, MaxCPF-growth is computationally inexpensive than MCPF-growth [6], and practicable in very large real-world databases.

The limitations of these two approaches and the proposed model are discussed in next section.

3 Extended Model of Periodic-Frequent Patterns

3.1 Limitations of Existing Approaches

An open problem that is common to above two studies [6,11] is the methodology to specify items' *minIS* and *maxIP* values. Kiran et al. [6] have described the following methodology to address this problem:

$$\begin{aligned} minIS(i_j) &= max(\gamma \times S(i_j),\ LS) \\ &\quad and \\ maxIP(i_j) &= max(\beta \times S(i_j) + Per_{max},\ Per_{min}) \end{aligned} \tag{5}$$

where $i \in I$ and $S(i)$ is the *support* of the item i. In Eq. 5, LS is the user-specified lowest *minimum item support* allowed and $\gamma \in [0, 1]$ is a parameter that controls how the $minIS$ values for items should be related to their *supports*. In Eq. 5, Per_{max} and Per_{min} are the user-specified maximum and minimum *periodicities* such that $Per_{max} \geq Per_{min}$ and $\beta \in [-1, 0]$ is a user-specified constant.

Although Eq. 5 facilitates every item to have different $minIS$ and $maxIP$ values, it suffers from the following limitations: (i) This methodology requires several input parameters from the user. (ii) Equation 5 determines the $maxIP$ of an item by taking into account only its *support*. As a result, this equation implicitly assumes that all items having the same *support* will also have similar *periodicities* in a transactional database. However, this is seldom the case as items with similar *support* can have different *periodicities*. We have observed that employing this methodology to specify items' $maxIP$ values in the transactional databases, where items can have similar *support* but different *periodicities* can still lead to the *rare-item problem*.

Example 4. Consider a hypothetical transactional database containing 100 transactions. Let 'x' and 'y' be two items in the database having the same *support* (say, $sup(x) = sup(y) = 40$), but different *periodicities* (say, $per(x) = 11$ and $per(y) = 30$). Since Eq. 5 determines the $maxIP$ values by taking into account only the *support* of the items, both 'x' and 'y' will be assigned a common $maxIP$ value although their actual *periodicity* is different from one another. This can result either in missing interesting patterns or generating too many patterns. For instance, if we set $\beta = -0.5$, $Per_{min} = 10$ and $Per_{max} = 50$, then $maxIP(x) = maxIP(y) = 20$. In this case, we miss the periodic-frequent patterns containing 'y' because $per(y) \not\leq maxIP(y)$. In order to find the periodic-frequent patterns containing both 'x' and 'y' items, we have to set a high β value. When β is set at -0.375, we derive $maxIP(x) = maxIP(y) = 35$. In this case, we find periodic-frequent patterns containing 'y' because $per(y) \leq maxIP(y)$. However, we may also witness too many patterns containing the item 'x' because its $maxIP$ value is three times higher than its *periodicity*.

We now describe the proposed model that do not suffer from this problem.

3.2 Proposed Model

To address the *rare-item problem*, we need an approach that extracts interesting periodic-frequent patterns involving both frequent and rare items yet filtering out uninteresting patterns. After conducting the initial investigation on the nature of interesting patterns found in various databases, we have made a key observation that most of the interesting periodic-frequent patterns discovered in a database have their *support* and *periodicity* close to that of its individual items. The following example illustrates our observation.

Example 5. In a supermarket, cheap and perishable goods (e.g., bread and butter) are purchased more frequently and periodically than the costly and durable goods (e.g., bed and pillow). Among all the possible combinations of the above four items, we normally consider {bread, butter} and {bed, pillow} as interesting patterns, because only these two patterns generally have *support* and *periodicity* close to the *support* and *periodicity* of its individual items. All other uninteresting patterns, {bread, bed}, {bread, pillow}, {butter, bed} and {butter, pillow}, generally have *support* and *periodicity* relatively far away from the *support* and *periodicity* of its individual items as compared against the above two patterns.

Henceforth, in this paper we consider a pattern as interesting if its *support* and *periodicity* are close to the *support* and *periodicity* of its individual items. In this context, we need two measures to determine the interestingness of a pattern with respect to both *support* and *periodicity* dimensions.

In the literature, researchers have discussed several measures to address the *rare-item problem* in *support* dimension [12,16]. Each measure has a selection bias that justifies the significance of a knowledge pattern. As a result, there exists no universally acceptable best measure to judge the interestingness of a pattern for any given database. In this paper, we use *all-confidence* to address the *rare-item problem* in frequency dimension. The reasons for choosing this measure over other measures are as follows: (*i*) The *all-confidence* assesses the interestingness of a pattern by determining how close is its *support* with respect to the *support* of all of its items in a database. (*ii*) The *all-confidence* satisfies the *anti-monotonic property* [9]. This property plays a key role in the practicable ability of our model. (*iii*) The *all-confidence* satisfies the *null-invariance property* [5]. This property facilitate us to discover genuine interesting patterns without being influenced by the item co-absence in the database.

Continuing with the basic model of periodic-frequent patterns (discussed in Sect. 1), the proposed model is as follows.

Definition 1. ***(All-confidence of X)*** *The all-confidence of X, denoted as $allConf(X)$, is the ratio of support of X to the maximal support of an item $i_j \in X$. That is, $allConf(X) = \frac{sup(X)}{max(sup(i_j)|\forall i_j \in X)}$.*

For a pattern X, $allConf(X) \in (0, 1]$. As per the *all-confidence* measure, a pattern is interesting in *support* dimension if its *support* is close to the *support* of all of its items. The parameter *minAllConf* indicates the user-specified minimum *all-confidence* threshold value. Based on *minSup* and *minAllConf* thresholds, all the interesting patterns involving rare items in *support* dimension are extracted.

The usage of *all-confidence* alone is insufficient to completely address the *rare-item problem* in periodic-frequent pattern mining. The reason is this measure does not take into account the *periodicity* dimension of a pattern.

Example 6. The column titled **II** in Table 2 shows the periodic-frequent patterns discovered when *all-confidence* is used along with *support* and *periodicity* measures. *The minSup, minAllConf and maxPer values used to find these patterns are 2, 0.6 and 6, respectively.* It can be observed from the discovered periodic-frequent patterns that though *all-confidence* is able to prune the uninteresting pattern '*ce*,' it has failed to prune another uninteresting pattern '*cd*' from the list of periodic-frequent patterns generated by the basic model. Henceforth, the *rare-item problem* has to be addressed with respect to both *support* and *periodicity* dimensions.

As there exists no measure in the literature that determines the interestingness of a pattern with respect to the *periodicities* of all of its items, we propose a new measure, ***periodic-all-confidence***, to extract interesting patterns in *periodicity* dimension involving rare items, which is defined as follows.

Definition 2. *(**Periodic-all-confidence of X**) The periodic-all-confidence of X, denoted as perAllConf(X), is the ratio of periodicity of X to the minimal periodicity of an item* $i_j \in X$. *That is,* $perAllConf(X) = \frac{per(X)}{min(per(i_j)|\forall i_j \in X)}$.

For a pattern X, $perConf(X) \in [1, \infty)$. As per the *periodic-all-confidence* measure, a pattern is interesting in *periodicity* dimension, if the *periodicity* of a pattern is close to the *periodicity* of all of its items. The parameter *maxPerAllConf* indicates the maximum *periodic-all-confidence* threshold set by the user. Based on *maxPer* and *maxPerAllConf* thresholds, the interesting patterns involving rare items in *periodicity* dimension are extracted.

Henceforth, the periodic-frequent pattern is defined as follows.

Definition 3. *(**Periodic-frequent pattern X**) The pattern X is said to be periodic-frequent if* $sup(X) \geq minSup$, $allConf(X) \geq minAllConf$, $per(X) \leq maxPer$ *and* $perAllConf(X) \leq maxPerAllConf$. *The terms minSup, minAllConf, maxPer and maxPerAllConf, respectively represent the user-specified minimum support, minimum all-confidence, maximum periodicity and maximum periodic-all-confidence.*

Example 7. If the user-specified $minSup = 2$, $minAllConf = 0.6$, $maxPer = 6$ and $maxPerAllConf = 1.5$, then the pattern '*ab*' is said to be a periodic-frequent pattern, because $sup(ab) \geq minSup$, $allConf(ab) \geq minAllConf$, $per(ab) \leq maxPer$ and $perAllConf(ab) \leq maxPerAllConf$.

Example 8. The column titled **III** in Table 2 shows the complete set of periodic-frequent patterns discovered from Table 1. It can be observed that the proposed model has not only discovered the periodic-frequent patterns containing rare items but also pruned the uninteresting patterns '*cd*' and '*ce*.' *This clearly demonstrates that the proposed model discovers periodic-frequent patterns containing rare items without generating too many uninteresting patterns.*

Property 1. If $X \subset Y$, then $TID^X \supseteq TID^Y$. Therefore, $sup(X) \geq sup(Y)$ and $allConf(X) \geq allConf(Y)$.

Property 2. If $X \subset Y$, then $per(X) \leq per(Y)$. Therefore, $perAllConf(X) \leq perAllConf(Y)$ as $\frac{per(X)}{min(per(i_j) \forall i_j \in X)} \leq \frac{per(Y)}{min(per(i_j) \forall i_j \in Y)}$.

The discovered periodic-frequent patterns satisfy the *anti-monotonic property*. The correctness is straightforward to prove from Properties 1 and 2.

Definition 4. ***Problem Definition:*** *Given the database (TDB) and the user-specified minimum support (minSup), minimum all-confidence (minAllConf), maximum periodicity (maxPer) and maximum periodic-all-confidence (maxPerAllConf), the problem of finding periodic-frequent patterns involve discovering all patterns that satisfy the minSup, minAllConf, maxPer and maxPerAllConf thresholds. Please note, the support and periodicity of a pattern can also be expressed in percentage of $|TDB|$.*

4 Proposed Algorithm

Tanbeer et al. [13] have proposed Periodic-Frequent pattern-growth (PF-growth) to discover periodic-frequent patterns using *support* and *periodicity* measures. Unfortunately, this algorithm cannot be directly used for finding the periodic-frequent patterns with our model. The reason is PF-growth does not determine the interestingness of a pattern using *all-confidence* and *periodic-all-confidence* measures. In this paper, we extend PF-growth to determine the interestingness of a pattern using these two measures. We call the proposed algorithm as Extended Periodic-Frequent pattern-growth (EPF-growth). The proposed algorithm involves two steps: (i) construction of Extended Periodic-Frequent pattern-tree (EPF-tree), (ii) recursively mining EPF-tree to discover periodic-frequent patterns. Before we describe these two steps, we explain the structure of EPF-tree.

4.1 Structure of EPF-Tree

The structure of EPF-tree consists of a prefix-tree and a EPF-list. The EPF-list consists of three fields: item name (i), *support* (s) and *periodicity* (p). The structure of prefix-tree in EPF-tree is similar to that of the prefix-tree in FP-tree [4]. However, to obtain both *support* and *periodicity* of the patterns, the nodes in EPF-tree explicitly maintain the occurrence information for each transaction by maintaining an occurrence transaction-id list, called *tid-list*, only at the last node of every transaction. Complete details on prefix-tree are available in [13].

4.2 Construction of EPF-Tree

Since the periodic-frequent patterns generated by the proposed model satisfy the *anti-monotonic property*, periodic-frequent items (or 1-patterns) play a key

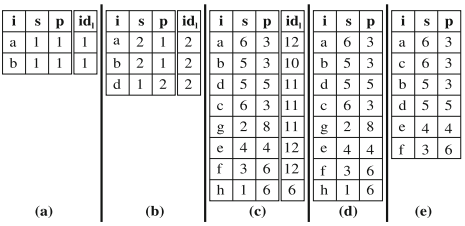

Fig. 2. Construction of EPF-list for Table 1. (a) After scanning the first transaction (b) After scanning the second transaction (c) After scanning the last transaction (d) Updated EPF-list (e) Final EPF-list with sorted list of periodic-frequent items

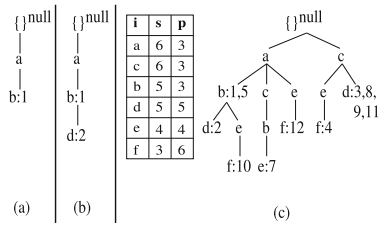

Fig. 3. Construction of EPF-tree for Table 1. (a) After scanning first transaction (b) After scanning second transaction (c) After scanning every transaction

role in efficient discovery of higher order periodic-frequent patterns. Periodic-frequent items are discovered by populating the EPF-list (lines 1 to 13 in Algorithm 1). Figures 2(a), (b), (c), (d) and (e) show the steps involved in finding periodic-frequent items from EPF-list. The user-specified $minSup$, $minAllConf$, $maxPer$ and $maxPerAllConf$ values are 2, 0.6, 6 and 1.5, respectively.

Algorithm 1. Construction of EPF-tree(TDB: Transactional database, $minSup$: minimum support, $minAllConf$: minimum all-confidence, $maxPer$: maximum periodicity, $maxPerAllConf$: maximum periodic-all-confidence)

1: Let id_l be a temporary array that records the tid of the last appearance of each item in the TDB. Let $t = \{tid_{cur}, X\}$ denote the current transaction with tid_{cur} and X representing the transaction-id of the current transaction and pattern, respectively.
2: **for** each transaction $t \in TDB$ **do**
3: **if** an item i occurs for the first time **then**
4: Insert i into the EPF-list with $sup^i = 1$, $per^i = tid_{cur}$ and $id_l^i = 1$.
5: **else**
6: $sup^i = sup^i + 1$.
7: **if** $(tid_{cur} - id_l^i) > per^i$ **then**
8: $per^i = tid_{cur} - id_l^i$.
9: **for** each item i in EPF-list **do**
10: **if** $(|TDB| - id_l^i) > per^i$ **then**
11: $per^i = |TDB| - id_l^i$.
12: Remove items from the EPF-list that do not satisfy $minSup$ and $maxPer$.
13: Sort the remaining items in EPF-list in descending order of their $support$. Let this sorted list of items be EPF.
14: Create a root node in EPF-tree, T, and label it "$null$."
15: **for** each transaction $tr \in TDB$ **do**
16: Sort the items in t in EPF order. Let this list of sorted periodic-frequent items in t be $[p|P]$, where p is the first item and P is the remaining list. Call $insert\_tree([p|P], tid_{cur}, T)$, which is the same as in [4].

After finding periodic-frequent items, prefix-tree is constructed by performing another scan on the database (lines 14 to 16 in Algorithm 1). The construction of prefix-tree in EPF-tree is similar to the construction of prefix-tree in FP-tree [4]. However, it has to be noted that leaf nodes in EPF-tree maintain the transaction-ids of the transactions. Figures 3(a), (b) and (c) show the construction of EPF-tree after scanning first, second and every transaction in the transactional database, respectively. In EPF-tree, an item header table is built so that each item points to its occurrences in the tree via a chain of node-links, to facilitate tree traversal. For simplicity, we do not show these node-links in trees, however, they are maintained as in FP-tree.

4.3 Mining EPF-Tree

Algorithm 2 describes the procedure for mining periodic-frequent patterns from EPF-tree. The EPF-tree is mined by calling EPF-growth as (EPF-tree, *null*). This algorithm resembles FP-growth. However, the key difference is that once the pattern-growth is achieved for a suffix 1-pattern (or item), it is completely pruned from the EPF-tree by pushing its tid-list to respective parent nodes.

Table 3 summarizes the working of this algorithm. First, we consider item 'f,' which is the bottom-most item in the EPF-list, as a suffix pattern. This item appears in three branches of the EPF-tree (refer Fig. 3(c)). The paths formed by these branches are $\{cef : 4\}$, $\{abef : 10\}$ and $\{aef : 12\}$ (format of these branches is $\{nodes : time\text{-}stamps\}$). Therefore, considering 'f' as a suffix item, its corresponding three prefix paths are $\{ce : 4\}$, $\{abe : 10\}$ and $\{ae : 12\}$, which form its conditional pattern base (refer Fig. 4(a)). Its conditional EPF-tree contains only a single path, $\langle e : 4, 10, 12\rangle$; '$a$,' '$b$' and '$c$' are not included because their *all-confidence* and *periodic-all-confidence* do not satisfy the $minAllConf$ and $maxPerAllConf$ respectively. Figure 4(b) shows the conditional EPF-tree of 'f.' The single path generates the pattern $\{ef : 3, 0.75, 6, 1.5\}$ (format is $\{pattern$: *support*, *all-confidence*, *periodicity*, *periodic-all-confidence*$\}$). The same process of creating prefix-tree and its corresponding conditional tree is repeated for further extensions of 'ef.' Next, 'f' is pruned from the original EPF-tree and its *tid*-lists are pushed to its parent nodes, as shown in Fig. 4(c). All the above processes are once again repeated until EPF-list $= \emptyset$.

5 Experimental Results

In this section, we show that the proposed model discovers interesting patterns pertaining to both frequent and rare items by pruning uninteresting patterns. We also evaluate the proposed model against the existing models of periodic-frequent patterns [6,11,13].

The algorithms, *PF-growth*, *MCPF-growth*, *MaxCPF-growth* and *EPF-growth* are written in C++ and run with Fedora 22 on a 2.66 GHz machine with 8 GB of memory. We have conducted experiments using both synthetic (**T10I4D100K**) and real-world (**Retail** and **FAA-accidents**) databases. The T10I4D100K data-base is generated using the IBM data generator [2]. This

Algorithm 2. EPF-growth($Tree$, α)

1: **for** each a_i in the header of $Tree$ **do**
2: Generate pattern $\beta = a_i \cup \alpha$. Construct an array TID^{β}, which represents the set of transaction-ids at which β has appeared in TDB. Next, compute from TID^{β}, $sup(\beta)$, $allConf(\beta)$, $per(\beta)$ and $perAllConf(\beta)$ and compare them with $minSup$, $minAllConf$, $maxPer$ and $maxPerAllConf$, respectively.
3: **if** $sup(\beta) \geq minSup$, $allConf(\beta) \geq minAllConf$, $per(\beta) \leq maxPer$ and $perAllConf(\beta) \leq maxPerAllConf$ **then**
4: Output β as a periodic-frequent pattern as {β: sup, allConf, per, perAllConf}.
5: Traverse $Tree$ using the node-links of β, and construct β's conditional pattern base and β's conditional EPF-tree $Tree_{\beta}$.
6: **if** $Tree_{\beta} \neq \emptyset$ **then**
7: call EPF-growth($Tree_{\beta}$, β);
8: Remove a_i from the $Tree$ and push a_i's tid-list to its parent nodes.

database contains 878 items with 100,000 transactions. The **Retail** database contains the market basket data from a Belgian retail store. This database contains 16,471 items with 88,162 transactions. The **FAA-accidents** database is constructed from the accidents data recorded by FAA from 1-January-1978 to 31-December-2014. This database contains 9,290 items with 98,864 transactions.

5.1 Patterns Generated by the Proposed Model

Figure 5 shows the number of patterns generated at different *minAllConf* and *maxPerAllConf* values. The *minSup* and *maxPer* are set at 0.01 % and 40 %. The following observations can be drawn: (*i*) The increase in *minAllConf* results in decrease of periodic-frequent patterns. The reason is that as *minAllConf* increases, the *support* threshold value of a pattern increases. (*ii*) The increase in *maxPerAllConf* results in increase of patterns. The reason is that as *maxPerAllConf* increases, the *periodicity* threshold value of a pattern increases.

Table 4 shows some of the interesting patterns discovered in FAA database. The *minSup*, *minAllConf*, *maxPer* and *maxPerConf* values used are 0.01 %, 0.01, 40 % and 9, respectively. *It can be observed from their support values that*

Table 3. Mining EPF-tree by creating conditional (sub -) pattern bases

Item	sup	per	Cond. Pattern Base	Cond. EPF-tree	Per. Freq. Patterns
f	3	6	$\{ce:4\}$, $\{abe:10\}$, $\{ae:12\}$	$\langle e:4,10,12\rangle$	{ef: 3, 0.75, 6, 1.5}
e	4	4	$\{c:4\}$, $\{abc:7\}$, $\{ab:10\}$, $\{a:12\}$	–	–
d	5	5	$\{ab:2\}$, $\{c:3,8,9,11\}$	–	–
b	5	3	$\{a:1,2,5,10\}$, $\{ac:7\}$	$\langle a:1,2,5,7,10\rangle$	{ab: 5, 0.833, 3, 1}
c	6	3	$\{a:7\}$	–	–

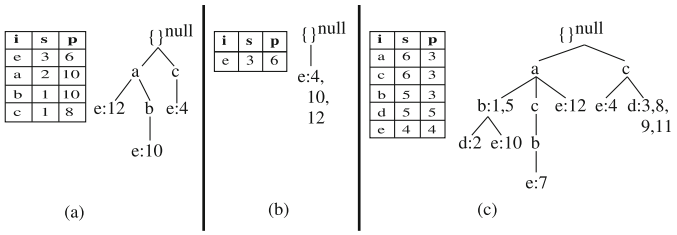

Fig. 4. Mining of EPF-tree for Table 1. (a) Prefix-tree of suffix item '*f*,' i.e., PT_f (b) Conditional tree of suffix item '*f*,' i.e., CT_f (c) EPF-tree after pruning item '*f*.'

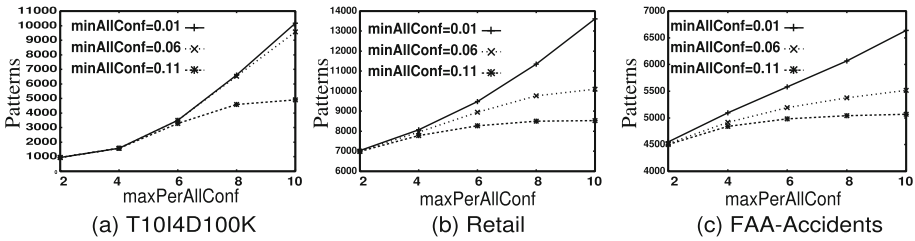

Fig. 5. Periodic-frequent patterns discovered in various databases

our model has discovered interesting patterns involving both frequent and rare items effectively. Please note that the *periodicity* (*per*) is expressed in days.

Table 4. Some of the interesting patterns discovered in FAA-accidents database

S. No.	Patterns	sup	allConf	per	perAllConf
1	{Ultralight Vehicles, Destroyed}	18	0.12	4904	8.01
2	{Starting engines, Destroyed}	13	0.06	4756	7.77
3	{General Operating Rules, Commercial Pilot, Minor}	10,399	0.15	32	6.4

5.2 Comparison of Proposed Model Against the Existing Models

For *MCPF-growth* and *MaxCPF-growth*, we use Eq. 5 to specify items' $minIS$ and $maxIP$ values. Setting the α and β values in this equation has been a non-trivial task as the patterns discovered by these algorithms can be different from the patterns discovered by *EPF-growth*. After conducting several experiments, we have empirically set the following values for *MCPF-growth* and *MaxCPF-growth* algorithms, such that both algorithms discover almost all periodic-frequent patterns discovered by *EPF-growth*.

Figure 6 shows the number of periodic-frequent patterns generated at different *minSup* values (*Y-axis* is plotted on logscale). For *EPF-growth*, we have fixed $minAllConf = 0.01$, $maxPer = 40\,\%$ and $maxPerAllConf = 9$ and vary *minSup* values. For *MCPF-growth* and *MaxCPF-growth*, we have set $\gamma = 0.01$, $LS = minSup$, $\beta = -0.4$, $Per_{max} = 40\,\%$ and $Per_{min} = 10\,\%$. For *PF-growth*, we have set $maxPer = 40\,\%$ and vary *minSup* values.

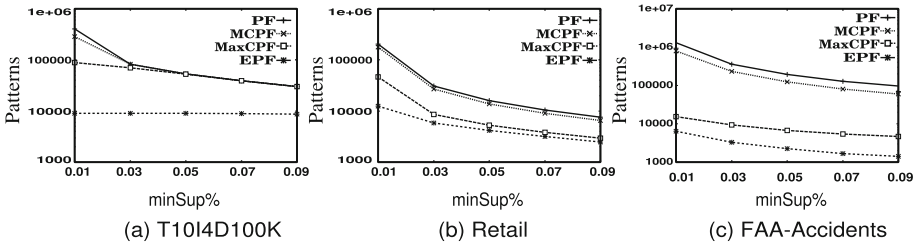

Fig. 6. Periodic-frequent patterns generated at different minSup values

Figure 7 shows the number of periodic-frequent patterns generated at different *maxPer* values (*Y-axis* is plotted on logscale). For *EPF-growth*, we have fixed $minSup = 0.01\,\%$, $minAllConf = 0.01$ and $maxPerAllConf = 9$ and vary *maxPer* values. For *MCPF-growth* and *MaxCPF-growth*, we have set $\gamma = 0.01$, $LS = 0.01\,\%$, $\beta = -0.4$, $Per_{max} = maxPer$ and $Per_{min} = 10\,\%$. For *PF-growth*, we have set $minSup = 0.01\,\%$ and vary *maxPer* values.

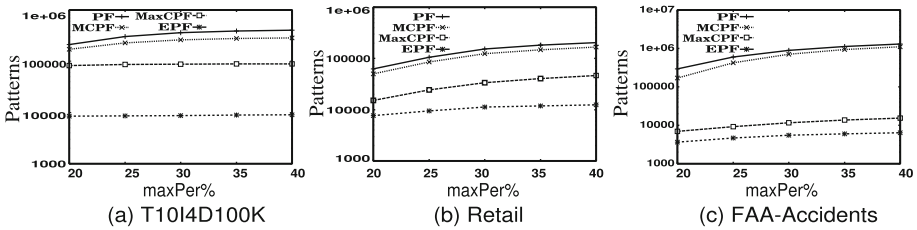

Fig. 7. Periodic-frequent patterns generated at different maxPer values

From Figs. 6 and 7, it can be observed that the proposed model has generated lesser number of periodic-frequent patterns than the other models, because the existing models have suffered from the *rare-item problem*.

Figure 8 shows the runtime taken by various models at different *maxPer* values (*Y-axis* is plotted on logscale). It can be observed that, in all the databases the proposed model takes lesser runtime to find periodic-frequent patterns than *PF-growth* and *MCPF-growth*. But the proposed model takes slightly more runtime than *MaxCPF-growth*. So the proposed model is not adding any significant overhead in mining periodic frequent patterns. Similar observations can be drawn when *minSup* is varied. Due to space restrictions, we are not reporting it.

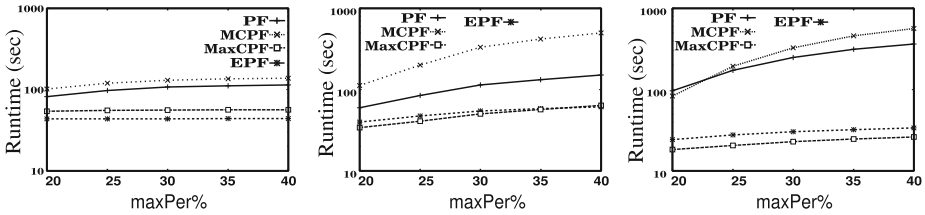

Fig. 8. Runtime requirements of various models at different maxPer values

6 Conclusions and Future Work

This paper introduces a model to address the rare item problem in both *support* and *periodicity* dimensions. A new interestingness measure, *periodic-all-confidence*, is proposed to address the problem in *periodicity* dimension. An efficient pattern-growth algorithm has been proposed to discover all periodic-frequent patterns in a database. Experimental results demonstrate that the proposed model is efficient. As a part of future work, we would like to study the change in periodic behavior of rare items due to noise.

References

1. Faa accidents dataset. http://www.asias.faa.gov/pls/apex/f?p=100:1:0::NO
2. Agrawal, R., Srikant, R.: Quest Synthetic Data Generator. IBM Almaden Research Center
3. Han, J., Gong, W., Yin, Y.: Mining segment-wise periodic patterns in time-related databases. In: KDD, pp. 214–218 (1998)
4. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. DMKD **8**(1), 53–87 (2004)
5. Kim, S., Barsky, M., Han, J.: Efficient mining of top correlated patterns based on null-invariant measures. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part II. LNCS, vol. 6912, pp. 177–192. Springer, Heidelberg (2011)
6. Uday Kiran, R., Krishna Reddy, P.: Towards efficient mining of periodic-frequent patterns in transactional databases. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part II. LNCS, vol. 6262, pp. 194–208. Springer, Heidelberg (2010)
7. Kiran, R.U., Shang, H., Toyoda, M., Kitsuregawa, M.: Discovering recurring patterns in time series. In: EDBT, pp. 97–108 (2015)
8. Ma, S., Hellerstein, J.: Mining partially periodic event patterns with unknown periods. In: ICDE, pp. 205–214 (2001)
9. Omiecinski, E.R.: Alternative interest measures for mining associations in databases. IEEE Trans. Knowl. Data Eng. **15**(1), 57–69 (2003)
10. Özden, B., Ramaswamy, S., Silberschatz, A.: Cyclic association rules. In: ICDE, pp. 412–421 (1998)

11. Surana, A., Kiran, R.U., Reddy, P.K.: An efficient approach to mine periodic-frequent patterns in transactional databases. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD Workshops 2011. LNCS, vol. 7104, pp. 254–266. Springer, Heidelberg (2012)
12. Tan, P.N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: KDD, pp. 32–41 (2002)
13. Tanbeer, S.K., Ahmed, C.F., Jeong, B.-S., Lee, Y.-K.: Discovering periodic-frequent patterns in transactional databases. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 242–253. Springer, Heidelberg (2009)
14. Tanbeer, S.K., Hassan, M.M., Alrubaian, M., Jeong, B.-S.: Mining regularities in body sensor network data. In: Di Fatta, G., Fortino, G., Li, W., Pathan, M., Stahl, F., Guerrieri, A. (eds.) IDCS 2015. LNCS, vol. 9258, pp. 88–99. Springer, Heidelberg (2015)
15. Weiss, G.M.: Mining with rarity: a unifying framework. SIGKDD Explor. **6**(1), 7–19 (2004)
16. Wu, T., Chen, Y., Han, J.: Re-examination of interestingnessmeasures in pattern mining: a unified framework. DMKD **21**(3), 371–397 (2010)
17. Yang, R., Wang, W., Yu, P.: Infominer+: mining partial periodic patterns with gap penalties. In: ICDM, pp. 725–728 (2002)