Discovering Periodic-Correlated Patterns in Temporal Databases

J. N. Venkatesh[1(✉)], R. Uday Kiran[2,3], P. Krishna Reddy[1], and Masaru Kitsuregawa[2,4]

[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 Information and Communication Technologies, Tokyo, Japan
[4] National Institute of Informatics, Tokyo, Japan

Abstract. The *support* and *periodicity* are two important dimensions to determine the interestingness of a pattern in a dataset. Periodic-frequent patterns are an important class of regularities that exist in a dataset with respect to these two dimensions. Most previous models on periodic-frequent pattern mining have focused on finding all patterns in a transactional database that satisfy the user-specified minimum support ($minSup$) and maximum periodicity ($maxPer$) constraints. These models suffer from the following two obstacles: (i) Current periodic-frequent pattern models cannot handle datasets in which multiple transactions can share a common time stamp and/or transactions occur at irregular time intervals (ii) The usage of single $minSup$ and $maxPer$ for finding the patterns leads to the *rare item problem*. This paper tries to address these two obstacles by proposing a novel model to discover periodic-correlated patterns in a temporal database. Considering the input data as a temporal database addresses the first obstacle, while finding periodic-correlated patterns address the second obstacle. The proposed model employs *all-confidence* measure to prune the uninteresting patterns in support dimension. A new measure, called *periodic-all-confidence*, is being proposed to filter out uninteresting patterns in periodicity dimension. A pattern-growth algorithm has also been discussed to find periodic-correlated patterns. Experimental results show that the proposed model is efficient.

Keywords: Data mining · Pattern mining · Periodic patterns
Rare item problem · Pattern-growth technique

1 Introduction

Periodic-frequent pattern[1] mining is an important model in data mining. It involves discovering all patterns in a transactional database that satisfy the

[1] A set of items represents a pattern (or an itemset).

A. Hameurlain et al. (Eds.): TLDKS XXXVIII, LNCS 11250, pp. 146–172, 2018.
https://doi.org/10.1007/978-3-662-58384-5_6

user-specified minimum support ($minSup$) and maximum periodicity ($maxPer$) constraints [35]. The $minSup$ controls the minimum number of transactions that a pattern must cover, and the $maxPer$ controls the maximum interval within which a pattern must reoccur in the entire database. Finding periodic-frequent patterns is a significant task with many business applications. A classic application is market-basket analytics. It analyzes how regularly the itemsets are being purchased by the customers. An example of a periodic-frequent pattern is as follows:

$$\{Bat, Ball\} \quad [support = 5\%, \quad periodicity = 1 \text{ h}].$$

The above pattern says that 5% of the customers have purchased the items 'Bat' and '$Ball$,' and the maximum duration between any two consecutive purchases containing both of these items is no more than an hour. This predictive behavior of the customers' purchases may facilitate the users in product recommendation and inventory management.

Fournier-Viger et al. [11] extended the periodic-frequent pattern model to find periodic-utility patterns in a transactional database. Amphawan et al. [2] extended the model to find top-k periodic-frequent patterns in a transactional database. Uday et al. [24] extended the periodic-frequent pattern model to find partial periodic-frequent patterns in a transactional database. Nofong [27] extended the periodic-frequent pattern model to find productive periodic-frequent patterns. In this model, a periodic-frequent pattern is considered productive if its support is greater than the product of its subsets. The popular adoption and successful industrial application of this model suffers from the following obstacles: (i) Since the model accepts the transactional database as an input, the model implicitly assumes that all transactions in a database occur at a fixed time-interval. This assumption limits the applicability of the model as transactions in a database may occur at irregular time intervals. (ii) The $minSup$ and $maxPer$ play a key role in periodic-frequent pattern mining. They are used to prune the search space and limit the number of patterns being generated. Since only a single $minSup$ and $maxPer$ are used for the whole data, the model implicitly assumes that all items in the data have uniform *support* and *periodicity*. However, this is seldom the case in many real-world applications. In many applications, some items appear very frequently in the data, while others rarely[2] appear. Moreover, rare items typically have high *periodicity* (i.e., inter-arrival times) as compared against the frequent items. If the *support* and *periodicity* of items vary a great deal, we will encounter the following two problems:

- If the $maxPer$ is set too low and/or the $minSup$ is set too high, we will miss the periodic patterns involving rare items.
- To find the periodic patterns involving both frequent and rare items, we have to set a high $maxPer$ and a low $minSup$. However, this may result

[2] Classifying the items into either frequent or rare is a subjective issue that depends upon the user and/or application requirements.

in combinatorial explosion, producing too many patterns, because frequent items can combine with one another in all possible ways and many of them may be meaningless.

This dilemma is known as the "*rare item problem*." This paper tries to address both of these problems.

Prior to our study, researchers have tried to address the *rare item problem* using the concept of multiple $minSup$ and $maxPer$ constraints [20,33]. In this concept, each item in the database is specified with a *minimum item support* ($minIS$) and *maximum item periodicity* ($maxIP$). Next, the $minSup$ and $maxPer$ of a pattern are specified depending on its items $minIS$ and $maxIP$ values, respectively. Although this concept facilitates every pattern to satisfy a different $minSup$ and $maxPer$ values, it still suffers from an open problem of determining the items' $minIS$ and $maxIP$ values.

In this paper, we propose a model to discover periodic-correlated patterns in a temporal database. A temporal database is a collection of transactions ordered by their timestamps. A temporal database facilitates multiple transactions to share a common timestamp and time gaps in-between consecutive transactions. Thus, considering the input data as a temporal database addresses the first obstacle in periodic-frequent pattern model. In the literature, correlated pattern model was discussed to address the *rare item problem* in frequent pattern mining [28]. We extend this model to find periodic-correlated patterns in a temporal database. Thus, addressing the second obstacle of periodic-frequent pattern model. The proposed model considers 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. The renowned *all-confidence* [28] measure is used to determine how close is the *support* of a pattern with respect to the *support* of its individual items. To the best of our knowledge, there exists no measure to determine how close is the *periodicity* of a pattern with respect to the *periodicity* of its items. So forth, we introduce a new measure, called *periodic-all-confidence*, to determine the interestingness of a pattern. The *periodic-all-confidence* measure is used to determine how close is the *periodicity* of a pattern with respect to the *periodicity* of its individual items. These two measures facilitate us to achieve the objective of generating periodic-correlated patterns containing both frequent and rare items yet without causing frequent items to generate too many uninteresting patterns. A pattern-growth algorithm, called Extended Periodic-Correlated pattern-growth (EPCP-growth), has also been described to find all periodic-correlated patterns. Experimental results demonstrate that the proposed model can discover useful information and EPCP-growth is runtime efficient and highly scalable as well.

In [38], we have studied the problem of finding periodic-correlated patterns in transactional databases. In this paper, we first extend this study to temporal databases and provide theoretical correctness for EPCP-growth algorithm. We also evaluate the performance of EPCP-growth by conducting extensive experiments on both synthetic and real-world databases.

The rest of the paper is organized as follows. Section 2 describes related work on frequent pattern mining, periodic pattern mining and periodic-frequent pattern mining. Section 3 extends the (full) periodic-frequent pattern model to handle the temporal databases. Section 4 introduces the proposed model of finding periodic-correlated patterns in a temporal database. Section 5 describes EPCP-growth algorithm. Section 6 reports on experimental results. Finally, Sect. 7 concludes the paper with future research directions.

2 Related Work

2.1 Frequent Pattern Mining

Agrawal et al. [1] introduced the problem of finding frequent patterns in a transactional database. Since then, the problem of finding these patterns has received a great deal of attention [9,10,12,13,30,36,42]. The basic model used in most of these studies remained the same. It involves discovering all frequent patterns in a transactional database that satisfy the user-specified minimum support (*minSup*) constraint. The usage of single *minSup* for the entire database leads to the *rare item problem* (discussed in previous section). When confronted with this problem in real-world applications, researchers have tried to address it using the concept of multiple *minSups* [16,21,26]. In this concept, each item in the database is specified with a *support*-constraint known as *minimum item support* (*minIS*). Next, the *minSup* of a pattern is represented with the lowest *minIS* value of its items. Thus, every pattern can satisfy a different *minSup* depending upon its items. A major limitation of this concept is that it suffers from an open problem of determining the items' *minIS* values.

Brin et al. [6] introduced correlated pattern mining to address the *rare item problem*. The statistical measure, χ^2, was used to discover correlated patterns. Since then, several interestingness measures have been discussed based on the theories in probability, statistics, or information theory. Examples include *all-confidence*, *any-confidence*, *bond* [28] and *kulc* [5]. Each measure has its own selection bias that justifies the rationale for preferring one pattern over another. As a result, there exists no universally acceptable best measure to discover correlated patterns in any given database. Researchers are making efforts to suggest an appropriate measure based on user and/or application requirements [28,32,34,37,39].

Recently, *all-confidence* is emerging as a popular measure to discover correlated patterns [17,18,25,40,44,45]. It is because this measure satisfies both the *anti-monotonic* (see Definition 1) and *null-invariance* (see Definition 2) properties. The former property says that all non-empty subsets of a correlated pattern must also be correlated. This property plays a key role in reducing the search space, which in turn decreases the computational cost of mining the patterns. In other words, this property makes the correlated pattern mining practicable in real-world applications. The latter property discloses genuine correlation relationships without being influenced by the object co-absence in a database. In other words, this property facilitates the user to discover interesting patterns

involving both frequent and rare items without generating a huge number of uninteresting patterns. In this paper, we use this measure to address the *rare item problem* in *support* dimension.

Definition 1. *(**Anti-monotonic property** [1]). A measure C is anti-monotone if and only if whenever a pattern (or an itemset) X violates C, so does any superset of X.*

Definition 2. *(**Null-invariance property** [34]). Let us consider a 2 x 2 contingency table (shown in Table 1) as a contingency matrix, $M = [f_{11}\ f_{10}; f_{01}\ f_{00}]$. Let an interestingness measure be a matrix operator, O, that maps the matrix M into a scalar value, k, i.e., $OM = k$. A binary measure of association is null-invariant if $O(M + C) = O(M)$, where $C = [00; 0k]$ and k is a positive constant.*

Table 1. A 2×2 contingency table for variables A and B

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

Table 1 is an example of a contingency table for rule $A => B$, where A and B are the frequent itemsets and N denotes the total number of records. The number of records in which A and B occurs together is denoted by f_{11}. Similarly, f_{10} denotes the number of records in which A doesn't occur with B, f_{01} denotes the number of records in which B doesn't occurs with A and f_{00} denotes the number of records in which neither A nor B occurs.

2.2 Periodic Pattern Mining

Han et al. [14] introduced (partial) periodic pattern[3] model to find temporal regularities in time series data. The model involves the following two steps: (i) segment the given time series into multiple period-segments such that the length of each period-segment is equal to the user-specified *period* (*per*), and (ii) discover all patterns that satisfy the user-specified *minSup*.

Example 1. Let $I = \{abcde\}$ be the set of items and $S = a\{bc\}baebacea\{ed\}d$ be a time series data generated from I. If the user-defined *period* is 3, S is segmented into four period-segments such that each period-segment contains only 3 events (or itemsets). That is, $PS_1 = a\{bc\}b$, $PS_2 = aeb$, $PS_3 = ace$ and $PS_4 = a\{ed\}d$. Let $a * b$ be a pattern, where '$\star$' denotes a wild (or do not care) character that can represent any itemset. This pattern appears in the period-segments of PS_1

[3] The term 'pattern' in a time series represents a set of itemsets (or sets of items).

and PS_2. Therefore, its *support* is 2. If the user-specified *minSup* is 2, then $a \star b$ represents a periodic pattern as its *support* is no less than *minSup*. In this example, braces for singleton itemsets have been eliminated for brevity.

Han et al. [13] have discussed Max-sub-pattern hitset algorithm to find periodic patterns. Chen et al. [8] developed a pattern-growth algorithm, and showed that it outperforms the Max-sub-pattern hitset algorithm. Aref et al. [4] extended Han's model for the incremental mining of partial periodic patterns. Yang et al. [41] studied the change in periodic behavior of a pattern due to noise, and enhanced the basic model to discover a class of periodic patterns known as asynchronous periodic patterns. Zhang et al. [43] enhanced the basic model of partial periodic patterns to discover periodic patterns in character sequences like protein data. Cao et al. [7] discussed a methodology to determine the *period* using auto-correlation. The popular adoption and successful industrial application of partial periodic pattern model suffers from the following two issues:

- **Rare item problem:** The usage of single *minSup* for the entire time series leads to the *rare item problem*.
- **Inability to consider temporal occurrence information of the items within a series:** The basic model of periodic patterns implicitly considers the data as an evenly spaced time series (i.e., all events within a series occur at a fixed time interval). This assumption limits the applicability of the model as events in many real-world time series datasets occur at irregular time intervals.

Yang et al. [41] used "*information gain*" as an alternative interestingness measure to address the problem. Chen et al. [8] extended Liu's model [26] to find periodic patterns in time series using multiple *minSups*. It has to be noted that these studies have focused on finding periodically occurring sets of itemsets in time series data, while the proposed study focuses on finding periodically occurring correlated itemsets in temporal databases.

2.3 Periodic-Frequent Pattern Mining

Ozden et al. [29] enhanced the transactional database by a time attribute that describes the time when a transaction has appeared, investigated the periodic behavior of the patterns to discover cyclic association rules. In this study, a database is fragmented into non-overlapping subsets with respect to time. The association rules that are appearing in at least a certain number of subsets are discovered as cyclic association rules. By fragmenting the data and counting the number of subsets in which a pattern occurs greatly simplifies the design of the mining algorithm. However, the drawback is that patterns (or association rules) that span multiple windows cannot be discovered.

Tanbeer et al. [35] discussed a model to find periodic-frequent patterns in a transactional database. This model eliminates the need of data fragmentation, and discovers all patterns in a transactional database that satisfy the user-specified *minSup* and *maxPer* constraints. A pattern-growth algorithm,

called Periodic-Frequent Pattern-growth (PFP-growth), was also discussed to find these patterns. Uday et al. [19] have discussed a greedy search technique to efficiently compute the *periodicity* of a pattern. Anirudh et al. [3] have proposed an efficient pattern-growth algorithm based on the concept of period summary. In this concept, the *tid*-list of the patterns are compressed into partial periodic summaries, and later aggregated to find periodic-frequent patterns efficiently. The popular adoption and successful industrial application of periodic-frequent pattern model suffers from the following two issues: (*i*) cannot handle databases in which transactions are occurring at irregular time intervals and (*ii*) the rare item problem (both issues are discussed in Sect. 1). This paper tries to address both of these issues.

Uday et al. [20] extended Liu's model [26] to address the *rare item problem* in periodic-frequent pattern mining. In this model, every item $i_j \in I$ in the database is specified with $minIS$ and $maxIP$ values. The $minSup$ and $maxPer$ for a pattern $X \subseteq I$ are specified as follows:

$$\begin{aligned} minSup(X) &= min(minIS(i_j)|\forall i_j \in X) \\ &\quad and \\ maxPer(X) &= max(maxIP(i_j)|\forall i_j \in X) \end{aligned} \tag{1}$$

where, $minSup(X)$ represents the *minimum support* of X, $maxPer(X)$ represents the *maximum periodicity* of X, $minIS(i_j)$ denotes the *minimum item support* of an item $i_j \in X$ and $maxIP(i_j)$ denotes the *maximum item periodicity* of an item $i_j \in X$.

The usage of item-specific $minIS$ and $maxIP$ values facilitates every pattern to satisfy a different $minSup$ and $maxPer$ depending on its items. However, the major limitation of this model is the computational cost because the generated periodic-frequent patterns do not satisfy the *anti-monotonic property*. Akshat et al. [33] proposed an alternative periodic-frequent pattern model using the item-specific $minIS$ and $maxIP$ values. In this model, the $minSup$ and $maxPer$ for a pattern X are specified as follows:

$$\begin{aligned} minSup(X) &= max(minIS(i_j)|\forall i_j \in X) \\ &\quad and \\ maxPer(X) &= min(maxIP(i_j)|\forall i_j \in X) \end{aligned} \tag{2}$$

The periodic-frequent patterns discovered by this model satisfy the anti-monotonic property. Henceforth, this model is practicable in real-world applications.

An open problem that is common to above two studies [20,33] is the methodology to specify items' $minIS$ and $maxIP$ values. Uday et al. [20] have described the following methodology to address this problem:

$$\begin{aligned} minIS(i_j) &= max(\gamma \times S(i_j),\ LS) \\ &\quad and \\ maxIP(i_j) &= max(\beta \times S(i_j) + Per_{max},\ Per_{min}) \end{aligned} \tag{3}$$

where $i \in I$ and $S(i)$ is the *support* of the item i. In Eq. 3, 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. 3, 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. 3 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 3 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 temporal 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 temporal databases, where items can have similar *support* but different *periodicities* can still lead to the *rare item problem*.

Example 2. 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. 3 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*.

Rashid et al. [31] introduced standard deviation as an alternative measure of $maxPrd$. Nofong [27] employed mean as an alternative measure to determine the periodic interestingness of a pattern. Unfortunately, these alternative interestingness measures are impracticable on very large databases because the discovered patterns do not satisfy the downward closure property.

Recently, Uday et al. [22,23] have studied the problem of finding (partial) periodic patterns in temporal databases. In this paper, we extended the (full) periodic-frequent pattern model to handle the temporal databases.

3 Periodic-Frequent Pattern Model

In this section, we redefine the periodic-frequent pattern model [35] by taking into account the temporal databases. Care has been taken such that the nomenclature of redefined model is consistent with the nomenclature of the basic model of periodic-frequent patterns.

Let I be the set of items. Let $X \subseteq I$ be a **pattern** (or an itemset). A pattern containing β number of items is called a β**-pattern**. A **transaction**, $t_k = (tid, ts, Y)$ is a tuple, where $tid \in \mathbb{R}$ represents transactional-identifier, $ts \in \mathbb{R}$ represents the timestamp at which the pattern Y has occurred. A **temporal 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. Let ts_{min} and ts_{max} denote the minimum and maximum timestamps in TDB, respectively. For a transaction $t_k = (tid, ts, Y)$, $k \geq 1$, such that $X \subseteq Y$, it is said that X occurs in t_k and such timestamp is denoted as ts^X. Let $TS^X = \{ts^X_j, \cdots, ts^X_k\}$, $j, k \in [1, m]$ and $j \leq k$, be an **ordered set of timestamps** where X has occurred in TDB. In this paper, we call this list of timestamps of X as **ts-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, $sup(X) = |TS^X|$. Let ts^X_q and ts^X_r, $j \leq q < r \leq k$, be the two consecutive timestamps in TS^X. The time difference (or an inter-arrival time) between ts^X_r and ts^X_q is defined as a ***period*** of X, say p^X_a. That is, $p^X_a = ts^X_r - ts^X_q$. Let $P^X = (p^X_1, p^X_2, \cdots, p^X_r)$ be the set of all *periods* for pattern X. The ***periodicity*** of X, denoted as $per(X) = max(p^X_1, p^X_2, \cdots, p^X_r)$. 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 redefined **problem definition** of periodic-frequent pattern mining involves discovering all patterns in TDB that satisfy the user-specified $minSup$ and $maxPer$ constraints. The *support* of a pattern can be expressed in percentage of $|TDB|$. Similarly, the *period* and *periodicity* of a pattern can be expressed in percentage of $(ts_{max} - ts_{min})$.

Example 3. Table 2 shows the temporal 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, $t_1 = (101, 1, ab)$, 101 (from the left hand side) represents the transaction identifier of the transaction, 1 denotes the timestamp at which the transaction has occurred and 'ab' represents the itemset occurring in this transaction. In the entire database, this pattern appears at the timestamps of $1, 2, 5, 7$ and 10. Therefore, $TS^{ab} = \{1, 2, 5, 7, 10\}$. The *support* of '$ab$,' i.e., $sup(ab) = |TS^{ab}| = |1, 2, 5, 7, 10| = 5$. If the user-specified $minSup = 5$, then 'ab' is a frequent pattern because $sup(ab) \geq minSup$. The minimum and maximum timestamps of all transactions in this database are 1 and 12, respectively. Therefore, $ts_{min} = 1$ and $ts_{max} = 12$. All periods for this pattern are: $p^{ab}_1 = 0$ $(= 1 - ts_{min})$, $p^{ab}_2 = 1$ $(= 2 - 1)$, $p^{ab}_3 = 3$ $(= 5 - 2)$, $p^{ab}_4 = 2$ $(= 7 - 5)$, $p^{ab}_5 = 3$ $(10 - 7)$ and $p^{ab}_6 = 2$ $(= ts_{max} - 10)$. Therefore, $P^{ab} = (0, 1, 3, 2, 3, 2)$. The *periodicity* of '$ab$,' i.e., $per(ab) = max(0, 1, 3, 2, 3, 2) = 3$. If the user-defined $maxPer = 3$, then the frequent pattern 'ab' is a periodic-frequent pattern because $per(ab) \leq maxPer$.

The two key differences between the Tanbeer's model [35] and the above model of periodic-frequent patterns are as follows: (i) In the above model, the

first period of a pattern is calculated using ts_{min}, where as the first period of a pattern in Tanbeer's model is calculated with reference to initial time, which is zero. (*ii*) Since the Tanbeer model considers input data as a transactional database with transactions occurring at a fixed time interval, the *periodicity* of a pattern is expressed in percentage of $|TDB|$, whereas in the above model the *periodicity* of a pattern is expressed in percentage of $(ts_{max} - ts_{min})$.

Table 2. Running example: A temporal database

tid	ts	Items	tid	ts	Items
101	1	a, b	107	7	a, b, c, e
102	3	a, b, d	108	8	c, d
103	3	c, d, g	109	9	c, d
104	4	c, e, f	110	10	a, b, e, f
105	5	a, b	111	11	c, d, g
106	7	h	112	12	a, e, f

The above redefined model of periodic-frequent patterns still suffers from the *rare item problem* (refer Example 4). In the next section, we describe periodic-correlated pattern model to address this problem.

Example 4. Consider the rare items 'e' and 'f' in Table 2. 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 3. 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*).

4 Periodic-Correlated Pattern Model

To address the *rare item problem* in periodic-frequent pattern mining, we need a model that extracts interesting 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 patterns discovered in a database have their *support* and *periodicity* close to that of its individual items. The following example illustrates our observation.

Table 3. Periodic-frequent patterns discovered from Table 2. The terms *Pat*, *sup*, *allConf*, *per* 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	sup	allConf	per	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	✓	✓	✓
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	✓	✓	×

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 [34,39]. In this paper, we use *all-confidence* to address the *rare item problem* in support dimension. (The reason for choosing this measure for finding periodic-correlated patterns has been described in Sect. 2).

Continuing with the model of periodic-frequent patterns (discussed in the previous section), the proposed model of periodic-correlated patterns is as follows.

Definition 3. ***(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.

Definition 4. *(**Correlated pattern X**). The pattern X is said to be correlated if $sup(X) \geq minSup$ and $allConf(X) \geq minAllConf$. The terms minSup and minAllConf, respectively represent the user-specified minimum support and minimum all-confidence.*

Example 6. In Table 2, the *support* of the patterns a, b and ab are 6, 5 and 5, respectively. Therefore, the *all-confidence* of ab, i.e., $allConf(ab) = \frac{sup(ab)}{max(sup(a),sup(b))} = \frac{5}{max(6,5)} = 0.833$. If the user-specified $minSup = 2$ and $minAllConf = 0.6$, then ab is a correlated pattern because $sup(ab) \geq minSup$ and $allConf(ab) \geq minAllConf$.

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

Example 7. The column titled **II** in Table 3 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 5. *(**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, $perAllConf(X) = \frac{per(X)}{min(per(i_j)|\forall i_j \in X)}$.*

Example 8. In Table 2, the *periodicity* of the patterns a, b and ab are 3, 3 and 3, respectively. Therefore, the *periodic-all-confidence* of ab, i.e., $perAllConf(ab) = \frac{per(ab)}{min(per(a),per(b))} = \frac{3}{min(3,3)} = 1$.

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 can be extracted.

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

Definition 6. *(**Periodic-correlated pattern X**). The pattern X is said to be periodic-correlated 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 9. 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-correlated pattern, because $sup(ab) \geq minSup$, $allConf(ab) \geq minAllConf$, $per(ab) \leq maxPer$ and $perAllConf(ab) \leq maxPerAllConf$.

Example 10. The column titled **III** in Table 3 shows the complete set of periodic-correlated patterns discovered from Table 2. It can be observed that the proposed model has not only discovered the periodic-correlated patterns containing rare items but also pruned the uninteresting patterns '*cd*' and '*ce*.' *This clearly demonstrates that the proposed model discovers periodic-correlated patterns containing rare items without generating too many uninteresting patterns.*

The discovered periodic-correlated patterns satisfy the *anti-monotonic property* (see Lemma 1). The correctness is straightforward to prove from Properties 1 and 2.

Property 1. If $X \subset Y$, then $TS^X \supseteq TS^Y$. 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 $\frac{per(X)}{min(per(i_j) \forall i_j \in X)} \leq \frac{per(Y)}{min(per(i_j) \forall i_j \in Y)}$.

Lemma 1. *If $X \subset Y$, then $TS^X \supseteq TS^Y$. Therefore, $sup(X) \geq sup(Y)$, $allConf(X) \geq allConf(Y)$, $per(X) \leq per(Y)$ and $perAllConf(X) \leq perAllConf(Y)$.*

Definition 7. ***Problem Definition:** Given the temporal 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-correlated patterns involves discovering all patterns that satisfy the minSup, minAllConf, maxPer and maxPerAllConf thresholds. The support of a pattern can be expressed in percentage of $|TDB|$. The periodicity of a pattern can be expressed in percentage of $(ts_{max} - ts_{min})$.*

5 Proposed Algorithm

Tanbeer et al. [35] 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-correlated patterns with our model. The reason is that 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-Correlated pattern-growth (EPCP-growth). The proposed algorithm involves two steps: (*i*) construction of Extended Periodic-Correlated pattern-tree (EPCP-tree), (*ii*) recursively mining EPCP-tree to discover periodic-correlated patterns. Before we describe these two steps, we explain the structure of EPCP-tree.

5.1 Structure of EPCP-tree

The structure of EPCP-tree consists of a prefix-tree and a EPCP-list. The EPCP-list consists of three fields: item name (*i*), *support* (*s*) and *periodicity* (*p*). The structure of prefix-tree in EPCP-tree is similar to that of the prefix-tree in FP-tree [15]. However, to obtain both *support* and *periodicity* of the patterns, the nodes in EPCP-tree explicitly maintain the occurrence information for each transaction by maintaining an occurrence timestamp list, called *ts-list*, only at the tail node of every transaction. Complete details on prefix-tree are available in [35].

One can assume that the structure of the prefix-tree in an EPCP-tree may not be memory efficient since it explicitly maintains timestamps of each transaction. However, it has been argued that such a tree can achieve memory efficiency by keeping transaction information only at the tail nodes and avoiding the support count field at each node [35].

5.2 Construction of EPCP-tree

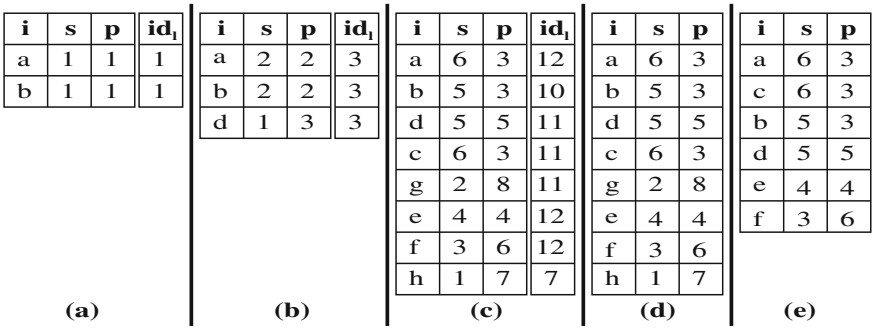
Since the periodic-correlated patterns generated by the proposed model satisfy the *anti-monotonic property*, periodic-correlated items (or 1-patterns) play a key role in efficient discovery of higher order periodic-correlated patterns. These items are discovered by populating the EPCP-list (lines 1 to 18 in Algorithm 1). Figures 1(a), (b), (c), (d) and (e) show the steps involved in finding periodic-correlated items from EPCP-list. The user-specified $minSup$, $minAllConf$, $maxPer$ and $maxPerAllConf$ values are 2, 0.6, 6 and 1.5, respectively.

After finding periodic-correlated items, prefix-tree is constructed by performing another scan on the database (lines 19 to 23 in Algorithm 1). A EPCP-tree is constructed as follows. First, create the root node of the tree and labeled it as "*null*." Scan the database a second time. The items in each transaction are processed in $EPCP$ order (i.e., sorted according to descending support count), and a branch is created for each transaction such that only the tail-nodes record the timestamps of transactions. For example, the scan of the first transaction,

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

Let id_l be a temporary array that records the ts of the last appearance of each item in the TDB. Let $t = \{tid, ts_{cur}, X\}$ denote the current transaction with tid, ts_{cur} and X representing the transanction identifier, timestamp of the current transaction and pattern, respectively.
for each transaction $t \in TDB$ **do**
  if an item i occurs for the first time **then**
    Insert i into the EPCP-list with $sup^i = 1$, $per^i = ts_{min} - ts_{cur}$ and $id_l^i = 1$.
  else
    $sup^i = sup^i + 1$.
    if $(ts_{cur} - id_l^i) > per^i$ **then**
      $per^i = ts_{cur} - id_l^i$.
    end if
  end if
end for
for each item i in EPCP-list **do**
  if $(ts_{max} - id_l^i) > per^i$ **then**
    $per^i = ts_{max} - id_l^i$.
  end if
end for
Remove items from the EPCP-list that do not satisfy $minSup$ and $maxPer$.
Sort the remaining items in EPCP-list in descending order of their $support$. Let this sorted list of items be $EPCP$.
Create a root node in EPCP-tree, T, and label it "$null$."
for each transaction $t \in TDB$ **do**
  Sort the items in t in $EPCP$ 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 $insert\_tree([p|P], ts_{cur}, T)$.
end for

"$101 : 1 : ab$," which contains two items (a, b in $EPCP$ order), leads to the construction of the first branch of the tree with two nodes, $\langle a \rangle$ and $\langle b : 1 \rangle$, where a is linked as a child of the root and $b : 1$ is linked to a. The EPCP-tree generated after scanning the first transaction is shown in Fig. 2(a). The scan on the second transaction, "$102 : 3 : abd$," containing the items a, b and d in CI order, would result in a branch where a is linked to the root, b is linked to a and $d : 3$ is linked to b. However, this branch would share a common prefix, ab, with the existing path for first transaction. Therefore, we create a single new node $\langle d : 3 \rangle$, and link $d : 3$ to b as shown in Fig. 2(b). A similar process is repeated for the remaining transactions and the tree is updated accordingly. Figure 2(c) shows the EPCP-tree constructed after scanning the entire database. In EPCP-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.



i	s	p	id_l
a	1	1	1
b	1	1	1

(a)

i	s	p	id_l
a	2	2	3
b	2	2	3
d	1	3	3

(b)

i	s	p	id_l
a	6	3	12
b	5	3	10
d	5	5	11
c	6	3	11
g	2	8	11
e	4	4	12
f	3	6	12
h	1	7	7

(c)

i	s	p
a	6	3
b	5	3
d	5	5
c	6	3
g	2	8
e	4	4
f	3	6
h	1	7

(d)

i	s	p
a	6	3
c	6	3
b	5	3
d	5	5
e	4	4
f	3	6

(e)

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

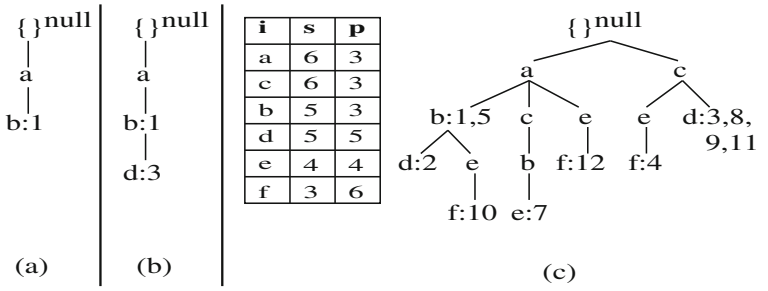


Fig. 2. Construction of EPCP-tree for Table 2. (a) After scanning first transaction (b) After scanning second transaction (c) After scanning every transaction

The EPCP-tree maintains the complete information of all periodic-correlated patterns in a database. The correctness is based on Property 3 and shown in Lemmas 2 and 3. For each transaction $t \in TDB$, EPCP(t) is the set of all candidate items in t, i.e., $EPCP(t) = item(t) \cap EPCP$, and is called the candidate item projection of t.

Property 3. An EPCP-tree maintains a complete set of candidate item projections for each transaction in a database only once.

Lemma 2. *Given a TDB and user-defined minSup, minAllConf, maxPer, and maxPerAllConf thresholds, the complete set of all periodic-correlated item projections of all transactions in the TDB can be derived from the EPCP-tree.*

Proof. Based on Property 3, each transaction $t \in TDB$ is mapped to only one path in the tree, and any path from the root up to a tail node maintains the complete projection for exactly n transactions (where n is the total number of entries in the ts-list of the tail node).

Algorithm 2. Insert_tree($[p|P]$, ts_{cur}, T)

1: **if** T does not have a child N satisfying $p.itemName = N.itemName$ **then**
2: Create a new node N and set its parent as T. Let its node-link be linked to the nodes with the same item via the node-link structure.
3: **end if**
4: Remove p from P.
5: **if** P is non-empty **then**
6: Call Insert_tree($[P]$, ts_{cur}, N)
7: **else**
8: Add ts_{cur} to T (i.e., leaf node).
9: **end if**

Lemma 3. *The size of the EPCP-tree (without the root node) on a TDB for user-defined minSup, minAllConf, maxPer, and maxPerAllConf thresholds, is bounded by* $\sum_{t \in TDB} |EPCP(t)|$.

Proof. According to the EPCP-tree construction process and Lemma 2, each transaction t contributes at most one path of size $|EPCP(t)|$ to an EPCP-tree. Therefore, the total size contribution of all transactions can be $\sum_{t \in TDB} |EPCP(t)|$ at best. However, since there are usually many common prefix patterns among the transactions, the size of an EPCP-tree is normally much smaller than $\sum_{t \in TDB} |EPCP(t)|$.

Before we discuss the mining of EPCP-tree, we explore the following important property and lemma of an EPCP-tree.

Property 4. A tail node in an EPCP-tree maintains the occurrence information for all the nodes in the path (from the tail node to the root) at least in the transactions in its ts-list.

Lemma 4. *Let* $Z = \{a_1, a_2, \cdots, a_n\}$ *be a path in an EPCP-tree where node* a_n *is the tail node carrying the ts-list of the path. If the ts-list is pushed-up to node* a_{n1}*, then* a_{n1} *maintains the occurrence information of the path* $Z' = \{a_1, a_2, \cdots, a_{n1}\}$ *for the same set of transactions in the ts-list without any loss.*

Proof. Based on Property 4, a_n maintains the occurrence information of path Z' at least in the transactions in its ts-list. Therefore, the same ts-list at node a_{n1} maintains the same transaction information for Z' without any loss.

5.3 Mining EPCP-tree

Algorithm 3 describes the procedure for mining periodic-correlated patterns from EPCP-tree. The EPCP-tree is mined by calling EPCP-growth as (EPCP-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 EPCP-tree by pushing its ts-list to respective parent nodes.

The working of this algorithm is as follows. We proceed to construct the prefix tree for each candidate item in the EPCP-list, starting from the bottom most item, say i. To construct the prefix-tree for i, the prefix sub-paths of node i are accumulated in a tree-structure, PT_i. Since i is the bottom-most item in the EPCP-list, each node labeled i in the EPCP-tree must be a tail node. While constructing PT_i, based on Property 4, we map the ts-list of every node of i to all items in the respective path explicitly in the temporary array (one for each item). This temporary array facilitates the calculation of sup, $allConf$, per, $perAllConf$ of each item in PT_i (line 2 in Algorithm 3). If an item j in PT_i has $sup \geq minSup$, $allConf \geq minAllConf$, $per \leq maxPer$ and $perAllConf \leq maxPerAllConf$, then we construct its conditional tree and mine it recursively to discover the recurring patterns (lines 3 to 9 in Algorithm 3). Moreover, to enable the construction of the prefix-tree for the next item in the EPCP-list, based on Lemma 4, the ts-lists are pushed-up to the respective parent nodes in the original EPCP-tree and in PT_i as well. All nodes of i in the original EPCP-tree and is entry in the EPCP-list are deleted thereafter (line 10 in Algorithm 3).

Using Properties 3 and 4, the conditional tree CT_i for PT_i is constructed by removing all those items from PT_i that have $sup \leq minSup$, or $allConf \leq minAllConf$, or $per \geq maxPer$ or $perAllConf \geq maxPerAllConf$. If the deleted node is a tail node, its ts-list is pushed-up to its parent node. The contents of the temporary array for the bottom item j in the EPCP-list of CT_i represent TS_{ij} (i.e., the set of all timestamps where items i and j have appeared together in the database). The same process of creating a prefix-tree and its corresponding conditional tree is repeated for further extensions of "ij". The whole process of mining for each item is repeated until EPCP-list $\neq \emptyset$.

Table 4 summarizes the working of this algorithm. First, we consider item 'f,' which is the bottom-most item in the EPCP-list, as a suffix pattern. This item appears in three branches of the EPCP-tree (refer Fig. 2(c)). The paths formed by these branches are $\{cef : 4\}$, $\{abef : 10\}$ and $\{aef : 12\}$ (format of these branches is $\{nodes : timestamps\}$). 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. 3(a)). Its conditional EPCP-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 3(b) shows the conditional EPCP-tree of 'f.' The single path generates the pattern $\{ef : 3, 0.75, 6, 1.5\}$ (format is $\{pattern$: $support$, $all\text{-}confidence$, $periodicity$, $periodic\text{-}all\text{-}confidence\}$). 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 EPCP-tree and its ts-lists are pushed to its parent nodes, as shown in Fig. 3(c). All the above processes are once again repeated until EPCP-list $\neq \emptyset$.

Algorithm 3. EPCP-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 TS^{β}, which represents the set of timestamps at which β has appeared in TDB. Next, compute from TS^{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-correlated 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 EPCP-tree $Tree_{\beta}$.
6: **if** $Tree_{\beta} \neq \emptyset$ **then**
7: call EPCP-growth($Tree_{\beta}$, β);
8: **end if**
9: **end if**
10: Remove a_i from the $Tree$ and push a_i's ts-list to its parent nodes.
11: **end for**

Table 4. Mining EPCP-tree by creating conditional (sub -) pattern bases

Item	sup	per	Cond. Pattern Base	Cond. EPCP-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	$\{ab:3\}$, $\{c:3,8,9,11\}$	–	–
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	$\{a:7\}$	–	–

6 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-correlated patterns [20,33,35].

The algorithms, *PF-growth*, *MCPF-growth*, *MaxCPF-growth* and *EPCP-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 [1]. This 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

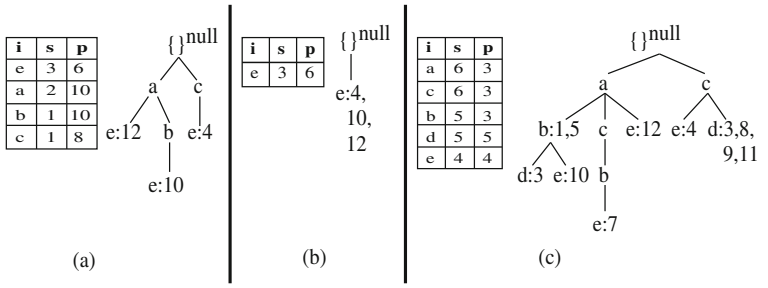


Fig. 3. Mining of EPCP-tree for Table 2. (a) Prefix-tree of suffix item '*f*,' i.e., PT_f (b) Conditional tree of suffix item '*f*,' i.e., CT_f (c) EPCP-tree after pruning item '*f*.'

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.

6.1 Patterns Generated by the Proposed Model

Figure 4(a)–(c) shows the number of patterns generated at different *minAllConf* and *maxPerAllConf* values in T10I4D100K, Retail and FAA-Accidents databases. *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-correlated 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.

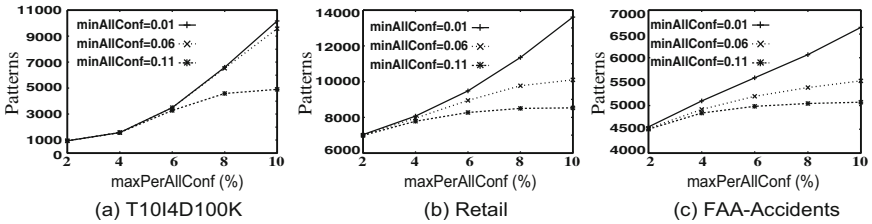


Fig. 4. Periodic-correlated patterns generated at different *maxAllConf* and *maxPerAllConf* values

Figure 5 show the runtime requirements of EPCP-growth at different *maxPerAllConf* and *minAllConf* values in T10I4D100K, Retail and FAA-Accidents databases. The following observations can be drawn: (*i*) The increase in *minAllConf* decreases the runtime of EPCP-growth. The reason is that increase in *minAllConf* decreases the number of periodic-correlated patterns. (*ii*) The increase in *maxPerAllConf* results in increase of runtime of EPCP-growth.

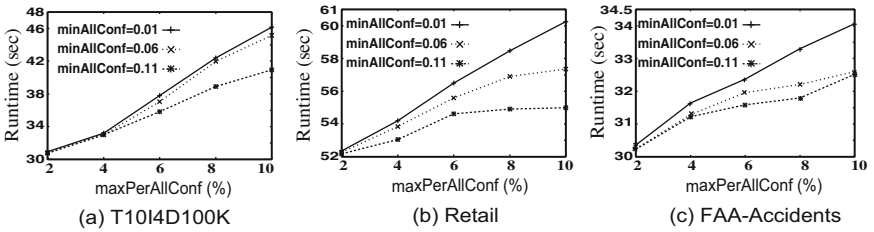


Fig. 5. Runtime requirements of EPCP-growth at different *maxAllConf* and *maxPerAllConf* values

Table 5 shows some of the interesting patterns discovered in FAA database. The *minSup*, *minAllConf*, *maxPer* and *maxPerAllConf* values used are 0.01%, 0.01, 40% and 9, respectively. *It can be observed from their support values that our model has discovered interesting patterns involving both frequent and rare items effectively.* Please note that the *periodicity* (*per*) is expressed in days.

Table 5. Some of the interesting patterns discovered in FAA-Accidents database

S. No.	Patterns	sup	allConf	per	perAllConf
1	{Pilot Not Certificated, Destroyed}	13	0.06	4756	7.77
2	{Student, Substantial}	136	0.02	893	29.77
3	{Boeing, Substantial}	214	0.02	214	26.35
4	{Private-Pilot, Cessna, CE-172, Minor}	1,661	0.03	117	23.4
5	{General Operating Rules, Commercial Pilot, Minor}	10,399	0.15	32	6.4

The first pattern in this table reveals interesting information that 13 aircrafts have been 'destroyed' when piloted by a non-certified pilot. The *periodicity* of this event is 4756 days (≈13 years). The second pattern indicates 136 aircrafts driven by student pilots have suffered substantial damages at least once in every ≈2.5 years. The third pattern indicates that Boeing aircrafts have suffered substantial damages at least once in every ≈7 months. The fourth pattern reveals the information that Cessna airlines CE-172 driven by private pilots have encountered minor damages at least once in every ≈4 months. The last pattern reveals the information that at least once in every 32 days, an aircraft driven by commercial pilots has witnessed minor damages during general operating rules. It can be observed that the first three patterns have low *support* and high *periodicity*. These patterns are often difficult to find with existing approaches due to combinatorial explosion. Thus, the proposed model can efficiently discover useful information pertaining to both frequent and rare items.

6.2 Comparison of Proposed Model Against the Existing Models

For *MCPF-growth* and *MaxCPF-growth*, we use Eq. 3 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 *EPCP-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 *EPCP-growth*.

Figure 6 shows the number of periodic-frequent patterns generated at different $minSup$ values (Y-$axis$ is plotted on logscale). For *EPCP-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. It can be observed that the proposed model has generated less of number of patterns because *all-confidence* has pruned the uninteresting patterns having support much less than the support of individual items.

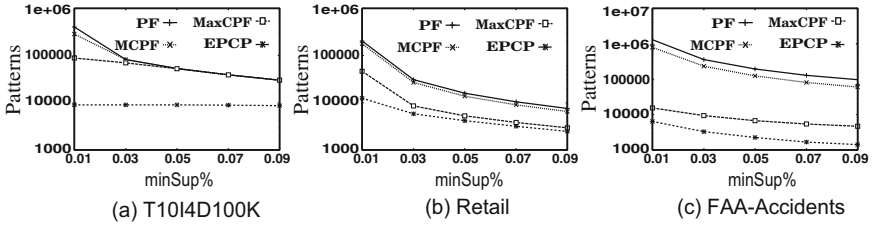


Fig. 6. Periodic-Correlated patterns generated at different minSup values

Figure 7 shows the number of periodic-frequent patterns generated at different $maxPer$ values (Y-$axis$ is plotted on logscale). For *EPCP-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. It can be observed that the proposed model has generated less number of patterns at different $maxPer$ values. It is because *periodic all-confidence* has pruned out those uninteresting patterns whose *periodicity* was much higher than the periodicity of its individual items.

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*.

Figures 8 show the runtime taken by various models at different $minSup$ values (Y-$axis$ is plotted on logscale). It can be observed that, in all the databases

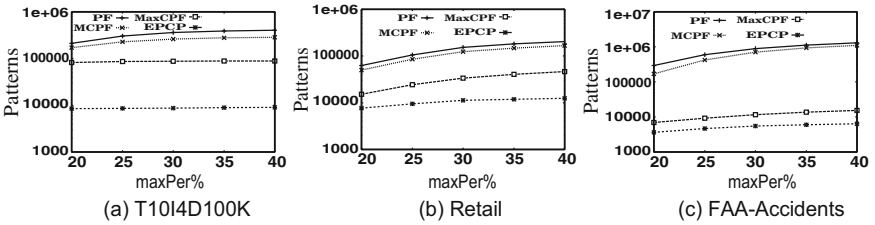


Fig. 7. Periodic-Correlated patterns generated at different maxPer values

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-correlated patterns.

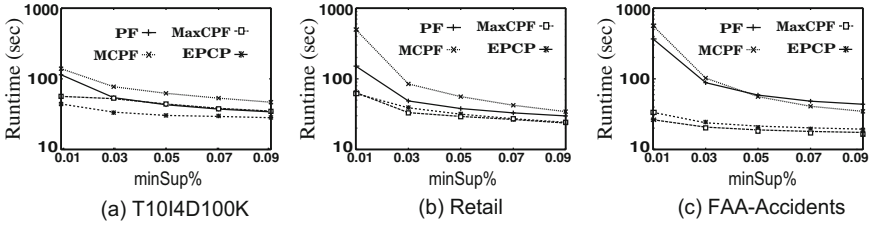


Fig. 8. Runtime requirements of various models at different minSup values

Figure 9 shows the runtime taken by various models at different *maxPer* values (*Y-axis* is plotted on logscale). Similar observations to that of varying *minSup* can be drawn.

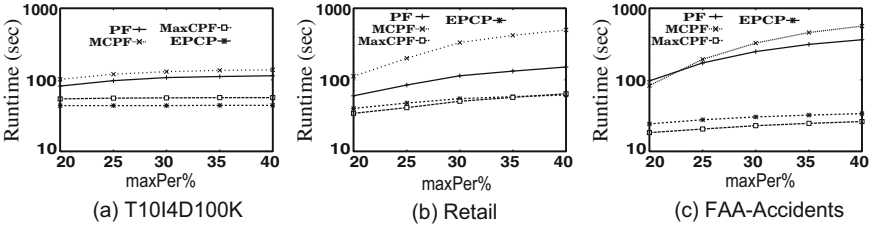


Fig. 9. Runtime requirements of various models at different maxPer values

6.3 Scalability

We studied the scalability of EPCP-growth on execution time by varying the number of transactions in a database. We used Kosarak, T10I4D1000K and T25I6D1000K datasets for this experiment. We divided the database into five equal parts, i.e., 20% transactions in each part. Then we investigated the performance of EPCP-growth by accumulating each part with previous parts and running EPCP-growth each time. Figure 10(a), (b) and (c) show the graph of runtime requirements of EPCP-growth in Kosarak, T10I4D1000K and T25I6D1000K databases, respectively. It is clear from the graphs that as the database size increases, overall tree construction and mining time increase. However, the figure shows stable performance of about linear increase in runtime with respect to the database size.

Figure 11(a), (b) and (c) show the graph of memory requirements of EPCP-growth in Kosarak, T10I4D1000K and T25I6D1000K databases, respectively. Similar observations to that of runtime requirements can be drawn. Therefore, it can be observed from the scalability test that EPCP-growth scales linearly with the increase in database size.

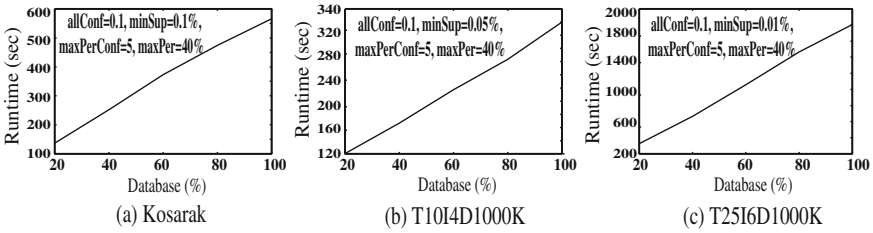


Fig. 10. Runtime requirements of *EPCP*-growth in various databases

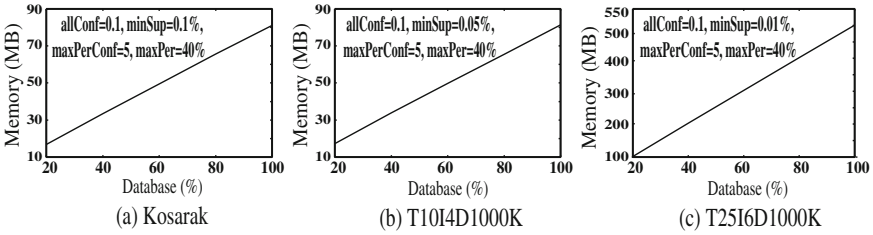


Fig. 11. Memory requirements of *EPCP*-growth in various databases

7 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-correlated 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.

Acknowledgements. This research was partly supported by Real World Information Analytics project of National Institute of Information and Communications Technology, Japan.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207–216 (1993)
2. Amphawan, K., Lenca, P., Surarerks, A.: Mining Top-K periodic-frequent pattern from transactional databases without support threshold. In: Papasratorn, B., Chutimaskul, W., Porkaew, K., Vanijja, V. (eds.) IAIT 2009. CCIS, vol. 55, pp. 18–29. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10392-6_3
3. Anirudh, A., Kirany, R.U., Reddy, P.K., Kitsuregaway, M.: Memory efficient mining of periodic-frequent patterns in transactional databases. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8, December 2016
4. Aref, W.G., Elfeky, M.G., Elmagarmid, A.K.: Incremental, online, and merge mining of partial periodic patterns in time-series databases. IEEE TKDE **16**(3), 332–342 (2004). Mar
5. Bradshaw, J.: Yams - yet another measure of similarity. In: EuroMUG (2001). http://www.daylight.com/meetings/emug01/Bradshaw/Similarity/YAMS.html
6. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. In: SIGMOD, pp. 265–276 (1997)
7. Cao, H., Cheung, D.W., Mamoulis, N.: Discovering partial periodic patterns in discrete data sequences. In: Dai, H., Srikant, R., Zhang, C. (eds.) PAKDD 2004. LNCS (LNAI), vol. 3056, pp. 653–658. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24775-3_77
8. Chen, S.S., Huang, T.C.K., Lin, Z.M.: New and efficient knowledge discovery of partial periodic patterns with multiple minimum supports. J. Syst. Softw. **84**(10), 1638–1651 (2011). Oct
9. Deng, Z.H.: Diffnodesets: an efficient structure for fast mining frequent itemsets. Appl. Soft Comput. **41**, 214–223 (2016). http://www.sciencedirect.com/science/article/pii/S156849461600017X
10. Deng, Z.H., Lv, S.L.: Prepost+: an efficient n-lists-based algorithm for mining frequent itemsets via childrenparent equivalence pruning. Expert Syst. Appl. **42**(13), 5424–5432 (2015). http://www.sciencedirect.com/science/article/pii/S0957417415001803
11. Fournier-Viger, P., Lin, J.C.-W., Duong, Q.-H., Dam, T.-L.: PHM: mining periodic high-utility itemsets. In: Perner, P. (ed.) ICDM 2016. LNCS (LNAI), vol. 9728, pp. 64–79. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41561-1_6
12. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: Current status and future directions. DMKD **14**(1) (2007)
13. Han, J., Dong, G., Yin, Y.: Efficient mining of partial periodic patterns in time series database. In: ICDE, pp. 106–115 (1999)

14. Han, J., Gong, W., Yin, Y.: Mining segment-wise periodic patterns in time-related databases. In: KDD, pp. 214–218 (1998)
15. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. **8**(1), 53–87 (2004). Jan
16. Hu, Y.H., Chen, Y.L.: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decis. Support Syst. **42**(1), 1–24 (2006)
17. Kim, S., Barsky, M., Han, J.: Efficient mining of top correlated patterns based on null-invariant measures. In: PKDD, pp. 177–192 (2011)
18. Kim, W.Y., Lee, Y.K., Han, J.: Ccmine: efficient mining of confidence-closed correlated patterns. In: Advances in Knowledge Discovery and Data Mining, pp. 569–579 (2004)
19. Kiran, R.U., Kitsuregawa, M.: Novel techniques to reduce search space in periodic-frequent pattern mining. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014. LNCS, vol. 8422, pp. 377–391. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05813-9_25
20. 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. LNCS, vol. 6262, pp. 194–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15251-1_16
21. Uday Kiran, R., Krishna Reddy, P.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: EDBT, pp. 11–20 (2011)
22. Uday Kiran, R., Shang, H., Toyoda, M., Kitsuregawa, M.: Discovering partial periodic itemsets in temporal databases. In: Proceedings of the 29th International Conference on Scientific and Statistical Database Management, Chicago, IL, USA, 27–29 June 2017, pp. 30:1–30:6 (2017). http://doi.acm.org/10.1145/3085504.3085535
23. Uday Kiran, R., Venkatesh, J.N., Fournier-Viger, P., Toyoda, M., Reddy, P.K., Kitsuregawa, M.: Discovering periodic patterns in non-uniform temporal databases. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10235, pp. 604–617. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57529-2_47
24. Uday Kiran, R., Venkatesh, J., Toyoda, M., Kitsuregawa, M., Reddy, P.K.: Discovering partial periodic-frequent patterns in a transactional database. J. Syst. Softw. **125**, 170–182 (2017). http://www.sciencedirect.com/science/article/pii/S0164121216302382
25. Lee, Y.K., Kim, W.Y., Cao, D., Han, J.: Comine: efficient mining of correlated patterns. In: ICDM, pp. 581–584 (2003)
26. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: KDD, pp. 337–341 (1999)
27. Nofong, V.M.: Discovering productive periodic frequent patterns in transactional databases. Ann. Data Sci. **3**(3), 235–249 (2016)
28. Omiecinski, E.R.: Alternative interest measures for mining associations in databases. IEEE Trans. Knowl. Data Eng. **15**, 57–69 (2003)
29. Özden, B., Ramaswamy, S., Silberschatz, A.: Cyclic association rules. In: ICDE, pp. 412–421 (1998)
30. Pyun, G., Yun, U., Ryu, K.H.: Efficient frequent pattern mining based on linear prefix tree. Knowl. Based Syst.**55**, 125–139 (2014). http://www.sciencedirect.com/science/article/pii/S0950705113003249

31. Rashid, M.M., Karim, M.R., Jeong, B.-S., Choi, H.-J.: Efficient mining regularly frequent patterns in transactional databases. In: Lee, S., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7238, pp. 258–271. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29038-1_20
32. Surana, A., Uday Kiran, R., Krishna Reddy, P.: Selecting a right interestingness measure for rare association rules. In: International Conference on Management of Data, pp. 105–115 (2010)
33. Surana, A., Uday Kiran, R., Krishna Reddy, P.: An efficient approach to mine periodic-frequent patterns in transactional databases. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 254–266. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28320-8_22
34. Tan, P.N., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Knowledge Discovery and Data Mining, pp. 32–41 (2002)
35. 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 (LNAI), vol. 5476, pp. 242–253. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01307-2_24
36. Uno, T., Kiyomi, M., Arimura, H.: LCM ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining. In: Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations, OSDM 2005, pp. 77–86. ACM, New York (2005). http://doi.acm.org/10.1145/1133905.1133916
37. Vaillant, B., Lenca, P., Lallich, S.: A clustering of interestingness measures. In: Suzuki, E., Arikawa, S. (eds.) DS 2004. LNCS (LNAI), vol. 3245, pp. 290–297. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30214-8_23
38. Venkatesh, J.N., Uday Kiran, R., Krishna Reddy, P., Kitsuregawa, M.: Discovering periodic-frequent patterns in transactional databases using all-confidence and periodic-all-confidence. In: Hartmann, S., Ma, H. (eds.) DEXA 2016. LNCS, vol. 9827, pp. 55–70. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-44403-1_4
39. Wu, T., Chen, Y., Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework. DMKD **21**(3), 371–397 (2010)
40. Xiong, H., Tan, P.N., Kumar, V.: Hyperclique pattern discovery. Data Mining Knowl. Discov. **13**(2), 219–242 (2006)
41. Yang, J., Wang, W., Yu, P.S.: Mining asynchronous periodic patterns in time series data. IEEE Trans. Knowl. Data Eng. **15**, 613–628 (2003)
42. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. Technical report, Rochester, NY, USA (1997)
43. Zhang, M., Kao, B., Cheung, D.W., Yip, K.Y.: Mining periodic patterns with gap requirement from sequences. ACM Trans. Knowl. Discov. Data **1**(2), August 2007
44. Zhou, Z., Wu, Z., Wang, C., Feng, Y.: Efficiently Mining Mutually and Positively Correlated Patterns. In: Li, X., Zaïane, O.R., Li, Z. (eds.) ADMA 2006. LNCS (LNAI), vol. 4093, pp. 118–125. Springer, Heidelberg (2006). https://doi.org/10.1007/11811305_12
45. Zhou, Z., Wu, Z., Wang, C., Feng, Y.: Mining both associated and correlated patterns. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3994, pp. 468–475. Springer, Heidelberg (2006). https://doi.org/10.1007/11758549_66