Towards Efficient Mining of Periodic-Frequent Patterns in Transactional Databases

R. Uday Kiran and P. Krishna Reddy

Center for Data Engineering
International Institute of Information Technology-Hyderabad
Hyderabad, India - 500032
uday_rage@research.iiit.ac.in, pkreddy@iiit.ac.in
http://research.iiit.ac.in/~uday_rage,http://iiit.ac.in/~pkreddy

Abstract. Periodic-Frequent patterns are an important class of regularities that exist in a transactional database. A pattern is *periodic-frequent* if it satisfies both minimum support (*minsup*) and maximum periodicity (*maxprd*) constraints. *Minsup* constraint controls the minimum number of transactions that a pattern must cover in a database. *Maxprd* constraint controls the maximum duration between the two transactions below which a pattern should reoccur in a database. In the literature an approach has been proposed to extract periodic-frequent patterns using single *minsup* and single *maxprd* constraints. However, real-world databases are mostly non-uniform in nature containing both frequent and relatively infrequent (or rarely) occurring items. Researchers are making efforts to propose improved approaches for extracting frequent patterns that contain rare items as they contain useful knowledge. For mining periodic patterns that contain frequent and rare items we have to specify low *minsup* and high *maxprd*. It is difficult to mine periodic-frequent patterns because the low *minsup* and high *maxprd* can cause combinatorial explosion. In this paper we propose an improved approach which facilitates the user to specify different *minsup* and *maxprd* values for each pattern depending upon the items within it. Also, we present an efficient pattern growth approach and a methodology to dynamically specify *maxprd* for each pattern. Experimental results show that the proposed approach is efficient.

Keywords: Data mining, frequent pattern, rare periodic-frequent pattern, multiple constraints.

1 Introduction

Periodic-frequent patterns [3] are an important class of regularities that exist in a database. In many real-world applications, these patterns provide useful information regarding the patterns which are not only occurring frequently, but also appearing periodically (or regularly) throughout a transactional database. The basic model of periodic-frequent patterns is as follows [3].

Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items. A set $X = \{i_j, \cdots, i_k\} \subseteq I$, where $j \leq k$ and $j, k \in [1, n]$, is called a **pattern** (or an itemset). A transaction

$t = (tid, Y)$ is a tuple, where tid represents a transaction-id (or a timestamp) and Y is a pattern. A transactional database T over I is a set of transactions, $T = \{t_1, \cdots, t_m\}$, $m = |T|$, where $|T|$ is the size of T in total number of transactions. If $X \subseteq Y$, it is said that t contains X or X occurs in t and such transaction-id is denoted as t_j^X, $j \in [1, m]$. Let $T^X = \{t_k^X, \cdots, t_l^X\} \subseteq T$, where $k \leq l$ and $k, l \in [1, m]$ be the ordered set of transactions in which pattern X has occurred. Let t_j^X and t_{j+1}^X, where $j \in [k, (l-1)]$ be two consecutive transactions in T^X. The number of transactions or time difference between t_{j+1}^X and t_j^X can be defined as a ***period*** of X, say p^X. That is, $p^X = t_{j+1}^X - t_j^X$. Let $P^X = \{p_1^X, p_2^X, \cdots, p_r^X\}$, be the set of periods for pattern X. The ***periodicity*** of X, denoted as $Per(X) = max(p_1^X, p_2^X, \cdots, p_r^X)$. The ***support*** of X, denoted as $S(X) = |T^X|$. The pattern X is said to be periodic-frequent pattern, if $S(X) \geq minsup$ and $Per(X) \leq maxprd$, where $minsup$ and $maxprd$ are user-specified minimum support and maximum periodicity constraints. Both periodicity and support of a pattern can be described in percentage of $|T|$.

Table 1. Transaction database. Transactions are ordered based on timestamp.

TID	Items	TID	Items
1	bread, jam, pencil	7	bread, jam, ball, bat
2	ball, bat, pen		
3	bread, jam, ball	8	bed, pillow
4	bed, pillow	9	bread, jam
5	bread, jam	10	ball, bat pencil
6	ball, bat		

Table 2. Periodic-Frequent patterns having support ≥ 2 and *periodicity* ≤ 4

Pattern	S	P	Pattern	S	P
bread	5	2	**{bread,ball}**	2	4
ball	5	3	{bread, jam}	5	2
jam	5	2	**{ball,bat}**	4	4
bat	4	4	{bed, pillow}	2	4
bed	2	4			
pillow	2	4			

Example 1. Consider the transactional database shown in Table 1. Each transaction in it is uniquely identifiable with a transactional-id (tid) which is also a timestamp of that transaction. Timestamp indicates time of occurrence of the transaction. The set of items, $I = \{bread, jam, ball, bat, bed, pillow, pencil, pen\}$. The set of *bread* and *jam* i.e., $\{bread, jam\}$ is a pattern. This pattern occurs in *tids* of $1, 3, 5, 7$ and 9. Therefore, $T^{\{bread,jam\}} = \{1, 3, 5, 7, 9\}$. Its support count (or support), $S(bread, jam) = |T^{\{bread,jam\}}| = 5$. The periods for this pattern are $1 (= 1 - t_i)$, $2 (= 3 - 1)$, $2 (= 5 - 3)$, $2 (= 7 - 5)$, $2 (= 9 - 7)$ and $1 (= t_l - 9)$, where $t_i = 0$ represents the initial transaction and $t_l = 10$ represents the last transaction in the transactional database. The periodicity of $\{bread, jam\}$, $Per(bread, jam) = maximum(1, 2, 2, 2, 2, 1) = 2$. If the user-specified $minsup = 4$ and $maxprd = 2$, the pattern $\{bread, jam\}$ is a periodic-frequent pattern because $S(bread, jam) \geq minsup$ and $Per(bread, jam) \leq maxprd$.

For this model, an efficient pattern growth approach based on a tree-structure, called Periodic-Frequent tree (PF-tree) was also discussed to discover complete set of periodic-frequent patterns [3]. The structure of PF-tree is different from the FP-tree [2]. It is because FP-tree was not proposed to consider the periodicity of a pattern.

Using only a single *minsup* and single *maxprd* constraints, it is easy to discover periodic-frequent patterns consisting of frequent items. However, real-world databases are mostly non-uniform in nature containing both frequent and relatively infrequent (or rarely) occurring items. More important, periodic-frequent patterns consisting of rare items i.e., rare periodic-frequent patterns can provide useful information.

Example 2. Generally in a supermarket, the set of items *bed* and *pillow* are rarely purchased than the set of items *bread* and *jam*. Also, the duration of two consecutive purchases of '*bed* and *pillow*' is relatively longer than the two consecutive purchases of '*bread* and *jam*'. However, the former set of items is more interesting as it generates more revenue per unit as in this case.

Rare periodic-frequent patterns have relatively low support (or frequency) and high periodicity (due to their sporadic nature). It is difficult to mine these patterns with a "single *minsup* and single *maxprd* model" because this model suffers from "rare item problem." That is, to mine rare periodic-frequent patterns, one has to specify low *minsup* and high *maxprd*. This may cause combinatorial explosion, producing too many periodic-frequent patterns, because, those frequent items will be associated with one another in all possible ways and many of them are uninteresting. Uninteresting periodic-frequent patterns are the patterns having low support and/or high periodicity and consist of only frequent items.

Example 3. Consider the transactional database shown in Table 1. To mine periodic-frequent patterns consisting of the rare items (*bed* and *pillow*), one has to set low *minsup* and high *maxprd*. Let $minsup = 2$ and $maxprd = 4$. Table 1 presents the discovered periodic-frequent patterns. It can be observed that along with the interesting patterns $\{bread, jam\}$ and $\{bed, pillow\}$, the uninteresting patterns i.e., $\{\boldsymbol{bread, ball}\}$ and $\{\boldsymbol{ball, bat}\}$ (patterns represented in bold letters) have also been generated as periodic-frequent patterns. The patterns $\{\boldsymbol{bread, ball}\}$ and $\{\boldsymbol{ball, bat}\}$ are uninteresting because they contain only frequent items and have low support and/or high periodicity. These patterns can be considered as interesting if they have satisfied high *minsup* and low *maxprd*, say $minsup = 4$ and $maxprd = 2$. Like, the periodic-frequent pattern $\{bread, jam\}$.

In the literature, "rare item problem" was also confronted while mining frequent patterns using "single *minsup* model." Efforts are being made to propose improved approaches using "multiple *minsup* model" [4,5,6,7,8]. In this model, each item is specified with a support constraint, called minimum item support (MinIS), and *minsup* of a pattern is represented with the minimal $MinIS$ value among all its items. Thus, each pattern can satisfy a different *minsup* depending upon the items within it. In [4,6], methodologies have been discussed to specify items' $MinIS$ values depending upon their respective supports.

In this paper, we extend the existing "multiple *minsup* model" to "multiple *minsup* and multiple *maxprd* model" to efficiently mine periodic-frequent patterns consisting of both frequent and rare items. In the proposed model, each pattern can satisfy a different *minsup* and *maxprd* values depending upon the items within it. Specifically, the user specifies two types of constraints: (*i*) support constraint, called *minimum item support* (*MinIS*) and (*ii*) periodicity constraint, called *maximum item periodicity* (*MaxIP*). Thus, different patterns may need to satisfy different *minsup* and *maxprd* values depending upon the items within it.

The periodic-frequent patterns mined using the proposed model do not satisfy *downward closure property.* That is, not all non-empty subsets of a periodic-frequent pattern need be periodic-frequent. This increases the search space to discover complete set of periodic-frequent patterns. However, we propose an efficient pattern growth approach, which uses various techniques to minimize the search space for efficient mining of periodic-frequent patterns. Experimental results on both synthetic and real-world databases show that the proposed approach efficiently discover periodic-frequent patterns consisting of both frequent and rare items. However, it requires more runtime because the periodic-frequent patterns mined using the proposed model do not satisfy *downward closure property.*

The rest of the paper is organized as follows. In Section 2, we introduce the extended model of mining periodic-frequent patterns. For the proposed model, a pattern-growth approach based on a tree structure, called Multi-Constraint Periodic-Frequent tree (MCPF-tree) has been discussed in Section 3. We report our experimental results in Section 4. Finally, Section 5 concludes the paper.

2 The Extended Model

In the new model, each item in a transactional database has two types of constraints: a support constraint, called *minimum item support* (*MinIS*) and a periodicity constraint, called *maximum item periodicity* (*MaxIP*). A pattern is *periodic-frequent* if it satisfies lowest *MinIS* and maximum *MaxIP* values of all the items within it.

Continuing with the basic model of periodic-frequent patterns, let $MinIS(i_j)$ and $MaxIP(i_j)$ be the *minimum item support* and *maximum item periodicity* specified for an item $i_j \in I$. Then, a pattern $X = \{i_1, i_2, \cdots, i_k\} \subseteq I$ is periodic-frequent if:

$$\begin{aligned} S(X) &\geq minimum(MinIS(i_1), MinIS(i_2), \cdots, MinIS(i_k)) \\ &and \\ Per(X) &\leq maximum(MaxIP(i_1), MaxIP(i_2), \cdots, MaxIP(i_k)) \end{aligned} \tag{1}$$

Minimum item supports and maximum item periodicities enable us to achieve the goal of specifying higher *minsup* and lower *maxprd* for patterns that only involve frequent items, and specifying lower *minsup* and higher *maxprd* for patterns involving rare items.

2.1 Specifying *MaxIP* for an Item

If there exists numerous items within a transactional database, it will be very difficult for the user to manually specify *MinIS* and *MaxIP* values for every item. In the literature (multiple *minsup* based frequent pattern mining), there exists methodologies to specify items' *MinIS* values dynamically depending upon their respective supports [4,6]. In this paper, we propose a methodology to specify items' *MaxIP* values dynamically depending upon their support values. The methodology is as follows:

$$\begin{aligned} mip(i_j) &= \beta \times S(i_j) + Per_{max} \\ MaxIP(i_j) &= mip(i_j) \quad if \;\; mip(i_j) \geq Per_{min} \\ &= Per_{min} \qquad otherwise \end{aligned} \tag{2}$$

where, $S(i_j)$ is the support of the item i_j, 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. The above methodology has the following three properties.

Property 1. If $\beta = 0$ and $Per_{max} > Per_{min}$, each items' *MaxIP* value will be equal to P_{max}. In such a scenario, the proposed model is same as mining periodic-frequent patterns with a single *maxprd* constraint, where $maxprd = Per_{max}$.

Property 2. If $Per_{max} = Per_{min}$, each items' *MaxIP* value will be equal to Per_{max} or Per_{min}. The proposed model is same as mining periodic-frequent patterns with a single *maxprd* constraint, where $maxprd = Per_{max} = Per_{min}$.

Property 3. It is an order-reversing function. That is, in I, if $S(i_1) \leq S(i_2) \leq \cdots S(i_n)$, then $MaxIP(i_1) \geq MaxIP(i_2) \geq \cdots \geq MaxIP(i_n)$. Thus, as compared with frequent items, rare items will have high *MaxIP* values.

2.2 Nature of the Periodic-Frequent Patterns

The periodic-frequent patterns mined using "single *minsup* and single *maxprd* model" satisfy *downward closure property.* That is, all non-empty subsets of a periodic-frequent pattern are periodic-frequent. However, the periodic-frequent patterns mined using the proposed model do not have to satisfy *downward closure property.*

Example 4. Let a, b and c be the three items in a transactional database. The user-specified *MinIS* values for these items be 10, 9 and 3 respectively. The user-specified *MaxIP* values for these items be 5, 6 and 5 respectively. After scanning the database, let the support of these respective items be 9, 8 and 4. Let the periodicity of these items be 4, 4 and 6 respectively. Clearly, the items a, b and c are non-periodic-frequent items (or 1-patterns) because $S(a) < MinIS(a)$, $S(b) < MinIS(b)$ and $P(c) > MaxIP(c)$. However, their superset i.e., $\{a, b, c\}$ with support=3 and periodicity=6 can be still be generated as a periodic-frequent pattern because supports of a, b and c are greater than or equal to $MinIS(c)$ and periodicities of a, b and c are less than or equal to $MaxIP(b)$. So, if we do not discard these non-periodic-frequent items, the *downward closure property* is lost.

2.3 Problem Definition

Given a transactional database T, items' $MinIS$ and $MaxIP$ values, discover complete set of periodic-frequent patterns that satisfy lowest $MinIS$ and maximum $MaxIP$ values of all the items within the respective pattern.

3 MCPF-Tree: Design, Construction and Mining

In this section, we describe the structure, construction and mining of periodic-frequent patterns using Multi-Constraint Periodic-Frequent Pattern-tree (MCPF-tree).

3.1 Structure of MCPF-Tree

The MCPF-tree consists of two components: MCPF-list and a prefix-tree. MCPF-list is a list with four fields: item, support (S), periodicity (P), $MinIS$ (mis) and $MaxIP$ (mip). The node structure of prefix-tree in MCPF-tree is same as the prefix-tree in PF-tree [3], which is as follows.

The prefix-tree in MCPF-tree explicitly maintains the occurrence information for each transaction in the tree structure by keeping an occurrence transaction-id list, called *tid*-list, only at the last node of every transaction. Two types of nodes are maintained in a MCPF-tree: ordinary node and *tail*-node. The ordinary node is similar to the nodes used in FP-tree, whereas the latter is the node that represents the last item of any sorted transaction. The structure of a *tail*-node is $N[t_1, t_2, ..., t_n]$, where N is the node's item name and $t_i, i \in [1, n]$, (n be the total number of transactions from the root up to the node) is a transaction-id where item N is the last item. Like the FP-tree [2], each node in a MCPF-tree maintains parent, children, and node traversal pointers. However, irrespective of the node type, no node in a MCPF-tree maintains support count value in it. We now explain construction and mining of MCPF-tree.

3.2 Constructing MCPF-Tree

Let id_l be a temporary array to record the *tids* of the last occurring transactions of all items in the MCPF-list. Let t_{cur} and p_{cur} respectively denote the *tid* of current transaction and the most recent period for an item $i_j \in I$. The MCPF-tree is, therefore, maintained according to the process given in Algorithm 1.

Consider the transactional database shown in Table 1. Let the user-specified $MinIS$ values for the items $bread, ball, bat, jam, bed, pillow, pen$ and $pencil$ be 4, 4, 4, 4, 2, 2, 2 and 2, respectively. Let the user-specified $MaxIP$ values for these items be 2, 2, 2, 2, 4, 4, 4 and 4, respectively. Then, $L = \{bread, ball, bat, jam, bed, pillow, pen, pencil\}$.

In Fig. 1, we show how the MCPF-list is populated for the transactional database shown in Table 1. Fig. 1(a) shows the MCPF-list populated after inserting items in L order (Line 1 of Algorithm 1). Fig. 1(b), Fig. 1(c) and Fig. 1(d)

Algorithm 1. MCPF-tree (T: Transactional database, I: set of items, $MinIS$: items' minimum item support, $MaxIP$: items' maximum item periodicity)

```
1: Sort the items in descending order of their $MinIS$ values. Let this order of items be
   $L$. In $L$ order, insert each item $i_j \in I$ into the MCPF-list with $S(i_j) = 0$, $P(i_j) = 0$,
   $mis(i_j) = MinIS(i_j)$ and $mip(i_j) = MaxIP(i_j)$.
2: for each transaction $t_{cur} \in T$ do
3:    for each $i_j \in t_{cur}$ do
4:       $S(i_j) + +$; $p_{cur} = t_{cur} - id_l(i_j)$;
5:       if ($p_{cur} > P(i_j)$) then
6:          $P(i_j) = p_{cur}$;
7:       end if
8:    end for
9: end for
10: At the end of $T$, calculate $p_{cur}$ for each item by considering $t_{cur}$ equal to the $tid$
    of the last transaction in $T$, and update their respective $p$ value if $p_{cur} \geq P$. The
    purpose of this step is to reflect correct periodicity of each item in the MCPF-list.
11: repeat
12:    Let $i_k \in I$ be the item having lowest $MinIS$ ($MIS_{min}$) value among all frequent
       items.
13:    for each item $i_j$ in MCPF-list do
14:       if ($S(i_j) < MIS_{min}$) then
15:          Remove $i_j$ from the MCPF-list.
16:       end if
17:    end for
18:    Let $i_l \in I$ be the periodic item that has maximum $MaxIP$ ($MIP_{max}$) value
       among all the remaining items.
19:    for each item $i_j$ in MCPF-list do
20:       if (($P(i_j) > MIP_{max}$) then
21:          Remove $i_j$ from the MCPF-list.
22:       end if
23:    end for
24: until MCFP-list does not contain $i_k$
25: /* The above repeat step is necessary because $i_k$ can be pruned if its periodicity
    is greater than the $MIP_{max}$ value. */
26: Let $L'$ be the sorted list of items in MCPF-list.
27: for each transaction $t \in T$ do
28:    Sort the items in $L'$ order and create a branch in MCPF-tree as in PF-tree.
29: end for
30: For tree-traversal, maintain node-links in MCPF-tree as in PF-tree.
```

show the MCPF-list generated after scanning first (i.e., $t_{cur} = 1$), second (i.e., $t_{cur} = 2$) and every transaction (i.e., $t_{cur} = 10$), respectively (lines 2 to 9 in Algorithm 1). To reflect the correct periodicity for each item in the MCPF-tree, the whole MCPF-list is refreshed as mentioned in line 10 of Algorithm 1. The resultant MCPF-tree is shown in Fig. 1(e). In this figure, it can be observed that the periodicity of *pen* is changed from 2 to 8 as it did not appear in the database after $tid = 2$.

The periodic-frequent patterns mined using the proposed model do not satisfy *downward closure property.* This increases the search space for mining these patterns. To minimize the search space, we explore Lemma 1 and Lemma 2. The lowest MinIS value (MIS_{min}) among all frequent items in the MCPF-list is 2 (Line 12 in Algorithm 1). The item that has MIS_{min} value is *pencil*. Using $MIS_{min} = 2$, *pen* is pruned from the MCPF-list because its support is less than the MIS_{min} value (Line 13 to 17 in Algorithm 1). Among the remaining items, the periodic items are *bread*, *jam*, *ball*, *bat*, *bed* and *pillow*. The maximum $MaxIP$ value (MIP_{max}) among all these items is 4 (Line 18 in Algorithm 1). Using $MIP_{max} = 4$, *pencil* is pruned from the MCPF-list because its periodicity is greater than MIS_{min} (Line 19 to 23 in Algorithm 1). As the item *pencil* that represents the frequent item having lowest $MinIS$ is pruned, the above steps of finding new $MinIS$ is repeated (Line 14 in Algorithm 1). The new MIS_{min} is 2. The item *pillow* has MIS_{min} among all the remaining frequent items. Every item in MCPF-list has support than or equal to MIS_{min}. Therefore, no item is pruned from the MCPF-list. The MIP_{max} value among the remaining in MCPF-list is 4. Every item in the MCPF-list satisfies MIP_{max}. Therefore, no item is pruned from the MCPF-list. As the item *pillow* that represents the item having MIS_{min} is not pruned from the MCPF-list, the pruning process is completed. The resultant (compact) MCPF-list is shown in Fig. 1(f). Let L' be the new list of items in MCPF-list that are sorted in descending order of their MIS values.

Using L', perform second scan on the transactional database to construct prefix-tree in MCPF-tree. The construction of prefix-tree in MCPF-tree is same as the construction of prefix-tree in PF-tree (or FP-tree). Figure 2(a), Figure 2(b) and Figure 2(c) show the construction of MCPF-tree after scanning first, second and every transaction in the transactional database. In MCPF-tree, node-links are maintained as in FP-growth. For simplicity of figures, we are omitting them.

Lemma 1. *An item is* frequent *if its support is greater than or equal to its MinIS value. Any item which has support less than the lowest MinIS value among all* ***frequent items*** *cannot generate any periodic-frequent pattern.*

Proof. In the proposed model, an item which has the lowest $MinIS$ value within a periodic-frequent pattern is a frequent item (apriori property [1]). Therefore, every periodic-frequent pattern will have support greater than or equal to the lowest $MinIS$ value among all frequent items. Thus, any item which has support less than the lowest $MinIS$ value among all frequent items cannot generate any periodic-frequent pattern.

item	S	P	MIS	MIP	ld_1
bread	0	0	4	2	0
jam	0	0	4	2	0
ball	0	0	4	2	0
bat	0	0	4	2	0
bed	0	0	2	4	0
pillow	0	0	2	4	0
pencil	0	0	2	4	0
pen	0	0	2	4	0

a) Before scanning the transactional dataset

item	S	P	MIS	MIP	ld_1
bread	1	1	4	2	1
jam	1	1	4	2	1
ball	0	0	4	2	0
bat	0	0	4	2	0
bed	0	0	2	4	0
pillow	0	0	2	4	0
pencil	1	1	2	4	1
pen	0	0	2	4	0

b) After scanning first transaction

item	S	P	MIS	MIP	ld_1
bread	1	1	4	2	1
jam	1	1	4	2	1
ball	1	2	4	2	2
bat	1	2	4	2	2
bed	0	0	2	4	0
pillow	0	0	2	4	0
pencil	1	1	2	4	1
pen	1	2	2	4	2

c) After scanning second transaction

item	S	P	MIS	MIP	ld_1
bread	5	2	4	2	9
jam	5	2	4	2	9
ball	5	3	4	2	10
bat	4	4	4	2	10
bed	2	4	2	4	8
pillow	2	4	2	4	8
pencil	2	9	2	4	10
pen	1	2	2	4	2

d) After scanning last transaction

item	S	P	MIS	MIP
bread	5	2	4	2
jam	5	2	4	2
ball	5	3	4	2
bat	4	4	4	2
bed	2	4	2	4
pillow	2	4	2	4
pencil	2	9	2	4
pen	**1**	**8**	**2**	**4**

e) Updated the MCPF-list

item	S	P	MIS	MIP
bread	5	2	4	2
jam	5	2	4	2
ball	5	3	4	2
bat	4	4	4	2
bed	2	4	2	4
pillow	2	4	2	4

f) Final MCPF-list

Fig. 1. Construction of MCPF-list

Lemma 2. *An item is periodic if its periodicity is less than or equal to its $MaxIP$ value. Let I' be the set of items which have support greater than or equal to lowest $MinIS$ value among all frequent items. Any item which has periodicity greater than the maximum $MaxIP$ value among all periodic items in I' cannot generate any periodic-frequent pattern.*

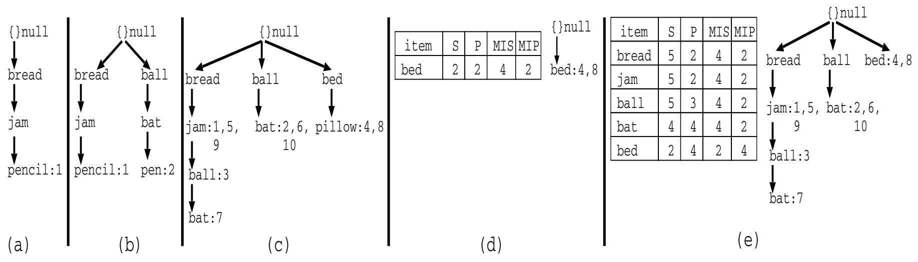
Proof. In the proposed model, an item which has maximum $MaxIP$ value in a periodic-frequent pattern is a periodic item (apriori property). Therefore, every periodic-frequent item will have periodicity less than or equal to maximum $MaxIP$ of all periodic items in I'. Hence, any item which has periodicity greater than the maximum $MaxIP$ value among all periodic items in I' cannot generate any periodic-frequent pattern.

The correctness of MCPF-tree is based on Property 4 and Lemma 3.

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

Lemma 3. *Let $P = \{i_1, i_2, \cdots, i_n\}$ be a path in a MCPF-tree where node i_n is the tail-node carrying the tid-list of the path. If the tid-list is pushed-up to node i_{n-1}, then i_{n-1} maintains the appearance information of the path $P' = \{i_1, i_2, \cdots, i_{n-1}\}$ for the same set of transactions in the tid-list without any loss.*

Proof. Based on Property 4, i_n maintains the occurrence information of the path P' at least in the transactions in its *tid*-list. Therefore, the same *tid*-list at node i_{n-1} exactly maintains the same transaction information for P' without any loss.



item	S	P	MIS	MIP
bed	2	2	4	2

item	S	P	MIS	MIP
bread	5	2	4	2
jam	5	2	4	2
ball	5	3	4	2
bat	4	4	4	2
bed	2	4	2	4

Fig. 2. Construction and mining MCPF-tree. (a) Prefix-tree after scanning first transaction (b) Prefix-tree after scanning second transaction (c) Prefix-tree after scanning every transaction (d) PT_{pillow} and CT_{pillow} and (e) MCPF-tree after pruning item *pillow*.

3.3 Mining of *MCPF-Tree*

Consider the bottom-most item, say i, of the *MCPF-list*. For i, construct a prefix-tree, say PT_i by accumulating only the prefix sub-paths of nodes i. Since i is the bottom-most item in the MCPF-list, each node labeled i in the MCPF-tree must be a tail-node. While constructing the PT_i, based on Property 4 we map the *tid*-list of every node of i to all items in the respective path explicitly in a temporary array (one for each item). It facilitates the periodicity and support calculation for each item in the MCPF-list of PT_i.

Lemma 4. *Let $S_i(j)$ be the support of an item j in PT_i. If $S_i(j) < MinIS(i)$, j cannot generate any periodic-frequent pattern in PT_i.*

Proof. The MCPF-tree is constructed in $MinIS$ descending order of items. So, all items in PT_i will have $MinIS$ values greater than or equal to the $MinIS$ of i. Thus, based on *apriori property*, an item j having $S_i(j) < MinIS(i)$ cannot generate any periodic-frequent pattern in PT_i.

Lemma 5. *In PT_i, let I'' be the set of items which have their support greater than or equal to $MinIS$ of i. In I'', any item which has periodicity greater than the maximum $MaxIP$ value among all $\{I'' \cup i\}$ items cannot generate any periodic-frequent pattern in PT_i.*

Proof. Any periodic-frequent pattern that can be generated from PT_i cannot have periodicity greater than the maximum $MaxIP$ value among the set of $\{I'' \cup i\}$. Therefore, any item in I'', which has periodicity greater than maximum $MaxIP$ value among all $\{I'' \cup i\}$ items cannot generate any periodic-frequent pattern in PT_i (*apriori property*).

Using Lemma 4 and Lemma 5, the conditional tree CT_i for PT_i is constructed by removing all those items which have support less than the $MinIS$ of i (say, $minsup_i$) or periodicity greater than the maximum $MaxIP$ of all items in $\{PT_i \cup i\}$ (say, $maxprd_i$). If a deleted node is a *tail*-node, its *tid*-list is pushed-up to its parent node. The contents of temporary array for the bottom item j in the MCPF-list of CT_i represents T^{ij} (i.e., set of all *tids* where items i and j occur together). From T^{ij}, it is a simple calculation of $S(ij)$ and $Per(ij)$ for the pattern $\{i, j\}$.

1. If $S(ij) \geq minsup_i$ and $Per(ij) \leq maximum(MaxIP(i), MaxIP(j))$, then $\{i, j\}$ is considered as a periodic-frequent pattern and the same process of creating prefix-tree and its corresponding conditional tree is repeated for further extensions of "ij".
2. Else, we will verify whether $S(ij) \geq MinIS(i)$ and $Per(ij) \leq maxprd_i$. If the above condition is satisfied, then the same process of creating prefix-tree and its corresponding conditional tree is repeated for further extensions of "ij" even though it is a non-periodic-frequent pattern. It is because higher orders of $\{i, j\}$ can still be periodic-frequent.

Next, i is pruned from the original *MCPF-tree*. (The procedure for pruning i is same as pruning non-periodic-frequent items in earlier steps.) The whole process of mining each item is repeated until MCPF-list $\neq \emptyset$. The correctness of this procedure is based Property 4 and Lemma 3.

Consider the item *pillow*, which is the last item in *MCPF-tree* of Fig. 1(f). The PT_{pillow} generated from *MCPF-tree* is shown in Fig. 2(d). Since, there is only one item *bed*, which is satisfying $MinIS_{pillow}$ and $MaxIP_{bed}(= minimum($-$MaxIP_{pillow}, MaxIP_{bed}))$, $CT_{pillow} = PT_{pillow}$. From $T^{\{bed,pillow\}}$, the pattern $\{bed, pillow\}$ will be generated a periodic-frequent pattern. The MCPF-tree generated after pruning *pillow* is shown in Fig. 2(e). Similar process is repeated for other items in the MCPF-tree. Finally, the set of periodic-frequent patterns generated are $\{\{bread\}, \{jam\}, \{bed\}, \{pillow\}, \{bread, jam\}, \{bed, pillow\}\}$. The patterns $\{bread, ball\}$ and $\{ball, bat\}$, which are generated as periodic-frequent patterns at low *minsup* and high periodicity ($minsup = 2$ and $maxprd = 4$) in Example 3 have failed to be periodic-frequent patterns in the proposed model. It is because they failed to satisfy $minsup = 4$ and $maxprd = 2$.

In this way the proposed model is able to efficiently address *rare item problem* while mining rare periodic-frequent patterns.

One can assume that the structure of a MCPF-tree may not be memory efficient, since it explicitly maintains *tids* of each transaction. It was proven in [3] that such a tree achieves the memory efficiency by keeping transaction information only at the tail-nodes and avoiding the support field at each node.

3.4 Relationship among the Patterns Generated in Various Models

For a given transactional database T, let F be the set of frequent patterns generated at $minsup = x\%$. Let PF be the set of periodic-frequent patterns generated at $minsup = x\%$ and $maxprd = y\%$. Let $MCPF$ be the set of set of periodic-frequent patterns generated in the proposed model when items' $MinIS$ values were greater than or equal to $x\%$ and $MaxIP$ values were less than or equal to $y\%$. The relation between these patterns is $MCPF \subseteq PF \subseteq F$.

4 Experimental Results

In this section, we present the performance comparison of the proposed model against basic model discussed in [3]. All programs are written in $C++$ and run with Ubuntu 8.1 on a 2.66 GHz machine with 2GB memory. The runtime specifies the total execution time.

The experiments are pursued on two types of datasets: synthetic dataset ($T10I4D100k$) and real-world datasets (retail and mushroom). $T10I4D100k$ [1] is a large sparse dataset with 100,000 transactions and 870 distinct items. The mushroom dataset is a dense dataset containing 8,124 transactions and 119 distinct items. The retail dataset [9] is also a large sparse dataset with 88,162 transactions and 16,470 items. All of these datasets are available at Frequent Itemset MIning (FIMI) repository (http://fimi.cs.helsinki.fi/data/).

To specify items' $MaxIP$ values we used the methodology discussed in Equation 2. To specify items' $MinIS$ values we used the methodology proposed in [4]. It is as follows:

$$MinIS(i_j) = maximum(\gamma \times S(i_j),\ LS) \tag{3}$$

where, 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.

4.1 Experiment 1

In synthetic, retail and mushroom datasets, both *minsup* and LS values are fixed at 0.1%, 0.1% and 25% respectively. With $\gamma = \frac{1}{\alpha}$, and varying α, we present the number of periodic-frequent patterns generated in these databases at different $maxprd$ ($maxprd = P_{max} = P_{min} = x\%$) values in Fig. 3(a), Fig. 3(b) and Fig. 3(c) respectively.

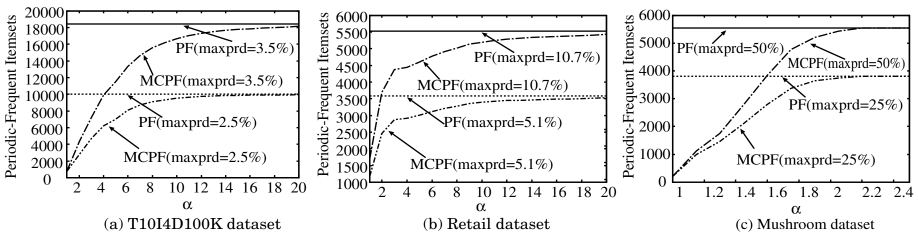


Fig. 3. Periodic-Frequent patterns generated at different $MinIS$ values

It can be observed that the number of periodic-frequent patterns significantly reduced by our model when α is not too large. When α becomes larger, the number of periodic-frequent patterns found by our model gets closer to that found by the traditional model (single *minsup-maxprd* model). The reason is because when γ becomes larger more and more items' $MinIS$ values reach LS.

4.2 Experiment 2

In synthetic, retail and mushroom datasets, *minsup* and *LS* values are fixed at 0.1%, 0.1% and 25% respectively. Next, γ was set at 0.5%. With $\beta = -\frac{1}{\alpha}$ and varying α from 0 to 20, we present the number of periodic-frequent patterns generated in these databases at different *maxprd* values in Fig. 4(a), Fig. 4(b) and Fig. 4(c) respectively.

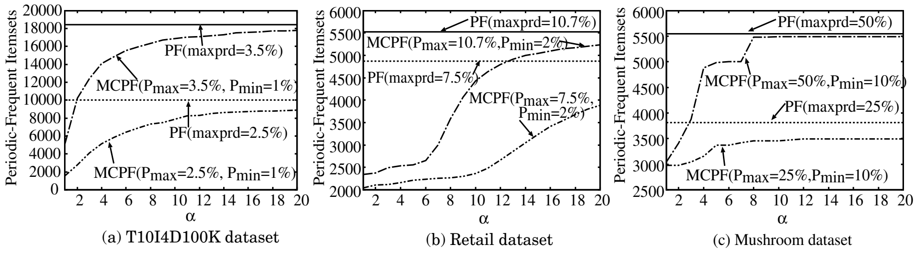


Fig. 4. Periodic-Frequent patterns generated at different *MaxIP* values

It can be observed that the number of periodic-frequent patterns significantly reduced by our model when α is small. When α becomes larger, the number of periodic-frequent patterns found by our model gets closer to that found by the traditional model (single *minsup-maxprd* model). The reason is because when α increases more and more items' *MaxIP* values reach P_{max}.

Fig. 5(a), Fig. 5(b) and Fig. 5(c) show the runtime taken by both of these approaches at different *MaxIP* values. It can be observed that the runtime taken by the proposed approach to generate periodic-frequent patterns increases with increase in α value. It is because of the increase in number of periodic-frequent patterns. (Similar observations can also be drawn for the previous experiments. Due to page limitations, we are not discussing them in this paper.) Also, it can be observed that the proposed approach requires more runtime to find periodic-frequent patterns. The reason is due to the increased search space as periodic-frequent patterns mined using the proposed model do not follow *downward closure property.*

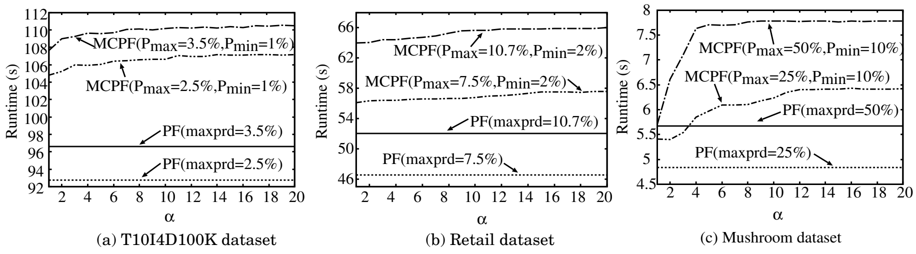


Fig. 5. Runtime taken for generating periodic-frequent patterns at different *maxprd* values

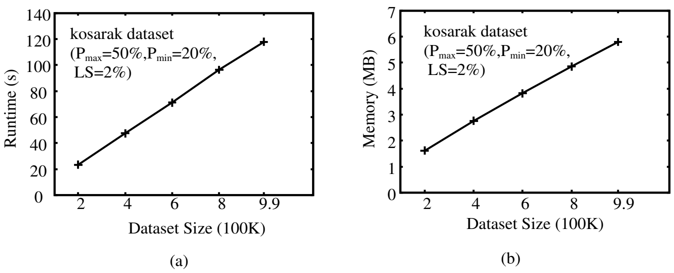


Fig. 6. Scalability of MCPF-tree. (a) Runtime requirements of MCPF-tree and (b) Memory requirements of MCPF-tree.

4.3 Experiment 3: Scalability Test

We studied the scalability of our MCPF-tree on execution time and required memory by varying the number of transactions in a database. We use real *kosarak* dataset for the scalability experiment, since it is a huge sparse dataset with 990,002 transactions and 41,270 distinct items. We divided the dataset into five portions of 0.2 million transactions in each part. Then we investigated the performance of MCPF-tree after accumulating each portion with previous parts and performing periodic frequent itemset mining each time. For calculation of items' $MinIS$ values in each experiment, the LS and γ values are fixed at 2% and 0.5%, respectively. Similarly, items' $MaxIP$ value, P_{min}, P_{max} and β values are fixed as 20%, 50% and -0.1 respectively. The experimental results are shown in Figure 6. The runtime and memory in y-axes of Figure 6(a) and Figure 6(b) respectively specify the total memory and execution time with the increase in database size. It is clear from the graphs that as the database size increases, overall tree construction and mining time, and memory requirement increase. However, MCPF-tree shows stable performance of about linear increase in runtime and memory consumption with respect to the database size. Therefore, it can be observed from the scalability test that MCPF-tree can mine periodic-frequent itemsets over large datasets and distinct items with considerable amount of runtime and memory.

5 Conclusion

In many real-world applications, rare periodic-frequent patterns can provide useful information. It is difficult to mine rare periodic-frequent patterns with a "single *minsup* and single *maxprd* model" due to rare item problem. In this paper, we have proposed an improved model to extract rare periodic-frequent patterns. In the proposed model each pattern can satisfy different *minsup* and *maxprd* values depending upon the items within it. For this model, we have also proposed an efficient pattern growth approach based on a tree structure, called MCPF-tree. As the periodic-frequent patterns mined using the proposed model do not satisfy *downward closure property*, the proposed pattern growth approach uses various other techniques to minimize the search space. We have also discussed a methodology to specify *maxprd* for each pattern dynamically. The experimental results demonstrate that the proposed approach efficiently finds periodic-frequent patterns consisting of frequent and rare items. However, it requires more runtime than single *minsup* and *maxprd* model, because the periodic-frequent patterns mined using the proposed model do not satisfy *downward closure property*.

References

1. Agrawal, R., Imielinski, T., and Swami, A.: Mining association rules between sets of items in large databases. In: SIGMOD, pp. 207-216 (1993)
2. Jiawei, H., Jian, P., Yiwen, Y., and Runying, M.: Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*. In: ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 53-87 (2004)
3. Tanbeer, S. K., Ahmed, C. F., Jeong, B., and Lee, Y.: Discovering Periodic-Frequent Patterns in Transactional Databases. In: Pacific Asia Knowledge Discovery in Databases (2009)
4. Liu, B., Hsu, W., and Ma, Y.: Mining Association Rules with Multiple Minimum Supports In: Knowledge Discovery and Databases, pp. 337–241 (2009)
5. Hu, Y.-H., Chen, Y.-L.: Mining Association Rules with Multiple Minimum Supports: A New Algorithm and a Support Tuning Mechanism. Decision Support Systems 42(1), 1–24 (2006)
6. Uday Kiran, R., Krishna Reddy, P.: An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules. In: IEEE Symposium on Computational Intelligence and Data Mining (2009)
7. Uday Kiran, R., Krishna Reddy, P.: An Improved Frequent Pattern-growth Approach to Discover Rare Association rules. In: International Conference on Knowledge Discovery and Information Retrieval (2009)
8. Uday Kiran, R., Krishna Reddy, P.: Mining Rare Association Rules in the Datasets with Widely Varying Items' Frequencies. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010. LNCS, vol. 5981, pp. 49–62. Springer, Heidelberg (2010)
9. Brijs, T., Swinnen, G., Vanhoof, K., Wets, G.: The use of association rules for product assortment decisions - a case study. In: Knowledge Discovery and Data Mining (1999)