Mining Periodic-Frequent Patterns with Maximum Items' Support Constraints

by

R. Uday kiran, P Krishna Reddy

in

ACM Compute 2010
(*ACM Compute 2010*)

Report No: IIIT/TR/2010/38

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
January 2010

Mining Periodic-Frequent Patterns with Maximum Items' Support Constraints

R. Uday Kiran
International Institute of Information Technology-Hyderabad
Hyderabad
Andhra Pradesh, India
uday_rage@research.iiit.ac.in

P. Krishna Reddy
International Institute of Information Technology-Hyderabad
Hyderabad
Andhra Pradesh, India
pkreddy@iiit.ac.in

ABSTRACT

The single minimum support (*minsup*) based frequent pattern mining approaches like Apriori and FP-growth suffer from "rare item problem" while extracting frequent patterns. That is, at high *minsup*, frequent patterns consisting of rare items will be missed, and at low *minsup*, number of frequent patterns explode. In the literature, efforts have been made to extract rare frequent patterns under "multiple minimum support framework". In this framework, "rare frequent patterns" can be extracted by specifying *minsup* of the pattern using two models: minimum constraint model and maximum constraint model. In the literature, an approach has been proposed to extract only those frequent patterns which occur periodically. The basic model of periodic-frequent patterns is based on single *minsup* constraint. It was observed that the periodic-frequent pattern mining approach also suffers from the "rare item problem". An effort has been made to extract rare periodic-frequent patterns using minimum constraint model. In this paper, we have proposed a pattern-growth approach to extract rare periodic-frequent patterns by specifying *minsup* under maximum constraint model. Experiment results show that the proposed approach is efficient.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

Periodic-Frequent patterns, multiple minimum supports, *minimum constraint model* and *maximum constraint model*.

COMPUTE 2010 January 22-23, 2010, Bangalore, India.

1. INTRODUCTION

Frequent pattern mining is an important model in data mining. The basic model of frequent patterns is as follows [1]:

Let $I = \{i_1, i_2, \cdots, i_n\}$ be a set of items. A set $X = \{i_k, \cdots, i_{k'}\} \subseteq I$, where $k \leq k'$ and $k, k' \in [1, n]$, is called a pattern (or an itemset). A transaction $t_{tid} = (tid, Y)$ is a tuple where *tid* represents a transactional-id (or timestamp) and Y is a pattern. A transactional database T is a set of transactions i.e., $T = \{t_1, t_2, \cdots, t_m\}, m = |T|$, where $|T|$ represents the size or total number of transactions in transactional database. If $X \subseteq Y$, it is said that X occurs in t and such transactional-id is denoted as t^X_{tid}, $tid \in [1, m]$. Let $T^X = \{t^X_l, \cdots, t^X_{l'}\}$, where $l \leq l'$ and $l,\ l' \in [1, m]$, denote the set of all transaction-id's where pattern X occurs. Support of X, denoted as $S(X)$, refers to the proportion of transactions that contain X in the transaction dataset. The pattern X is frequent, if $S(X) \geq minsup$, where *minsup* refers to user-specified minimum support.

Table 1: Transaction dataset

TID	Items	TID	Items
1	*Bread, Jam*	6	*Bread, Jam*
2	*Ball, Bat*	7	*Ball, Bat, Bed, Pillow*
3	*Bed, Pillow*	8	*Bread, Ball, Bat*
4	*Bread, Jam, Ball*	9	*Bread, Ball, Jam, Bat*
5	*Bread, Jam*	10	*Ball, Bat*

Example 1: (Running Example) Let our running example be the transaction dataset shown in Table 1. The set of items, $I = \{Bread, Ball, Jam, Bat, Pillow, Bed\}$. The transactional dataset, T contains 10 transactions. So size of the transactional dataset, $m = 10$. The items in the set $\{Bread, Jam\}$ is a pattern (or an itemset). This pattern occurred in 5 transactions. The *tids* in which this pattern has occurred, $T^{\{Bread,\ Jam\}} = \{1, 4, 5, 6, 9\}$. Thus, $S(Bread, Jam) = \frac{5}{10} = 0.5$. If $minsup = 0.5$, then this pattern is a frequent pattern because $S(Bread, Jam) \geq minsup$. (For this pattern, the support in terms of count is 5. For ease of explanation, throughout this paper we discuss this example in terms of support count.)

Minsup corresponds to minimal number of transactions in which a pattern should appear in a transaction dataset. Using only a single *minsup* to discover frequent patterns, frequent pattern mining techniques like Apriori [1] and FP-

growth [3] implicitly assume that all items within a dataset have uniform or similar frequencies. This is seldom not the case in most of the real-world applications. In many applications, some items appear frequently in the data, while others appear relatively infrequent or rare. In such datasets, using a single *minsup* constraint to mine frequent patterns consisting of both frequent and rare items raises the dilemma, called *rare item problem* [4]. This dilemma is as follows.

1. If *minsup* is set very high, we miss the frequent patterns consisting of rare items because rare items fail to satisfy high *minsup*.
2. In order to find frequent patterns consisting of both frequent and rare items, *minsup* should be set very low. However, this may cause combinatorial explosion, producing too many frequent patterns, because those frequent items will be associated with one another in all possible ways and many of them are meaningless or uninteresting to the user.

Example 2: Continuing with Example 1, it can be observed that at high *minsup*, say $minsup = 4$ (in support counts), rare items like *Bed* and *Pillow* fail to participate in generating frequent patterns because their support is less than *minsup*. To find frequent patterns consisting of both frequent and rare items, let us specify low *minsup*, say $minsup = 2$. At this *minsup*, frequent patterns generated are shown in the fifth column of Table 2 with the term 'Y'. It can be observed that among the frequent patterns generated, the patterns {*Bread, Ball*}, {*Bread, Bat*} and {*Jam, Bat*} (patterns shown in bold-italics) are uninteresting because these patterns contain only frequent items occurring together in very less number of transactions. These patterns can be interesting if they have satisfied high *minsup*, say $minsup = 4$.

Uninteresting patterns are described as those patterns which have low support and contain only frequent items. To address the "rare item problem" i.e., to prevent uninteresting patterns from becoming frequent patterns, efforts have been made in [4] [5] [6] [7] [8] to discover frequent patterns using multiple *minsup* constraints.

Independent to the detailed implementation technique, the model used in [4] [6] [7] [8] is as follows. For each item specify a support constraint, called *minimum item support* (MIS) such that it reflects the respective items' frequency (or support). Next, represent *minsup* of a pattern with minimal MIS among all its items. Thus, each pattern, depending upon the items within it can satisfy a different *minsup* constraint. We call this model as *minimum constraint model*.

In *minimum constraint model*, a pattern has to satisfy only the lowest MIS value among all its items. Hence, a pattern can become a frequent pattern though it fails to satisfy the MIS values of all other remaining items within it. However, in certain scenarios, if the user specifies a MIS value for an item, it may mean that any frequent pattern consisting of that item should have support no less than its MIS value. That is, a pattern must satisfy the MIS values of every item within it. With this requirement, another model which specifies *minsup* of a pattern as maximum MIS among all its items has been discussed in [5]. We call this model, as *maximum constraint model*.

Example 3: Continuing with Example 2, Let MIS values for the items *Bread, Ball, Jam, Bat, Pillow and Bed* be 5, 5, 4, 4, 2 and 2 respectively. The MIS value for an item *Bread* is 5. It means any frequent pattern consisting of item *Bread* should have support no less 5. Similarly, for the other items. The *minsup* for the pattern {*Bread, Jam*} is $max(MIS(Bread), MIS(Jam)) = max(5, 4) = 5$. As $S(Bread, Jam) \geq 5$, this pattern is a frequent pattern. The frequent patterns generated using *maximum constrain model* are shown in sixth column of Table 2 with the term 'Y'. It can be observed that the uninteresting patterns {*Bread, Ball*}, {*Bread, Bat*} and {*Jam, Bat*}, which were generated as frequent patterns at low *minsup* ($minsup = 2$) have failed to become frequent patterns. It is because they have failed to satisfy their *minsup* values which are 5, 5 and 4 respectively.

Table 2: Frequent patterns generated with different approaches for the transaction dataset shown in Table 1. The second column, titled $TIDs$ represents the *tids* of the transactions where the respective patterns have appeared. The third column, titled S represents the support count of the respective pattern. The fourth column, titled Per represents the periodicity of the respective pattern. The columns, titled with I, II, III and IV represent the patterns mined using single *minsup* model, maximum constraint model, periodic-frequent pattern model and the proposed model respectively. In these columns, frequent or periodic-frequent patterns generated are represented with the term 'Y'. For the other case, they are represented with the term 'N'.

Pattern	$TIDs$	S	Per	Approaches			
				I	II	III	IV
{Bread}	1, 4, 5, 6, 8, 9	6	3	Y	Y	Y	Y
{Ball}	2, 4, 7, 8, 9, 10	6	3	Y	Y	Y	Y
{Jam}	1, 4, 5, 6, 9	5	3	Y	Y	Y	Y
{Bat}	2, 7, 8, 9, 10	5	5	Y	Y	N	N
{Bed}	3, 7	2	4	Y	Y	Y	Y
{Pillow}	3, 7	2	4	Y	Y	Y	Y
{Bread, Jam}	1, 4, 5, 6, 9	5	3	Y	Y	Y	Y
{***Bread, Ball***}	4, 8, 9	3	4	***Y***	N	***Y***	N
{***Bread, Bat***}	8, 9	2	8	***Y***	N	N	N
{Bread, Bed}	-	0	10	N	N	N	N
{Bread, Pillow}	-	0	10	N	N	N	N
{Ball, Jam}	9	1	9	N	N	N	N
{Ball, Bat}	2, 7, 8, 9, 10	5	5	Y	N	N	N
{Ball, Bed}	7	1	7	N	N	N	N
{Ball, Pillow}	7	1	7	N	N	N	N
{***Jam, Bat***}	4, 9	2	5	***Y***	N	N	N
{Jam, Bed}	-	0	10	N	N	N	N
{Jam, Pillow}	-	0	10	N	N	N	N
{Bat, Bed}	7	1	7	N	N	N	N
{Bat, Pillow}	7	1	7	N	N	N	N
{Bed, Pillow}	3, 7	2	4	Y	Y	Y	Y

By enabling different patterns to satisfy different *minsup* values depending upon the items within the respective pat-

tern, both of these two models try to efficiently address the "rare item problem." Among these two models, which model to use for finding frequent patterns depends upon the user-requirement, application type etc. It has to noted that for a given set of items' MIS values, the set of frequent patterns generated by the *maximum constraint model* are always subset or equivalent to the set of frequent patterns generated by *minimum constraint model* [5].

In this paper, we try to address the "rare item problem" while mining the special kind of frequent patterns, called periodic-frequent patterns [9]. Periodic-frequent patterns are those frequent patterns which occur periodically (or regularly) within a transaction dataset. The significance of these patterns in real-world applications like stock markets and supermarkets has been discussed in [9]. The basic model of periodic-frequent patterns is as follows:

Continuing with the basic model of frequent patterns, a period of pattern X, say p^X, is the time difference between the *tid*'s of t^X_{l+1} and t^X_l, $l \in [1, (m-1)]$, divided by size of the transaction dataset. Let, P^X denote the set of all periods of X i.e., $P^X = \{p^X_o, \cdots, p^X_{o'}\}$. Then, $Per(X) = \text{Max}(p^X_o, \cdots, p^X_{o'})$ is called the periodicity of pattern X. A pattern is said to be periodic-frequent, if (i) its periodicity is no greater than the user-specified maximum periodicity ($maxper$) constraint and (ii) its support is no less than the user-given minimum support ($minsup$) constraint.

> **Example 4:** Continuing with Example 1, it can be observed that the pattern {*Bread, Jam*} has occurred in five transactions. The *tids* in which this pattern has occurred, $T^{\{Bread,\ Jam\}} = \{1, 4, 5, 6, 9\}$. A period for this pattern is $(4-1) = 3$. The set of periods for this pattern, $P^{\{Bread, Jam\}} = \{\mathbf{1}, 3, 1, 1, 3, \mathbf{1}\}$. The first and last periods for this pattern (periods shown in bold) are derived with respect to the first and last transactions in the transaction dataset. Therefore, periodicity of this pattern, $Per(Bread, Jam) = max(1, 3, 1, -1, 3, 1) = 3$. Let the user-specified $minsup$ and $maxper$ values be 5 and 4 respectively. Then, this pattern is a periodic-frequent pattern because $S(bread, jam) \geq minsup$ and $Per(bread, jam) \leq maxper$.

The basic model of periodic-frequent patterns is also based on single $minsup$ constraint. Hence, this model also implicitly assumes that all items in the dataset have similar frequencies, therefore, faces "rare item problem" while mining periodic-frequent patterns consisting of both frequent and rare items.

> **Example 5:** For the transaction dataset shown in Table 1, let us specify high $minsup$, say $minsup = 4$, and $maxper = 4$. At this $minsup$, the rare items *Bed* and *Pillow* fail to participate in generating periodic-frequent patterns. To facilitate these rare items in generating periodic-frequent patterns, let us specify low $minsup$, say $minsup = 2$, and $maxper = 4$. The seventh column in Table 2 presents the periodic-frequent patterns generated at $minsup = 2$. It can be observed that the frequent pattern *Bat* has failed to be a periodic-frequent pattern because it failed to satisfy $maxper$ constraint. Therefore, all its supersets, including the frequent pattern {*Ball, Bat*} have failed to be periodic-frequent patterns. It can also be observed that the uninteresting pattern {*Bread, Ball*} has still qualified as periodic-frequent pattern even though $maxper$ acts as one of the pruning constraint.

In [10], *minimum constraint model* was extended to the basic model of periodic-frequent patterns so that "rare item problem" can be addressed while mining periodic-frequent patterns consisting of both frequent and rare items. But this model cannot satisfy the user requirements which are same as those of *maximum constraint model*. That is, generate all periodic-frequent patterns whose support satisfies the support constraints (MIS values) of all the items within it and periodicity no greater than $maxper$.

With this motivation, in this paper, we extend the *maximum constraint model* to the basic model of periodic-frequent patterns so that "rare item problem" can be addressed while mining periodic-frequent patterns consisting of both frequent and rare items. The proposed model facilitates patterns to satisfy $maxper$ and different $minsup$'s depending upon the items in the respective pattern. For this extended model, we also discuss a pattern growth approach, called **Max**imum **C**onstraint **P**eriodic-**F**requent **P**attern-growth (MaxCPFP-growth). Experimental results show that the proposed approach is very efficient.

Remaining part of the paper is organized as follows. In Section 2, we discuss the related work pertaining to the efforts made to address "rare item problem" while mining frequent and periodic-frequent patterns. In Section 3, we discuss the proposed model and the MaxCPFP-growth approach. Experimental results conducted on synthetic and real world datasets are presented in Section 4. Section 5 concludes the paper.

2. RELATED WORK

2.1 Frequent pattern mining

Frequent pattern mining approaches like Apriori [1] and FP-growth [3] use only a single $minsup$ constraint to discover frequent patterns. With a single $minsup$, it is difficult capture items' frequencies in a dataset. Hence such single $minsup$ based frequent pattern mining approaches face the dilemma called "rare item problem" while mining frequent patterns consisting of both frequent and rare items. That is, at high $minsup$, frequent patterns involving rare items will be missed, and at low $minsup$, number of frequent patterns explode.

To address the "rare item problem," efforts have been made in [4] [5] [6] [7] [8] to discover frequent patterns using multiple $minsup$ constraints. In [4], *minimum constraint model* (discussed in Section 1) has been proposed and extended to Apriori approach to discover frequent patterns consisting of both frequent and rare items. In [5], *maximum constraint model* (discussed in Section 1) has been proposed and extended to Apriori approach to discover frequent patterns. In [6], *minimum constraint model* has been extended to FP-growth approach because the Apriori-like approach which was discussed in [4] faces the performance problems like generating huge number of candidate patterns and multiple scans on the dataset. In [7], an effort has been made to specify items' MIS values dynamically. In [8], an effort has been made to improve the performance of [6] by efficiently identifying only those items which can generate frequent patterns.

2.2 Periodic-Frequent pattern mining

Periodic-frequent patterns are introduced in [9]. A pattern-growth approach which is based on a tree structure, called Periodic Frequent-tree (PF-tree) has also been discussed for mining such patterns. Technically, a pattern is said to be periodic-frequent if it satisfies *minsup* and maximum periodicity (*maxper*) constraints. As the basic model of periodic-frequent patterns is based on single *minsup* constraint, this model also faces "rare item problem" while mining periodic-frequent patterns consisting of both frequent and rare items.

To address "rare item problem" while mining periodic-frequent patterns consisting of both frequent and rare items, *minimum constraint model* has been extended to the basic model of periodic-frequent patterns [10]. Also, a multiple *minsup* based periodic-frequent pattern-growth, called Multiple Support Periodic-Frequent pattern-growth (*MSPFP-growth*) has also been discussed. This extended model cannot serve the requirements which are those of *maximum constrain model*.

With this motivation, we try to extend *maximum constraint model* to the basic model of periodic-frequent patterns. The proposed approach differs from the existing approaches in the following ways:

1. The proposed approach is different from all of the multiple *minsup* based frequent pattern mining approaches ([4] [6] [5] [7] [8]) because the latter approaches do not consider *tids* (or timestamp), whereas proposed approach has to consider *tids*.

2. The model discussed in [9] is based on single *minsup*, whereas the proposed model is based on multiple *minsups*. Also, to mine periodic-frequent patterns, the approach discussed in [9] requires two scans on the dataset. But the proposed approach requires only a single scan on the dataset because it utilizes the prior knowledge regarding items' MIS values.

3. The model discussed in [10] is an extension of *minimum constraint model* to basic model of periodic-frequent patterns. The model discussed in this paper is an extension of *maximum constraint model* to basic model of periodic-frequent patterns. The *minimum constraint model* follows *sorted closure property*. Briefly, this property says, all non-empty subsets of a periodic-frequent pattern need not necessarily be periodic-frequent. (For detailed explanation on this property refer to [4].) So the approach discussed in [10] has to consider both infrequent and frequent periodic-frequent patterns for generating further (or higher order) periodic-frequent patterns. The *maximum constraint model* follows *downward closure property*. Hence the proposed approach has to consider only periodic-frequent patterns for generating further (or higher order) periodic-frequent patterns.

3. PROPOSED MODEL AND APPROACH

3.1 Basic model

Given the user-specified *maximum periodicity* (*maxper*) constraint and *minimum item support* (MIS) for every item, a pattern is said to be periodic-frequent if (i) its support is greater than or equal to the maximum MIS value among all its items and (ii) its periodicity less than or equal to *maxper* (see Equation 1).

$$\begin{array}{lcl} S(i_1, i_2, ..., i_k) & \geq & max \left(\begin{array}{c} MIS(i_1), MIS(i_2) \\ ..., MIS(i_k) \end{array} \right) \\ & and & \\ Per(i_1, i_2, ..., i_k) & \leq & maxper \end{array} \quad (1)$$

where, $MIS(i_j)$ represents the *minimum item support* for an item $i_j \in I$, $S(i_1, i_2, \cdots, i_k)$ and $Per(i_1, i_2, ..., i_k)$ represents the respective support and periodicity of the pattern $\{i_1, i_2, \cdots, i_k\}$, $1 \leq k \leq n$.

This model follows *downward closure property*. The *downward closure property* says, "all non-empty subsets of a frequent pattern are also frequent."

3.2 Proposed Approach: Maximum Constraint Periodic-Frequent Pattern-growth

Given the *maxper* constraint and items' MIS values, the proposed approach utilizes the prior knowledge regarding the items' MIS values and discovers complete set of periodic-frequent patterns with a single scan on the dataset. In particular, using the items' MIS values, the proposed approach constructs a tree known as MIS-PF-tree with a single scan on the dataset and uses this tree for mining periodic-frequent patterns. The approach is as follows.

3.2.1 Step 1: Construction of MIS-PF-tree

The MIS-PF-tree consists of two components: MIS-PF-list and prefix-tree. The MIS-PF-list is a list having four fields - item name (i), total support (f), periodicity (p) and minimum item support (MIS). Structurally, the prefix-tree used in MIS-PF-tree is similar to the prefix-tree used in PF-tree. However, the construction procedure is different.

1. In the PF-tree, construction of prefix-tree requires *two scans* on the dataset. In the MIS-PF-tree, prefix-tree is constructed with a *single scan* on the dataset because the proposed approach utilizes the prior knowledge regarding MIS values of the items.

2. In the prefix-tree of PF-tree, items are arranged in *support* descending order. In the prefix-tree of MIS-PF-tree, items are arranged in MIS descending order.

The structural description of prefix-tree is as follows. The prefix-tree explicitly maintains the occurrence information of each transaction in the tree structure by keeping an occurrence transaction-id list, called *tid*-list, only at the last node of every transaction. Hence, there are two types of nodes maintained in the prefix-tree. They are: ordinary node and *tail*-node. Ordinary node is similar to that used in FP-tree, whereas *tail*-node represents the last item of any sorted transaction. Therefore, the structure of a *tail*-node is $N[t_1, t_2, \cdots, t_n]$, 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, each node in a PF-tree maintains parent, children, and node traversal pointers. However, irrespective of the node type, no node in prefix-tree maintains support count value in it.

Given the items' MIS values, the construction of MIS-PF-tree is as follows. First, the items are sorted in descending

order of their MIS values. Let the sorted list of items be L. Second, in L order, each item is added into the MIS-PF-list by setting their respective f, p and MIS values equivalent to 0, 0 and MIS value of the respective item. Third, in the prefix-tree, a root node is created and labeled as *null*.

Let id_l be the temporary array which records the *tids* of last occurring transactions of the items in the MIS-PF-list. Let t_{cur} and p_{cur} denote the *tid* of current transaction and the most recent period for an item. The id_l value of every item is set to 0. Next, by scanning each transaction in the transaction dataset the MIS-PF-tree is updated as follows. For each transaction t_{cur}, identify the items in it. For each item in the respective transaction, perform the following steps in the MIS-PF-list: (i) increment the f value of the respective item by 1 (ii) calculate p_{cur} for the respective item as $t_{cur} - id_l$ and update its p value as p_{cur} only when $p_{cur} > p$ and (iii) set id_l value of the respective item with t_{cur}. For the same transaction a branch is created in the prefix-tree and the *tid* of the respective transaction is added to *tid*-list of the tail node represented by this transaction. (The creation procedure of a branch in the prefix-tree is same as that in FP-tree; however, we do not maintain frequency at each node.)

After scanning all the transactions, update the p value of every item in the MIS-PF-list by calculating the p_{cur} value with t_{cur} equivalent to the *tid* of the last transaction in the transaction dataset. In Algorithm 1, lines from 1 to 15 describe the construction of MIS-PF-tree.

For the transaction dataset shown in Table 1, the construction of MIS-PF-tree is as follows. Let the MIS values for the items *Bread*, *Jam*, *Bat*, *Ball*, *Bed* and *Pillow* be 5, 4, 4, 5, 2 and 2 respectively. The sorted list of items in descending order of their MIS values is *Bread, Ball, Jam, Bat, Bed* and *Pillow*. Let this sorted list be L. Next, MIS-PF-list is created by adding each item in L order and by setting their respective f and p and MIS values equivalent with 0, 0 and the user-specified MIS value for the respective item. In prefix-tree, a root node is created and labeled as *null*.

Initially, set id_l values of every item equal to 0. When first transaction in Table 1 is scanned with $t_{cur} = 1$, the items in t_{cur} are sorted in L order. Thus the sorted transaction is {*Bread, Jam*}. The item *Bread* is chosen and its f value is incremented by 1. The p_{cur} value of this item ($p_{cur} = t_{cur} - id_l = 1 - 0$) is calculated and is equivalent to 1. As the p_{cur} value is greater than its respective p value ($1 > 0$), its p value is set with its respective p_{cur} value. That is, $p = p_{cur} = 1$. The id_l value of the item *Bread* is updated with t_{cur} i.e., $id_1 = 1$. Similarly, for the item *Jam*, the derived f, p and id_l values are 1, 1 and 1 respectively. Next, in the prefix-tree, a branch is created (in L order) with two nodes, ⟨Bread⟩ and ⟨Jam⟩, where *Bread* is linked to *root* and *Jam* is linked to *Bread*. For the node *Jam*, which represents the *tail*-node for this transaction, the *tid* of this transaction is added to its *tid*-list. Similar procedure is repeated for remaining transactions. After scanning the complete transaction dataset, the p values of each item in the MIS-PF-list are updated by calculating p_{cur} value with $t_{cur} = 10$. The resultant MIS-PF-tree is shown Figure 1.

3.2.2 *Constructing the* compact MIS-PF-tree

The MIS-PF-tree is constructed with every item in the transaction dataset. However, as the proposed model follows *downward closure property*, the items whose support is less

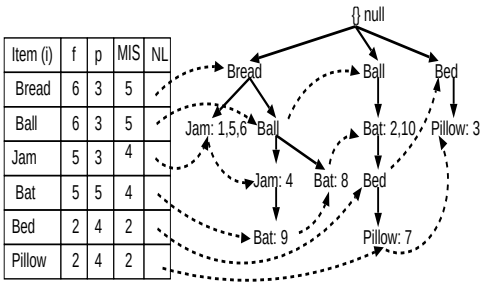


Figure 1: MIS-PF-tree.

Algorithm 1 MIS-PF-tree(Tran:transaction dataset, I: itemset containing n items, MIS: minimum item support values for n items, *maxper*: maximum periodicity)

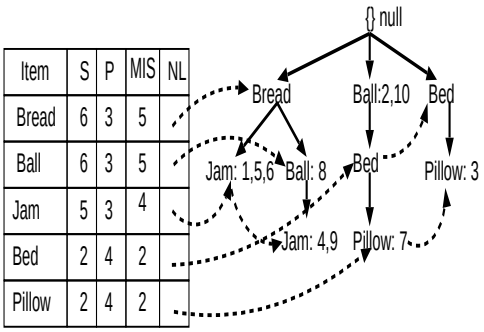
1: Let L represent the set of items sorted in nondecreasing order of their MIS values.
2: In L order, insert each item into the MIS-PF-list by setting f, p and MIS values equivalent to 0, 0 and MIS value of the respective item.
3: Create the root node in the MIS-PF-tree, T, and label it as *null*;
4: Let id_l be the temporary array which records the *tids* of last occurring transactions of the items i in the MIS-PF-list. Let t_{cur} and p_{cur} denote the *tid* of current transaction and the most recent period for an item.
5: **for** each transaction t_{cur} in $Tran$ **do**
6: Sort the items in t_{cur} in L order;
7: **for** each item i in t_{cur} **do**
8: $f = f + 1$; $p_{cur} = t_{cur} - id_l$; $id_l = t_{cur}$;
9: **if** $p_{cur} > p$ **then**
10: $p = p_{cur}$;
11: **end if**
12: **end for**
13: Let the sorted items in t be $[a|A]$, where a is the first element and A is the remaining list. Call InsertTree($[a|A]$, T,t_{cur}), which is performed as follows. If T has a child N such that N.item-name=p.item-name, then move into the node; else create a new node N, and its parent link be linked to T, and its node-link to the nodes with the same item-name via the node-link structure. If A is nonempty, call InsertTree(A,N) recursively; else add t_{cur} to the *tid*-list of N.
14: **end for**
15: At the end of $Tran$, calculate p_{cur} for each item by considering t_{cur} = the *tid* of the last transaction in $Tran$, and update the respective p_i according to step 9 and 11.
16: **for** each item i in MIS-PF-list **do**
17: **if** $f < MIS_i$ or $p > maxper$ **then**
18: Call MisPruning(T, i).
19: Remove item i from the MIS-PF-list;
20: **end if**
21: **end for**
22: Let the tree (*compact MIS-PF-tree*) derived after tree pruning operation is *cMIS-PF-tree*.

Procedure 2 MisPruning (Tree, i)

```
1: for each node in the node-link of i in Tree do
2:   if the node is a leaf then
3:     Remove the node and move its tid-list to its respective parent node;
4:   else
5:     Remove the node and link its parent node with its child node(s);
6:     if the node has tid-list then
7:       Move the nodes tid-list to its respective parent node;
8:     end if
9:   end if
10: end for
```

than their respective MIS value or periodicity greater than *maxper* constraint do not generate any periodic-frequent pattern. Therefore, such items are pruned from the MIS-PF-tree by performing tree-pruning operation so that the resulting MIS-PF-tree is compact. We call such resultant MIS-PF-tree as *compact MIS-PF-tree*. In Algorithm 1, lines from 16 to 21 and Procedure 2 describe the pruning of items from the MIS-PF-tree.

Continuing with the example, it can be observed from the MIS-PF-list of Figure 1 that item *Bat* has its periodicity greater than *maxper* constraint. Therefore, this item is pruned from the prefix-tree and MIS-PF-list of MIS-PF-tree. The resultant *compact MIS-PF-tree* after pruning item *Bat* is shown in Figure 2. It can be observed that the *tid*-lists of the item node *Bat* are pushed to respective parent nodes. Also, its child nodes are pushed to its parent node.



Item	S	P	MIS	NL
Bread	6	3	5	
Ball	6	3	5	
Jam	5	3	4	
Bed	2	4	2	
Pillow	2	4	2	

Figure 2: *compact MIS-PF-tree* generated after pruning item *Bat*.

3.2.3 *Mining the* compact MIS-PF-tree

Mining the *compact MIS-PF-tree* is similar to mining of PF-tree [9], which is based on pattern growth-based bottom-up mining technique involving (i) counting length-1 periodic-frequent items, (ii) constructing the prefix-tree for each periodic-frequent pattern, and (iii) constructing the conditional-tree from each prefix-tree. Before, we discuss the prefix-tree construction process we explore the following property and lemma in the *compact MIS-PF-tree*.

PROPERTY 3.1. *In a branch of the* compact MIS-PF-tree*, an item at level 'l' always have its MIS value less than or equal to MIS value of all the items located from level '1' to '$l-1$' (in the same branch).*

LEMMA 3.1. *Let α be a pattern in* compact MIS-PF-tree*. Let B be α conditional pattern base, and β be an item in B. Let $S_B(\beta)$ be the support of β in the B. Let $MIS(\beta)$ be the* minimum item support *of β. If α is periodic-frequent and $S_B(\beta) \geq MIS(\beta)$ and $P^{<\alpha,\beta>} \leq maxper$ then the pattern $< \alpha, \beta >$ is a periodic-frequent pattern.*

PROOF. According to the definition of conditional pattern base and MIS-PF-tree, each subset in B occurs under the condition of the occurrence of α in the transactional database. If an item β appears in B at n number of transactions, it appearers with α in the same n number of transactions. From the Property 3.1, we know that the $MIS(\beta) \geq minsup(\alpha)$. If $S_B(\beta) \geq MIS(\beta)$ then $<\alpha, \beta>$ is a frequent pattern. If $< \alpha, \beta >$ has periodicity less than or equal to *maxper* then $< \alpha, \beta >$ is a periodic-frequent pattern. □

Using the features revealed by the above property and lemma, we proceed to construct each prefix-tree starting from the bottom-most item, say i, of the MIS-PF-list in *compact MIS-PF-tree*. Only the prefix sub-paths of nodes labeled i in the *compact MIS-PF-tree* are accumulated as the prefix-tree for i, say MIS-PT$_i$. Since, i is the bottom-most item in the MIS-PF-list, each node labeled i in the MIS-PF-tree must be a tail-node. While constructing the MIS-PT$_i$, 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 MIS-PF-list of MIS-PT$_i$. Moreover, to enable the construction of the prefix-tree for the next item in the MIS-PF-list, the *tid*-lists are pushed-up to respective parent nodes in the original MIS-PF-tree and in MIS-PT$_i$ as well. All nodes of i in the MIS-PF-tree of i's entry in the MIS-PF-list are deleted thereafter. The correctness of pushing the *tid*-list to the parent nodes is discussed in [9].

Using the MIS of the item i, based on the Lemma 3.1 we construct conditional tree MIS-CT$_i$ from MIS-PT$_i$ by removing all non-periodic-frequent nodes (items which have support value less than their respective MIS value or periodicity greater than *maxper*). If the deleted node is a tail-node, its *tid*-list is pushed-up to its parent-node. The contents, of the temporary array for the bottom item j in the MIS-PF-list of MIS-CT$_i$ represents T^{ij} (i.e., the set of all *tid*s where items i and j occur together). Therefore, it is rather simple calculation to compute $Per(i,j)$ and $S(i,j)$ from $T^{i,j}$ by generating $P^{i,j}$. Then the pattern "i, j" is generated as periodic-frequent pattern with the support and periodicity values of $S(i,j)$ and $P(i,j)$, respectively. The same process of creating prefix-tree and its corresponding conditional tree is repeated for further extensions of "i, j". The whole process of mining for each item is repeated until MIS-PF-list $\neq \emptyset$. The above bottom-up mining technique on MIS-descending MIS-PF-tree is efficient, because it shrinks the search space dramatically with the progress of mining process.

Continuing with the example, mining the periodic-frequent patterns from the *compact MIS-PF-tree* shown in Figure 2 is as follows. The bottom most item in the MIS-PF-list, *Pillow* is initially chosen for mining periodic-frequent patterns. Its prefix-tree, MIS-PT$_{Pillow}$, shown in Figure 3(a) is constructed with the prefix sub-paths of nodes labeled with item *Pillow*. From MIS-PT$_{Pillow}$, the conditional-tree, MIS-CT$_{Pillow}$ (see Figure 3(b)) is constructed by removing

all non-periodic frequent nodes. From MIS-CT$_{Pillow}$, the pattern {Bed, Pillow} is generated as periodic-frequent pattern because $S(Bed, Pillow) \geq MIS(Bed)$ and $P(Bed, Pillow) \leq maxper$.

To enable construction of prefix-tree for the remaining items in the *MIS-PF-list* of *compact MIS-PF-tree*, the *tid*-lists of the item *Pillow* are pushed-up to respective parents nodes in the original *compact MIS-PF-tree* and all nodes of item *Pillow* in *compact MIS-PF-tree* are deleted thereafter. Similar process is carried out for remaining items in the *compact MIS-PF-tree*. In Table 2, eight column presents the periodic-frequent patterns generated using the proposed approach. It can be observed that the uninteresting periodic-frequent pattern {*Bread, Ball*} which was generated at $minsup = 2$ and $maxper = 4$ (discussed in Example 5) has failed to be a periodic-frequent pattern in the proposed model because it failed to satisfy the high $minsup$ i.e., $minsup = 5$.

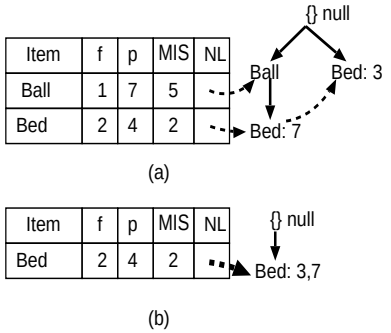


Figure 3: Mining periodic-frequent patterns for the item *Pillow*. (a) MIS-PT$_{Pillow}$ and (b) MIS-CT$_{Pillow}$.

3.3 Relationship Between the Patterns

At $minsup = x$, let F be the set of frequent patterns generated by the model discussed in [1]. At $minsup = x$ and $maxper = y$, let PF be the set of periodic-frequent patterns generated by the model discussed in [9]. Let the items be specified with MIS values such that no item has MIS value less than $minsup$. For the given items' MIS values and $maxper = y$, let $MinPF$ be the set of periodic-frequent patterns generated by the model discussed in [10]. Similarly, for the given items' MIS values and $maxper = y$, let $MaxPF$ be the set of periodic-frequent patterns generated by the proposed model. The relationship between these set of patterns is $MaxPF \subseteq MinPF \subseteq PF \subseteq F$.

4. EXPERIMENTAL RESULTS

In this section, we present the performance comparison of the proposed approach with the approaches discussed in [9] [10]. We have evaluated the performance results pertaining to the generation of periodic-frequent patterns by considering two kinds of datasets: synthetic and real world datasets. The synthetic dataset T10.I4.D100K, is generated with the data generator [1], which is widely used for evaluating association rule mining algorithms. It contains 1,00,000 number of transactions and 886 items. Another dataset is a real world dataset referred as retail dataset [11]. It contains 88,162 number of transactions and 16,470 items. The

Procedure 3 MaxCPFP-growth(cMIS-PF-tree: *compact MIS-PF-tree*, *maxper*: maximum periodicity)

1: **repeat**
2: Construct MIS-PT$_i$ for the bottom most item i in the MIS-PF-list of cMIS-PF-tree. MIS-PT$_i$ consists of prefix sub-paths of nodes labeled i in the cMIS-PF-tree. Since i is the bottom-most item in the cMIS-PF-list, each node labeled i in the cMIS-PF-tree must be a *tail*-node. So while constructing MIS-PT$_i$, we map the *tid*-list of every node of i to all items in the respective path explicitly in a temporary array so that it facilitates the periodicity and support calculation of each item in the MIS-PF-list of MIS-PT$_i$.
3: From MIS-PT$_i$, conditional tree CT_i is constructed by removing all non-periodic-frequent nodes from the $MIS\text{-}PT_i$. If the deleted node is a *tail*-node, its *tid*-list is pushed up to its parent node.
4: The contents of the temporary array for the bottom item j in the MIS-PF-list of $MIS\text{-}CT_i$ represent the T^{ij} (i.e., set of all *tids* where items i and j occur together).
5: From T^{ij}, the pattern "i, j" is generated as a periodic-frequent pattern if $S(i,j) \geq MIS(j)$ and $Per(i,j) \leq maxper$.
6: The same process of creating prefix-tree and its corresponding conditional tree is repeated for further extension of "i, j".
7: After finding all periodic-frequent patterns consisting of item i, all nodes of i in the prefix-tree of cMIS-PF-tree and i's entry in the MIS-PF-list are deleted thereafter.
8: **until** MIS-PF-list is empty

experimental procedure followed in this paper is similar to experimental procedure discussed in [4] [7].

For our experiments, we used the method discussed in [7] for specifying the MIS values for the items (see Equation 2).

$$MIS(i_j) = S(i_j) \times \beta \quad if \quad (S(i_j) \times \beta) > LS \qquad (2)$$
$$= LS \qquad otherwise$$
$$\beta = \frac{1}{\alpha} \qquad (3)$$

where, $S(i_j)$ refers to support of an item $i_j \in$ I, LS refers to user-specified "least support" value, $MIS(i_j)$ refers to minimum item support for an item $i_j \in$ I and β is a constant which ranges from 0 to 1 i.e., $\beta \in [0, 1]$. α is another constant which is varying from 1 to 20 throughout our experiments.

For both synthetic and Retail datasets, the MIS values are calculated by fixing LS value at 0.1% and varying the α values from 1 to 20. The *maxprd* value used in T10I4D100k and retail datasets are 3.5% and 5% respectively.

Both Figure 4(a) and Figure 4(b) show how the number of periodic-frequent patterns generated at different α values with the different approaches. The approaches are MSPFP-growth [10] and the proposed approach (MaxCPFP-growth). In these figures, the X-axis represents the SD (β) value and the Y-axis represents the number of periodic-frequent patterns generated. The think line in Figure 4(a) represents the number of periodic-frequent patterns generated using the approach discussed in [9], with $minsup = 0.1\%$ and $maxper = 3.5\%$. The think line in Figure 4(b) rep-

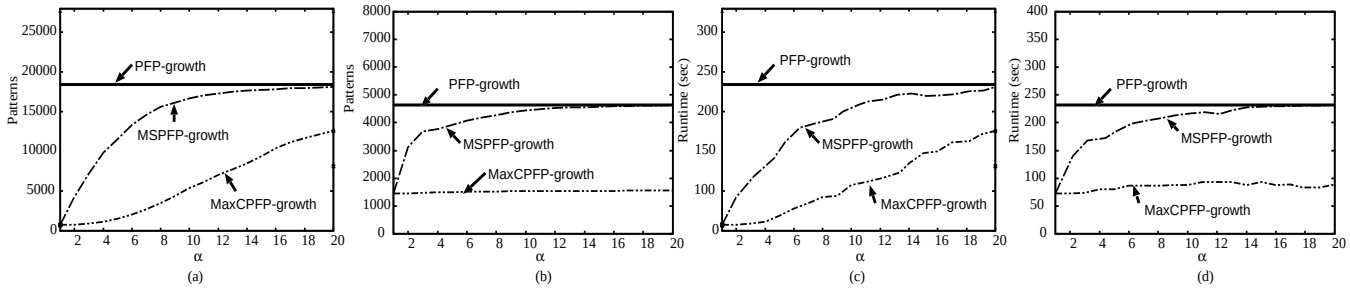


Figure 4: Periodic-frequent patterns generated at different SD values in (a) Synthetic and (b) Retail datasets. Runtime at different SD values in (c) Synthetic and (d) Retail datasets.

resents the number of periodic-frequent patterns generated using the approach discussed in [9], with $minsup = 0.1\%$ and $maxper = 5\%$. It can be observed that the number of periodic-frequent patterns is significantly reduced by our method with low α value. When α value becomes larger, the number of periodic-frequent patterns found by our method gets closer to that found by the [9]. The reason is, when α becomes larger more and more items' MIS values reach LS. Also, it can be observed that the proposed model discovered less number of periodic-frequent patterns as compared with the model discussed in [10]. The reason is, for a pattern, $minsup$ specified by the proposed model is always greater than or equal to the $minsup$ specified by the model discussed in [10].

Both Figure 4(c) and Figure 4(d) show the runtime taken by the different approaches for generating periodic-frequent patterns at different α values. The think line in this graph represents the runtime taken by the approach discussed in [9]. It can be observed that as the α value increases, runtime of the proposed approach also increases and gets closer to that of [9]. The reason is, when α increases, more number of periodic-frequent patterns are getting generated.

5. CONCLUSION

The basic model of periodic-frequent patterns uses only a single $minsup$ as one of its constraint. Hence, while mining periodic-frequent patterns consisting of both frequent and rare items this model faces the dilemma, called "rare item problem." With this motivation, we have proposed an efficient model by extending the *maximum constraint model* to basic model of periodic-frequent patterns. The reason for choosing this model is that this model efficiently address the "rare item problem" while mining frequent patterns consisting of both frequent and rare items. For this model, a pattern growth has also been presented in this paper. The effectiveness of the proposed model and approach have been shown experimentally.

6. REFERENCES

[1] Agrawal, R., Imielinski, T. and Swami, A., Mining association rules between sets of items in large databases, ACM SIGMOD International Conference on Management of Data', Vol. 22, ACM Press, Washington DC, USA, pp. 207–216, 1999.

[2] Hipp, J., Guntzer, U., and Nakhaeizadeh, G., Algorithms for Association Rule Mining A General Survey and Comparision, ACM Special Interest Group on Knowledge Discovery and Data Mining. Vol. 2, Issue 1, 2000.

[3] Jiawei, H., Jian, P., Yiwen, Y., and Runying, M., Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, 2004.

[4] Liu, B., Hsu, W., and Ma, Y., Mining Association Rules with Multiple Minimum Supports, ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations, 1999.

[5] Lee, Y., Hong, T., and Lin, W., Mining Association Rules with Multiple Minimum Supports using Maximum Constraints, International Journal of Approximate Reasoning (2005).

[6] Ya-Han Hu, and Yen-Liang Chen, Mining Association Rules with Multiple Minimum Supports: A New Algorithm and a Support Tuning Mechanism, Decision Support Systems, Vol. 42 , Issue 1, pp. 1 - 24, 2006.

[7] Uday Kiran, R., and Krishna Reddy, P., An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules, IEEE Symposium on Computational Intelligence and Data Mining, 2009.

[8] Uday Kiran, R., and Krishna Reddy, P., An Improved Frequent Pattern-growth Approach To Discover Rare Association rules, International Conference on Knowledge Discovery and Information Retrieval, 2009.

[9] Tanbeer, S. K., Ahmed, C. F., Jeong, B., and Lee, Y., Discovering Periodic-Frequent Patterns in Transactional Databases, Pacific Asia Knowledge Discovery in Databases, 2009.

[10] Uday Kiran, R., and Krishna Reddy, P., Mining Periodic-Frequent Patterns using Multiple Minimum Supports, International Conference on Management of Data, 2009.

[11] Brijs T., Swinnen G., Vanhoof K., and Wets G., The use of association rules for product assortment decisions - a case study, Knowledge Discovery and Data Mining, 1999.