MEMORY EFFICIENT FREQUENT PATTERN MINING USING TRANSPOSITION OF DATABASE

Mukesh Bathre
Govt. College of Engineering, Keonjhar, Odisa, India

Vivek Kumar Vaidya
Lecturer, CSE, J.M. Govt. Polytechnic College, Burhanpur M.P

Alok Sahelay
Lecturer, CSE, J.M. Govt. Polytechnic College, Burhanpur M.P

ABSTRACT

Frequent pattern can be extract from any dataset by using Apriori algorithm. Apriori algorithm is first choice of all researchers to find frequently occurs pattern from any binary dataset. Dataset contain record of user purchase item as transaction record. This paper improves existing apriori algorithm performance by extract frequent patterns from binary transaction data. New approach is applied for dataset implementation in form of transposed database of user’s record for fast data access. New work has done to mine frequent patterns using transposition of dataset, if database is large and contains thousands of attributes but having only some objects. This work analytically investigates the search space problem of frequent patterns mining and characterize database in transposed form and proposes an algorithm for mining frequent patterns based on Apriori algorithm by space reduced longest common sequence (LCS). This method makes apriori algorithm space efficient. The space complexity of proposed algorithm is \(O(n)\) while the dynamic approach like longest common subsequence space complexity is \(O(n^2)\) memory for given items in dataset.

Index Terms: Apriori Algorithm, Frequent Itemset, Data Mining, Space Complexity, Transposition of Database.


http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=7&IType=2
1. INTRODUCTION

Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions in binary format. Many other techniques are there to extract association rules from no transaction or no timestamps. Main work of apriori algorithm is to find association rules between different data sets of items this also known as “Market basket problem”. Each set of data has a number of items and is called a transaction. The output of Apriori is sets of rules that tell us how often items are contained in sets of data. Frequent item set mining and association rule induction are powerful methods for so-called market basket analysis, which aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies, online shops etc. With the induction of frequent item sets and association rules one tries to find sets of products that are frequently bought together, so that from the presence of certain products in a shopping cart one can infer (with a high Probability) that certain other products are present. Such information, especially if expressed in the form of rules, can often be used to increase the number of items sold, for instance, by appropriately arranging the products on the shelves of a supermarket or on the pages of a mail-order catalog (they may, for example, be placed adjacent to each other in order to invite even more customers to buy them together) or by directly suggesting items to a customer, which may be of interest for him/her.

2. RELATED WORKS

Determination of association rule mining, e-mail (emails about criminal activity) is suspected. Negative emotion words betray theory, a new person pronoun, in addition to simple words, the high-frequency words and special words were written in the body is characterized by deceptive e-mail writing pre-processed. Terms of apriori algorithm [1] is used to make. Data generated in the mail soon. It is used for automated analysis and evaluation to identify criminal activities and the announcement. Apriori algorithm for association rule mining, and all e-mail messages using the action verbs, past tense, using futures and evaluated. It’s an action verb, such kind of emails in the future tense suffix and another with a message by e-mail if you are suspicious. Warning email to "‘kill and bomb” future tense of words such as "‘will and shall," which refers to such terms. Step number.

In order to classify the e-mail box, all HTML from the text element, header, body, etc removed, before the words are stop words tokenizing. After separation of the body, begins to move e-mail classification. Training data “Bomb/Blast/Kill " key and " will/may " they, important that the class information, e-mails, a move “attacked/terrorist” and tense “was” in them using apriori algorithm. The training set apriori algorithm to find the e-mail database of words frequently used in the mining frequent item sets. Apriori algorithm for association rules and the rules used to set this item as follows.

Tense=past, Attack= Y, Bomb =Y -> Email = Suspicious informative Email.
Tense=future, Attack= Y, Bomb =Y -> Email = Suspicious alert Email.
Tense=future, Attack= N, Bomb =N -> Email = Normal Email.

An improved frequent pattern tree based on the technique named dynamic frequent pattern tree is proposed by Gyorodi [2]. The new method is efficiently applied on real world size database. A comparison between classical frequent pattern mining algorithms that are candidate set generation, test and without candidate generation is proposed in paper. Apriori algorithm, frequent pattern growth, dynamic frequent pattern growth are compared and presented together. Apriori algorithm in
used to rule mining in huge transaction database and Apriori algorithm is a bottom up approach. Frequent pattern growth is used to novel, compact data structure, referred to as frequent pattern tree, fp tree based ones are partition based, divide and conquer methods.

Optimization of association rule mining and apriori algorithm Using Ant colony optimization [3]. This paper is on Apriori algorithm and association rule mining to improved algorithm based on the Ant colony optimization algorithm. ACO was introduced by dorigo and has evolved significantly in the last few years. Many organizations have collected massive amount data. This data set is usually stored on storage database systems. Two major problems arise in the analysis of the information system. One is reducing unnecessary objects and attributes so as to get the minimum subset of attributes ensuring a good approximation of classes and an acceptable quality of classification. Another one is representing the information system as a decision table which shows dependencies between the minimum subset of attributes and particular class numbers without redundancy. In Apriori algorithm, is working process explained in steps. Two step processes is used to find the frequent item set to join and prune. ACO algorithm was inspired from natural behavior of ant colonies. ACO is used to solve to numerous hard optimizations including the traveling salesman problem. ACO system contains two rules. One is local pheromone update rule, which is applied in constructing solution. Another one is global pheromone update rule which is applied in ant construction. ACO algorithm includes two more mechanisms, namely trail evaporation and optionally deamonactions. ACO algorithm is used for the specific problem of minimizing the number of association rules. Association rule mining template is guided from XML document. XML is used in all areas of Internet application programming and is giving large amount of data encoded in XML [4]. With the continuous growth in XML data sources, the ability to extract knowledge from them for decision support becomes increasingly important and desirable. Due to the inherent flexibilities of XML, in both structure and semantics, mining knowledge in the XML Era is faced with more challenges than in the traditional structured world. This paper is a practical model for mining association rules from XML document. XML enabled association rule frame work that was introduced by Feng. XML AR frame works better than simple structured tree. The framework is flexible and powerful enough to represent simple and difficult structured association rules in XML document. But the best level of XML document model which is not yet implemented in has been proposed. The problem of mining XML association rules from the content of XML documents is based on user provided rule template. An implementation model is already introduced by Feng. Practical model consists of the following steps Filtering, Generating Virtual Transactions, Database reverse engineering association rule mining [5] is based on. Classification of the document database system could not be found in the poorly written, or even a particularly difficult task. The concept design of database reverse engineering to recover the database in an attempt to exit. Mining technique uses to detect the concept plan proposing a strategy paper. They used the normal form. Classical database is a valuable asset to the organization. New technologies were developed in 1970 as a COBOL and the database, and mini - computer platforms, file systems using older programming languages. Even some of the databases are outdated concepts such as the hierarchical data model, designed, and maintained and adjusted to serve the current needs of modern companies was difficult for them. Classical databases, messaging systems and their structures are related to the contents of the move and changing. This document is no longer in my approach to system design,
however, is hard to achieve; most companies in general are rising. Problems of migrating legacy databases to retrieve and database structures, database reverse engineering has been proposed. Reverse engineering process design and manufacturing process from the first to explore the objective devices and other hardware.

Generalized association rule mining based on tree structure. Into account [6] the use of specific information useful knowledge than ordinary flat association rules mining to detect this possibility. It employs a tree structure to compress the database Fp-line method, based on the paradigm proposing to mine development. There are two methods to the study of tree structure below and above are below. There are only a few studies have given way to common rules. Even some of the SQL queries and reports to the general association rule proposed parallel machine performance ratings. Apriori algorithm is a level-wise approach. A frequent pattern mining paradigm in the field of research has become a new trend.

Efficient association rules mining properties by using the apriori algorithm. Apriori algorithm to obtain the means of association rules from the dataset. It's [7] the case of a large dataset is a time consuming procedure apriori algorithm is an efficient algorithm. Time changes many long-term activities to increase the number of paths apriori and the proposed database. Disadvantages and apriori algorithm apriori algorithm can improve performance; this paper describes the application properties. Customers who buy products at the beginning of an association rule mining is market basket analysis to find out how. Minimum support of all frequent items finding all association rules at uvallatu.

1. 1. Find they considered the two-stage process. Enumeration of all frequent item sets the size of the search space is 2^n.
2. 2. To create strong rules. Used to create an association rule that satisfies any of the gate. Apriori and apriori algorithm using data mining tool and then running to write pseudo code. Association rule mining is the limit. ARM algorithm encounters a problem that does not return by the end user in a reasonable time. Activity in the presence and absence of an item is the only database that tells us a lot of shortcomings, and it is not efficient in the case of large dataset.ARM. The weight and size can be removed using such properties. Some of the limitations associated with large database apriori algorithm Searches. Its easy to implement using apriori exists. Association rule mining as well as potential customers for commercial gain valuable information, much improved by the use of such properties.

Association rule mining has wide applicability in many areas, efficient algorithm as it is a time consuming algorithm in case of large dataset [7]. With the time a number of changes are proposed in Apriori to enhance the performance in term of time and number of database passes. This paper illustrates the apriori algorithm disadvantages and utilization of attributes which can improve the efficiency of apriori algorithm. Association rule mining is initially used for Market Basket Analysis to find how items purchased by customers are related. The problem of finding association rules can be stated as follows: Given a database of sales transactions, it is desirable to discover the important associations among different items such the presence of some items in a transaction will imply the presence of other items in the same transaction. Discovering all association rules is considered as two-phase processes which are Association rule mining efficiency can be improved by using attributes like profit, quantity which will give the valuable information to the customer as well as the business. Association rule mining has a wide range of applicability in many areas.
3. TRANSPOSITION OF DATABASE
To avoid confusion between rows (or columns) of the original database and rows (columns) of the “transposed” database, this work define a database as a relation between original and transposed representations of a database in Table 3.1.

### Table 1 Transposition of Database

**Database D**

<table>
<thead>
<tr>
<th>Object</th>
<th>Attribute Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>a1a2a3</td>
</tr>
<tr>
<td>O2</td>
<td>a1a2a3</td>
</tr>
<tr>
<td>O3</td>
<td>a2a3a4</td>
</tr>
</tbody>
</table>

**Transposed Database DT**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Object Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>O1O2</td>
</tr>
<tr>
<td>a2</td>
<td>O1O2O3</td>
</tr>
<tr>
<td>a3</td>
<td>O1O2O3</td>
</tr>
<tr>
<td>a4</td>
<td>O3</td>
</tr>
</tbody>
</table>

The attributes are A = {a1, a2, a3, a4} and the objects are O = {o1, o2, o3}. Here use a string notation for object sets or itemsets, e.g., a1a3a4 denotes the itemset {a1, a3, a4} and o2o3 denotes the object set {o2, o3}. This dataset is used in all the examples between two sets: a set of attributes and a set of objects.

4. DYNAMIC APPROACH
The longest common subsequence problem is one of the common problems which can be solved efficiently using dynamic programming. “The Longest common subsequence problem is, this work are given two sequences X=<x1,x2--------xn> and Y=<y1,y2--------ym> and wish to find a maximum length common subsequence of X and Y” for example : if X=<A,B,C,B,D,A,B> and Y=<B,D,C,A,B,A> then The sequence <B, C, B, A> longest common subsequence. Let us define C [i, j] to be the length of an LCS of the sequences xi and yj. If either i=0 or j=0, one of the sequence has length 0, so the LCS has length 0. The Optimal substructure of the LCS Problem gives the recursive formula in equation-1.

\[
C(i, j) = \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
C(i-1, j-1) + 1 & \text{if } i, j > 0 \text{ and } x_i = y_j \\
\max \{C(i-1, j), C(i, j-1), C(i-1, j-1)\} & \text{if } i > 0 \text{ and } x_i \neq y_j
\end{cases}
\]

4.1. DAPS and DFPMT Approach
The mining algorithm works over the entire database file, first transpose the database and count the number of item and transaction string generated for each item. Sort the item numbers. Now apply Apriori like Algorithm in which first now calculate
frequent pattern $C_1$. It reduces un-frequent pattern and its transaction details also. For each pass now apply following sequence of operation until condition occurred. First generate the candidate pattern and prune by Apriori method. To count the support, instead of whole database for each pruned pattern now find longest common subsequence and length of transaction string of pattern’s item and also stored new pattern and its transaction string so that next iteration now trace above string. To find longest common subsequence this work used dynamic programming approach which faster than traditional approach. Write pruned pattern list with transaction string. So that in next pass this work used this pattern list instead of all pattern list. An advantage of this approach is in each iteration database filtering and reduces, so each iteration is faster than previous iteration.

4.2. DAPS Algorithm

1. Compute $k-1$ Subset of $k$-itemset
2. Generate Itemset Transition string from Filter Transposed Database
3. Computer LCS for each item in itemset using Transition string.
4. If length of LCS $>$ $\theta$ then itemset is frequent.

4.3. DFPMT Algorithm (Dynamic Approach for Frequent Patterns Mining Using Transposition of Database)

5. Convert Database in Transpose Form DT
6. Compute $F_1$ of all Frequent Items
7. $C_1 :=$ DT
   (Only Frequent Item row with Transition id string )
9. While $L_{k-1} \neq \emptyset$ do
10. Compute $C_k$ of all candidate $k-1$ Itemsets
   /* New Algorithm which prune and count support using LCS */
11. Compute $L_k =$ DAPS ($C_k$)
12. $K := K + 1$

Back bone of this algorithm is finding LCS from dataset which is perform by dynamic algorithm in above work proposed Heuristic space efficient algorithm But it can also be thought of as a way of computing the entries in the array $L$. The recursive algorithm controls what order this work fill them in, but this work get the same results if now filled them in some other order. This work might as well use something simpler, like a nested loop that visits the array systematically. The only thing have to worry about is that when fill in a cell $L[i,j]$, need to already know the values it depends on, namely in this case $L[i+1,j]$, $L[i,j+1]$, and $L[i+1,j+1]$. For this reason have to traverse the array backwards, from the last row working up to the first and from the last column working up to the first. This is iterative (because it uses nested loops instead of recursion) or bottom up (because the order fill in the array is from smaller simpler sub problems to bigger more complicated ones).

Advantages of this method include the fact that iteration is usually faster than recursion, this work don’t need to initialize the matrix to all -1’s, and this work save three if statements per iteration since this work don’t need to test whether $L[i,j]$, 

http://www.iaeme.com/IJCET/index.asp 58 editor@iaeme.com
L[i+1,j], and L[i,j+1] have already been computed (this work know in advance that the answers will be no, yes, and yes). One disadvantage over memorizing is that this fills in the entire array even when it might be possible to solve the problem by looking at only a fraction of the array's cells.

5. PROPOSED WORK

One disadvantage of the dynamic programming methods this work've described, compared to the original recursion, is that they use a lot of space: O(mn) for the array L (the recursion only uses O(n+m). But the iterative version can be easily modified to use less space -- the observation is that once this work have computed row i of array L, this work no longer need the values in row i+1.

5.1. Space Efficient LCS Algorithm (SELCS)

1. Store string to A and B array, Allocate storage for one-dimensional arrays X and Y and initialize with 0 length of first string is i and second is j.
2. While m ≠ 0 do following
3. While n ≠ 0 do
4. Compare A[m] with B[n] and do
   If A[m] or B[n] having null value store X[n] = 0.
   Otherwise if A[m] equal to B[n] store X[n] to 1 + Y[n+1].
   Above two condition not true then store to X[n] = larger (Y[n] and X[n+1]).
5. Before update value of m replace array Y to X.
6. Return length of LCS with common string.

5.2. APS Algorithm

1. Compute k-1 Subset of k-itemset
2. Generate Itemset Transition string from Filter Transposed Database
3. Computer LCS for each item in itemset using Transition string and SELCS algorithm.
4. If length of LCS > δ then itemset is frequent.

5.3. Efficient Method for Frequent Patterns using Transposition of Database

1. Convert Database in Transpose Form DT
2. Compute F1 of all Frequent Items
3. C1:= DT
   (Only Frequent Item row with Transition id string)
5. While Lk-1≠{ } do
6. Compute Ck of all candidate k-1 Itemsets
   /* New Algorithm which prune and count support using SELCS */
7. Compute Lk=APS (Ck)
8. K:=K+1

This takes roughly the same amount of time as before, O(mn) or O(n2) it uses a little more time to copy X into Y but this only increases the time by a constant (and
can be avoided with some more care). The space is either O(m) or O(n), whichever is smaller (switch the two strings if necessary so there are more rows than columns). Unfortunately, this solution does not leave you with enough information to find the subsequence itself, just its length.

6. METHODOLOGY WITH EXAMPLE

Study the following transaction database. \( A = \{ A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9 \} \). Assume \( \sigma = 20\% \), Since \( T \) contains 15 records, it means that an itemset that is supported by at least three transactions is a frequent set.

<table>
<thead>
<tr>
<th>S. No.</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
<th>A9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Transaction id String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,14</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2,4,6,7,13,15</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4,6,10,11,14,15</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2,3,6,13</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1,3,5,8,10,11,12,14</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1,5,7,12,13</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>3,5,7,10,11,13,14</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1,2,9,12</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>7,15</td>
<td>2</td>
</tr>
</tbody>
</table>
Now Apply Algorithm

1) Pass 1

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Transaction id String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2,4,6,7,13,15</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>4,6,10,11,14,15</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2,3,6,13</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>1,3,5,8,10,11,12,14</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1,5,7,12,13</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>3,5,7,10,11,13,14</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1,2,9,12</td>
<td>4</td>
</tr>
</tbody>
</table>

$L_1 := \{2, 3, 4, 5, 6, 7, 8\}$

2) Pass 2

Generate candidate for k=2

$C_2 = \{\{2,3\},\{2,4\},\{2,5\},\{2,6\},\{2,7\},\{2,8\},\{3,4\},\{3,5\},\{3,6\},\{3,7\},\{3,8\},\{4,5\},\{4,6\},\{4,7\},\{4,8\},\{5,6\},\{5,7\},\{5,8\}, \{6,7\}, \{6,8\}, \{7,8\}\}$

After Apply DAPS Algorithm

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Transaction id String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,3</td>
<td>4,6,15</td>
<td>3</td>
</tr>
<tr>
<td>2,4</td>
<td>2,6,13</td>
<td>3</td>
</tr>
<tr>
<td>3,5</td>
<td>10,11,14</td>
<td>3</td>
</tr>
<tr>
<td>3,7</td>
<td>10,11,14</td>
<td>3</td>
</tr>
<tr>
<td>5,6</td>
<td>1,5,12</td>
<td>3</td>
</tr>
<tr>
<td>5,7</td>
<td>3,5,10,11,14</td>
<td>5</td>
</tr>
<tr>
<td>6,7</td>
<td>5,7,13</td>
<td>3</td>
</tr>
</tbody>
</table>

3) Pass 3

Generate Candidate for k=3

$C_3 = \{\{2,3,4\},\{3,5,7\},\{5,6,7\}\}$

After Apply DAPS Algorithm

<table>
<thead>
<tr>
<th>Item Id</th>
<th>Transaction id String</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,5,7</td>
<td>10,11,14</td>
<td>3</td>
</tr>
<tr>
<td>5,6,7</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

$L_3 := \{(3, 5, 7)\}$

$L := L_1 \cup L_2 \cup L_3$
7. RESULTS
This chapter demonstrates the experiments that this work have performed to evaluate the new scheme. For the evaluation purpose this work has conducted several experiments using the existing data set. Those experiments performed on computer with Core 2 Duo 2.0 GHZ CPU, 2.0 GB memory and hard disk 80 GB. Both the algorithms have developed in JAVA language. For unit of measurement this work considers time in seconds and memory in bytes respectively.

7.1. Datasets used for Results
This section introduced data sets taken from the UCI machine learning repository [26] and discretised using the KDD dataset [26] have been made available. Discretisation has been carried out assuming a maximum of 5 "divisions". Note that all their files have been "zipped" using .gzip. The files are intended for use with frequent pattern mining and frequent itemset extraction from dataset which require binary valued input data.


7.2. Comparisons of Proposed work in terms of Memory
Following table and figure1.shows the comparison between dynamic and efficient LCS algorithm. Here dataset size is measured according to items in respective dataset for generating frequent patterns.

7.3. Comparisons of Proposed work in terms of Time
The experimental result of time is shown in Figure 2 reveals that the proposed scheme outperforms the efficient Apriori approach for different minimum support (MS) value. Our algorithm takes minimum 0.016 sec time during pattern generation also takes more than 300 secs to generate frequent pattern for dataset size 48842 transitions so it’s difficult to show graph in this range, hence here this work show maximum limit
20 sec for graphical representation for comparison time taken to generate frequent pattern to understand comparison easily. Actual time taken for dataset is mention in figure 2. Here dataset size is measured as total no. of transaction in respective dataset.

![Time Comparison](image)

**Time comparison for different Minsup value**

7.4. **No. of items in 1 item frequent pattern list**

Size of frequent pattern list is changed according to minimum support given for this this work have comparison for different minsup for every dataset.

![Comparision of 1-itemset size](image)

**No. of items in 1 item frequent pattern list**

The above figure 3 show this comparison and it can be observed that according to minimum support as minimum support increases frequent item becomes decreases and as minimum support increased frequent items in set going to increase.

8. **CONCLUSION**

In this work consider following factors to generate our new scheme, are the time and the memory consumption. These factors are affected by the approach for finding the frequent itemsets. Paper gives improvement apriori and FP tree algorithm by space reduction method and fast processing by applying transactional dataset. The key idea
behind finding frequent pattern is to get common string between current combinations of itemset with every transition in transition dataset. Traditional algorithm such as dynamic algorithm is used 2 dimensional array for computing common sequence present in dataset which consumes $O(n^2)$ memory space, here this work proposed algorithm for LCS problem which takes $O(n)$ memory space for computing LCS for current items sequence with transition dataset. This proposed algorithm reduces space complexity a lot. Simulation work supports this statement. The proposed algorithm is efficient to reduced memory space almost 80% in compare to tradition algorithm without increasing time complexity for calculating frequent pattern. Overall time complexity can be observed by given time comparison in result and discussion result section. Therefore this work employed it in our scheme to guarantee the memory saving without effecting time in the case of transpositional structure of dense data sets. Thus this algorithm produces frequent itemsets completely. Thus it saves much space and time and considered as an efficient method as proved from the results.

REFERENCES


Memory Efficient Frequent Pattern Mining Using Transposition of Database


