Discovery of Frequent Closed Itemsets using Reduced Pattern Count Tree

Geetha M R.J. D'Souza

Abstract-- **In this paper, a new algorithm for mining frequent closed itemsets from large volumes of data is implemented. A frequent itemset is maximal if none of its proper supersets is frequent. The total number of maximal frequent itemsets *M* is much smaller than that of frequent itemsets *F*, and we can derive each frequent itemset from *M*. However, *M* does not contain information of the support of each frequent itemset unless it is a maximal frequent itemset. Thus, mining only maximal frequent itemsets causes loss of information. However, when a transaction database is very dense and the minimum support is very low, i.e., when the database contains a significant number of large frequent itemsets, mining all frequent itemsets might not be a good idea. The concept of closed frequent itemsets solves this problem. This approach, uses a tree based data structure called Reduced Pattern Count Tree, and discovers all closed frequent itemsets in one scan of the database. On the other hand, the current algorithms need at least two scans of the database except Pattern Count Tree based algorithm, which requires a single scan of the database, but uses Lexicographical Ordered FP-tree to discover all frequent patterns.**

Index Terms-- **Closed frequent itemsets, Path, RPC-tree, Support, Transaction head**

I. INTRODUCTION

An itemset is closed if none of its proper supersets has the same support as it has. The total number of closed frequent itemsets *C* is still much smaller than that of frequent itemsets *F*. Furthermore, we can derive *F* from *C*, because a frequent itemset *I* must be a subset of one or more closed frequent itemsets, and *I*'s support is equal to the maximal support of those closed itemsets that contain *I*.

Geetha M is Selection Grade Lecturer in the Computer Engineering Department of Manipal Institute of Technology, Manipal, 576104, Karnataka India (Phone : 91-9845784282
E-mail maiya_geetha@yahoo.com).

R.J. D'Souza is Professor in the Department of Mathematical and Computational Sciences, National Institute of Technology Karnataka, Surathkal, India

In summary, the relation among F, C, and M is $M \subseteq C \subseteq F$.

The discovery of association rules is the most well studied problem and is an important problem in data mining. Let $I= \{i_1, i_2, i_3, \ldots, i_m\}$ be a set of items. Let D be a set of transactions, where each transaction T contains a set of items. A transaction t is said to support an item i_i, if i_i, is present in t. t is said to support a subset of items *X* contained in I, if t supports each item in *X*. An itemset X contained in I has a support s in D, if *s*% of transactions in D supports *X*. An itemset with at least the user defined minimum support is called a frequent itemset. In order to discover all frequent itemsets for a given database, it is enough to find all its closed frequent itemsets since $M \subseteq C \subseteq F$.

An association rule [2] is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \varphi$. The association rule $X \Rightarrow Y$ [2] holds in the database D with support s if s% of transactions in D contains $X \cup Y$. The association rule $X \Rightarrow Y$ [2] holds in the database D with confidence c if c% of transactions in D that contains X also contains Y. Mining of association rules is to find all association rules that have support and confidence greater than or equal to the user-specified minimum support and minimum confidence respectively [1]. The first step in the discovery of association rules is to find all frequent itemsets with respect to the user specified minimum support. The second step in forming the association rules from the frequent itemsets is straight forward as described in [1]. There are many interesting algorithms for finding frequent itemsets. The FP- Tree Growth algorithm [2] as proposed by Han *et al* requires two scans of the database to discover all frequent itemsets. The Pattern Count tree (PC-tree) is the contribution of V.S.Ananthanarayana *et al*, which is a complete and compact representation of the database. They discovered frequent itemsets by converting PC-tree to LOFP tree [4] in a single database scan. PC-tree [3] is a data structure, which is used to store all the patterns occurring in the tuples of a transaction database, where a count field is associated with each item in every pattern, which is responsible for a compact realization of the database. The

completeness property of the PC-tree [3] motivated us to discover all closed frequent itemsets using Reduced Pattern Count-tree.

II PROPOSED ALGORITHM

The Reduced Pattern Count -tree algorithm (RPCA) does not generate any candidate itemsets, and through this algorithm it is possible to discover all closed frequent itemsets. The steps to be followed for finding the closed frequent itemsets using Reduced Pattern Count tree (RPC –tree) are

Step 1: Construction of Pattern Count tree [4].

Step 2: Reduction of Pattern Count tree to contain only frequent items:

A. Removal of Infrequent items:

The items, which are not frequent, are removed from the PC -tree.

```
for all items I in the PC –tree do begin
   if I is not frequent , delete node I from the PC -tree
end.
```

B. Removal of Repeated Transactions heads:

It is assumed that the transaction head in the tree is the node through which one or more transactions originate and there are n transactions heads in the PC –tree. The above step may leave PC-tree with repeated transactions heads. The merging algorithm removes such transactions heads.

```
for all transactions head h from 1 to n-1 do begin
   for all transactions head j from 2 to n do begin
      if (h = = j)
         append(h, j); // Appending the transactions head h to j.
    end
 end
```

C . Removal of repeated Siblings:

The PC-tree, which is reduced in the above step, may contain repeated sibling heads. i.e. even though the PC–tree possesses different transaction heads, for a particular head any two siblings may have same labels. The following algorithm removes repeated siblings.

```
for all transactions head h from 1 to n do begin
  Remove a transaction head h from PC–tree, which results in
  a tree called XPC –tree.
  Append all transactions beginning with h from the removed
  branch of PC-tree, to the transaction head h1 having the label
  of h.
  The transaction head h1 which is obtained in the above step
  is attached to XPC –tree.
end.
```

The resulting tree at the end of this step is called Reduced PC –Tree (RPC -tree).

Step 3: Discover closed frequent itemsets by finding all possible paths and its frequency for all one large frequent itemsets.

This step involves several sub-steps.

Reverse the set of frequent 1-itemsets f in descending order of their item number. It is assumed that there are n one frequent itemsets in f; p is a path structure to store all possible paths; h a transactions head in a reduced PC -tree; I an item in the set f; T holds the elements which are frequent with I; m the number of elements in T; Count[I] contains frequency of an item I; r is a path structure to store closed frequent itemsets; fcount[k] represents the frequency of reduced path r_k; N the number of reduced paths in r and s a user defined support.

```
    for every item I in f do begin
        Do a preorder scan of the Reduced PC-tree to get all
        possible paths ending with item number I.

        for all transactions heads h in the reduced PC-tree do
        begin

     if (h> = I) remove h from the reduced PC -tree;
     else {store count[I ] as first element, all other elements
   before I and element I, as next in path p }
continue search for I in Reduced PC –tree until the end of the
tree
 end.
/*Reduce all the paths in p to contain only frequent itemset
associated with I. */
    for all paths p do begin
        for all I in p do begin
         If I is not frequent, remove I from p
           else, add I to the set T.
    end
   end
/* Finding the association amongst the elements in T. */
   for all Ij in T, j from 1 to m do begin
   for all reduced paths p do begin
      if Ij € p, put all elements after Ij and before I in r.
                  count[ j ]=count[ Ij ]
  end
/*Reduce r*/
       for all paths r do begin
              for all I in r do begin
                   if I is not frequent with Ij remove I from r
                end
                if r contains more than two frequent items then
                  count[ r ]=count[ I ]
       end
/*Finding the frequency of reduced paths.*/
  for all reduced paths rk , k from 1 to N do begin
       for all reduced paths rl , l from 1 to N, k ≠ l do
            begin
                 If (rk= = rl)
                   N=N-1(Delete rl)
                  count[ k ]= count [k ]+ count [ l ]
                  fcount[k]=count[ k]
                If ( rk is contained in rl )
     count[ k ]= count [k ]+ count [ l ]
    fcount[k]=count[ k]
   end
end
/*finding Potential closed frequent itemsets*/
```

Initialize i to 1
for all reduced paths r_k , k from 1 to N do begin
if (fcount[k] >= s) and there exists no other path containing r_k with the same support as that of r_k then move r_k to Ci and increment i by 1;
else ignore r_k
end
end
end
/* finding frequent closed itemsets*/
n=i
for all reduced paths C_k , k from 1 to i-1 do begin
for all reduced paths Cj , j from 2 to i do begin
if Ck is contained in any of the Cj with the same support then ignore Ck, n=n-1;
else
end
end
/* Recovering proper subsets which are closed frequent itemsets. */
for all reduced paths C_k , k from 1 to n-1 do begin
for every subset s of Ck containing 2 or more items begin
if s is properly contained in any of the Cj, j from 2 to n and s is not equal to any of the Cj then
add s to closed frequent itemset if count of s is not equal to count of Cj;
end
end

The working of above algorithm is explained with the help of the following example.
Consider a sample database given in Table1. The user defined minimum support is 20% of 15 transactions. This means 3 transactions.

Table1: A Sample Database

Transaction ID	Item numbers of items Purchased(Transactions)
1	1, 5, 6, 8
2	2, 4, 8
3	4, 5, 7
4	2, 3
5	5, 6, 7
6	2, 3, 4
7	2, 6, 7, 9
8	5
9	8
10	3, 5, 7
11	3, 5, 7
12	5, 6, 8
13	2, 4, 6, 7
14	1, 3, 5, 7
15	2, 3, 9

PC-tree corresponding to the above database is given in Figure1.

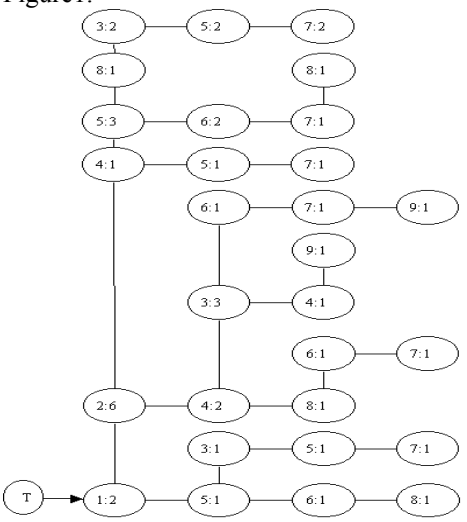

Figure1: PC-Tree corresponding to a Sample Database given in Table1.

REMOVAL OF INFREQUENT ITEMSETS:
The item numbers 2, 3, 4, 5, 6, 7, 8 become frequent–1 itemsets since their frequencies are greater than or equal to user defined minimum support. Also, item numbers 1 and 9 are considered to be infrequent since their frequencies are less than the user defined minimum support value. Hence the nodes labelled with 1 and 9 are removed from the PC-tree given in Figure1. The resulting PC-tree is as shown in Figure 2.

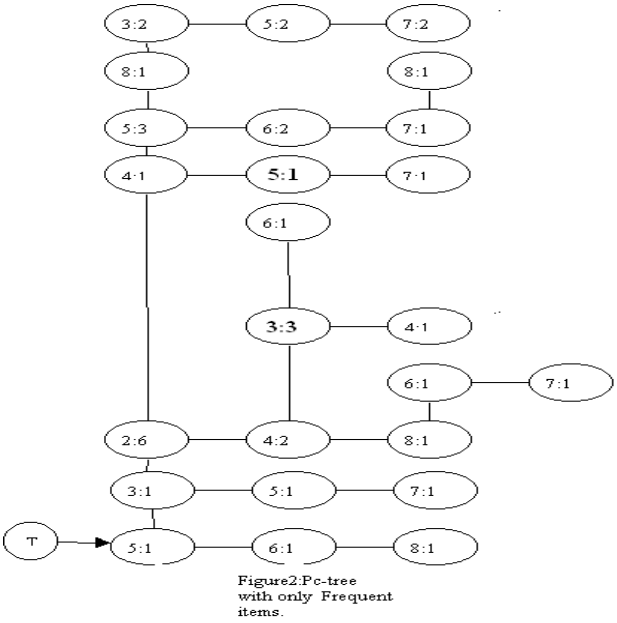

Figure2:Pc-tree with only Frequent items.

REMOVAL REPEATED TRANSACTION HEADS:
The above PC-tree contains repeated transaction heads labeled with item numbers 5 and 3. They are merged at this level and the resulting PC- tree is shown in Figure 3.

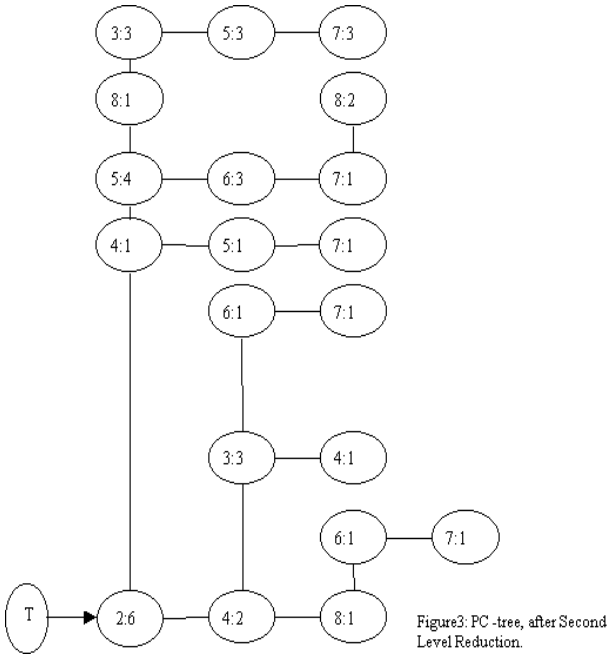

Figure3: PC -tree, after Second Level Reduction.

REMOVAL OF REPEATED SIBLINGS:

Since Figure 3 does not contain repeated sibling heads, it represents the Reduced Pattern Count Tree.

Step 3: We reverse the array *f* of frequent 1- itemsets i.e. *f* contains item numbers 8, 7, 6, 5, 4, 3, 2.

Iteration 1

The first element in *f* is 8. i.e. let the current item number I be equal to 8.

The Reduced PC-tree contains a node labeled 8 as a transaction head. Therefore, it is removed from the reduced PC -tree. The resulting tree is shown in Figure 4.

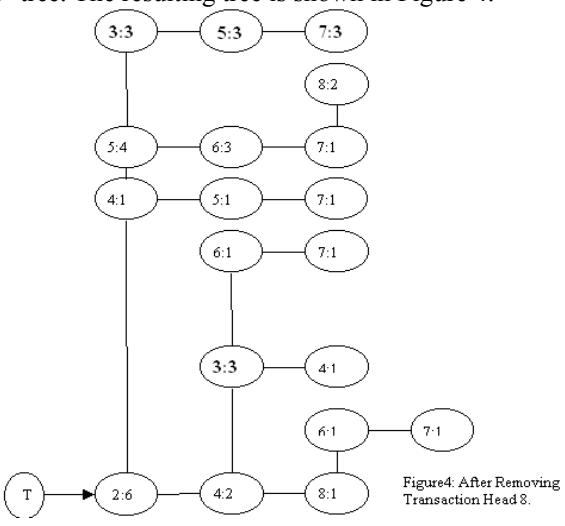

Figure4: After Removing Transaction Head 8.

The possible paths *p* ending with 8 are {2:5,6,8} {1:2,4,8}. Since item numbers 2, 4, 5, and 6 do not have user defined minimum support, there is no frequent path ending with item number 8.

Iteration 2

The second element in *f* is 7. It can be seen from Figure 4, that any of the transactions heads of reduced PC-tree is not labeled 7.

The possible paths *p* ending with 7 are {1: 2, 4, 6, 7}, {1: 2, 6, 7}, {1: 4, 5, 7}, {1: 5, 6, 7}, {3: 3, 5, 7}. We see that the item numbers 3, 5, 6 have user defined minimum support. Let *T*={3, 5, 6 }. The only potentially frequent 2-closed itemsets are C1={5:5, 7} and C2 = {3: 6, 7}. The frequent itemset {3: 3, 7} is not a closed itemset since support of {3: 3, 7} is the same as that of its superset {3: 3, 5, 7}.

Now eliminating all item numbers from *p* except those from T, the reduced paths in *p* are

{1: 6, 7}, {1: 6, 7,} {1: 5, 7}, {1: 5, 6, 7}, {3: 3, 5, 7}.

The reduced paths in *p* are

{3:6, 7}, {5:5,7}, {1: 5, 6, 7},{3: 3, 5, 7}.

Now consider the first element of T, i.e. 3. It is clear that there is only one path having elements after 3 and before 7 and its support is greater than user defined minimum support. Therefore, the path C3={3: 3, 5, 7} is a potential closed frequent 3-item set beginning with 3 and ending with 7. By similar arguments, it can be shown that {5:5,7} and {3:6,7}are potential closed frequent itemsets , which begin with 5 and 6, and end with 7 respectively.

Iteration 3

The third element in *f* is *6*. The possible paths *p* ending with 6 are {1: 2, 6},{1:2, 4, 6},{3: 5, 6}. Since the item number 5 occurs 3 times, the set *T* = {5}. Therefore, C4={3: 5 , 6} is the only one potential closed frequent 2- itemset ending with 6.

Iteration 4:

The next element in *f* is *5* and is the transactions head of Reduced PC-tree. Therefore, that particular branch is removed from the RPC-tree. The resulting tree after removing 5 is shown in Figure 5.

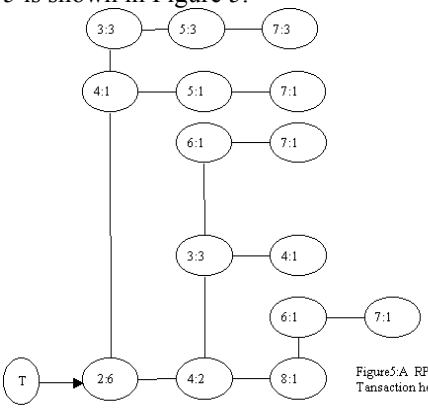

Figure5:A RPC-tree after removal of Tansaction head 5

The possible paths *p* ending with 5 are {1: 4, 5}, {3: 3, 5}. Since the item number 3 has minimum support, we have T= {3}. Therefore, C5={3: 3, 5} is the only one potential frequent 2-itemset ending with 5 .

Iteration 5

The next element in *f* is 4. The resulting RPC –tree after removing the transaction head labeled 4 from Figure 5 is shown in Figure 6.

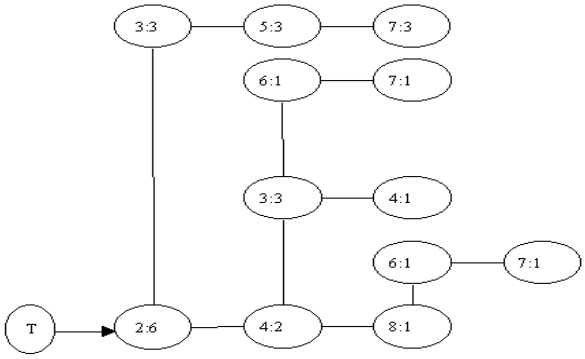

Figure6: RPC -tree After Removing Transaction Head 4

The possible paths *p* ending with 4 are {2: 2, 4}, {1: 2, 3, 4}. Since the item number 2 satisfies user defined minimum support, we have *T*= {2}.Therefore, the potentially frequent-2 closed itemset obtained is C6={3: 2, 4}.

Iteration 6

The next element in *f* is 3. The resulting RPC -tree after removing transaction head labeled 3 is shown in Figure 7.

The path ending with three is C7={3: 2, 3}. Since this path satisfies user defined minimum support, and there exists no other path with the same support containing it, C7 is a potentially closed frequent itemset.

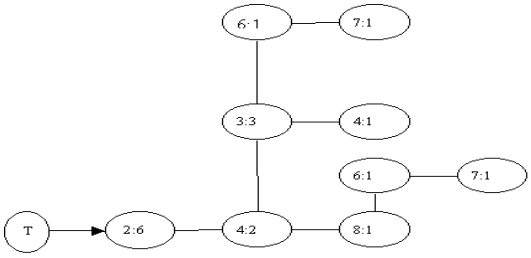

Figure7: A RPC -treea fter removal of Transaction Head 3.

Iteration 7

The Reduced PC-tree obtained in the previous iteration, contains only one transaction head, which is 2. The removal of this node results in an empty tree. Therefore, there is no path ending with 2. This completes all iterations.

At the end of this iteration we have 6 potentially closed frequent itemsets. They are C1={5:5, 7} C2={3: 6, 7}. C3={3: 3, 5, 7} C4={3: 5 , 6}, C5={3: 3, 5} C6={3: 2, 4}, C7={3:2,3}

Applying the property of closed frequent itemsets we get only C1={5:5, 7}, C2={3: 6, 7}. C3={3: 3, 5, 7} C4={3: 5 , 6} and C6={3: 2, 4} C7={3:2,3}as frequent closed itemsets along with closed frequent 1-itemsets such as {2}, {3}, {4}, {5}, {6}, {7}, {8}.

III PERFORMANCE ANALYSIS

The above algorithm is checked for data sets having 1K, 5K, 10K, 15K, 20K, 25K and 50K transactions against the support value .75 % and the result is compared with FP –tree to discover all frequent itemsets. The time graph is shown in Figure 8.

Table2: Parameters and data set Used

|D|	Number of Transactions
|T|	Average size of a transactions
|I|	Average size of maximal potentially large itemsets.
|L|	Number maximal potential large itemsets.
NOI	Number of items.

Table3: Data sets used

Sets	|D|	|T|	|I|	|L|	NOI
Set_1	1K	10	4	1000	1000
Set_2	5K	10	4	1000	1000
Set_3	10K	10	4	1000	1000
Set_4	15K	10	4	1000	1000
Set_5	20K	10	4	1000	1000
Set_6	25K	10	4	1000	1000
Set_7	50K	10	4	1000	1000

Table4 : Execution Time in seconds

Sets	FP-tree	RPC-tree
1	23	19
2	28	24
3	42	38
4	61	52
5	74	67
6	95	85
7	188	174

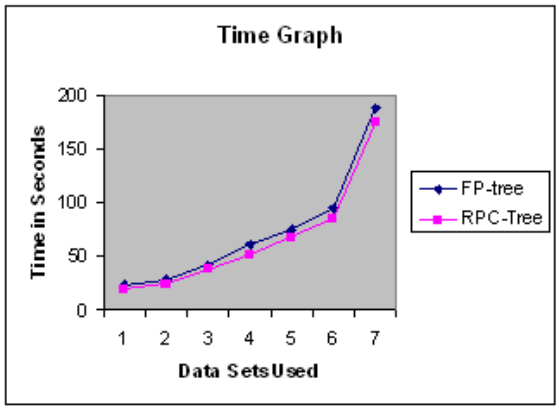

Figure 8: Time Graph

Theoretical Comparisons with some Algorithms:

1. We know that for Apriori algorithm, the database has to be read at least k times if we have to find frequent itemsets of length k . For our method, only one scan is required to discover all closed frequent itemsets. This will save considerable execution time.
2. If N is the number of transactions in the given database, our method requires exactly one scan to read the N transactions and construct Reduced Pattern Count Tree. Therefore our algorithm is in O(N), where as Apriori, which depends on size of large itemsets Lk, is in O(k*N).
3. Even though the algorithms Max Miner[10] or Pincer search[9] can discover closed frequent itemsets by examining obtained maximal frequent itemsets, they require more than one database scan and involve candidate generation method. Since our approach uses a single data structure which is a compact and complete representation of the given database, and does not use any candidates generation method, it is clearly space efficient.
4. The algorithm CHARM, is found be efficient for discovering closed association rules [12] but requires multiple passes to discover all frequent closed itemsets. It is calculated by taking the sum of the lengths of all tidsets scanned from disks, and then dividing the sum by the tidset lengths for all items in the database.
5. The Algorithm AClose [11] is an Apriori –like algorithm that directly mines closed frequent itemsets, but involves candidate generation method. This algorithm can perform an order of magnitude better than Apriori for low support values, but for high support values, it can in fact be worse than Apriori. This is because for high support the number of frequent itemsets is not too much, and the closure computing step of AClose dominates computation time. Like Apriori, Aclose could not be run for every low values of support. The generator finding step finds many generators to be kept in the memory.
6. In view of all the above, our approach is extremely effective in efficiently mining all the closed frequent itemsets, and is able to gracefully handle very low support values, even in dense datasets.

IV CONCLUSION

In this paper, we proposed a novel method for discovering closed frequent itemsets directly by reducing PC-tree, which requires only one scan of the database. Also, it is found that the algorithm is more time efficient than FP-tree, Apriori, Aclose etc since these algorithms require more than one scan of the database. In addition, our algorithm works for any small support value. However, execution time increases a little as the size of the database increases. As a future investigation a method may be developed to improve the time efficiency of our method.

REFERENCES

[1] Jiwaei Han, Micheline Kamber "Data Mining Concepts and Techniques".

[2] Arun K Pujari – "Data mining Techniques. "

[3] Ananthanarayana V. S, Subramanian, D.K., Narasimha Murthy, M- Scalable, distributed and dynamic mining of association rules pp559-566, 2000.

[4] Ananthanarayana V. S, Subramanian, D.K., Narasimha Murthy, M- "Scalable, distributed and dynamic mining of association rules using PC – tree" IISc –CSA, Technical Report, (2000).

[5] Rakesh Agrawal, Tomasz Imielinski, Arun Swami, "Mining Association Rules between Sets of Items in Very Large databases." Proc of, ACM SIGMOND Conf Management of Data , pp,207-216,1993

[6] Rakesh Agarwal, Ramakrishnan Srikant,- "Fast algorithms for mining Association Rules in Large databases" Proc of, 20th Int'l conf very large Databases, pp, 478-499,1994

[7]Fayyad U. M., Piatetsky-Shapiro G.,Smith P., Uthurusamy R. (Eds.):Advances in Knowledge discovery and Data mining.

[8]Han, J., Pei, J., Yin, Y. Mining Frequent Patterns without Candidate Generation, Proc. Of ACM-SIGMOD, (2000).

[9] D-I. Lin and Z.M. Kedem. Pincer-Search: A new algorithm for discovering the maximum frequent set. In the 6th Intl. conf. Extending Data base Technology, March 1998.

[10] R.J Bayardo." Efficiently mining long patterns from databases". In ACM SIGMOD conf. Managemnet of Data, June 1998.

[11]N. Pasqquier, Y Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In 7th Intl. Conf. on Database Theory, January 1999.

[12] Mohammed J.Zaki and Ching-Jui Hsiao "CHARM:An Efficent Algorithm for closed Assocaition Rule Mining"

[13] IBM/Quest/Synthetic data.