ALGORITHM FOR MINING TIME VARYING FREQUENT ITEMSETS

D.SUJATHA[1], PROF.B.L.DEEKSHATULU[2]
[1]HOD, Department of IT, Aurora's Technological and Research Institute, Hyderabad
[2]Visiting Professor, Department of C&IS, University of Hyderabad

E-Mail: sujata_dandu@yahoo.com, deekshatulu@hotmail.com

ABSTRACT

A popular subfield of data mining is the area of association mining that searches for frequently co-occurring items in a market-basket type database. A market basket is the list of items a customer purchases at the super market. We can analyze past transaction data to discover customer behaviors such that quality of business decision can be improved. The Input of Association mining is large set of transactions each consisting of a list of items a customer has purchased at a supermarket. This paper focuses on the composition of the list of frequent itemsets that may change in time as the purchasing habits get affected by season, fashion and introduction of new products. The output of this algorithm is to get the frequent items which occur after a particular item or item sets.

Keywords: *Data Mining, Association Mining, Algorithm, Frequent Items*

1. INTRODUCTION

This algorithm is used for mining frequent item sets which is based on prefix tree representation of given database of transaction (called an FP tree), which can save considerable amount of memory for storing the transactions. An FP-tree is basically a prefix tree for the transactions. That is, each path represents a set of transactions that share the same prefix, each node corresponds to one item. In addition, all nodes referring to the same item are linked together in a list, so that all transactions containing a specific item can easily be found and counted by traversing this list. The list can be accessed through a head element, which also states the total number of occurrences of the item in the database.

In my implementation the initial FP-tree is built from a main memory representation of the (preprocessed) transaction database as a simple list of integer arrays. This list is sorted lexicographically (thus respecting the order of the items in the transactions, which reflects their frequency). The sorted list can easily be turned into an FP-tree with a straightforward recursive procedure: at recursion depth k, the k-th item in each transaction is used to split the database into sections, one for each item. For each section a node of the FP-tree is created and labeled with the item corresponding to the section. Each section is then processed recursively, split into subsections, a new layer of nodes (one per subsection) is created etc. Note that in doing so one has to take care that transactions that are only as long as the current recursion depth are handled appropriately, that is, are removed from the section before going into recursion.

2. BACKGROUND

During the process of mining frequent itemsets, when the minimum support is little, the production of candidate sets is a kind of time consuming and frequent operation in the mining algorithm. The FP-growth algorithm does not produce the candidate sets, the database is scanned twice. During the first database scan the number of occurrences of each item is determined and infrequent ones are discarded. Then the frequent items are ordered descending their support. During the second database scan the transactions are read and the frequent items of them are inserted into a so called FP-tree structure. In this way the database is pruned and is compressed into the memory.

In FP growth algorithm the order in which the items are taken is considered and accordingly the FP tree is constructed and from this FP tree frequent items are generated. The FP growth algorithm mines the frequent items which occur with the particular item or set of items but this algorithm does mining to get the frequent items which occur after a particular item or item sets.

3. METHODOLOGY

By using this FP tree will mine the entire time varying frequent item set. The idea of this algorithm is to travel through the tree on the basis of greater count there by getting the next frequent items. The algorithm reduces the total number of candidate itemsets by producing a compressed version of the database in terms of FP-tree. The FP-tree stores relevant information and allows for the efficient discovery of frequent itemsets.

The algorithm is constructed in two steps:
1. Building of the FP-tree.
2. Mining the tree to find the frequent itemsets.

3.1. Building of the FP-Tree:

steps 1. Frequent itemsets along with the count of transactions containing each item are computed. The itemsets are sorted in non-decreasing order. The root of the FP-tree is created with a “null” label.

steps 2. For each transaction T in the database, place the frequent itemsets in T in sorted order. Designate T as consisting of a head and the remaining items i.e. the tail.

steps 3. Insert itemset information recursively into the FP-tree with an item name=head, increment the count associated with N by 1 else create new node, N with a count of 1, link N to its parent and link N with the item header table. If tail is nonempty, repeat the above step using only the tail i.e. the old head is removed and the new head is the first item from the tail and the remaining items become the new tail.

Example:

FP-tree usually contains the null node as a root node. The construction of FP-tree is shown below. Before building the tree, the items in a transaction are arranged in the increasing order. Each node in the tree contains the following information: content, count and pointers of the sub tree. Similar nodes in different transactions are linked together using pointers.

A. Primitive Transactional Database

Let the transactional database have 5 transactions. We use Table 1 to illustrate the algorithm for finding frequent item-sets.

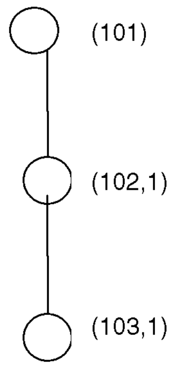
Table 1: Sample Database

T1 D	Items Bought
1	101,102,103
2	101,105
3	101,104
4	101,104,105,106
5	101,104

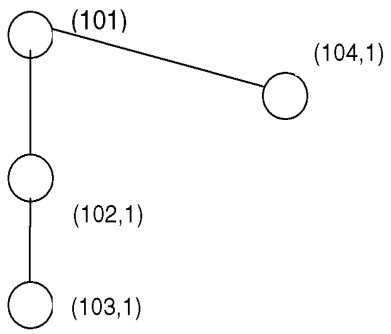
B. Construction of FP-Tree

With the above observations, one may construct the frequent-pattern tree as follows: Null node is taken as the root node, after that we take the first transaction. In that transaction we select the item which is brought first, we will check if that item is present as a child node to the root node, if it is already present then we increase the count or else we create a new sub tree.

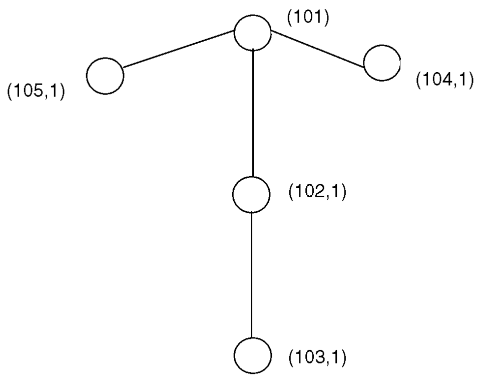
Step 1 : TD1



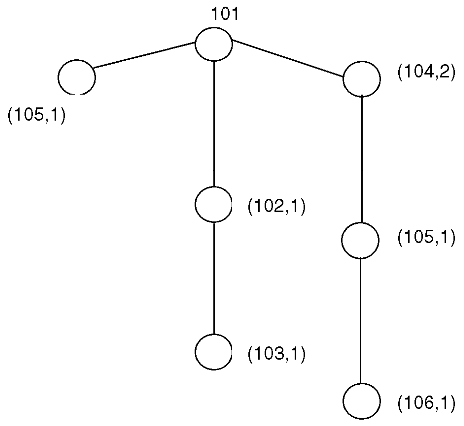
Step 2 : TD1+TD2



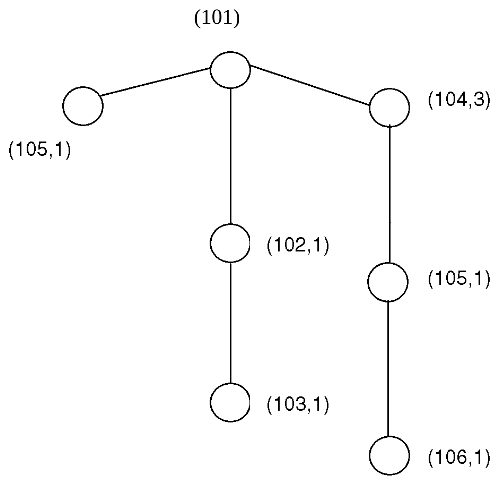
STEP 3 : TD1 + TD2 + TD3



STEP 4 : TD1 + TD2 + TD3 + TD4



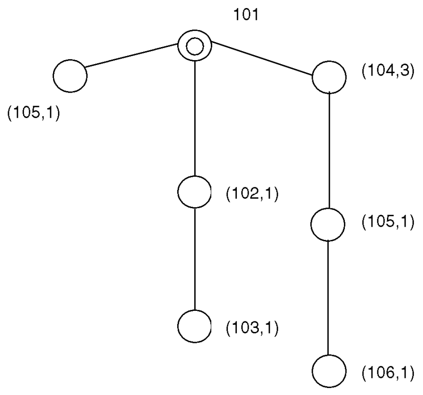
STEP 5 : TD1 + TD2 + TD3 + TD4 + TD5



3.2. Mining the tree to find the frequent itemsets.

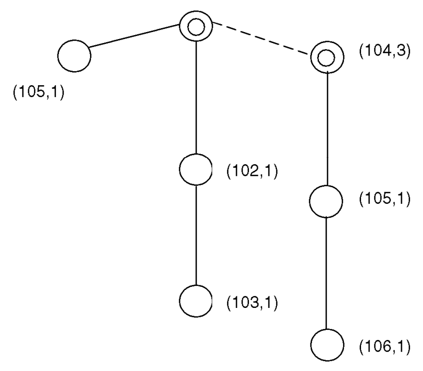
Once the FP-tree is constructed we mine the frequent time varying items by applying the algorithm. The frequent items are known as the basis of count of the particular node. For a particular item if you want to known which item is been frequently after that is known by comparing the count of the child node, the child node which has highest count is to be considered to be brought frequently after that item.

If you want to know which three items are purchased after a particular item, we will make the item as the root. For Ex: if we want to know the items that are purchased after 101

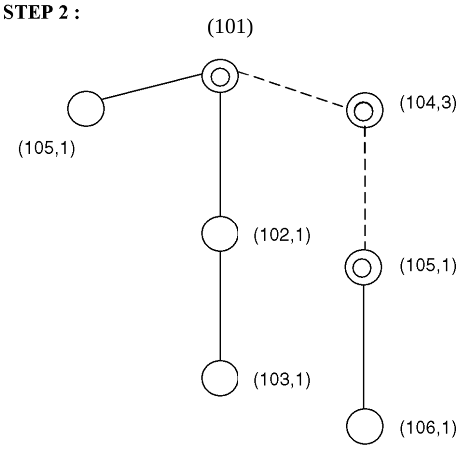


Then we will recursively search for the nodes which has greater count, save that value and move to that particular node.

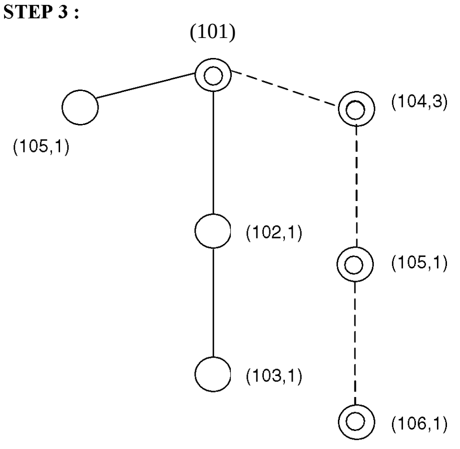
STEP 1 :



STEP 2 :



STEP 3 :



Algorithm

Input: D, Transactions list; item or itemsets, the count of frequent itemsets needed

Output: Constructed fp tree and frequent itemsets in D

Method:

```
IP_Page();/* To display the input screen */
Insert(int Value);/* To insert into the tree */
Mining();/* perform mining on the tree */
Display();/* To show the output */
```

Mining Algorithm

```
        Main
        Begin
                Initialize global variables;
                Initialize graphics mode;
        If graphics mode is not initialized then
                Display bgi error "graphics
                mode is not initialized"; /*
                error message */
        Endif
         Call IP_page(); /* to display the input
        screen */
        Initialize root;
        Open Transaction_file(DB);
        If unable to open then
                Display error "cannot open
                file";
        Endif
        Read transaction from file
        While  not EOF then
        Begin
            If transaction contains given
            IP_items then
                Insert (next item);
             Endif
             Else
                Continue
             End else
        End
        Call Mining(); /* after constructing the
        tree start the mining */
        Call Display(); /* display the frequent
        items */
        End
```

```
/*****************IP_Page**************/

Void IP_Page() /*  input page display */
Begin
        Display input page using graphics;
End


/*****************Insert () **************/

Int Insert (value)
Begin
I=0;
While i<no of child of root then
If root contains value as child node then
        Increment count
        Update root to that particular child;
End if
Else
        Create new node;
        Initialize variable of that particular
        node;
        Add this node to the root node;
End else
End

/************Mining() *****************/
Void mining ()
Begin
        Start from root
        Check the child node which contains
        the greatest count
        Begin
                Store that node in an array
        End
        Repeat the process until you get the
        required frequent items
End


/*************Display()***************/
Void Display ()
Begin
        Display the tree;
        Display the frequent time varying
        items with count
End
```

4. ALGORITHM PERFORMANCE ANALYSIS

4.1 Space complexity analysis

The space cost of this algorithm is mainly storing the FP-Tree and the size of the FP-Tree is decided by the length of affair database. Initially the FP-Tree grows as the database increases but after reaching a particular stage the tree remains constant but the count of node only increases. Even if the database increases, the tree remains constant therefore less space is required to store.

4.2 Time complexity analysis

The time complexity of the algorithm depends on the scanning of the database and then mining it. As the algorithm requires only one scan of the database which is decided by its length, So the average time is O(n).The time complexity for mining is similar to searching of an n-ary tree. For a n-ary tree with height h, the upper bound for the maximum number of leaves is n^h.

5. CONCLUSION

In this paper, we discuss the algorithm for time varying frequent itemsets which allows using FP-tree more effectively for mining frequent itemsets. The results illustrate that the algorithm can be implemented to know the item or set of items that are purchased immediately after a particular item.

REFERENCES:

[1] J.Han, J.Pei, and Y.Yin, "Mining Frequent Patterns without Candidate Generation," Proc, ACM-SIGMOD Int'1 Conf. Management of Data, pp. 1-12, May 2000.

[2] R.Gopalan, and Y.G.Suchayo, *TreeTL-Mine: Mining Frequent Itemsets Using Pattern Growth, Tid Intersection and Prefix Tree, in Proceeding of 15th Australian Joint Conference on Artificial Intelligence,* Canberra, LNAL, 2557, Springer, 2002

[3] K. Wang, L, Tang, J. Him, and J.Liu, 'Top Down FP-Growth for Association Rule Mining," *Proceedings of the 6th Pac$c-Asia Conference on Advances in Knowledge Discovery and Data Mining, 2002*

[4] R. Agarwal, C. Agarwal, and V.V.V.Prasad, "A Tree Projection Algorithm for Generation of Frequent Itemsets," *Journal on Puraflel and Distributed Computing,* 2000, Vol.61, pp, 350-371

[5] S.Brin, R.Motwani, J.Ullman, and Tsur, S.Dynamic Itemset counting and implication

rules for market basket data, In Proc. Of ACM SIGMO, pp.225-264, 1997

[6] R.Gopalan, and Y.G.Sucahyo, mining Frequent ItemsetsMore Efficiently, in Proceedings of 2002 International Conference of Fuzzy Systems and Knowledge Discovery Singapore, 2002

[7] Y.G.Sucahyo, and R.Gopalan, Efficient Frequent ItemSet Mining using a Compressed Prefix Tree with PatternGrowth, in Proceedings of 14th Australian Database Conference, Adeliade, Australia, 2003

[8] S.Lu, Z, Lu, "Fast mining maximum frequent itemsets", Journal of Software, 2001, 12(2):293-297

[9] Y.Song, Y.Zhu, Z.Sun, "An Algorithm and Its Updating Algorithm Based on FP-Tree for Mining Maximum Frequent Itemsets", Journal of Software, 2003, 14(9): 1586-1592.

[10] D.Lin, Z.M.Kedem, "A new Algorithm for discovering the maximum frequent set", IEEE Transactions on Knowledge and Data Engineering, 2002, 14(5): 553-566