# Divide and Conquer Approach to Mine High Utility Itemsets Represented in Tree Data Structure

**Ganesh Sawant**
Department of Computer Engineering
Annai Velankanni College Tholayavattam

*Abstract*— The Process of mining high utility of itemsets refers to identification of profitable and important items from set of transactions. Unit transaction consists of transaction id, quantity and name of purchased item. To access transactions, they are stored in tree data structure because tree data structure allows transactions to store in compact and memory efficient way. However for large transaction database loading of whole transaction tree into memory is expensive task. In this paper divide and conquer technique is discussed which avoids loading of large transaction tree into memory and also helps to gets some seasonal profitable items.

***Key words:*** Divide and Conquer Approach, Tree Data Structure

## I. INTRODUCTION

Basically in high utility itemsets mining, transactions are mined to find out profitable and important itemsets. Such profitable itemsets helps to meets growing business demand and earn profit. There are many technique used for mining these can be categorized as into frequent pattern mining, weighted frequent pattern mining and high utility patter mining [1]. In Frequent pattern mining hidden patterns are found depending on its probability. Each item is associated with binary values in frequent pattern mining i.e. whether item is present or absent. This will leave items with low occurrences as less importance, which might not be a case. In next technique weighted association rule mining unit profit of each item in transactional dataset is considered and hence even if items are less frequent, they will be considered as high utility itemsets if they possess high weights. Even though weighted mining considers weight in form of profit it doesn't considers quantity of items. Therefore cannot satisfy requirements of users who are interested in discovering itemsets with high sales profit. The utility mining has become a useful technique to address above issues. Utility can be imagined as multidimensional term, which has different meanings according to context. For instance utility can refer to interestingness of itemsets, in another case it may signify high profit, while in some other perspective it may be related to importance of item in database.

Here transactions considered are item sales transaction which comprises of following fields named Transaction id, Item, Frequency/quantity, Time. Transaction id is unique id for transaction. Item is name of single item which was purchased. Frequency is number of item purchased and time is period when transaction was done, which is many date of transaction. There are many ways to store this transaction in memory for processing. Simple way could be just load all transaction into memory as it is from database. Which is not very space efficient because of memory requirement to store each transaction. More efficient way includes use of tree data structure to represent transaction. Tree data structure stores transactions in compressed form and thus memory requirement is much less. To further reduce the memory requirement divide and conquer technique is used which iteratively constructs tree for small chunks of transactions. Due to this for particular time instance, tree representing only part of database is loaded into memory. Here data structure used to store transaction is called UP-tree [10].

## II. LITERATURE REVIEW

This section goes through existing techniques used in mining itemsets from database. At first Apriori [2] algorithm was used to find association among itemsets. But for large set of transaction apriori loads all transactions into memory and thus increases the memory requirement, also it generated large amount of intermediate candidate itemsets. Next technique to overcome shortcoming of Apriori algorithm was frequent pattern growth mining algorithm also called as FP-Growth[3]. One of advantage of FP-Growth algorithm is it uses tree structure to store transactions thus we get benefit of less memory requirement. Second benefit is it mines frequent itemsets in just two scan without generating large number of candidate itemsets. Even though frequent pattern technique out performs Apriori, importance of item to user is not taken into consideration . The importance of items to user is one factor which needs to get considered, Weighted association rule mining [4], [5] is such technique later came into existence. In this technique relative importance of each item to user can be traced to identify high utility items in which user is interested. One drawback present here is weighted association rule mining does not have downward closure property which states that if itemset is infrequent then all of its superset are also infrequent itemsets. The downward closure property was addressed in [6] which states if itemset is not high utility then its superset is also not high utility itemset. The solution proposed makes use of transaction weight which considers importance of itemset and also maintain downward closure property. There is another algorithm called as Two phase algorithm which consist of two mining Phases [7]. In first phase Apriori based level-wise method is used to produce High Transaction Weighted Utility Itemsets (HTWUIs). In second phase high utility itemsets are produced. Two phase algorithm make use of transaction weighted downward closure property (TWDC) property and thus reduces the search space of itemsets. The pitfall of two phase was it generates lot of candidate itemsets in first phase. The Tree-Based algorithm named IHUP was later emerged which overcomes problem of large itemset generation and to store transaction in compressed form[8]. Tree-based algorithms makes use of tree structure to maintain information about itemsets. Tree structure helps to store transactions in compressed form. General node structure consist of item name, Transaction weighted utility (TWU) value and support count of item. Algorithm works in three steps, first

 

it constructs IHUP-Tree. While construction of tree transactions are rearranged in descending order of TWU or Support count or lexicographic order. Reordering of transactions helps in limiting long tree traversals. In second step HTWUIs are obtained by applying FP-Growth technique [3]. Then in last step high utility itemsets are gathered by applying one additional database scan. Although Tree-based algorithm reduces generation of large number of intermediate candidate itemsets but this count can be reduced further. Latest techniques such as Discarding Global Unpromising items (DGU) and Decreasing Global node Utilities (DGN) further reduces intermediate itemset generation [1]. Here we see how DGU and DGN can be applied for efficiently mining of high utility itemsets from transactional database. In short, earlier association rule mining technique such as Apriori technique considers all items in database as equal by only considering whether item is present or not. It doesnt consider how items are important to user. Then next frequent itemset mining technique named FPGrowth just considers frequency of occurrence of item without considering importance of item to user and hence contribute to small percentage of overall profit. Further weighted association rule mining itemset considers importance of itemset to user and contributes well to overall profit. DGU and DGN are kind kind of weighted association rule mining technique which try to reduce large intermediate itemset generation. But one Problem related with DGU and DGN strategies is, it misses out seasonal itemset which occur for small period of time but provides profit. Here divide and conquer method shows how seasonal profitable products can be mined without having high memory requirements.

## III. TRANSACTION STRUCTURE

As discussed earlier each transaction consist of Transaction id, Item name, Frequency of item and time period. Following table shows transaction format.

| TID | ITEM | FREQUENCY | TIMEPERIOD |
|---|---|---|---|
| T1 | A | 10 | 2014-01-01 01:00:00 |
| T1 | B | 10 | 2014-01-01 01:00:00 |
| T1 | D | 10 | 2014-01-01 20:00:00 |
| T2 | A | 7 | 2014-02-01 11:00:00 |
| T2 | B | 7 | 2014-02-01 09:00:00 |
| T2 | D | 10 | 2014-02-01 08:00:00 |
| T3 | A | 10 | 2014-03-01 07:00:00 |
| T3 | B | 10 | 2014-03-01 22:00:00 |

Fig. 1: Transactions structure

Fig 1 shows transaction structure in which for each TID say T1 we have items A, B, D with frequency/ Quantity 10,10,10 respectively. Each transaction also maintains the time period. Which specifies occurrence of transaction.

## IV. UP-TREE CONSTRUCTION

UP-Tree is tree data structure used to store transactions. UP-tree has root node {R} and remaining nodes in UP-Tree comprises of item name stored as N.name, its utility N.nu, its parent N.parent, its horizontal link N.hlink, N.link node and set of child nodes. Fig 2 shows diagrammatic representation of UP-Tree [1]. For item name{C} in fig 2, '11' is node utility and '2' is occurrence count. Detailed procedure to calculate N.nu, is show in [1].

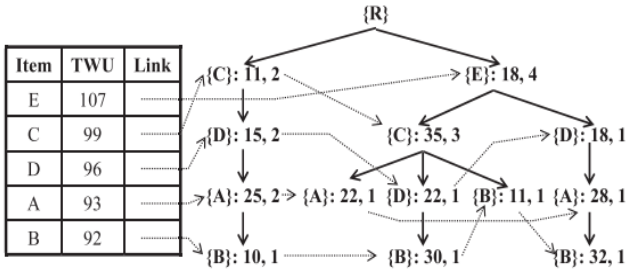


Fig. 2: UP-Tree structure.

N.linkNode is a boolean attribute use as indicator to attach child node. For example if value of N.linkNode is found to be true then it indicates that Node to be inserted in tree must be attached as child of N node. Following algorithm shows steps to construct UP-tree.

Algorithm - UP-Tree Construction.
for each transaction 'Ti' in database do
for each item 'Xi' in 'Ti' do
Calculate the utility' Ui 'of item 'Xi'.
insertNodeToTree(Xi, Ui);
end for
end for
Procedure insertNodeToTree (Xi, Ui)
{
Nr = getRootNodeOfTree();
If Nr.linkNode == true
Set Xi as Child of Nr
Nr.linkNode = false
Xi.linkNode = true
return
else
Nchild = Get child nodes of root element Nr.
for each child Nc in Nchild
setRootOfTree(Nc)
insertNodeToTree(Nc)
end for
}

## V. DIVIDE AND CONQUER TECHNIQUE

Instead of processing whole items in transaction at one go, dividing transaction into period wise chunks saves lot of memory and processing time. This is because at particular instance of time only small part of transaction is into database. Mining process applied in [1] processes all transactions at one time and hence is memory expensive. Also seasonal profitable products are missed in this mining process. To overcome this issues process transactions on the basis of time period. UP-Tree would be constructed only for transaction in particular duration and hence size of UP-Tree will also be small. Next mine this UP-Tree for particular duration and gather profitable itemset. In next iteration process transactions for next interval of time period and gather profitable transactions into this time period and add to previous set of profitable transactions. Do this procedure until all transactions are processed.

Algorithm – Divide and conquer.
Tlimit = Get Time Interval limit
ProfitableTrans = []
while transactions in database do
*1) Calculate time interval range Tij ,*
Where i – starting date.
j- starting date + Tlimit

 

*2) Construct UP-Tree for transaction in interval Tij.*
*3) Mine UP-Tree to produce profitable itemset Pi.*
*4) Store Pi into ProfitableTrans*
*5) ProfitableTrans = ProfitableTrans + Pi*
end while

## REFERENCES

[1] Vincent S. Tseng et.al ”Efficient algorithms for mining high utility Itemsets from transactional databases”, IEEE transaction, vol 25, Aug 2013.
[2] R. Agrawal and R.Srikant, ”Fast Algorithms for Mining Association Rules”, Proc. 20th Intl Conf. Very Large DataBases (VLDB), pp. 487-499,1994.
[3] J. Han, J. Pei, and Y.Yin, ”Mining Frequent Patterns without Candidate Generation”, Proc. ACM-SIGMOD Intl Conf. Management of Data pp. 1-12, 2000.
[4] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong, ”Mining Association Rules with Weighted Items”, Proc. Intl Database Eng. And Applications Symp. (IDEAS 98), pp. 68-77, 1998.
[5] K.Sun and F.Bai, ”Mining Weighted Association Rules without Preassigned Weights”, IEEE Trans. Knowledge and Data Eng.,vol. 20, no. 4, pp. 489-495, Apr. 2008.
[6] F. Tao, F. Murtagh, and M. Farid, ”Weighted Association Rule Mining Using Weighted Support and Significance Framework”, Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD 03), pp.661-666, 2003.
[7] Y. Liu, W. Liao, and A. Choudhary, ”A Fast High Utility Itemsets Mining Algorithm”, Proc. Utility-Based Data Mining Workshop, 2005.
[8] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, ”Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases”,IEEE Trans. Knowledge and Data Eng., vol. 21, no. 12, pp. 1708-1721,Dec. 2009.
[9] M.J. Zaki, ”Scalable Algorithms for Association Mining”, IEEE Trans.Knowledge and Data Eng., vol. 12, no. 3, pp. 372-390, May 2000.
[10] S.J. Ye n and Y.S. Lee, ”Mining High Utility Quantitative Association “, sep 2007

 