Extraction of Highly Utilized Itemsets from large Transactional Databases

K.Madhavi
Department of Computer Science and Technology
VR Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India

CH.Nanda Krishna
Department of Computer Science and Technology
VR Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India

Sai Harshita.Gutta
Department of Computer Science and Technology
VR Siddhartha Engineering College, Vijayawada, Andhra Pradesh, India

Abstract- The main aim of this project is to generate the high utility (profitable) itemsets in a parallel environment, by considering the quantity of each item. The parallel mining of high utility itemsets will take very less time than mining with the single system over large number of transactions. A Utility-pattern tree(UP_tree) data structure (Memory-Resident) is used to store the information about the transactions. By UP_Growth algorithm the candidate itemsets are generated by scanning the database only twice. The above Utility-Pattern tree is duplicated at every node in the parallel system and at each node the conditional pattern base tree is constructed for the assigned high utility itemsets, from this tree mined the high utility itemsets according to the given minimum utility threshold. The performance of UP_Growth algorithm is observed by applying on the dataset.

Keywords: Utility Mining, Quantitative Database.

I. INTRODUCTION

Association rules mining (ARM) [1] is one of the most widely used techniques in data mining and knowledge discovery and has many applications like business, medicine and other domains. Make the decisions about marketing activities such as, e.g., in supermarkets.

Data mining is the extraction of potentially useful information from the available data. Discovery of the knowledge is the data mining goal, which is used to predict the feature behavior for taking the structural decisions. Mining the frequent patterns is a former research topic in data mining. Frequent pattern mining is the finding of interesting patterns hidden in a database. Frequent pattern mining doesn't consider the profit for each item. To overcome this problem weighted frequent pattern mining [2] is proposed. Where as in weighted frequent pattern mining doesn't consider the quantity of each item. To overcome this problem utility mining is proposed. Utility mining consider the both profit and quantity of each item. Utility mining supports the quantitative database.

Utility mining finds the most valuable frequent itemsets. The Utility of an item refers to the users interestingness for item. The multiple of itemsets external and internal utilities defines it's utility. External utility is the profit of the item, internal utility is quantity of the item present in the transaction. For the given utility threshold if the determined itemset utility is greater than the given utility threshold then it is called promising itemset else it is called unpromising itemset. The process of mining high utility itemsets requires two inputs first one is transactional database and second one is profit for each item.

II. BACKGROUND

Definition 1: Itemset X utility in

D DataBase is represented by

U(X) = $\sum_{X \subseteq T_d \wedge T_d \in D} u(X, T_d).$

Definition 2: For the D database the itemset X Transaction weighted utility is

TWU(X)=$\sum_{X \subseteq T_d \wedge T_d \in D} \mathrm{TU}(T_d).$

When mining the high utility itmsets from very large transactions takes much time, this mining time can be reduced by implementing in a parallel environment.

III. RELATED WORK

For mining the frequent itemsets basically Apriori Algorithm [1] is used. The disadvantage of this algorithm is it requires too many scans of the entire database and it generates test the each candidate itemset knowing for promising or unpromising itemset. To overcome this Frequent pattern growth (FP_Growth)[3] algorithm is proposed. High utility itemsets are mined by Up_Growth algorithm and it is a tree-based approach as same as FP-Growth[3] algorithm. Up_Growth algorithm requries only two scans of database[4] for creating the Up_tree from this the large utility itemsets are mined. For mining the interesting itmesets first Up_tree is created then the conditional pattern base (CPB) trees are derived for each high utility item assigned to the corresponding each parallel node. From each parallel node the corresponding high utility items are mined by using the Up_Growth algorithm.

For constricting the Up_Tree the node structure is

```
Class N
{
    String name;
    int count;
    int utility;
    String parent;
    Class N hlink;
}
```

The link pointer is used to represent the last appearance of the node in UP_Tree which has same node name in the entry header table.

3.1 Steps for Construction of UP_Tree:

Database is scanned only twice for Construction of UP_Tree

First scan

For each transaction the transaction utility (Tu) is computed.

At the same time the transaction weighted utility(TWU) of each item is also computed.

The items with TWU's less than the given threshold then those items are called unpromising items and those items are discarded.

For the promissing items compute the retrasaction utilities.

The promising items are arranged in the decreasing order of their TWU's.

Second scan

Each transaction is inserted as a branch into a UP_Tree.

After construction of UP_Tree the following strategies are used to reduce the time and the search space. The strategies are as follows

3.2 Strategy 1: Discarding global unpromising items (DGU).

Since the unpromising items does not involved in the high utility itemsets, these are not consider in the creation of Global UP_tree.

Strategy 2: Discarding global node utilities (DGN).

In the construction of a Global Utility pattern tree for a node the successor utilities are deleted from the current utility of the node.

Strategy 3: Discarding local unpromising items (DLU).

In the construction of a local Utility pattern tree the unpromising items minimum utilities are dropped from path utilities of paths.

Strategy 4: Decrease the local node utilities (DLN).

For construction of a local UP_tree for any node the item's minimum utilities of descendant's are decreased.

The pseudo code for UP-Growth is as follows:

Method: U-PGrow(Ax, Bx, Z)

Input: Ax as UP_Tree, Bx header table for Ax and Z itemset.

Output: PHUI's in given tree.

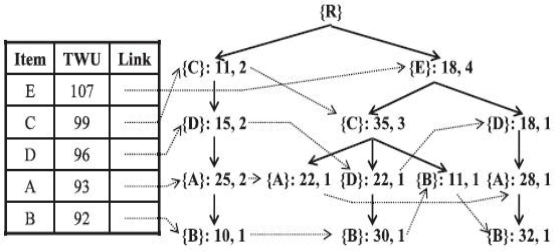
Method: U-PGrow (Ax, Bx, Z)

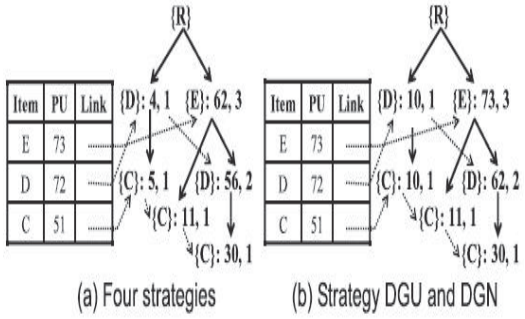
1. For every ci in Bx do
2. Create PHUI P = Z ∪
3. Assign P's utility= utility of ci's in Bx.
4. Create P's CPB;
5. Place local items which are having greater min_util in P-CPB into By.
6. Implement DLU.
7. Implement DLN, add branches to Ay;
8. If Ay! = { } call U-PGrow (Ay, By, P);
9. End of for.

Utility Pattern Tree is used for maintaining the information about the transactions. From the UP_Tree the PHUI's are generated. UP_Tree will be replicated at each parallel node and the corresponding the CPB (Conditional Pattern Base) trees are generated at each node for all the items in the header table. For each items CPB's the high utility itemssets are mined.

The UP_Tree constructed as follows:

Figure 1: The UP_Tree for table 1 and conditional tree





IV. PROPOSED SYSTEM

In the proposed system the mining of highly utilized itemsets will done in parallel. Here the construction of Up_tree[5] will done at server and the highly utilized itemsets are mined at each parallel node from its conditional pattern base tree's.The structure of the proposed strategies includes three phases: 1) Scan the database twice to develop a worldwide Tree with the underlying two techniques 2) recursively create PHUIs from worldwide UP-Tree[6] and close-by UP-Trees by UP-Growth with the third and fourth philosophies or by UP-Growth+ with the last two frameworks and 3) recognize real high utility itemsets from the plan of PHUIs.

4.1 The Proposed Data Structure: UP-Tree

To facilitate the mining performance and avoid scanning original database repeatedly, we use a compact tree structure, named UP-Tree, to maintain the information of transactions and high utility itemsets. Two strategies are applied to minimize the overestimated utilities stored in the nodes of global UP-Tree.

4.1.1 The Proposed Mining Method: UP-Growth

After constructing a global UP-Tree, a general method for generating PotentialyHUIs is to mine UP-Tree by FP-Growth [5]. So many candidate itemsets will be generated. Thus, we use another algorithm UP-Growth by implementing two more strategies into the framework of FP-Growth. By the strategies, overestimated utilities of itemsets can be decreased and thus the number of PHUIs can be further reduced.

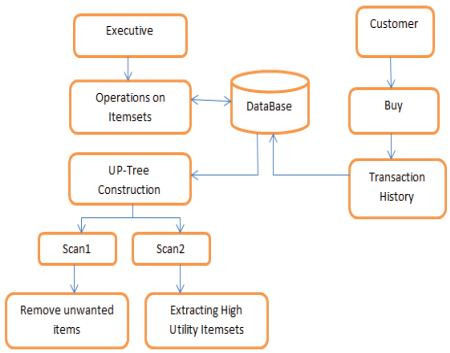


Figure2: Architecture of UP-Growth

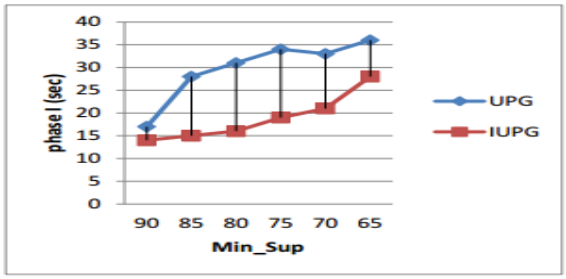
4.2 An Improved Mining Method: UP-Growth+

UP-Growth accomplishes less time for execution over FP-Growth by utilizing DLU and DLN to diminish overestimated utilities of itemsets. In any case, the overestimated utilities can be nearer to their genuine utilities by disposing of the assessed utilities that are nearer to real utilities of unpromising things and relative hubs. In this paper, we propose another important strategy, named UP-Growth+, for lessening over-evaluated utilities all the more successfully.

In UP-Growth, least thing utility table is utilized to decrease the overestimated utilities. In UP-Growth+, insignificant hub utilities in every way are utilized to make the evaluated pruning values nearer to genuine utility estimations of the pruned things in database

V. EXPERIMENTAL EVOLUTIONS

The performance of mining high utility itemsets is rapidly increased in the parallel environment by minimizing the memory space and the generation of unwanted candidate itemsets. The new methods will give a batter performance for the large Databases and for the low utility thresholds. By the below experimental results, the Improved UP-Growth is efficiently reducing the execution time. Here compare the performance of UP_Growth and UP_Growth plus on datasets[6]. Table shows the execution times on various min_sup values from 65% to 90%.



VI. CONCLUSION

This paper proposes a parallel system which takes very less turnaround time for mining large utility itemsets.Converting the transactions into UP_tree two scans are sufficient. Conditional pattern based tress for each high utility item set is replicated at each parallel node. From all the parallel nodes we get the high utility itemsets. With this we can also create a large synthetic datasets.

REFERENCES

[1] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W.Kwong, "Mining Association Rules with Weighted Items".
[2] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K.Lee, "Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases".
[3] Agrawal and Srikant, "Fast Algorithms for Mining Association Rules".
[4] "A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets" Ying Liu, Wei-keng Liao, and Alok Choudhary
[5] Vincent. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu.: UP-Growth: An Efficient Algorithm for High UtilityItemsetMining.
[6] Frequent itemset mining implementations repository, http://fimi.cs.helsinki.fi/
[7] Y. C. Li, J. S. Yeh, and C. C. Chang.: Isolated items discarding strategy for discovering high utility itemsets. *In Data & Knowledge Engineering, Vol. 64, Issue 1, pp. 198-217, Jan., 2008.*
[8] R. Chan, Q. Yang, and Y. Shen.: Mining high utility itemsets. *In Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26, 2003.*
[9] Jiawei. Han, Jian. Pei, and Y. Yin.: Mining frequent patterns without candidate generation. *In Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12, 2000.*
[10] H. Yao, H. J. Hamilton, and L. Geng.: A unified framework for utility-based measures for mining itemsets. *In Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining, pp. 28-37, USA*
[11] S. J. Yen and Y. S. Lee.: Mining high utility quantitative association rules. In *Proc. of 9th Int'l Conf. on Data Warehousing and Knowledge Discovery, Lecture Notes in Computer Science 4654*, pp. 283-292, Sep., 2007