International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 12 (2014), pp. 1137-1143
© International Research Publications House
http://www. irphouse.com

A Scalable Algorithm Using Up-Growth for Mining High Utility Itemsets

C. Mamatha Devi

*Sree Vidyanikethan Engineering College,
Tirupati, Andhra Pradesh*

ABSTRACT

Now-a-days Patterns hidden in the databases are discovered efficiently in several data mining tasks some among them frequent pattern mining and high utility pattern mining. The number of useful algorithms has been proposed in present years, there is a problem of producing a large number of candidate itemsets for high utility itemsets. Such large number of candidate item sets degrades the mining performance in terms of execution time and space requirements. But this situation in the when the database contains lots of long transaction this situation may become worse. Mining high utility itemsets by cropping candidates based on the estimated utility values, and based on the transaction weighted utilization values. In this paper propose a method for Up-Growth from transactional databases.

The information of high utility itemsets is maintained in a tree based data structure named up-tree such that candidate itemsets can be generated efficiently with only two scans of database. first propose the strategies are discarding global unpromising items during constructing a global up-tree is quite effective especially when the transaction contain lots of unpromising items, such as those in sparse data sets. our second proposed strategy for decreasing global node utilities during overestimated utilities is to remove the utilities of descendant nodes from their node utilities in global up-tree the main goal of this project UP-tree based pattern mining utilizes the pattern growth method to avoid the costly generation of a large number of candidate sets and reduces the search space dramatically.

Keywords Frequent itemset, high utility itemset, utility mining, downword closure property, Datamining.

1. Introduction

Frequent patterns are itemsets, subsequences, or substructures that appear in a data set with frequency no less than a user-defined threshold. For example, a set of items, such as milk and bread, that appear repeatedly combined in a transaction data set, is a frequent itemset. Sequences that contain another sequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs repeatedly in a graph database, it is called a (frequent) structural pattern. Finding frequent patterns plays an essential role in mining associations, core relations, and many other interesting relationships among data. Frequent pattern mining for market basket analysis in the form of association rule mining. Market basket analysis might tell a retailer that customers often purchase shampoo and conditioner with each other, so putting both items on promotion at the same time would not create a significant increase in profit, while a promotion involving just one of the items would likely drive sales of the other

Apriori-based approach*:*

Apriori-based frequent substructure mining algorithms share near characteristics with Apriori-based frequent itemset mining algorithms. the search for frequent graphs starts with graphs of small "size", and proceeds in a bottom-up manner. at each iteration, the size of newly finding frequent substructures is increased by one. The AGM algorithm uses a vertex-based candidate generation method that increases the substructure size by one vertex at each iteration. Two size-*k* frequent graphs are joined only when the two graphs have the same size-*(k* 1*)*

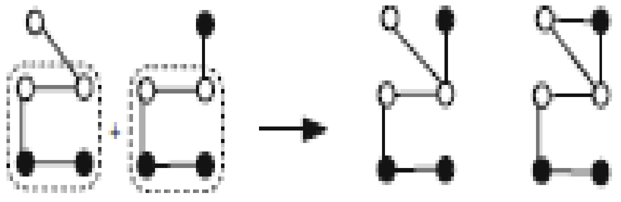


Fig. 1 Two substructures joined by two chains

II. UTILITY MINING

Mining high utility itemsets from Transactional databases refers to finding the itemsets with high profit utility of an items in transactional databases Consists of two aspects.

External utility Importance of different items is called external utility or unit profit value. Internal utility

Importance of items in transactions is called internal utility or support count.

III. In this paper drawback of IHUP algorithm from transactional databases IHUP algorithm produce too many HTWUI since the overestimated utility calculated by TWU is too long. Such a number of HTWUIs will reduce the mining performance in substantially in terms of execution time and memory consumption and also if requires too many database scans for discovering the high utility itemsets. The problem of mining high utility itemsets from D is to find the complete set of the itemsets whose utilities are greater than or equal to min_utility. in this paper propose a facilitate the mining performance and avoid scanning original database repeatedly, using a compact tree structure, named UP-tree. To maintain the information of transactions and high utility itemsets, four strategies are applied to minimize the overestimated utilities stored in the nodes of global UP-Tree. UP-Growth algorithm consists of three steps:1. Scan the database two times to construct a global UP-Tree with the strategy DGU and DGN.

Recursively generates potential high utility itemsets from global and local UP-Tree by UP-Growth with the strategy DLU and DLN.

Identify actual high utility itemsets from the set of PHUIs

There are several modules involved in finding high utility item sets from transactional databases. They are as follows

1) Data Collection
2) Calculation of TU and TWU Value
3) Construction of RTU and UP-Tree.
4) Construction of CPB for each item in Reorganized Transaction table and finding high utility item set

3. 1 DATA COLLECTION

The data has been collected A FAST ALGORITHM FOR MINING HIGH UTILITY ITEM SETS.

$u(\{A\}, T_1)=5*1=5$ item $u(i_p, T_d) = q(i_p, T_d) \times p(i_p)$ Table 1:Transactional database

TID	***Transaction***	***TU***
T1	(A, 1)(C, 1)(D, 1)	8
T2	(A, 2)(C, 6)(E, 2)(G, 5)	27
T3	(A, 1)(B, 2)(C, 1)(D, 6)(E, 1)(F, 5)	30
T4	(B, 4)(C, 3)(D, 3)(E, 1)	20
T5	(B, 2)(C, 2)(E, 1)(G, 2)	11

Table 2. Profit table

Item	A	B	C	D	E	F	G
Profit	5	2	1	2	3	1	1

4. 2 CALCULATION OFTU AND TWU VALUE:

$TWU(\{AD\})=TU(T_1)+TU(T_3)=8+30=38$

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

Item	A	B	C	D	E	F	G
TWU	65	61	96	58	88	30	38

CONSTRUCTION OF RTU AND UP-TREE

Two strategies are used for decreasing the overestimated utility of each item during the construction of Reorganized Transaction Table and a global UP-Tree.

Strategy 1: DGU

Discarding Global Unpromising items and their actual utilities from transactions and transaction utilities of the database. Any ordering can be used such as the lexicographic, support or TWU order. Each new TU after cropping anti cropping items is called reorganized transaction utility denoted as RTU.

Reorganized Transaction table is created by using this new TU value and the items. that the subroutine of Insert_Reorganized_Transaction is given below. Reorganized Transaction Table is shown below

Table 2. 1:Reorganized Transactions and their RTUs

TID	Reorganizationtransaction	RTU
T1’	(C, 1)(A, 1)(D, 1)	8
T2’	(C, 6)(E, 2)(A, 2)	22
T3’	(C, 1)(E, 1)(A, 1)(B, 2)(D, 6)	25
T4’	(C, 3)(E, 1)(B, 4)(D, 3)	20
T5’	(C, 2)(E, 1)(B, 2)	9

Strategy 2: DGN

Decreasing Global Node utilities for the nodes of global UP-Tree by actual utilities of descendant nodes during the construction of global UP-Tree. By applying strategy DGN, the utilities of the nodes that are closer to the root of a global UP-Tree are further reduced. Its subroutine is given below

UP-Tree is constructed byTwo steps:

1. Finding cropping items
2. Constructing UP-Tree

1. Finding cropping items

Cropping items are found by comparing TWU values of each items to the minimum utility threshold value. After sorting the cropping items in the descending order header table was constructed. A table named header table is employed to facilitate the traversal of UP-Tree. In header table, each entry records an item name, an overestimated utility, and a link.

2. Creating UP-Tree

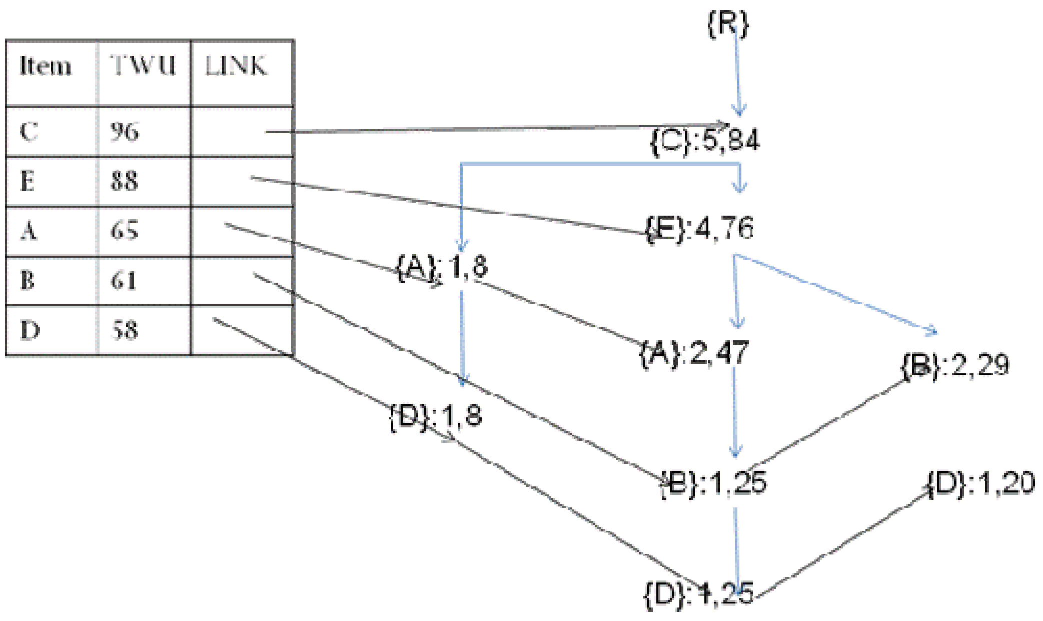
UP-Tree is using reorganized transactions with header table. An algorithm for creating UP-Tree is given below the construction of UP-Tree can be performed with two scans of the original data base. steps

First scan

1. TU of each transaction is computed.
2. TWU of each single item is also gathering.
3. Discarding global unpromising items.
4. Unpromising items are removed from the transaction and utilities are eliminated from the TU of the transaction.
5. The remaining promising items in the transaction are sorted in the descending order of TWU.

Second scan

Transactions are inserted into UP-Tree

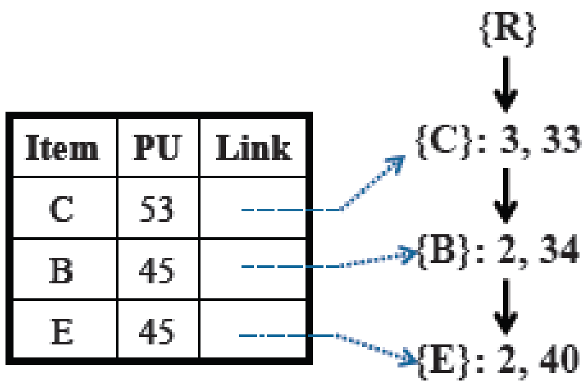


Item	TWU	LINK
C	96	
E	88	
A	65	
B	61	
D	58	

Figuare 3:UP TREE applying by DGU DNN Strategies

Conditional tree for item B by using CPB-b after applying DLN is show

Path	Reorganized path	supportcount	Path by DGU, DGN, DLU
{AC}	{C}	1	8
{BAEC}	{CBE}	1	25
{BEC}	{CBE}	1	20



(b) Four strategies

V. CONCLUSION

This report proposed tree-based algorithm, called UP-Growth, for scalable mining high utility itemsets from databases. It developed four effective strategies, DGU, DGN, DLU and DLN, to reduce search space and the number of candidates for utility mining. High utility itemsets can be identified by scaning reorganized transaction. since there is no unpromising item in the reorganized transactions, I/O cost and execution time can be further reduced. Experiments show that our UP-Growth outperforms the state-of-the-art algorithm substantially and has a good scalability for large database. In particular, our UP-Growth is over 10, 000 times faster than existing algorithms when database contains lots of long transactions

References

[1] C. F. Ahmed, S. K. Tanbeer, B. -S. Jeong and Y. -K. Lee(2009), “Efficient tree structures for high utility pattern mining in incremental databases, ” IEEE Transactions on Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708-1721.

[2] R. Chan, Q. Yang and Y. Shen(2003), “Mining high utility itemsets, ” in Proc. of Third IEEE Int'l Conf. on Data Mining, pp. 19-26.

[3] C. H. Cai, A. W. C. Fu, C. H. Cheng and W. W. K(1998), “Mining Association Rules with Weighted Items, ” in Proc. of the Int’l Database Engineering and Applications Symposium (IDEAS1998), pp.

[4] A. Erwin, R. P. Gopalan and N. R. Achuthan(2008), “Efficient mining of high utility itemsets from large datasets, ” in Proc. of PAKDD, LNA 5012, pp. 554-561.

[5] J. Han, J. Pei, Y. Yin(2000), “Mining frequent patterns without candidate generation, ”inProc. of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12.

[6] Y. Liu, W. Liao and A. Choudhary (2005), “A fast high utility itemsets mining algorithm, ” in Proc. of the Utility-Based Data Mining Workshop.

[7] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu(2012), “ Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases, ” IEEE Transactions on Knowledge and Data Engineering.