ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

Efficient Algorithm for Finding High Utility Itemsets from Large Transactional Databases Using R-Hashing Technique

D.Sathyavani, Prof. D.Sharmila

Department of IT, Maharaja Engineering College, Avinashi, India

Head, Dept. of EIE, Bannari Amman Institute of Technology, Sathy, India

ABSTRACT: Association rule mining is used to find the frequent item sets in large database. In the data mining field, utility mining emerges as an important topic that to mine the high utility itemsets from databases which refers to finding the itemsets with high profits. The huge number of high utility itemsets makes a challenging problem for the mining performance, due to generating more potential high utility item sets. So it consumes higher process in large database and decreases the mining efficiency. In existing system they have proposed two novel methods such as UP-growth and UP-growth$^+$ as well as a compact UP Tree data structure, which is used for efficiently discovering high utility itemsets from large transactional databases. In existing system random memory allocation is used to store the candidate that leads to high potential I/O operations. Also this approach is time consuming and requires high memory space. In order to solve this problem in proposed system, we used sorting with R-hashing technique for the memory allocation. Hence the candidate items are stored with their respective memory in UP tree. The experimental result shows that the proposed system is more effective than the existing methods according to the memory space and the number of candidate itemset generation and input and output operations.

KEYWORDS: candidate sets, frequent Item set, high utility itemset, utility mining, data mining

I. INTRODUCTION

Data mining is a technique to extracting the data's from the large database. Association rule mining find out the large transaction databases for association rules which provide the implicit relationship among data attributes. Data mining also known as knowledge discovery in databases. The most challenging data mining tasks is the mining of high utility itemsets efficiently. Identifying the itemsets with high utilities is called as Utility Mining. The utility can be measured based upon cost, profit or other expressions of user preferences. The frequent itemset is not sufficient to reflect the actual utility of an itemset. Frequent itemsets are the itemsets that occur frequently in the transaction data set. The main goal of frequent Itemset Mining is to identify the frequent itemsets in a transaction dataset. Utility mining gives an essential topic in the data mining field. Mining high utility itemsets from databases refers to finding the itemsets with high profits. The meaning of itemset utility is interestingness, importance, or profitability of an item to users. Utility of an itemset is defined as the product of its external utility and its internal utility[1].

II. BACKGROUND

The traditional association rule mining (ARM) model assumes that items have the same implication without considering their weight/attributes/characters within a transaction or within the whole frequent itemset mining. Many interesting and efficient algorithms have been proposed for mining association rules for Boolean attributes. The importance of items to users is not considered the concept of weighted items and weighted association rules. The concept of weighted association rules does not have downward closure property. During association rule mining process the mining performance cannot be improved transaction weight, and weighted support cannot only reveal the importance of an itemset. But maintains the downward closure property.

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

In existing system they have proposed two novel algorithms as well as a compact data structure for efficiently discovering high utility itemsets from transactional databases. That two algorithms, named as UP-Growth (Utility Pattern Growth) and UP-Growth+, and a compact tree structure, called UP-Tree (Utility Pattern Tree), for discovering high utility itemsets and maintaining important information related to utility patterns within databases. But in existing system requires high memory to store the candidate items. Also consumes more time while scanning the candidate item in UP-Tree. Even it maintains the essential information by both algorithm it takes high potential operations. These problems can be overcome by proposed technique. R-Hashing technique is used to allocate the memory for candidate itemsets.

High utility itemsets can be generated from UP-Tree efficiently with only two scans of original databases. Several strategies are proposed for facilitating the mining processes of UP-Growth and UP-Growth+ by maintaining only essential information in UP-Tree. By these strategies, overestimated utilities of candidates can be well reduced by discarding utilities of the items that cannot be high utility or are not involved in the search space. The proposed strategies can not only decrease the overestimated utilities of potential high utility itemsets but also greatly reduce the number of candidates. Different types of both real and synthetic datasets are used in a series of experiments to compare the performance of the proposed algorithms with the state-of the-art utility mining algorithms.

III.RELATED WORKS

A frequent itemset is a set of items that appears at least in a pre-specified number of transactions. Formally, let I = {i1, i2. . . im} be a set of items and DB = {T1, T2... Tn} a set of transactions where every transaction is also a set of items (i.e. itemset).
Given a minimum support threshold minSup an itemset. Frequent itemset mining is the first and the most time consuming step of mining association rules. During the search for frequent itemsets the anti-monotone property is used.

3.1 Problem Statement

Pruning search space for high utility itemset mining is difficult because of random memory allocation in an UP - Tree. With this reason the time taken to scan the candidate itemset UP-Tree is high and also it requires high potential I/O operations. So we are in need of an effective algorithm to overcome these problems.

3.2 High Utility Itemsets

A high-utility itemset mining model was defined by Yao, Hamilton and Butz [13]. It is a generalization of the share-mining model [3, 4]. The goal of high utility itemset mining process is to find all itemsets that give utility greater or equal to the user specified threshold. The following is the set of definitions given in [13] which we shall illustrate on a small example.

Definition 1: The external utility of an item ip is a numerical value yp defined by the user. It is transaction independent and reflects importance (usually profit) of the item. External utilities are stored in a utility table. For example, external utility of item B in Table 2 is 10.
Definition 2: The internal utility of an item ip is a numerical value xp which is transaction dependent. In most cases it is defined as the quantity of an item in transaction. For example, internal utility of item E in transaction T5 is 2 (see Table 1).

Table 1. Transactions databases

TID	A	B	C	D	E
1	0	0	18	0	1
2	0	6	0	1	1
3	2	0	1	0	1
4	1	0	0	1	1
5	0	0	4	0	2

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

6	1	1	0	0	0
7	0	10	0	1	1
8	3	0	25	3	1
9	1	1	0	0	0
10	0	6	2	0	2

In an example Table 2 shows that the profit of Utility mining given below,

Table 2. External utilities of items from database in Table 1.

Item	A	B	C	D	E
Profit(€)	3	10	1	6	5

Definition 3: Utility function f is a function of two variables:
$f(x, y) : (R^+,R^+) \rightarrow R^+$. The most common form also used in this paper is the product of internal and external utility: xp * yp.
Definition 4: The utility of item ip in transaction T is the quantitative measure computed with utility function from Definition 3u (ip, T) = f(xp, yp), ip ε T.
For example: utility of item E in transaction T5 is 2 * 5 = 10.

3.3 Fast Algorithm for Mining Utility-Frequent Itemsets

In the fast mining algorithm, 2P-UF utility-frequent itemset mining algorithm described in Section 2 is prove to find all utility-frequent itemsets. However, due to the monotone property of quasi support measure it has a few disadvantages which render it unusable for mining of large datasets. The first weak point is the reversed way of candidate generation. 2P-UF algorithm wastes time checking long itemsets that are highly unusual to be utility-frequent. For example, when mining a database with 1000 distict items (attributes) 2P-UF algorithm first generates and checks all itemsets of length 999, then itemsets of length 998 etc. Short itemsets which have fairly large probability to be utility-frequent come at the very end.
Candidate generation function is also slow and inefficient as it computes intersection of every pair of candidates in each iteration. Moreover, computation of quasi support measure is also inefficient because special data structures (hash trees) cannot be used and we have to scan database once for every candidate.
Finally, the two-phase form of the algorithm is space consuming since we have to store all quasi utility-frequent candidates from the first phase to filter them in the second phase. It is possible to avoid this waste of space by merging both phases

and filter non utility-frequent candidates in every iteration of the algorithm.

IV.PROPOSED METHOD

4.1 Discarding Global Unpromising Items during Constructing a Global UP-Tree

The construction of a global UP-Tree can be performed with two scans of the original database. In the first scan, TU of each transaction is computed. At the same time, TWU of each single item is also accumulated. By TWDC property, an item and its supersets are unpromising to be high utility item sets if its TWU is less than the minimum utility threshold. Such an item is called an unpromising item. An item ip is called a promising item if TWU (i_p) |min_util|. Otherwise it is called an unpromising item. Without loss of generality, an item is also called a promising item if its overestimated utility (which is different from TWU in this paper) is no less than min_util. Otherwise it is

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

called an unpromising item. New TU after pruning unpromising items is called reorganized transaction utility (abbreviated as RTU). RTU of a reorganized transaction T_r is denoted as RTU (T_r). By reorganizing the transactions, not only less information is needed to be recorded in UP-Tree, but also smaller overestimated utilities for item sets are generated. Strategy DGU uses RTU to overestimate the utilities of item sets instead of TWU. Since the utilities of unpromising items are excluded, RTU must be no larger than TWU. Therefore, the number of PHUIs with DGU must be no more than that of HTWUIs generated with TWU. DGU is quite effective especially when transactions contain lots of unpromising items, such as those in sparse datasets. Besides, DGU can be easily integrated into TWU-based algorithms. Moreover, before constructing a UP-Tree, DGU can be performed repeatedly till all reorganized transactions contain no global unpromising item. By performing DGU for several times, the number of PHUIs will be reduced; however, it needs several database scans.

4.2 Decreasing Global Node Utilities during Constructing a Global UP-Tree

It is shown that the tree-based framework for high utility item set mining applies the divide-and-conquer technique in mining processes. Thus, the search space can be divided into smaller subspaces. By applying strategy DGN, the utilities of the nodes that are closer to the root of a global UP-Tree are further reduced. DGN is especially suitable for the databases containing lots of long transactions. In other words, the more items a transaction contains, the more utilities can be discarded by DGN. On the contrary, traditional TWU mining model is not suitable for such databases since the more items a transaction contains, the higher TWU is. In following subsections, we describe the process of constructing a global UP-Tree with strategies DGU and DGN.

4.3 Constructing a global UP-Tree by Applying DGU and DGN

Recall that the construction of a global UP-Tree is performed with two database scans. In the first scan, each transaction's TU is computed; at the same time, each 1- item's TWU is also accumulated. Thus we can get promising items and unpromising items. After getting all promising items, DGU is applied. The transactions are reorganized by pruning the unpromising items and sorting the remaining promising items in a fixed order. Any ordering can be used such as the lexicographic, support or TWU order. Each transaction after the above reorganization is called a reorganized transaction. In the following paragraphs, we use the TWU descending order to explain the whole process since it is mentioned that the performance of this order is the best in previous study. Then a function Insert_Reorganized_Transaction is called to apply DGN during constructing a global UP-Tree.

4.4 UP-GROWTH

After constructing a global UP-Tree, a basic method for generating PHUIs is to mine UP-Tree by FP-Growth. However too many candidates will be generated. Thus, we propose an algorithm UP-Growth (Utility Pattern Growth) by pushing two more strategies into the framework of FP-Growth. By the strategies, overestimated utilities of item sets can be decreased and thus the number of PHUIs can be further reduced. In following subsections, we first propose the two strategies and then describe the process of UP-Growth.

4.5 Discarding Local Unpromising Items during Constructing a Local UP-Tree

The common method for generating patterns in tree based algorithms contains three steps: (1) Generate conditional pattern bases by tracing the paths in the original tree, (2) construct conditional trees (also called local trees in this paper) by the information in conditional pattern bases and (3) mine patterns from the conditional trees. However, strategies DGU and DGN cannot be applied into conditional UP-Trees since actual utilities of items in different transactions are not maintained in a global UP-Tree. We cannot know actual utilities of unpromising items that need to be discarded in conditional pattern bases unless an additional database scan is performed. To overcome this problem, a naïve solution is to maintain items' actual utilities in each transaction into each node of global UP-Tree. However, this is impractical since it needs lots of memory space. In view of this, we propose two strategies, named DLU and DLN, that are applied in the first two mining steps and introduced in this and next subsections, respectively. For the two strategies, we maintain a minimum item utility table to keep minimum item utilities for all global promising items in the database.

4.6 Decreasing Local Node Utilities during Constructing a Local UP-Tree

Since $\{i_m\}$-Tree must not contain the information about the items below i_m in the original UP-Tree, we can discard the utilities of descendant nodes related to i_m in the original UP-Tree while building $\{i_m\}$-Tree. (Here, original UP-Tree

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

means the UP-Tree which is used to generate $\{i_m\}$-Tree.) Because we cannot know actual utilities of the descendant nodes, we use minimum item utilities to estimate the discarded utilities. Path utility of item i_k in $\{i_m\}$-CPB is denoted as $pu(i_k,\{i_m\}\text{-CPB})$.

The same as DLU, DLN can be recognized as local version of DGN. By the two strategies, overestimated utilities for item sets can be locally reduced in a certain degree without losing any actual high utility item set. The whole process of UP-Growth with DLU and DLN will be addressed in detail in the next subsection.

4.7 UP-Growth: Mining a UP-Tree by Applying DLU and DLN

The process of mining PHUIs by UP-Growth is described as follows. First, the node links in UP-Tree corresponding to the item i_m, which is the bottom entry in header table, are traced. Found nodes are traced to root of the UP-Tree to get paths related to i_m. All retrieved paths, their path utilities and support counts are collected into i_m's conditional pattern base. A conditional UP-Tree can be constructed by two scans of a conditional pattern base. For the first scan, local promising and unpromising items are learned by summing the path utility for each item in the conditional pattern base. Then, DLU is applied to reduce overestimated utilities during the second scan of the conditional pattern base. When a path is retrieved, unpromising items and their estimated utilities are eliminated from the path and its path utility. Then the path is reorganized by the descending order of path utility of the items in the conditional pattern base.

4.8 An Improved Mining Method: UP-Growth+

UP-Growth achieves better performance than FP-Growth by using DLU and DLN to decrease overestimated utilities of item sets. However, the overestimated utilities can be closer to their actual utilities by eliminating the estimated utilities that are closer to actual utilities of unpromising items and descendant nodes. In this subsection, we propose an improved method, named UP-Growth+, for reducing overestimated utilities more effectively. In UP-Growth, minimum item utility table is used to reduce the overestimated utilities. In UP-Growth+, minimal node utilities in each path are used to make the estimated pruning values closer to real utility values of the pruned items in database. Minimal node utility for each node can be acquired during the construction of a global UP-Tree. First, we add an element, namely N.mnu, into each node of UP-Tree. N.mnu is minimal node utility of N. When N is traced, N.mnu keeps track of the minimal value of N.name's utility in different transactions. If N.mnu is larger than u (N.name, $T_{current}$), N.mnu is set to u(N.name, $T_{current}$). After introducing the modification of global UP-Tree, now we address the processes and two improved strategies of UP-Growth+, named DNU and DNN. When a local UP-Tree is being constructed, minimal node utilities can also be acquired by the same steps of global UP-Tree. In the mining process, when a path is retrieved, minimal node utility of each node in the path is also retrieved.

4.9 Enhanced UP Growth ++

In Enhanced UP-Growth$^+$, node utility count in each path are used to reduce the overestimated utilities that are closer to their actual utilities values of the unpromising items and descendant nodes. Node utility count for each path can be acquired during the construction of a local IMUP-Tree. First, the node links in IMUP-Tree corresponding to the item i_m, which is the bottom entry in header table, are traced. Found nodes are traced to root of the IMUP-Tree to get paths related to i_m. All retrieved paths, their minimum node utility and support count are collected into i_m's conditional pattern base. The two strategies are introduced to enhance the UP-Growth$^+$ named DPU and DPN. When a local IMUP-Tree is being constructed, minimal node utilities is retrieved from the global IMUP-Tree. In the mining process, when a path is retrieved, node utility count of the path is acquired.

The complete set of PHUIs is generated by recursively calling the procedure named UP-Growth. Initially, UPGrowth (TR, HR, null) is called, where TR is the global UP-Tree and HR is the global header table.

4.10 R-Hashing

The framework of the proposed methods consists of two steps. First one is we are using the sorting technique to sort the candidate items in descending order. Second is we are using the R-hashing technique to allocate the memory for candidate itemset.

The Enhanced IFP-Growth consists of three phases: In first phase, it scans the transactional database only once for generating equivalence classes of frequent items. In second phase, it consequently sorts the equivalence classes of

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

frequent items in descending order and filter out non-frequent items. Finally in third phase, the Enhanced IFP tree is constructed in order to extract the frequent itemsets.

Random Hashing algorithm which is a hash-based technique mines the potential high utility itemsets without any collision into the memory. The following are the steps which have been performed using RH algorithm.

V.PERFORMANCE EVALUATION

Performance about different sorting methods which were shows the results on Accidents dataset including following sorting methods on UPT&UPG+: lexicographical order (LEX), support descending order (SUP), TWU descending (TWU (D)) and ascending (TWU(A)) orders, and real utility descending (U(D)) and ascending (U(A)) orders. For U(D) and U(A), we use the order of actual utilities of 1-PHUIs after the first time database scan.

To show the performance of the proposed algorithms, we compared several compared methods and give them new notations as follows: IHUPTWU algorithm, which is proposed in [3] and composed of IHUPTWU-Tree and FPgrowth, is denoted as IHUPT&FPG. In the existing algorithms, there are two methods UPT&UPG and UPT&UPG+, that are composed with R-Hashing method.

To compare the performance of FP-Tree and UP-Tree, a method called UPT&FPG is also used with the proposed technique. It generates PHUIs from UP-Tree by FP-Growth directly, in other words, only DGU and DGN are applied. Since R-Hashing technique is used for following purposes. First, less memory required to store the candidate items. Second, less time consuming process while scanning the candidate item in the UP – Tree. Finally low potential I/O operations.

VI.CONCLUSION

In this paper, we have proposed an efficient algorithm R-Hashing technique for mining high utility itemsets from transaction databases. A data structure named UP-Tree is used for maintaining the information of high utility itemsets. though, the potential high utility itemsets can be efficiently generated from the UP-Tree with only two scans of the database, proposed method decreases the scanning process. In the experiments, both of synthetic and real datasets are used to evaluate the performance of our algorithm. The mining performance is improved significantly since both the search space and the number of candidates are effectively reduced by the proposed strategies. The experimental results show that R-Hashing structure performs the state-of-the-art algorithms significantly in large transactional database.

REFERENCES

[1] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases", *IEEE transaction on knowledge and data engineering*, Vol. 25, no.8, August 2013.

[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", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, Issue 12, pp. 1708-1721,2009.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. of the 20th Int'l Conf. on Very Large Data Bases*, pp. 487-499, 1994.

[4] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.*, pp. 3-14, Mar. 1995.

[5] 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 Engineering*, vol. 21, no. 12, pp. 1708-1721, Dec. 2009.

[6] Ke Sun and Fengshan Bai, "Mining Weighted Association Rules without Preassigned Weights", *IEEE transaction on knowledge and data engineering*, Vol.20, no.2.pp 489-495, 2008.

[7] R. Chan, Q. Yang, and Y. Shen, "Mining High Utility Itemsets," In *Proc. IEEE Third Int'l Conf. Data Mining*, pp. 19-26, Nov. 2003

[8] A. Erwin, R.P. Gopalan, and N.R. Achuthan, "Efficient Mining of High Utility Itemsets from Large Data Sets," *Proc. 12th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD)*, pp. 554-561, 2008.

[9] Mengchi Liu, Mengchi Liu, "**Mining** High Utility Itemsets without Candidate Generation", ACM, *CIKM'12*, October 29–November 2, 2012,

[10] H.F. Li, H.Y. Huang, Y.C. Chen, Y.J. Liu, and S.Y. Lee, "Fast and Memory Efficient Mining of High Utility Itemsets in Data Streams," *in Proc. of the 8th IEEE Int'l Conf. on Data Mining*, pp. 881-886, 2008.

[11] Praveen Arora, R.K.Chauhan, Ashwani Kush, "Frequent Itemsets from Multiple Datasets with Fuzzy data", *International Journal of Computer Theory and Engineering*, Vol.3, No. 2, 2011.

ISSN(Online):2320-9801
ISSN(Print): 2320- 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Special Issue 3, July 2014

[12] D.L. Olson, Yanhong Li, "Mining Fuzzy Weighted Association Rules", *Proceedings of the 40th Hawaii International Conference on System Sciences*, 2007.
[13] Sanobe Shaikh, Madhri Rao and S.S.Mantha, "A new Association Rule Mining Based on frequent Item set" , *David Bracewell*, AIAA 2011, CS & IT 03, pp.81-95, 2011.
[14] 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.
[15] V.S. Tseng, C.-W. Wu, B.-E. Shie, and P.S. Yu, "UP-Growth: An Efficient Algorithm for High Utility Itemsets Mining," *Proc. 16th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '10)*, pp. 253-262, 2010.
[16] Yao, H., Hamilton, H.J., Buzz, C.J, "A Foundational Approach to Mining Itemset Utilities from Databases", *In: 4th SIAM International Conference on Data Mining*, Florida USA, 2004.
[17] Yao, H., Hamilton, H.J, "Mining itemset utilities from transaction databases", *Data & Knowledge Engineering 59(3)*, 603–626 2006.
[18] Liu, Y., Liao, W.K., Choudhary, A, "A Fast High Utility Itemsets Mining Algorithm", In*: 1st Workshop on Utility-Based Data Mining*, Chicago Illinois, 2005.