The Solution associated with downward closure property was addressed in [6]. These solutions specifies use of transaction weight which reflects importance of itemset and also maintain downward closure property. Another algorithm named as Two-phase consist of two mining stages [7]. In phase one Apriori based level-wise method is used to generate High Transaction Weighted Utility Itemsets (HTWUIs). The second phase gives high utility itemsets. Two phase algorithm uses transaction weighted downward closure property (TWDC) property and thus reduces the search space, but still generates lot of candidate itemsets in phase one.

The Tree-Based algorithm called IHUP was soon introduced to overcome of large candidate itemset generation [8]. In tree-based algorithms a tree structure is used to maintain information about itemsets. Typical node structure consist of item name, Transaction weighted utility (TWU) value and support count. Algorithm works in three steps starting with first construction of IHUP-Tree. While construction of tree transactions are rearranged in descending order of TWU or Support count or lexicographic order. Reordering helps in limiting long tree traversals. In second step HTWUIs are generated by using FP-Growth technique [3]. Then in last step high utility itemsets are identified by applying additional database scan.Although Tree-based algorithm reduces generation of large number of candidate itemsets, this quantity can be further reduced.

Latest technique such as Discarding Global Unpromising items (DGU) and Decreasing Global node Utilities (DGN) further reduces intermediate candidate itemset generation [1]. In this paper will we see how DGU and DGN can be applied to efficiently mine high utility itemsets from transactional database.

In short traditional association rule mining such as Apriori technique treats all items in database equally by only considering item is present or not. It doesn't consider importance of item to user. Then later frequent itemset mining technique such as FP-Growth only considers frequency of occurrence of item without considering importance of item and thus contribute to small percentage of overall profit. Lastly weighted association rule mining itemset considers importance of itemset to user. There are many technique in weighted association rule mining such as DGU and DGN which try to reduce large on intermediate itemset generation.

Remainder of paper is as follows section III lists some basic definitions. Then section IV show working of Terms and Definitions. In section V we see two proposed strategies named DGU and DGN. Section VI goes through mining method used for mining frequent itemsets and last we have conclusion of topic in section VII.

3. Terms and Definitions

- *Data Mining*
  Data mining is a process of extracting useful information from large database.

- *Transactional Database*
  Database consisting of transactions, where write operations can be rolled back are called as transactional database.

- *Utility*
  Utility is defined as Interestingness, profitability or importance of item. There are two types of utilities one is internal utility and another external utility. Internal utility is importance of items in transaction, whereas External utility is importance of distinct individual item.

- *Utility of itemset*
  Utility of itemset is product of external and internal utility.
  $$u(i_p, T_d) = p(i_p) * q(i_p, T_d)$$

- *High utility itemset*
  Utility of itemset which is not less than user specified threshold.

- *Utility of itemset*
  Utility of itemset X in transaction $T_d$is sum of all utilities of items that belongs to itemset X and X is subset of T_d.
  $$u(X, T_d) = \sum_{i_p \in X \wedge X \subseteq T_d} u(i_p, T_d)$$

- *Utility of itemset in database*
  Utility of itemset X in database D is sum of all utilities of itemset X present in database D.
  $$u(X) = \sum_{X \subseteq T_d \wedge T_d \in D} u(i_p, T_d)$$

- *Utility of transaction*
  Utility of transaction is sum of all utilities of items in transaction.
  $$TU(T_d) = u(T_d, T_d)$$

- *Transaction weighted utility*
  Transaction weighted utility of itemset X is sum of all Transaction utilities TU that contains X.
  $$TWU(X) = \sum_{X \subseteq Td \wedge Td \in D} TU(T_d)$$

- *High transaction weighted utility*
  If TWU of itemset X is greater than user specified threshold then X is HTWUIs.

- *Transaction Weighted Downward Closure*
  If X is not HTWUIs then any Superset of X is a low utility itemset.

Example-

Table 1 : Transaction database

TID	Transaction	TU
T_1	(A,1) (C,10) (D,1)	17
T_2	(A,2) (C,6) (E,2) (G,5)	27
T_3	(A,2) (B,2) (D,6) (E,2) (F,1)	37
T_4	(B,4) (C,13) (D,3) (E,1)	30
T_5	(B,2) (C,4) (E,1) (G,2)	13
T_6	(A,1) (B,1) (C,1) (D,1) (H,2)	12

Table 2: Profit Table

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

From Table 1 and 2 we have,
$u(\{A\}, T_1) = 5 * 1 = 5$;
$u(\{AD\}, T_1) = u(\{A\}, T_1) + u(\{D\}, T_1) = 5 + 2 = 7$;
$u(\{AD\}) = u(\{AD\}, T_1) + u(\{AD\}, T_3) + u(\{AD\}, T_6)$
$= 7+22+7 = 36$;
$TWU(\{G\}) = TU(T_2) + TU(T_5) = 27 + 13 = 40$
If minimum utility is set to 30 then {G} is high transaction weighted utility itemset (HTWUI). If minimum utility is set to 50, then {G} and its superset are not HTWUI due to downward closure property.

4. Background Study: Working of IHUP-Tree

IHUP (Incremental high utility pattern) was proposed by Ahmed et al. [8] to overcome problem of scanning database too many times to generate HTWUIs. Each node of IHUP tree stores information in form of item name, TWU and support count. IHUP algorithm works in three steps. First one is construction of IHUP tree, second one is generation of HTWUIs and third and last one is identification of high utility itemsets.

In first step all items are arranged in descending order of TWU. In fig 1 min_util is set to 40 hence items who's TWU is below 40 are not included. This is because item and its superset will not be part of HTWUIs so it is better to exclude such kind of items. Items which are excluded are F, G and H. During insertion of transaction into tree itemsets are also sorted internally in descending order, for example in Table 1 Transaction T1 consist of item A,C and D. Now these items are also sorted in descending order of its TWU. After sorting we get sequence as C, D and A. So transaction T1 is inserted as follows. First C is inserted in left branch with entry as {C}:10,1 then D is inserted as child of C with entry as {D}:2,1 and lastly A is inserted as child of D with entry as {A}:5,1. Here first number after colon is profit of item and next one is support count.

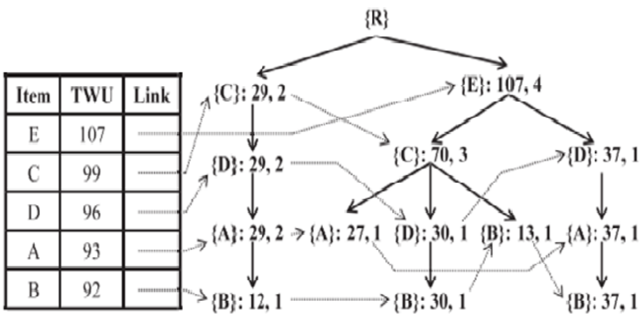

Figure 1: An IHUP- Tree when min_util = 40

In similar way all transactions are inserted into tree. With addition of an each entry profit and support count of item is updated. After tree construction in second step HTWUIs are collected by using FP-Growth [3] .In last step high utility itemsets are identified by scanning database once more.

As stated earlier IHUP generates too many of HTWUIs in phase I, thus reduces mining performance. To overcome this problem two strategies are proposed namely DGU (Discarding Global Unpromising Items) and DGN (Decreasing Global Node Utilities). In next section we will see working of above mentioned strategies.

5. DGU and DGN Strategies

DGU and DGN basically try to reduce intermediate large candidate itemset generation. DGU strategy works in two scan, in first scan TU of each transaction is computed along with TWU of each item. In second scan items who's TWU are less than threshold are removed from database. During removal TU of each transaction is also updated. Updated TU is called as RTU (Reorganized transaction utility).

DGN works according to following principle, since in construction of i_K – tree , i_{K+1}, $i_{K+2....}$ items are not involved they can be discarded. Brief explanation of DGU is given further.

Before using above strategies we will first see data structure used in these strategies. Here like IHUP-Tree another data structure named UP-Tree is used. Difference between IHUP-Tree and UP-Tree is, UP-Tree stores item name in N.name, its utility N.nu, its parent N.parent, its horizontal link N.hlink and set of child nodes. In addition to this there is table called header table to facilitate traversal of UP-Tree.

The construction of Global UP-tree by applying DGU and DGN works in three steps. First Unpromising items are removed and RTU is calculated. Second transactions are sorted according to TWU. At last transactions are inserted into UP-Tree and its utility N.Nu is updated. Now we will see illustration of DGU and DGN strategies with an example. Consider transaction database in table 1 and profit of each item in table 2. Suppose min_util is set to 50.In first scan TU of each transaction and TWU of each item is computed. Values of TWU are computed as follows,

A = 17 + 27 + 37 + 12 = 93
B = 37 + 30 + 13 + 12 = 92
C = 17 + 27 + 30 + 13 + 12 = 99
D = 17 + 37 + 30 + 12 = 96
E = 27 + 37 + 30 + 13 = 107
F = 37
G= 27 + 13 = 40
H=12

As min_util is set to 50, F, G and H are unpromising items and hence are discarded. Remaining items are promising ones and are used further. When unpromising items are discarded TU of each transaction is updated. The new TU is called RTU.

Table 3: Reorganized transaction utility

TID	Reorganized transaction	RTU
T_1'	(C,10) (D,1) (A,1)	17
T_2'	(E,2) (C,6) (A,2)	22
T_3'	(E,2) (D,6) (A,2) (B,2)	32
T_4'	(E,1) (C,13) (D,3) (B,4)	30
T_5'	(E,1) (C,4) (B,2)	11
T_6'	(C,1) (D,1) (A,1) (B,1)	10

After reorganization of transaction, they are inserted into global UP-Tree. For example consider transaction T_1'. When it is inserted first node N_C is created with N_C.item = {C} and N_C.count = 1. N_C.nu is calculated as RTU of T_1' minus utilities of remaining items i.e RTU(T_1') – *u*({D},T_1' , {A},T_1'). On substituting values we get 17 - (2+5) = 10. Similarly each transaction is inserted into global UP-Tree and N.nu of each node is updated. After inserting all transaction global UP-Tree will look like as below.

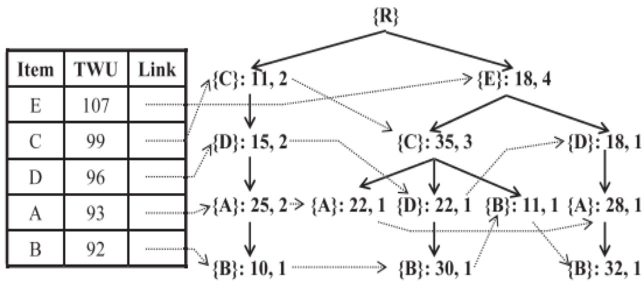

Figure 2: A UP-Tree after applying DGU and DGN

When we compare tree produced in IHUP and tree produced by applying DGU and DGN we can notice utilities of nodes in UP-Tree are less than those in IHUP-Tree. Thus we DGU can say DGU and DGN further reduces number of HTWUIs. After construction of UP-Tree, last thing we have to do is to mine the tree for high utility itemset. Consider UP-Tree in fig 2. Assume min_util = 50. The bottom entry in header table is of item 'B'. Now trace all {B}.hlinks and sum up {B}'s node utility. After summing we get nu_{sum}({B}) = 83. Thus new PHUIs (Potential High Utility Itemset) {B}:83 is generated because 83 is greater than min_util = 50. By tracing {B}.hlinks nodes related to {B} are found. Next by tracing these routes to root, four paths <ADC> : 10, <DCE> : 30, <CE> : 11 and <ADE> : 32 are found. Numbers besides path are PU (path utility) of path which are nothing but *nu* of {B}.

We have another local data structure called CPB (Conditional Path Base) to store local data set. For example {B}-CPB contains path ending with {B}. Hence we have following path in {B}-CPB tree.

Table 3: Path Utility

Retrieved path: Path utility	Reorganized path: Path utility (after DLU)	Support count
<ADC>: 10	<DC>: 5	1
<DCE>: 30	<EDC>: 30	1
<CE>: 11	<EC>: 11	1
<ADE>: 32	<ED>: 27	1

After retrieving all path we calculate PU of each item. PU is calculated same as TWU. Following are path utilities of item present in above table.

PU{A} = 10 + 32 = 42
PU{C} = 10 + 30 + 11 = 51
PU{D} = 10 + 30 + 32 = 72
PU{E} = 30 + 11 + 32 = 73

As we have considered min_util = 50, local unpromising item {A} is discarded and Reorganized path and path utility is calculated as shown in middle column of above table.

Now local UP-Tree is constructed as {B}-CPB from reorganized path. Consider insertion of reorganized path <DC> into {B}-Tree. First node N_D is created under root node with N_D.*nu* = 5 – minimum utility of {C} * <DC>.count = 5-1*1 = 4 Second node N_C is created under N_D with N_C.*nu* = 5 and N_C.Count = 1. After inserting all path tree looks like as follows.

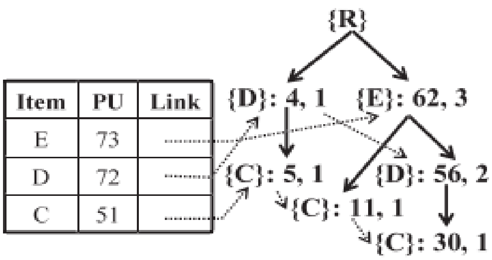

Figure 3: Mining of {B}-CPB tree

Remaining PHUIs from {B}-Tree are {BD}: 56+4 =60, {BDE}: 56 and {BE}: 62. After mining remaining entries in header table all PHUIs are obtained as {A}: 75, {B}:83, {BD}:60, {BDE}:56, {BE}:62 and {D}:55. Thus we have generated high utility itemsets efficiently.

6. Conclusion

We have seen many techniques for mining high utility itemsets starting with Apriori method till above discussed strategies. By far DGU and DGN strategies are been most efficient of all techniques. IHUP-Tree is efficient but produces lot many of intermediate itemsets. Thus two discussed strategies are efficient and can be used in various mining task where size of dataset is large.

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 Int'l 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 Int'l 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.

Int’l 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.