Generation of Potential High Utility Itemsets from Transactional Databases

Rajmohan.C
Asst.Prof/IT,
SREC, Coimbatore,INDIA
rajmohanresh@gmail.com

Priya.G
Dept of IT
,SREC,Coimbatore,.INDIA
pradeepriya21@gmail.com

Niveditha.C
Dept of IT
SREC,Coimbatore,.INDIA
smart.nivedetha@gmail.com

Pragathi.R
Dept of IT
SREC,Coimbatore,.INDIA
umailpragathi@gmail.com

***Abstract*— Mining high utility item sets from a transactional database refers to the discovery of item sets with high utility. Previous algorithm such as Apriori and Fp-Growth incurs the problem of producing a large number of candidate item sets for high utility item sets. Such large number of candidate item sets degrades the mining performance in terms of execution time. So, to improve the mining performance Up-Growth came into existence. Up-Growth effectively mines the potential high utility item sets from the Transactional database. The information of high utility item sets is maintained in a tree-based data structure named utility pattern tree (UP-Tree) such that candidate item sets can be generated efficiently with only two scans of database. The performance of UP-Growth is compared with the state-of-the-art algorithms on many types of both real and synthetic data sets.**

Index terms* –*Candidate Elimination, Frequent pattern, High Utility, Itemset Mining, Utility

I. INTRODUCTION

Data mining refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. Extracting Frequent Itemsets from large database is an important role in mining applications such as website click stream analysis, cross-marketing in retail stores, e-commerce management. The Apriori algorithms [1], which are used in association rule mining [14] produces the frequent itemsets from transactional database. In past, some previous studies are produced with candidate and without candidate generation. But it does not satisfy the customer requirements like profit, sales in particular item. Itemset mining with high interestedness or utility plays vital role in mining area. The unit profits and purchased quantities of items are not considered in the framework of mining frequent itemset. Hence, it cannot satisfy the requirement of the user who is interested in discovering the itemsets with high sales profits. In view of this, utility mining emerges as an important topic in data mining for discovering the itemsets with high utility like profits. The basic meaning of utility is interestedness / importance/ profitability of items to the users. To improve the mining performance and avoid scanning original database repeatedly, use a compact tree structure, named UP-Tree [16], to maintain the information of transactions and high utility itemsets.

II. MATHEMATICAL BACKGROUND

The definitions for UP-Tree construction are utility of an item, transaction utility, transaction weighted utility, High utility and their related works are as follows.

A. Preliminary

Given a set of items $I=\{i1,i2,...im\}$, each item $ip(1 \le p \le m)$ has a unit profit $pr(ip)$. A transaction database $D=\{T1,T2,....,Tn\}$ contains a set of transactions, and each transaction $Td(1 \le d \le n)$ has a unique identifier d, called TID. Each item ip in transaction *Td* is associated with a quantity *q(ip,Td)*, (i.e) the purchased quantity of *ip* in *Td.*

Definition 1: Utility of an item ip in a transaction *Td* is denoted as *u(ip,Td)* and defined as $pr(ip) \times q(ip,Td)$.

Definition 2: Utility of an itemset X in Td is denoted as u(X, Td) and defined as $\sum_{ip \in X \wedge X \subseteq Td} u(ip,Td)$.

Definition 3: Utility of an itemset X in D is denoted as u(X) defined as $\sum_{X \subseteq Td \wedge Td \in D} u(X,Td)$.

Definition 4: An itemset is called a high utility itemset if its utility is no less than user-specified minimum utility threshold which is denoted as min_util. Otherwise; it is called a low-utility itemset.

Definition 5: Transaction utility of a transaction Td is denoted as *TU(Td)* and defined as *u(Td,Td).*

Definition 6: Transaction-weighted utility of an itemset X is the sum of the transaction utilities of all the transactions containing X, which is denoted as TWU(X) and defined as $\sum_{X \subseteq Td \wedge Td \in D} TU(Td)$.

Definition 7: An itemset X is called a high-transaction weighted utility itemset (HTWUI) if TWU(X) is no less than min_util.

B. Related Work

The most famous mining techniques are association rule mining [1, 14] and sequential pattern mining [2]. Association rule mining extracts interesting correlations and frequent patterns from large set of transactional databases or other data repositories. Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. Association rules are sometimes very large. It is impossible for the end users to validate such large number of complex association rules, hence limiting the usefulness of the data mining results. There are two important basic measures for association rules, support and confidence. Since the database is large and users concern about only those frequently purchased items, mostly thresholds of support and confidence are predefined by users to drop those rules that are not so interesting or useful. The two thresholds are called minimal support and minimal confidence respectively. One of the well-known algorithms for mining association rule is Apriori [1]. Pattern growth based association rule mining algorithms [14] such as FP-Growth [14] were introduced later. FP-Growth achieves a better performance than Apriori-based algorithms since it finds frequent itemsets without generating any candidate itemset and scans database just twice. In the frequent itemset mining, the importance of items to users is not considered. Thus, weighted association rule mining was brought to attention [4]. Cai et al. first proposed the concept of weighted items and weighted association rules [4]. However, since the weighted association rules do not have downward closure property, mining performance can not be improved.

To address this problem, the concept of weighted downward closure property [16] is used.By using transaction weight, weighted support not only reflect the importance of an itemset but also maintain the downward closure property during the mining process. Thus, the issue of high utility itemset mining is raised and many studies [3, 5, 10, 16] have addressed this problem. Liu et al. proposed an algorithm named Two-Phase [15] which is mainly composed of two mining phases. In phase I, it uses an Apriori-based level-wise method to enumerate high utility items. Their TWUs are computed by scanning the database once in each pass. After these steps, the complete set of high utility items is collected in phase I. In phase II, high transaction weighted utility itemsets (HTWUIs) that are high utility itemsets are identified with an additional database scan. Eventhough Two-Phase algorithm reduces search space; it still generates too many candidates to obtain HTWUIs and requires multiple database scans. To overcome this problem, Li et al. [17] proposed an isolated items discarding strategy (abbreviated as IIDS) to reduce the number of candidates. By pruning isolated items during level-wise search, the number of candidate itemsets for HTWUIs in phase I can be reduced. This algorithm still scans database for several times and uses a candidate generation. Apriori is more efficient during the candidate generation process. Apriori uses pruning techniques to avoid measuring certain itemsets, while completeness. These are the itemsets that the algorithm can prove will not turn out to be large. However there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements. FP-Tree, frequent pattern mining, is another milestone in the development of association rule mining, which breaks the main bottlenecks of the Apriori. The frequent itemsets are generated with only two passes over the database and without any candidate generation process. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-Tree scales much better than Apriori because as the support threshold goes down, the number and the length of frequent itemsets increase dramatically. The frequent pattern generation process includes two sub processes: constructing the FT-Tree and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms.

III. PROPOSED WORK

The algorithm UP-Growth (Utility Pattern Growth) called UP-Tree (Utility Pattern Tree), for discovering high utility itemsets maintains important information related to utility patterns within databases. High utility itemsets can be generated from UP-Tree efficiently with only two scans of databases. Several strategies are proposed for facilitating the mining processes of UP-Growth by maintaining only essential information in UP-Tree. It is 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.DGU while constructing UP-Tree. The Global tree is constructed. The conditional pattern is generated by tracing the paths in the original tree. The Minimum item utility is evaluated by minimum utility threshold. Find the local promising items in the tree. Then Apply DLU to reduce path utilities of the paths; the path utility of an item is estimated.

The reorganized path construction process is done. This process is done after discarding the nodes. The local unpromising items and their estimated Node Utilities from the paths and path utilities of conditional pattern bases are extracted. The local Node utilities are decreased for the nodes of local UP-Tree by estimated utilities of descendant Nodes Reorganized path. Path utility (after DNU). Support count is estimated too. The steps are repeated certainly. After finding all PHUIs, the final step is to identify high utility itemsets and their utilities from the set of PHUIs by scanning original database once. 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.

To maintain the information from transactional database, UP-Tree is used. In UP-Tree, each node consists of item name, count, utility. A header table consists of information about the traversal of UP-Tree.

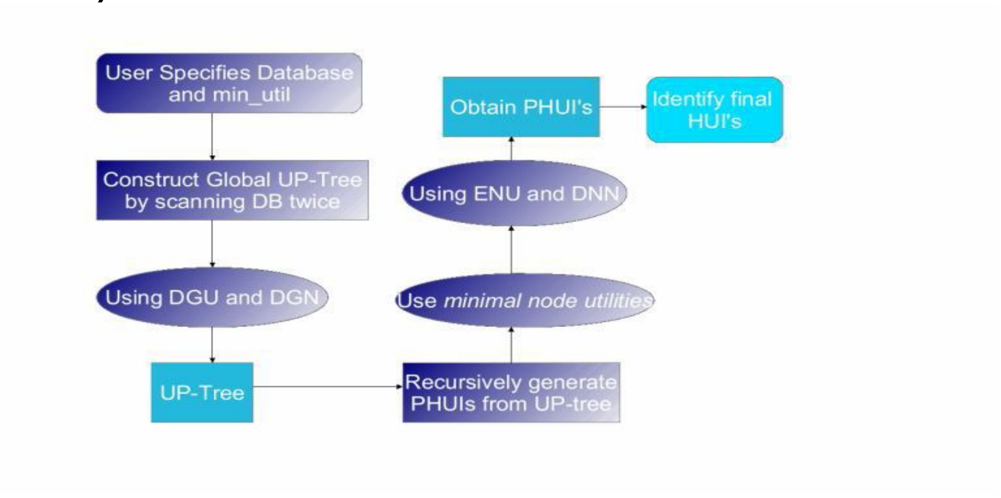

User Specifies Database and min_util
Construct Global UP-Tree by scanning DB twice
Using DGU and DGN
UP-Tree
Recursively generate PHUIs from UP-tree
Use minimal node utilities
Using ENU and DNN
Obtain PHUI's
Identify final HUI's

Figure 1- UP-Tree Flow

```
Subroutine: UP-Tree
Input: Transaction database D, user specified threshold.
Output: high utility itemsets.
1. Scan database of transactions Td ϵ D
2. Determine transaction utility of Td in D and TWU of itemset (X)
3. Get user specified threshold
4. If (TWU(X) ≤ min_sup) then Remove Items from transaction database
5. Else insert into header table H and to keep the items in the descending order.
6. Repeat step 4 & 5 until end of the D.
7. Insert Td into global UP-Tree
8. Apply DGU and DGN strategies on global UP- tree
9. Re-construct the UP-Tree
10. For each item $a_1$ in H do
11. Generate a PHUI Y=X U $a_1$.
12. Estimate utility of Y
13. Put local promising item in Y-CPB into H
14. Apply DLU Strategy to reduce path utility.
15. Apply DLN Strategy and insert path into $T_{d.}$
```

Figure 2. Subroutine of PHUI.

TRANSACTION TABLE TABLE1

TID	TRANSACTION	TU
T1	(1,1)(3,1)(4,1)	8
T2	(1,2)(3,6)(5,2)(7,5)	27
T3	(1,1)(2,2)(3,1)(4,6)(5,1)(6,3)	28
T4	(2,4)(3,3)(4,3)(5,1)	20

PROFIT TABLE TABLE II

ITEM	1	2	3	4	5	6	7
PROFIT	5	2	1	2	3	5	1

A. UP-Tree Elements:

In a UP-Tree, a node N constitutes of N.name, N.count, N.nu, N.parent, N.hlink and a set of child nodes. N.name is the node's item name. N.count is the node's support count. N.nu is the node's node utility, i.e., overestimated utility of the node. N.parent records the parent node of N. N.hlink is a node link which points to a node whose item name is the same as N.name.

B. Strategy DGU: Discarding Global Unpromising Items

The construction of a global UP-Tree can be performed with two scans of the original database. In the first scan, *Transaction Utility* (TU) is computed. At the same time, *Transaction-Weighted Utility* (TWU) of each single item is also accumulated. By *transaction-weighted downward closure* (TWDC) property, an item and its supersets are unpromising to be high utility itemsets if its also TWU is less than the minimum utility threshold. Such an item is called an unpromising item. An item is called a promising item if TWU >= min_util. Otherwise, it is called a un promising item. Without loss of generality, an item is also called a promising item if its *overestimated utility* is no less than min_util. Otherwise; it is called an unpromising item. Consider an example, the transaction database in Table 1 and the profit table in Table 2. Suppose the minimum utility Threshold (*min_util)* is 50. In the 1st scan of database, Transaction Utility and the TWUs of the items are computed.{2},{6} and {7} are unpromising items since their TWUs are less than *min_util*. The promising items are reorganized in header table in the descending order of TWU. Table 3 shows the reorganized transactions and their RTUs for the database in Table 1. As shown in Table 3, unpromising items {*2*},{6} and {*7*} are removed from the transactions second, third and fifth Transactions (*T2*, *T3* and *T4*) respectively. The utilities of {*2*},{6} and {*7*} are eliminated from the TUs of *T2*, *T3* and *T4* respectively. The remaining promising items {*1*}, {*3*}, {*4*} and {*5*} in the transaction are sorted in the descending order of TWU. Then, we insert reorganized transactions into the UP-Tree. After pruning items, sort the items in descending order based on TWU value.

TABLE 3 - RE-ORGANIZED TRANSACTIONS AND THEIR RTUS

ITEM	TWU
3	83
5	75
1	63
4	56

Table 4

TID	REORGANIZED TRANSACTION	RTU
T1	(1,1)(3,1)(4,1)	8
T2	(1,2)(3,6)(5,2)	22
T3	(1,1)(3,1)(4,6)(5,1)	23
T4	(3,3)(4,3)(5,1)	12

From the above table 4, UP-Tree will be constructed.
R-root node, {3}:4,83 –{item}:count,TWU

$$\begin{tikzcd}
 & \{R\} \arrow[d] & & \\
 & \{3\}:4,83 \arrow[dl] \arrow[dr] & & \\
\{1\}:1,8 \arrow[d] & & \{5\}:3,57 \arrow[dl] \arrow[dr] & \\
\{4\}:1,8 & \{1\}:2,45 \arrow[d] & & \{4\}:1,12 \\
 & \{4\}:1,23 & &
\end{tikzcd}$$

Figure 3. An UP-Tree when min_util=50

C. Strategy DGN: Decreasing Global Node Utilities

By actual utilities of descendant nodes during the construction of global UP-Tree we can decrease global node utilities. 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.
The potential high utility itemset from transactional database are as below

<3,1>	2
<3,5,1>	14
<3,5>	6

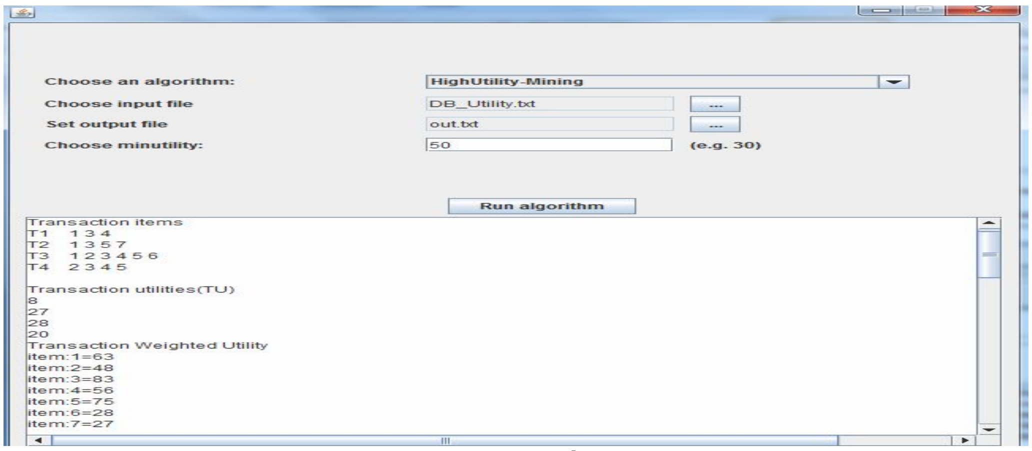
FIG 6 PHUI ITEMS

$$\begin{tikzcd}
 & \{R\} \arrow[d] & & \\
 & \{3\}:4,13 \arrow[dl] \arrow[dr] & & \\
\{1\}:1,5 \arrow[d] & & \{5\}:3,14 \arrow[d] \arrow[dr] & \\
\{4\}:1,2 & & \{1\}:2,17 \arrow[d] & \{4\}:1,6 \\
 & & \{4\}:1,1 &
\end{tikzcd}$$

Figure 5. A UP-Tree by applying Strategies DGU and DGN

IV. Implementation and Evaluation

The experiments were performed on a Intel CORE i3 processor with 1.5 GB memory. The operating system is Windows 7. The algorithm is implemented in Java language. The PHUI is identified by scanning the database twice. Input file, output file, minimum utility will be chosen. Input file consists of the items and their quantities in the transaction table. Profit table is also taken as input. Minimum utility value is taken is shown in figure 6. Output file consists of high utility items by eliminating unpromising items which are below the minimum threshold value. If the input is taken as transaction table and profit table (i.e) Table1 and Table 2, The Output is constructed by applying DGU and DGN strategies in Table 3. The output will be as potential high utility items.



Choose an algorithm: HighUtility-Mining
Choose input file: DB_Utility.txt
Set output file: out.txt
Choose minutility: 50 (e.g. 30)

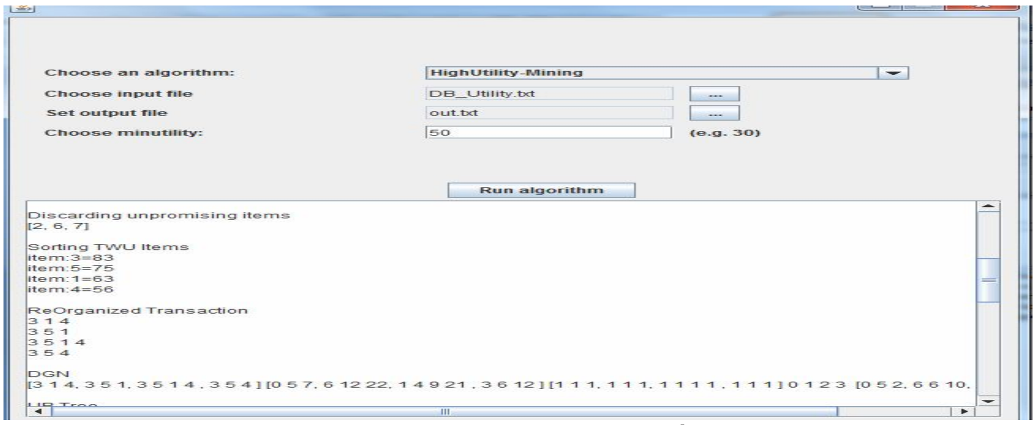
```
Transaction items
T1   1 3 4
T2   1 3 5 7
T3   1 2 3 4 5 6
T4   2 3 4 5

Transaction utilities(TU)
8
27
28
20
Transaction Weighted Utility
item:1=63
item:2=48
item:3=83
item:4=56
item:5=75
item:6=28
item:7=27
```

Figure 6. Input selection

A. UP-Tree construction:

Before constructing UP-Tree, utility for each item will be calculated. By using those item utilities, transaction utility can be calculated. Transaction weighted utility will be calculated for each item is shown in fig 8.



Choose an algorithm: HighUtility-Mining
Choose input file: DB_Utility.txt
Set output file: out.txt
Choose minutility: 50 (e.g. 30)

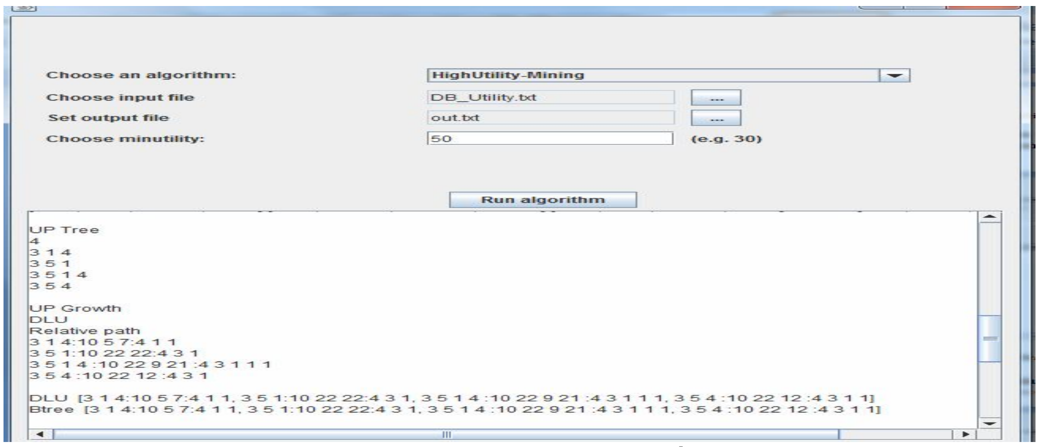
```
Discarding unpromising items
[2, 6, 7]

Sorting TWU Items
item:3=83
item:5=75
item:1=63
item:4=56

ReOrganized Transaction
3 1 4
3 5 1
3 5 1 4
3 5 4

DGN
[3 1 4, 3 5 1, 3 5 1 4 , 3 5 4 ] [0 5 7, 6 12 22, 1 4 9 21 , 3 6 12 ] [1 1 1, 1 1 1, 1 1 1 1 , 1 1 1 ] 0 1 2 3  [0 5 2, 6 6 10,
```

Figure 7. UP-Tree construction phase-I


Choose an algorithm: HighUtility-Mining
Choose input file: DB_Utility.txt
Set output file: out.txt
Choose minutility: 50 (e.g. 30)

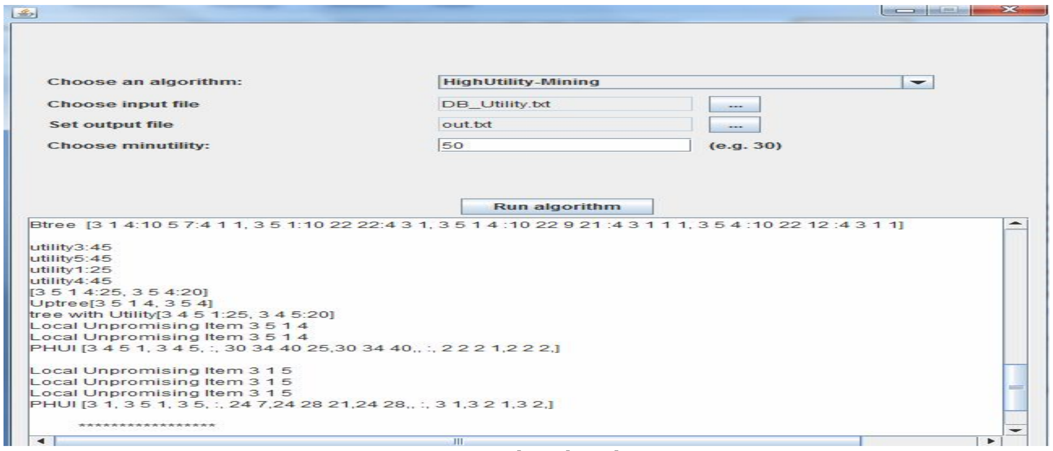
```
UP Tree
4
3 1 4
3 5 1
3 5 1 4
3 5 4

UP Growth
DLU
Relative path
3 1 4:10 5 7:4 1 1
3 5 1:10 22 22:4 3 1
3 5 1 4 :10 22 9 21 :4 3 1 1 1
3 5 4 :10 22 12 :4 3 1

DLU  [3 1 4:10 5 7:4 1 1, 3 5 1:10 22 22:4 3 1, 3 5 1 4 :10 22 9 21 :4 3 1 1 1, 3 5 4 :10 22 12 :4 3 1 1]
Btree  [3 1 4:10 5 7:4 1 1, 3 5 1:10 22 22:4 3 1, 3 5 1 4 :10 22 9 21 :4 3 1 1 1, 3 5 4 :10 22 12 :4 3 1 1]
```

Figure 8. UP-Tree construction phase-II

The transaction items are taken and UP-Tree is constructed by eliminating the unpromising items using DGU principles and it is implemented using Java language. Database scans are reduced by using UP-Growth algorithm.


Choose an algorithm: HighUtility-Mining
Choose input file: DB_Utility.txt
Set output file: out.txt
Choose minutility: 50 (e.g. 30)

```
Btree  [3 1 4:10 5 7:4 1 1, 3 5 1:10 22 22:4 3 1, 3 5 1 4 :10 22 9 21 :4 3 1 1 1, 3 5 4 :10 22 12 :4 3 1 1]

utility3:45
utility5:45
utility1:25
utility4:45
[3 5 1 4:25, 3 5 4:20]
Uptree[3 5 1 4, 3 5 4]
tree with Utility[3 4 5 1:25, 3 4 5:20]
Local Unpromising Item 3 5 1 4
Local Unpromising Item 3 5 1 4
PHUI [3 4 5 1, 3 4 5, :, 30 34 40 25,30 34 40,, :, 2 2 2 1,2 2 2,]

Local Unpromising Item 3 1 5
Local Unpromising Item 3 1 5
Local Unpromising Item 3 1 5
PHUI [3 1, 3 5 1, 3 5, :, 24 7,24 28 21,24 28,, :, 3 1,3 2 1,3 2,]

     ******************
```

Fig12 Potential High Utility Itemsets

V. Conclusion

Efficient algorithms named UP-Growth for mining high utility itemsets from transaction databases. A data structure named UP-Tree was proposed for maintaining the information of high utility itemsets. Potential high utility itemsets can be efficiently generated from UP-Tree with only two database scans. An adaptive Utility Mining algorithm called the UP-Growth is proposed, for mining high utility itemsets with a set of effective strategies for pruning potential high utility itemsets.
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. The UP-Growth algorithm builds UP-tree to reduce the memory consumption while storing the utility itemsets. An UP-tree is built only for pruned database that fit into main memory easily. Thus, the algorithm works efficiently for tree construction process in terms of time and memory space requirements. By simulation results, we have shown that the high utility items for the Transaction database by constructing UP-Tree.

References

[1]. 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.
[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. In *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, Issue 12, pp. 1708-1721, 2009.
[3]. 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, Nov., 2003.
[4]. A. Erwin, R. P. Gopalan, and N. R. Achuthan. Efficient mining of high utility itemsets from large datasets. In *Proc. of PAKDD 2008,LNAI 5012*, pp. 554-561.
[5]. J. Han, J. 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.
[6]. 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.
[7]. Y. Liu, W. Liao, and A. Choudhary. A fast high utility itemsets mining algorithm. In *Proc. of the Utility-Based Data Mining Workshop*, 2005.
[8]. B.-E. Shie, V. S. Tseng, and P. S. Yu. Online mining of temporal maximal utility itemsets from data streams. In *Proc. of the 25th Annual ACM Symposium on Applied Computing*, Switzerland, Mar., 2010.
[9]. H. Yao, H. J. Hamilton, 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, Aug., 2006.

[10]. 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

[11]. R. Agrawal and R. Srikant, "Mining Sequential Patterns," in Proc. of the 11th Int'l Conference on Data Engineering, pp. 3-14, Mar., 1995.

[12]. C. H. Lin, D. Y. Chiu, Y. H. Wu and A. L. P. Chen, "Mining frequent itemsets from data streams with a time-sensitive sliding window," in Proc. of the SIAM Int'l Conference on Data Mining (SDM 2005), 2005.

[13]. 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.

[14]. E. Georgii, L. Richter, U. Ru¨ ckert, and S. Kramer, "Analyzing Microarray Data Using Quantitative Association Rules," Bioinformatics, vol. 21, pp. 123-129, 2005.

[15]. J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," Proc. Int'l Conf. on Data Eng.,pp. 106-115, 1999.

[16]. J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," Proc. 21th Int'l Conf. Very Large Data Bases,pp. 420-431, Sept. 1995.