Paper ID: CSEIT10

FINDING HIGH UTILITY ITEMSETS OVER DYNAMIC TRANSACTIONAL DATABASE USING UP-GROWTH AND UP-GROWTH+ ALGORITHM

Ms. Anita A. Bhosale
Computer Sci and Engg Dept Annasaheb Dange College of Engg Ashta, Sangli, India
Mr. Siddheshwar V. Patil
Information Technology Dept Annasaheb Dange College of Engg Ashta, Sangli, India
Ms.Pallavi S.Kadam
Computer Sci and Engg Dept Annasaheb Dange College of Engg Ashta, Sangli, India

Abstract—Mining high utility itemsets is an important research area in data mining. Finding itemsets with high utility like profit from database is known as high utility itemset mining. Different algorithms have been work on this area, but some of them have problem of generating large number of Potential High Utility Itemsets (PHUIs). Due to this performance of mining is minimized in case of execution time. In this paper we have focus on UP-Growth and UP-Growth+ algorithm which overcomes this limitation. This technique uses tree based data structure, UP-Tree for generating high utility itemsets with two scan of database. In this paper we have extend the working of these algorithms on incremental database because these existing algorithms have consider only static database. This limitation has covered in our system. When database is updated, our system regenerates UP-tree efficiently and generates high utility itemsets from dynamic database. Also we have compared the performance of both algorithms.

Keywords—Data mining; High utility mining; Incremental mining;

I. INTRODUCTION

Frequent Itemset mining means finding items that occurs in a database above a user given frequency threshold. These techniques do not consider the quantity or profit of the purchased items. So it is not useful for the user who want find the importance of the items in database. However quantity and profit are basic terms for maximizing the profit of the organization. For this purpose new technique is introduced called as high utility itemsets mining [1]. This technique refers to finding itemsets from the database which gives high utility. Utility means importance or interestedness of items. Utility of items is calculated by multiplying internal utility and external utility. Itemset in a single transaction or quantity is called as internal utility and itemset in different transaction database or profit is called as external utility.

High utility itemsets is Itemset which have utility no less than a user-specified minimum utility threshold; otherwise, it is called a low-utility Itemset. In many applications like cross-marketing, in retail stores mining such high utility itemsets from databases is an important task.

Existing techniques [2, 3, 4, 5, 6, 7] used for utility pattern mining. However, some existing methods often generate a large set of potential high utility itemsets and the mining performance is degraded consequently. If database contain long transactions or low threshold value is set situation is more complicated for utility mining. The existing algorithms UP-Growth and UP-Growth+ [1] deal with these issues. In this paper this algorithms have work on dynamic transaction database.

II. RELATED WORK

Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee proposed three novel tree structures for efficiently perform incremental and interactive HUP mining[2]. The first tree structure is Incremental HUP Lexicographic Tree (IHUP$_L$-Tree). This tree arranges the items according to their lexicographic order and it can accepts the incremental data without any restructuring operation. The second tree structure is IHUP Transaction Frequency Tree (IHUP$_{TF}$-Tree), which arranging items according to item's transaction frequency in descending order. To minimize the mining time, the last tree IHUP-Transaction Weighted Utilization Tree (IHUP$_{TWU}$-Tree) is designed. It is based on the TWU value of items in descending order.

Alva Erwin, Raj P. Gopalan, and N. R. Achuthan, proposed CTU-PROL algorithm for efficient mining of high utility itemsets from large datasets[3]. This algorithm finds the large TWU items in the transaction database. If data sets is too large to be held in main memory, the algorithm creates subdivisions using parallel projections and for each subdivision, a Compressed Utility Pattern Tree (CUP-Tree) is used to mine the complete set of high utility itemsets. If the dataset is small, it creates a single CUP-Tree for mining high utility itemsets.

Shankar S., Purusothaman T., Jayanthi, S., suggested a novel algorithm for mining high utility itemsets[4]. This fast utility mining (FUM) algorithm finds all high utility itemsets within the given utility constraint threshold. The proposed FUM algorithm scales well as the size of the transaction database increases with regard to the number of distinct items available.

R. Chan, Q. Yang, and Y. Shen, suggested mining high utility itemsets[5]. They proposed a novel idea of top-K objective-directed data mining algorithm, which mines the top-K high utility closed patterns that directly support a given business objective. To association mining, they add the concept of utility to capture highly desirable statistical patterns and present a levelwise itemset mining algorithm. They develop a new pruning strategy based on utilities that allow pruning of low utility itemsets to be done by means of a weaker but antimonotonic condition.

Ramaraju C., Savarimuthu N., proposed a conditional tree based novel algorithm for high utility itemset mining[6]. A novel conditional high utility tree (CHUT) compress the transactional databases in two stages to reduce search space and a new algorithm called HU-Mine is proposed to mine complete set of high utility item sets.

Y. Liu, W. Liao, and A. Choudhary, proposed a fast high utility itemsets mining algorithm [7]. They are present a Two-Phase algorithm to efficiently prune down the number of candidates and can precisely obtain the complete set of high utility itemsets. First phase proposes a model that applies the "transaction-weighted downward closure property" on the search space to expedite the identification of candidates. Second phase performs one extra database scan to identify the high utility itemsets.

Adinarayanareddy B., O. Srinivasa Rao, MHM Krishna Prasad, suggested improved UP-Growth high utility itemset mining[8]. The compact tree structure, Utility Pattern Tree i.e. UP-Tree, maintains the information of transactions and itemsets and avoid scanning original database repeatedly. UP-Tree scans database only twice to obtain candidate items and manage them in an efficient data structured way. Applying this UP-Tree to the UP-Growth algorithm takes more execution time for Phase II. Hence they presents modified algorithm aiming to reduce the execution time by effectively identifying high utility itemsets.

P. Asha, Dr. T. Jebarajan, G. Saranya, presents a survey on efficient incremental algorithm for mining high utility itemsets in distributed and dynamic database[9]. This proposed system divided into one master node and two slave nodes. Database is partitioned for every slave node for computation. The slave node counts the occurrence of each item. These data's are stored in their local table. Then each slave node sends these tables to master node. This node maintain global table for storing these data. Based on the minimum utility threshold value it calculates the promising and unpromising itemsets.

III. Proposed System

UP-Growth and UP-Growth+[1] algorithm finds high utility Itemsets efficiently. By applying the strategies of these algorithms the number of candidate itemsets can be highly reduced in phase I. In phase II high utility itemsets can be identified. This technique used on fixed datasets. It did not consider the modification of database.
System Architecture as follows:

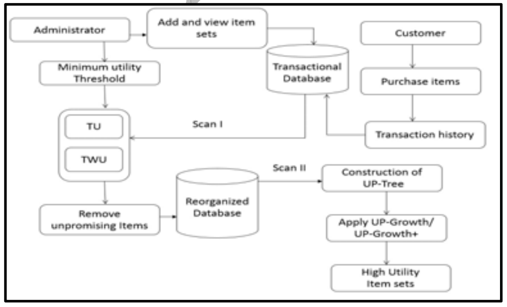


Fig. 1.System architecture

Our system works on incremental database i.e. deletion or insertion of one or more records from database is considered. It uses UP-growth and UP-Growth+ algorithm. This system generates UP-Tree within two scans of database.

IV. Methdology

A. Module 1: Administrator

Our system is works with Microsoft Food mart 2000 database. This database contains various products information for customer to purchase these products i.e. product name and prize etc. Database also has records of customer details and history of all transactions made by customers. The administrator works on this database. He has login into our system by using his authentication details. Administrator can see records of new customer and their transaction details in main database. Administrator also updates or adds new products for customers. Finally Administrator uses this database as input to the UP-construction module.

B. Module 2: Customer

In our system new customers register themselves by filling their details in registration form including their authentication details. After registration customers have to login to our system by using their username and password. Our system displays all product information to the customer for purchase. Among these products customer will purchase particular product. These transaction details of customer are saved in main Food mart database.

C. Module 3:Construction of UP-Tree[1]

Global UP-Tree is constructed by using two strategies DGU(Discarding Global Unpromising items) and DGN(Decreasing Global Node utilities).

1) Discarding Global Unpromising items

This approach constructs Global UP-Tree within two scans of database as follows:

a) First scan:-

- First Transaction Utility(TU) of each transaction of the database is computed by using formula(1)

$$TU(T_d) = u(T_d, T_d) \quad (1)$$

Where,
Td is Transaction in database having unique Identifier d.

- By using Transaction utility(TU) values for all transaction, Transaction Weighted Utility (TWU) of each item is also accumulated by using(2).

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

Where,
X is itemset and D is Database.

- We will get promising items and unpromising items depending on user specified minimum utility threshold.
- Then we have to remove unpromising items from the transaction and utilities are eliminated from the Transaction Utility (TU) of the transaction.

These transactions are called as Reorganized Transaction.

- Then all promising items in the Reorganized transaction are sorted in the descending order of Transaction Weighted Utility (TWU).

b) Second scan:-

- Resulting transactions of the database are inserted into UP-Tree.

Initially, UP-tree is created with root node R. In UP-Tree, each node N contains six elements i.e. name, support count, node utility value, parent, child nodes and link. Header table is used to facilitate the traversal of UP-Tree. It consists of item name, an estimate utility value and a link. The link points to the last occurrence of the node which has the same item as the entry in the UP-Tree. By following the link in the header table and the nodes in UP-Tree, the nodes whose item names are the same can be traversed efficiently.

2) Decreasing Global Node utilities

In this approach node utilities are reduced by real utilities of descendent node and these utilities are nearer to the root node of UP-tree. If transactions of the database contain more items, more utilities are eliminated by this approach.

D. Module 4: UP-Growth Algorithm[1]

After constructing Global UP-Tree it passes as input to the UP-Growth algorithm and it determines high utility itemsets. UP-Growth efficiently generates PHUIs from the global UP-Tree with two strategies, namely DLU (Discarding local unpromising items) and DLN(Decreasing local node utilities). For this Minimum Item Utility Table(MIUT) is used which contains minimum item utility for all global promising items.

1) Discarding local unpromising items

In this minimum item utilities of unpromising items are discarded from path utilities of the paths. Each path is traced from bottom node of header table and node which are found in tree traced to root. This approach generates pattern as follows:

- Generate Conditional Pattern Bases(CPB) by tracing the paths in the Global UP-Tree generated by UP-Construction module. CPB contains Path Utility(PU) values which is calculated by using formula(3),

$$\mathrm{pu}(p,\{i_m\}-\mathrm{CPB}) = N_{i_m}.\mathrm{nu} - \sum_{\forall i \in UI_{\{i_m\}-\mathrm{CPB}} \wedge i \subseteq p} \mathrm{miu}(i) \times p.\mathrm{count} \qquad (3)$$

Where,
pu is path utility of path p,
$\{i_m\} - \mathrm{CPB}$ is i_m's Conditional Pattern Base,
$N_{i_m}.\mathrm{nu}$ is the node's Utility,
$UI_{\{i_m\}} - \mathrm{CPB}$ is the set of unpromising items of $\{i_m\} - \mathrm{CPB}$,
$\mathrm{miu}(i)$ is minimum utility value of item i,
$p.\mathrm{count}$ is the count of path p

- Construct conditional tree known as local UP-Tree by using information of particular conditional pattern bases.
- Mine patterns from conditional trees or local UP-Trees

2) Decreasing local node utilities

In this strategy the minimum item utilities of descendant nodes are decreased during the construction of a local UP-Tree. Here we use minimum item utility because we don't know actual utilities of descedant nodes. Working of UP-Growth algorithm as follows

```
Subroutine: UP-Growth ($T_x$, $H_x$, X)
Input: A UP-Tree $T_x$, a header table $H_x$ and an itemsetX.
Output: All PHUIs in Tx.
Procedure UP-Growth ($T_x$, $H_x$, X)
  a. For each entry $a_i$ in $H_x$ do
  b.   Trace each node related to$a_i$ and calculated
       $a_{i.nu} = nu_{sum}(a_i)$;
       /* $nu_{sum}(a_i)$ is sum of node utilities of $a_i$ */
  c.   If $nu_{sum}(a_i) \geq$ mim_util, do
  d.     Generate a PHUI $Y = X \cup a_i$;
  e.     The estimate utility of Y is set as $a_i$'s utility value in $H_x$;
  f.     Construct Y's conditional pattern base Y -CPB;
  g.     Put local promising items in Y-CPB into $H_y$ ;
  h.     Use strategy DLU. It reduce path utilities of the paths;
  i.     Use strategy DLN and insert paths into $T_y$;
  j.     If $T_y$ != null then call UP-Growth($T_y$, $H_y$, Y );
  k.   End if
  l. End for
```

E. Module 6:UP-Growth+ algorithm[1]

UP-Tree generated from UP-construction module is also passed as input to the UP-Growth+ algorithm. Applying UP-Growth to UP-Tree takes more execution time. Therefore a newly introduced algorithm i. e. UP-Growth+ is used.

UP-Growth uses, minimum item utility table to reduce the overestimated utilities but in UP-Growth+, minimum 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 included during the construction of a Global UP-Tree. This Minimal node utility maintains minimal value of node utility in different transactions of the database, each time when that node is traced.

After constructing this modified global UP-Tree UP-Growth+ uses two improved strategies called DNU(Discarding local unpromising items)and DNN(Decreasing local node utilities).

1) Discarding local unpromising items

In DNU strategy we have to discard local unpromising items and their estimated Node Utilities from the paths and path utilities of conditional pattern bases.

When a local UP-Tree is being constructed, minimal node utilities can also be acquired similar to global UP-Tree. In this mining process, when a path is retrieved, we can calculate path utility by using minimum node utility with a formula (4)

$$\mathrm{pu}(p,\{i_m\}-\mathrm{CPB}) = N_{i_m}.\mathrm{nu} - \sum_{\forall i \in UI_{\{i_m\}-\mathrm{CPB}} \wedge i \subseteq p} \mathrm{mnu}(i) \times p.\mathrm{count} \; (4)$$

Where,
Mnu(i) is minimum node utility value of item i,

2) Decreasing local node utilities

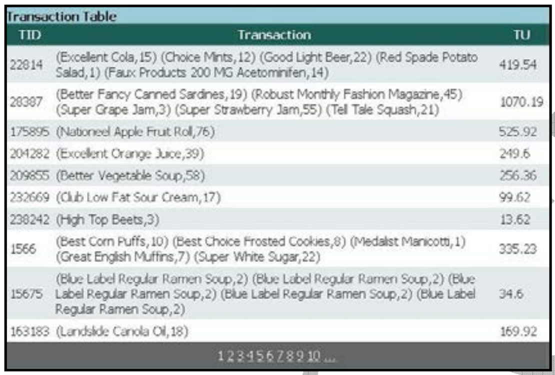
In second strategy we have to decrease local Node utilities for the nodes of local UP-Tree by estimated utilities of descendant Nodes. By applying this approach the utility of items further reduced. So it generates less PHUIs than UP-Growth.

F. Module 7: UP-Growth and UP-Growth+for dynamic Database

UP-Growth and UP-Growth+ algorithm find high utility itemsets from foodmart database. Every day new customer transaction details are added in the main database. Therefore each time these algorithms have to apply on updated database to find high utility itemsets. It scans database repeatedly. If the data is continuously added to the original transaction database, then the database size becomes larger and mining the entire lot would take more time. Our system regenerates UP-Tree for incremented dynamic database and gives high utility itemsets.

V. Experimental Result

Performance of the proposed algorithms is evaluated in this section. These experiments are performed on a 3.10 GHz Intel® Core™ i3-2100 Processor with 4 GB memory. Microsoft Windows 7 operating system is used. These algorithms are implemented in C# language. Microsoft Food mart dataset is used in these experiments. This database contains 2.5lac transactions and 1500 distinct items.

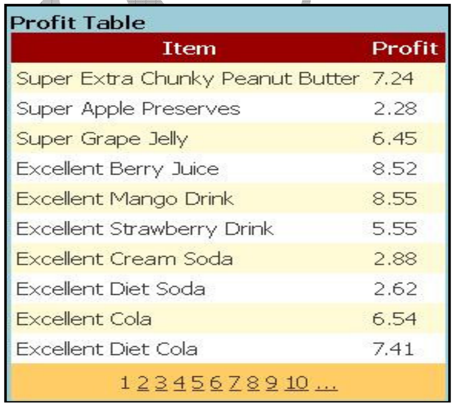

Transaction Table

TID	Transaction	TU
22814	(Excellent Cola,15) (Choice Mints,12) (Good Light Beer,22) (Red Spade Potato Salad,1) (Faux Products 200 MG Acetominifen,14)	419.54
28387	(Better Fancy Canned Sardines,19) (Robust Monthly Fashion Magazine,45) (Super Grape Jam,3) (Super Strawberry Jam,55) (Tell Tale Squash,21)	1070.19
175895	(Nationeel Apple Fruit Roll,76)	525.92
204282	(Excellent Orange Juice,39)	249.6
209855	(Better Vegetable Soup,58)	256.36
232669	(Club Low Fat Sour Cream,17)	99.62
238242	(High Top Beets,3)	13.62
1566	(Best Corn Puffs,10) (Best Choice Frosted Cookies,8) (Medalist Manicotti,1) (Great English Muffins,7) (Super White Sugar,22)	335.23
15675	(Blue Label Regular Ramen Soup,2) (Blue Label Regular Ramen Soup,2) (Blue Label Regular Ramen Soup,2) (Blue Label Regular Ramen Soup,2) (Blue Label Regular Ramen Soup,2)	34.6
163183	(Landslide Canola Oil,18)	169.92

1 2 3 4 5 6 7 8 9 10 ...

Fig. 2. Snapshot of Transaction Table

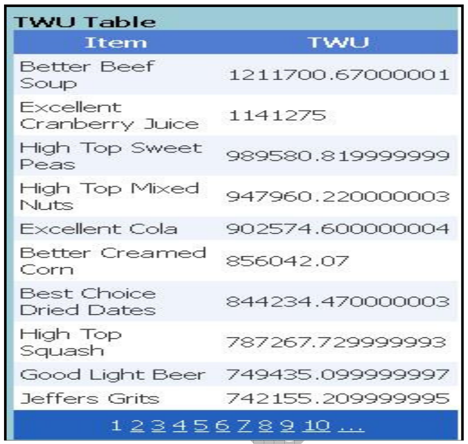
Fig. 2 shows transaction database which contains item name, quantity. Transaction utilities of all transactions are computed which displayed in last column. Fig. 3 shows 1500 items and their profit values.


Profit Table

Item	Profit
Super Extra Chunky Peanut Butter	7.24
Super Apple Preserves	2.28
Super Grape Jelly	6.45
Excellent Berry Juice	8.52
Excellent Mango Drink	8.55
Excellent Strawberry Drink	5.55
Excellent Cream Soda	2.88
Excellent Diet Soda	2.62
Excellent Cola	6.54
Excellent Diet Cola	7.41

1 2 3 4 5 6 7 8 9 10 ...

Fig. 3. Snapshot of Profit Table

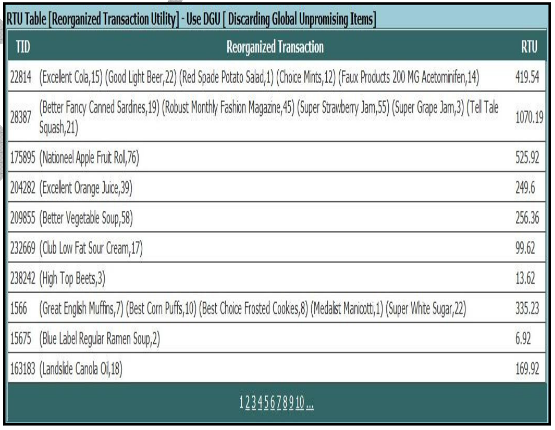

TWU Table

Item	TWU
Better Beef Soup	1211700.67000001
Excellent Cranberry Juice	1141275
High Top Sweet Peas	989580.819999999
High Top Mixed Nuts	947960.220000003
Excellent Cola	902574.600000004
Better Creamed Corn	856042.07
Best Choice Dried Dates	844234.470000003
High Top Squash	787267.729999993
Good Light Beer	749435.099999997
Jeffers Grits	742155.209999995

1 2 3 4 5 6 7 8 9 10 ...

Fig. 4 Snapshot of TWU Table

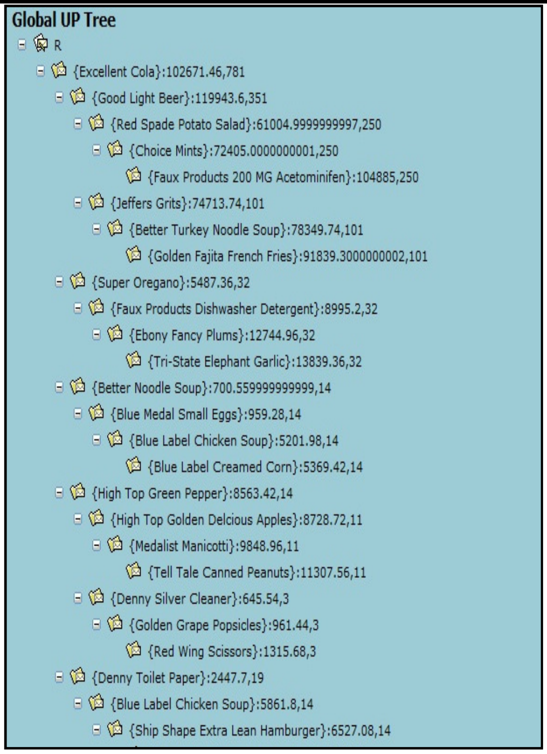
Transaction weighted utilities of all items are calculated and shown in fig. 4. Figure 5 shows the reorganized database after removing the unpromising items and their utilities.


RTU Table [Reorganized Transaction Utility] - Use DGU [Discarding Global Unpromising Items]

TID	Reorganized Transaction	RTU
22814	(Excellent Cola,15) (Good Light Beer,22) (Red Spade Potato Salad,1) (Choice Mints,12) (Faux Products 200 MG Acetominifen,14)	419.54
28387	(Better Fancy Canned Sardines,19) (Robust Monthly Fashion Magazine,45) (Super Strawberry Jam,55) (Super Grape Jam,3) (Tell Tale Squash,21)	1070.19
175895	(Nationeel Apple Fruit Roll,76)	525.92
204282	(Excellent Orange Juice,39)	249.6
209855	(Better Vegetable Soup,58)	256.36
232669	(Club Low Fat Sour Cream,17)	99.62
238242	(High Top Beets,3)	13.62
1566	(Great English Muffins,7) (Best Corn Puffs,10) (Best Choice Frosted Cookies,8) (Medalist Manicotti,1) (Super White Sugar,22)	335.23
15675	(Blue Label Regular Ramen Soup,2)	6.92
163183	(Landslide Canola Oil,18)	169.92

1 2 3 4 5 6 7 8 9 10 ...

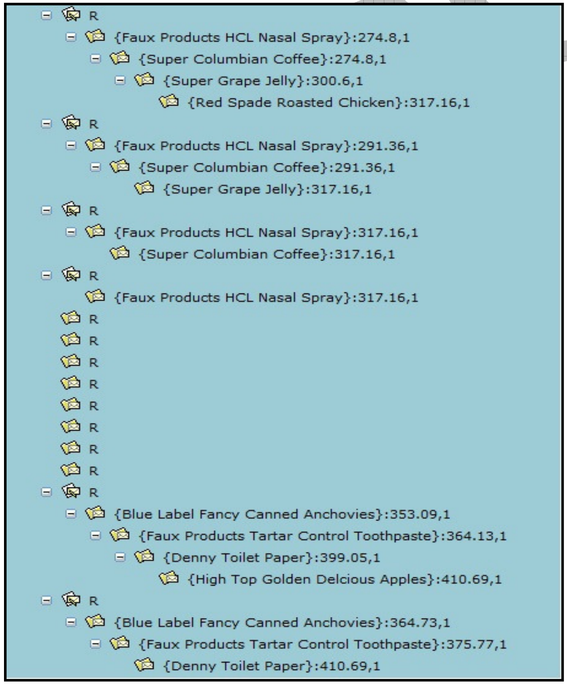
Fig. 5 Reorganized Transaction Table



```
Global UP Tree
⊟ R
  ⊟ {Excellent Cola}:102671.46,781
    ⊟ {Good Light Beer}:119943.6,351
      ⊟ {Red Spade Potato Salad}:61004.9999999997,250
        ⊟ {Choice Mints}:72405.0000000001,250
            {Faux Products 200 MG Acetominifen}:104885,250
      ⊟ {Jeffers Grits}:74713.74,101
        ⊟ {Better Turkey Noodle Soup}:78349.74,101
            {Golden Fajita French Fries}:91839.3000000002,101
    ⊟ {Super Oregano}:5487.36,32
      ⊟ {Faux Products Dishwasher Detergent}:8995.2,32
        ⊟ {Ebony Fancy Plums}:12744.96,32
            {Tri-State Elephant Garlic}:13839.36,32
    ⊟ {Better Noodle Soup}:700.559999999999,14
      ⊟ {Blue Medal Small Eggs}:959.28,14
        ⊟ {Blue Label Chicken Soup}:5201.98,14
            {Blue Label Creamed Corn}:5369.42,14
    ⊟ {High Top Green Pepper}:8563.42,14
      ⊟ {High Top Golden Delcious Apples}:8728.72,11
        ⊟ {Medalist Manicotti}:9848.96,11
            {Tell Tale Canned Peanuts}:11307.56,11
      ⊟ {Denny Silver Cleaner}:645.54,3
        ⊟ {Golden Grape Popsicles}:961.44,3
            {Red Wing Scissors}:1315.68,3
    ⊟ {Denny Toilet Paper}:2447.7,19
      ⊟ {Blue Label Chicken Soup}:5861.8,14
        ⊟ {Ship Shape Extra Lean Hamburger}:6527.08,14
```

Fig. 6 Sample of Global UP-Tree (using UP-Growth algorithm)

Above fig. 6 shows sample of Global UP-Tree for UP-Growth. This tree is very large. UP-Tree is constructed by using reorganized transactions and it requires just two scan of database. The items in a transaction are rearranged in descending order of the global TWU during the construction of UP-Trees.

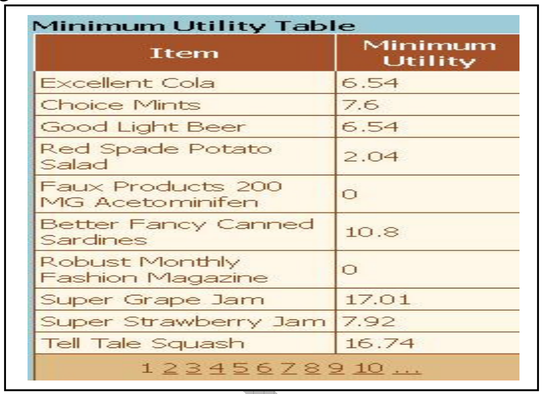


```
⊟ R
  ⊟ {Faux Products HCL Nasal Spray}:274.8,1
    ⊟ {Super Columbian Coffee}:274.8,1
      ⊟ {Super Grape Jelly}:300.6,1
          {Red Spade Roasted Chicken}:317.16,1
⊟ R
  ⊟ {Faux Products HCL Nasal Spray}:291.36,1
    ⊟ {Super Columbian Coffee}:291.36,1
        {Super Grape Jelly}:317.16,1
⊟ R
  ⊟ {Faux Products HCL Nasal Spray}:317.16,1
      {Super Columbian Coffee}:317.16,1
⊟ R
    {Faux Products HCL Nasal Spray}:317.16,1
  R
  R
  R
  R
  R
  R
  R
  R
⊟ R
  ⊟ {Blue Label Fancy Canned Anchovies}:353.09,1
    ⊟ {Faux Products Tartar Control Toothpaste}:364.13,1
      ⊟ {Denny Toilet Paper}:399.05,1
          {High Top Golden Delcious Apples}:410.69,1
⊟ R
  ⊟ {Blue Label Fancy Canned Anchovies}:364.73,1
    ⊟ {Faux Products Tartar Control Toothpaste}:375.77,1
        {Denny Toilet Paper}:410.69,1
```

Fig. 7 Sample of Local UP-Tree (using UP-Growth algorithm)

Fig. 7 shows sample of Local UP-Tree generated from global UP-Tree by UP-Growth algorithm.

UP-Growth algorithm uses minimum utility table as shown in fig. 8.



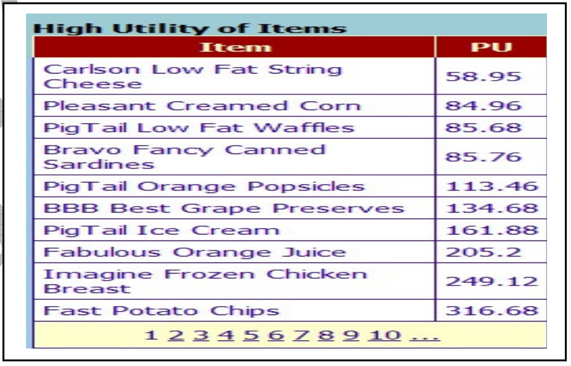
Minimum Utility Table

Item	Minimum Utility
Excellent Cola	6.54
Choice Mints	7.6
Good Light Beer	6.54
Red Spade Potato Salad	2.04
Faux Products 200 MG Acetominifen	0
Better Fancy Canned Sardines	10.8
Robust Monthly Fashion Magazine	0
Super Grape Jam	17.01
Super Strawberry Jam	7.92
Tell Tale Squash	16.74

1 2 3 4 5 6 7 8 9 10 ...

Fig. 8. Mininum utility Table

Following fig. 9 shows the High utility itemsets which are mined by using UP-Growth. There are 2020 high utility itemsets are generated from 2.5 lac transactions.



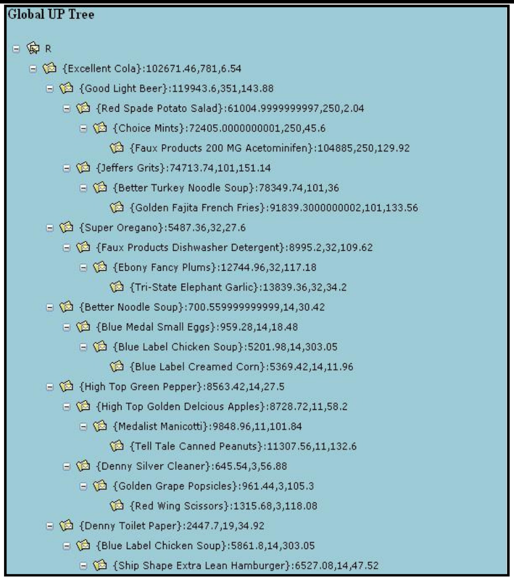
High Utility of Items

Item	PU
Carlson Low Fat String Cheese	58.95
Pleasant Creamed Corn	84.96
PigTail Low Fat Waffles	85.68
Bravo Fancy Canned Sardines	85.76
PigTail Orange Popsicles	113.46
BBB Best Grape Preserves	134.68
PigTail Ice Cream	161.88
Fabulous Orange Juice	205.2
Imagine Frozen Chicken Breast	249.12
Fast Potato Chips	316.68

1 2 3 4 5 6 7 8 9 10 ...

Fig. 9. Snapshot of High Utility Itemsets Table(UP-Growth)

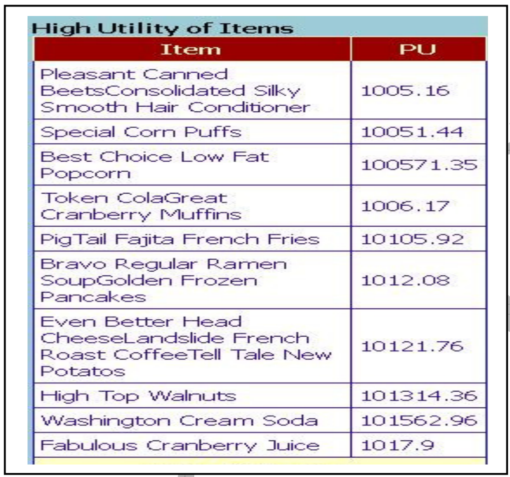
Fig. 10 shows Global UP-Tree for UP-Growth+ which contains node utilities of each node. Fig. 11 shows high utility itemsets generated by UP- Growth+. There are 1906 high utility itemsets are generated.



```
Global UP Tree

R
  {Excellent Cola}:102671.46,781,6.54
    {Good Light Beer}:119943.6,351,143.88
      {Red Spade Potato Salad}:61004.9999999997,250,2.04
        {Choice Mints}:72405.0000000001,250,45.6
          {Faux Products 200 MG Acetominifen}:104885,250,129.92
      {Jeffers Grits}:74713.74,101,151.14
        {Better Turkey Noodle Soup}:78349.74,101,36
          {Golden Fajita French Fries}:91839.3000000002,101,133.56
    {Super Oregano}:5487.36,32,27.6
      {Faux Products Dishwasher Detergent}:8995.2,32,109.62
        {Ebony Fancy Plums}:12744.96,32,117.18
          {Tri-State Elephant Garlic}:13839.36,32,34.2
    {Better Noodle Soup}:700.559999999999,14,30.42
      {Blue Medal Small Eggs}:959.28,14,18.48
        {Blue Label Chicken Soup}:5201.98,14,303.05
          {Blue Label Creamed Corn}:5369.42,14,11.96
    {High Top Green Pepper}:8563.42,14,27.5
      {High Top Golden Delcious Apples}:8728.72,11,58.2
        {Medalist Manicotti}:9848.96,11,101.84
          {Tell Tale Canned Peanuts}:11307.56,11,132.6
      {Denny Silver Cleaner}:645.54,3,56.88
        {Golden Grape Popsicles}:961.44,3,105.3
          {Red Wing Scissors}:1315.68,3,118.08
    {Denny Toilet Paper}:2447.7,19,34.92
      {Blue Label Chicken Soup}:5861.8,14,303.05
        {Ship Shape Extra Lean Hamburger}:6527.08,14,47.52
```

Fig. 10 Snapshot of Global UP-Tree(using UP-Growth+ algorithm)



High Utility of Items

Item	PU
Pleasant Canned BeetsConsolidated Silky Smooth Hair Conditioner	1005.16
Special Corn Puffs	10051.44
Best Choice Low Fat Popcorn	100571.35
Token ColaGreat Cranberry Muffins	1006.17
PigTail Fajita French Fries	10105.92
Bravo Regular Ramen SoupGolden Frozen Pancakes	1012.08
Even Better Head CheeseLandslide French Roast CoffeeTell Tale New Potatos	10121.76
High Top Walnuts	101314.36
Washington Cream Soda	101562.96
Fabulous Cranberry Juice	1017.9

Fig. 11. Snapshot of High Utility Itemsets Table (UP-Growth+ algorithm)

VI. CONCLUSION

UP Growth and UP-Growth+ are efficient for high utility itemsets mining. These techniques generate high utility itemsets within two scans of the transactional database. These algorithms works better if one or more transactions are deleted or inserted in transaction database than existing algorithm. Existing algorithms has work on only static transactions but our system works on dynamic database. Our experiments shows that performance of UP-Growth+ algorithm is better than UP-Growth. We conclude that UP-Growth+ algorithm required less execution time than UP-Growth. It also produced less high utility itemsets than UP-Growth.

REFERENCES

[1] Vincent S. Tseng, Bai-En Shie, Cheng-Wei Wu, and Philip S. Yu, Fellow, IEEE "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases" IEEE Trans. Knowledge and Data Engineering, vol. 25, no. 8, August 2013
[2] ChowdhuryFarhan Ahmed, Syed KhairuzzamanTanbeer, Byeong-SooJeong, and Young-Koo Lee, Member, IEEE "Efficient Tree Structures for High Utility Pattern Mining inIncremental Databases" IEEE Trans. Knowledge and Data Engineering, Vol. 21, Issue 12, pp. 1708-1721, December 2009.
[3] Alva Erwin, Raj P. Gopalan, and N. R. Achuthan, "Efficient Mining of High Utility Itemsets from Large Datasets", In Proc. of PAKDD 2008.
[4] Shankar, S.; Purusothaman, T.; Jayanthi, S. "Novel algorithm for mining high utility itemsets" International Conference on Computing, Communication and Networking, Dec. 2008.
[5] Raymond Chan; Qiang Yang; Yi-Dong Shen, "Mining high utility itemsets" In Proc. of Third IEEE Int'l Conf. on Data Mining ,November 2003.
[6] Ramaraju, C., Savarimuthu N. "A conditional tree based novel algorithm for high utility itemset mining", International Conference on Data mining, June 2011.
[7] Ying Liu, Wei-keng Liao, AlokChoudhary "A Fast High Utility Itemsets Mining Algorithm" In Proc. of the Utility-Based Data Mining Workshop, 2005.
[8] AdinarayanareddyB ,OSrinivasaRao, MHM Krishna Prasad, "An Improved UP-GrowthHigh Utility Itemset Mining" International Journal of Computer Applications (0975-8887) Volume 58-No.2, November 2012.
[9] P. Asha, Dr. T. Jebarajan, G. Saranya, "A Survey on Efficient Incremental Algorithm for Mining High Utility Itemsets in Distributed and Dynamic Database" IJETAEJournal, Vol.4, Issue 1, January 2014.