Mining High Utility Pattern in One Phase without Candidate Generation using up Growth+ Algorithm

P.Sri Varshini[#1], N.Saranya.N[*2], Uma Maheswari[#3], Prof.R.Sujatha[#4]

[#1]*B.E Department of Computer Science and Engineering, Sri Krishna College Of Technology, Kovaipudur, Coimbatore, India-641042*

[*2]*B.E Department of Computer Science and Engineering, Sri Krishna College Of Technology, Kovaipudur, Coimbatore, India-641042.*

[#3]*B.E Department of Computer Science and Engineering, Sri Krishna College Of Technology, Kovaipudur, Coimbatore, India-641042.*

[#4]*Professor, Department of Computer Science and Engineering, Sri Krishna College Of Technology, Kovaipudur, Coimbatore, India-641042.*

Abstract—*Utility mining developed to address the limitation of frequent itemset mining by introducing interestingness measures that satisfies both the statistical significance and the user's expectation. Existing high utility itemsets mining algorithms two steps: first, generate a large number of candidate itemsets and second, identify high utility itemsets from the candidates by an additional scan of the original transaction database. The performance holdup of these algorithms is the generate more no of candidates itemsets and increasing of the number of long transaction itemsets it cannot work minimum utility threshold, the situation may become worse and also creating more no tree. To overcome these problems, propose an efficient algorithm, namely UP-Growth (Utility Pattern Growth), for mining high utility itemsets with pruning techniques for pruning candidate itemsets. The information of high utility itemsets is stored in a special data structure named UP-Tree (Utility Pattern Tree) such that the candidate itemsets can be generated with only two scans of the database. The performance of UP growth+ was evaluated in comparison with the state-of-the-art algorithms on different types of datasets. The experimental results show that UP growth+ outperforms other algorithms in terms of both execution time and memory space under minimum utility threshold is, the more observable its advantage will be it can achieve the level of about two orders of magnitude faster than the state-of-the-art algorithms on dense dataset, and more than one order of magnitude on sparse datasets.*

Keywords—*Utility Pattern Growth, UP Tree, High Utility mining, reducing search space, Pruning.*

I. INTRODUCTION

Data mining is the process of extracting itemsets from the database that is useful for decision making process to increase awareness and market analysis. One of the important tasks is high utility mining which refers to the discovery of more profitable things. Mining the high utility itemset from the transaction database when the utility of an item is greater than or equal to user specified minimum utility threshold then item is profitable.

For example, assume the frequency of item A is 10, item B is 5, and itemset AB is 3. The profit of item A is 3, B is 2. The utility value of A is 10 * 3 = 30, B is 5* 2= 10, and AB is 3 * 3 + 5 * 2 = 19. If the minimum utility threshold is 20, A is a high utility itemset a number of algorithms have been proposed for high utility itemset mining. The existing algorithm based on two phase [1], TWU mining [2], UMMI algorithm [3]. Two phase algorithm mines the high utility itemset in two phases using the transaction weighted down closure property in phase I to find the HTWU (High Transaction Weighted Utility) item. In Phase II the high utility itemset is mined from first phase output of HTWU item. The TWU mining method uses a tree structure to store utility itemsets information. The exiting UMMI algorithm is introduced to overcome the shortcoming of two phases mining, TWU mining which consists of large number of HTWU itemsets hence it take more execution time. The High utility itemsets can be mined using two phase's algorithms: maximal phase and utility phase. Maximal phase mines the maximal transaction weighted utility (MTWU) item using maximal itemset property and utility phase discover the high utility itemset from MTWU item used Mlex tree. In exiting algorithm have some weakness to mining high utility itemset: a) Itemset with profit slightly less than the user defined threshold value is discarded. For example, if the minimum utility threshold is 40, itemsets with profit 30 will be pruning the lower threshold utility itemsets even though the itemset may be important. b). Quantity information about the high utility itemset is not reflect that a sales manager is interested. c) Profit quantity information is also not provided in the existing algorithm. These shortcomings influence the

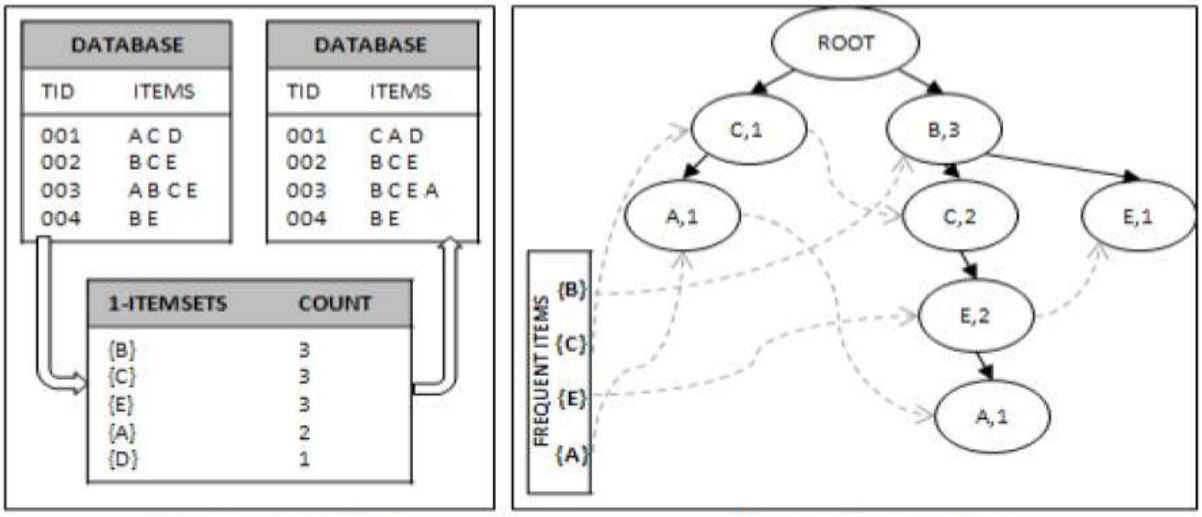
ability to select the more profitable and cost saving products.

II. EXISTING TECHNIQUE

The FP-Growth method adopts a divide and conquers strategy as follows: compress the database representing frequent items into a frequent-pattern tree, but retain the item set association information, and then divide the compressed database into a set of condition databases, each associated with one frequent item, and mine each such databases.

Firstly, database is read and frequent items are found which are the items are occurring in transactions less than minimum support. Secondly database is read again to build FP-tree. After creating the root, every transaction is read in an ordered way and pattern of frequent items in the transaction is added to FP-tree and nodes are connected to frequent items list and each other. This interconnection makes frequent pattern search faster avoiding the traversing of the entire tree. When considering the branch to be added for the transaction, the count of each and every node along the common prefix is incremented by 1. Nodes of same items are interconnected where most left one is connected to item in frequent items list. If the prefix of branch to be added does not exists then it is added as a new branch to root.

After constructing the tree mining proceeds as follows. Start from each frequent length 1 pattern, construct its conditional pattern base, then construct its conditional FP-tree and perform mining recursively on such tree. The support of a candidate (conditional) item set is counted traversing the tree thoroughly. The sum of count values at least frequent item's nodes i.e., base node gives the support value.



(a) First scan of database

(b) FP-Tree after second scan

ITEM	COND. PAT. BASE	COND. FP-TREE	FP GENERATED
A	(C:1), (BCE:1)	(C:2)	CA:2
E	(BC:2), (B:1)	(BC:2), (B:3)	BCE:2, CE:2, BE:3
C	(B:2)	(B:2)	BC:2

(c) Frequent pattern generation

Figure 1.1 FP-Growth Example

III. LITERATURE REVIEW

In many existing algorithms that are often unclear and also inaccurate to find the high utility itemsets from long transaction itemsets. Fuzzy logic is supportive to real life datasets that deal with uncertainty as it gives a flexible method to form a high-level concept of given problem. Fuzzy logic and data mining together present a means for generating more conceptual utility mining at a higher level. Ferdinando et al. presented a method for finding association rules from datasets based on fuzzy transforms technique.

Apriori Gen algorithm [6] was used to extract fuzzy association rules represented in the form of linguistic terms. A pre-processing phase was used to determine the optimal fuzzy partitions of quantitative attribute domains.

Utility Mining covers all aspects of "profitable" utilities that helps in detecting rare high utility itemsets from the transaction. High Utility Rare

Itemset Mining (HURI) is very useful in several real-life datasets. In Jyothi et al.presented a survey of different approaches and algorithms for mining the high-utility itemsets and rare itemset mining. Ashish et al [8] presented a fast and efficient fuzzy ARM algorithm on large datasets. The algorithm was 8 to 19 times faster than traditional fuzzy ARM on large standard datasets. In [2], unlike most two-phased ARM algorithms, the authors presented single itemset processing as opposite to concurrent itemset processing at each k no of level, recording some performance improvements.

In C. Saravanabhavan et al. proposed a tree based mining high utility itemsets. Firstly, the authors developed the utility frequent pattern tree structure to maintain the information of utility itemsets. Next the pattern growth methodology was used to mine the utility pattern sets entirely. Two algorithms, UP-Growth (Utility Pattern Growth) and UP-Tree (Utility Pattern Tree) are proposed [14] for mining high utility itemsets. Also, a set of effective strategies are discussed by SadakMurali et al, proposed for precisely pruning candidate itemsets.

In Adinarayanareddy.B, was proposed a modified UP-Growth (IUPG) algorithm for high utility itemset mining. The authors conclude that IUPG algorithm performs better than UP-Growth algorithm for different threshold values and also IUPG algorithm is highly scalable. Ruchi Patel was proposed a parallel and distributed method for mining the high utility patterns from very large transaction databases. The method also prunes the low utility itemsets from transactions at the starting level by using downward closure property.

Koh et al was proposed a modified apriori inverse algorithm to generate rare itemsets of user interest. The high utility itemsets as a measure usefulness or profitability of an itemset [12].These authors paying attention on the measures used for mining the utility-based itemset. Utility based measures uses the utilities of the patterns to reflect the user goals. The authors makes the official semantic meaning of utility measures and classify existing measures into one of the three categories: item level, transaction level and cell level. Existing system fused framework was proposed for incorporating utility based measures into the data mining process via a joined utility function.

One of the most needed areas of the application of fuzzy set theory is fuzzy rule-based systems [4]. These knowledge extraction tools discovers the essential associations contained in a database. Fuzzy techniques improves the understandability of buyer models. In Casillas et al presented a new approach for buyer behaviour modelling which is based on fuzzy association rules (FAS). A behavioural model was presented which cantered on consumer attitude towards shopping itemsets. The proposed algorithm efficiently finds the high utility itemsets and pruning the lower utility itemset if used defined threshold value is low.paragraphs must be indented.

.

IV. TERMS AND DEFINITIONS

Let D = {T1, T2… Tn} be a transaction dataset which contains n transaction itemsets and m distinct items, i.e. I= {i1, i2… im}. Each transaction itemset is represented as {i1: p1, i2: p2… iv:pv}, where {i1, i2, …, iv} is a subset of I, and pu (1≤u≤v) is the existential probability of item iuin a transaction itemset. The size of dataset D is the number of transaction itemsets and is denoted as |D|. An itemset X = {i1, i2… ik}, which contains k dissimilar items, is called a k-itemset, and k is the length of the itemset X.

Definition 1: The support value (SV) of an itemset X in a transaction dataset is defined by number of transaction itemsets containing X in it.

Definition 2: The expected support value(expSV) of an itemset X in a transaction dataset is denoted as expSN(X) and is defined by

$$expSV(X) = \sum_{T_d \supseteq X, T_d \in D} P(X, T_d) \quad (1)$$

Definition 3: Given a dataset D, the minimum expected support threshold $\tau$is a predefined percentage of |D|; correspondingly, the minimum expected support value (minExpSV) is defined by

$$minExpSV = |D| \times \tau (2)$$

anitemset X is called a utility itemset if its expected support value is not less than the value minExpSV.

Definition 4: The minimum support threshold $\tau$is a predefined percentage of |D|; correspondingly, the minimum support value (minSV) in a dataset D is defined by

$$minSV = |D| \times \tau (3)$$

Definition 5: The system mining high utility itemset and itemsets utility is equal to quantity value multiplied by profit. For example, item A occurs in different transaction T1 and T5. If the set of (A, T1) = {1/L, 0/M, 1/H} and (A, T5) = {0/L, 0/M, 1/H}. If their item utilities are obtaining by multiplying quantity with profit, item utilities of T1 and T5 will be same even though T5 defer a higher quantity value. The utility is defined as,

$$u(i_p, T_q) - f(i_p, T_q) \times S(i_p) (3)$$

Definition 6: The quantity The equation to find the quantity value of item ip in transaction Tq is denoted as f (ip,Tq) which is defined as,

$$U(i_p, T_q) - \sum q(i_p, T_q) \times weight(j) \ (4)$$

Where fq (ip,Tq, j) is the value of region j and weight (j) is a variable parameter defined by the region. If a region is low, then the weight should be low when compared to the region middle and high. Particular weights are assigned for area low, medium and high.

Definition 7: The transaction utility can be defined as the sum of the utilities of item occurring in the transaction. The equation denoting transaction utility is

$$tu(T_q) - \sum_{i_p \in T_q} u(i_p, T_q) (5)$$

Definition 8: The transaction weighted utilityit is the sum of transaction utilities of item occurring in the particular transaction for an item. The equation denoting transaction weighted utility can be defined as,

$$twu(x) - \sum_{xcT_q \in D} tu(T_q) (6)$$

V. PROPOSED SYSTEM

To improve the mining performance and to avoid scanning original database repeatedly, we use a compact tree structure, named UP-Tree, and to maintain the information of transactions and high utility itemsets. Two strategies are used to minimize the overestimated utilities stored in the nodes of global UP-Tree. In following sections, the elements of UP-Tree are first defined. Next, the two strategies are introduced.

A. The Elements in UP-Tree

Each node D consists of D.name, D.count, D.nu, D.parent, D.hlink and a set of child nodes. D.name is the item name of the node. D.count is the support count of that particular node. D.nu is the node utility, i.e., overestimated utility of the node. D.parent maintains the parent node of D. D.hlink is a node link which points to a node whose item name is the same as D.name. A table named header table is used to facilitate the traversal of UP-Tree. In header table, each entry stores an item name, an overestimated utility, and a link. The link points to the last occurrence of the node which has the same item as the entry. By following the links in header table and the nodes in UP-Tree, the nodes having the same name can be traversed in a better way.

B. Discarding Global Unpromising Items during Constructing a Global UP-Tree

The construction of a global UP-Tree can be done with two scans of the original database. In the first scan, TU of each transaction is calculated. simultaneously, TWU of each single item is also accumulated. By TWDC property, an item and its supersets are unpromising itemsets if its TWU is less than the minimum utility threshold. Such an item is called an unpromising item.

C. Discarding Global Unpromising Items during Constructing a Global UP-Tree

The construction of a global UP-Tree can be done with two scans of the original database. In the first scan, TU of each transaction is calculated. simultaneously, TWU of each single item is also accumulated. By TWDC property, an item and its supersets are unpromising itemsets if its TWU is less than the minimum utility threshold. Such an item is called an unpromising item.The tree-based framework for high utility itemset mining applies the divide-and-conquer technique in mining processes. Thus, the search space can be divided into smaller subspaces. By using strategy DGN, the utilities of the nodes that are closer to the root of a global UP-Tree are further decreased. DGN is suitable for the databases containing more number of long transactions. In other words, the more items a transaction has, the more utilities can be removed by DGN. On the other hand, traditional TWU mining strategy is not suitable for such databases since the more items a transaction contains, the higher TWU is.

D. Up Growth Plus Algorithm

UP-Growth is efficient than FP-Growth by using DLU and DLN to reduce overestimated utilities of itemsets. However, the overestimated utilities can be nearer to their actual utilities by eliminating the calculated utilities that are closer to the actual utilities of unpromising items. An improved method, named UP-Growth+, for reducing over estimated utilities more effectively is introduced. In UP-Growth, minimum item utility table is used to decrease the overestimated utilities. In UP-Growth+, minimal node utilities in each path are used to make estimated pruning values closer to real utility values of the pruned items.

After introducing the modification of global UP-Tree, the processes and two improved strategies of UP-Growth+, named DNU and DNN are performed. When a local UPTree is being constructed, minimal node utilities can also be mined by the same steps of global UP-Tree. In the mining process, when a path is retrieved, simultaneously minimal node utility of each node in the path is also retrieved.

Create Tree (D,τ)

INPUT: An uncertain dataset D consisting of n transaction itemsets and a predefined user minimum expected support thresholdτ.

OUTPUT: A global Tail Node-Tree *T*.

Step 1: Calculate the minimum expected support value *minExpSV*, i.e. $minExpSV = |D| \times \tau$; count the expected support value and support value of each transaction item by one scan of dataset.
Transaction utility can be defined as the sum of the utilities of item occurring in the transaction.
Transaction weighted utility it is the sum of transaction utilities of item occurring in the particular transaction for an item.
Step 2: place those items whose expected support value are not less than *minExpSV*to a global header table, and sort the items in the header table according to the descending order of their support value.
Step 3: Initially set the root node of the Tree *T* as null.
Step 4: Remove the items that are not in the header table from each transaction itemset, and sort the remaining items of each transaction itemset according to the order of the global header table, and get a sorted itemset*A*.
Step 5: If the length of itemset*A* is 0, process the next transaction itemset; otherwise insert the itemset*A* into the tail node tree *T* by the following steps:
a)Store the weight probability value of each item in itemset*A* sequentially to a list; save the list to an array (which is denoted as ***ProArr***); the equivalent series number of the list in the array is denoted as *ID*.
b) If there has not been a tail node for the itemset*X*, create a tail node *N* for this itemset, where *N.Tail_info.len*is the length of itemset*A*, and *N.Tail_info.Arr_ind*= (*ID*); otherwise, append the sequence number *ID* to *N.Tail_info.Arr_ind*.
Step 6: Process the next transaction itemset based on above steps until meet all the transaction from transaction weight dataset.

E. Mining high utility itemsets from a global tail node tree

After a tail node tree is constructed, the algorithm tail node tree can directly mine high utility itemsets from the tree without additional scan of dataset. The details of the mining techniques are described below. The algorithm TN-Mine is similar to the algorithm UP growth: it creates and processes sub trees recursively.

Mining high utility (*T*, G*H*, *minExpSV***)**

INPUT: An FTNT-Tree *T*, a global header table *H*, and a Minimum expected support value *minExpSV*.

OUTPUT: The high utility itemsets (HUIs).

Step 1: find the high utility items from header table one by one from the last item by the following steps.
Step 2: choose the current *base-itemset*(which is initialized as null); each new *base-itemset*is a high utility itemset.
Step 3: Example consider base item is "*Z*"Let*Z.link*sin the header table *H* contain *k* nodes whose item name is *Z*; we denote these *k* nodes as *N*1, *N*2, …, *Nk*; because item *Z* is the last one in the header table, all these *k* nodes are *tail nodes*, i.e., each of these nodes contains a *Tail_info*.The following sub steps.
a) Create a sub header table *subH*by scanning the *k* *base item* branches from these *k* nodes to the root.
b) Suppose the sub header table is zero, go to Step 4.
c): Create sub Tree each and every base item *subTree*= **CreateSubTree** (*Z.link*, *subH*).
Mining (*subTree*, *subH*, *minExpSV*).
Step 4: After find the high utility of that base item *Z* and remove that item from header table.
Step 5: For each of these *k* nodes denote as $ni \le N \le k$), modify its*Tail_info*by the following sub steps:
a) Alter *Ni.Tail_info.len*values: *Ni.Tail_info.len*= *Ni.Tail_info.len*-1.
b) Move *Ni.Tail_info*to the parent of node *Ni*.
Step 6: Process the next item of the header table *H*.

Subprograms:**CreateSubTree**(*all_baseitem_link*, *subH*)

INPUT: A list *link* which records tree nodes with the same item name, and a header table *subH*.

OUTPUT: A high utility itemsets from *subT*.

Step 1: Initially set the root node of the tree *subT*as zero.
Step 2: Process each node N_i in the list *link* by the following steps.
Step 3: Get the *tail-node-itemset*of node $N_i$from X item sets.
Step 4: Remove those items that are not in the header table *subH*from itemset*X*, and sort the remaining items in itemset*X* according to the order of the header table *subH*.
Step 5: If the length of the sorted itemsetk_i is 0, process the next node of the list *link*; otherwise insert the sorted itemset*X* into the FT-Tree *subT*by the following sub steps:
a) Get the original sequential *ID* of each item of the itemset*X* in the corresponding list of*ProArr*: *item_ind*= {*d*1, *d*2, *d*k} where *k* is the length of itemset*X*
b) Make a copy of *N.Tail_info*; denote the copy as *nTail_info*.
b): Alter *nTail_info*as the following:
(1) *nTail_info.len*= *k*.
(2) *nTail_info. Item_ind*= *item_ind*.
(3) if *nTail_info.bp*is null, set *nTail_info.bp*[*j*] to be the probability of item *Z*, i.e. *ProArr*[*nTail_info.Arr_ind*[*j*]]; otherwise, set *nTail_info.bp*[*j*] to be the product of *nTail_info.bp*[*j*] and the probability of item *Z* ($1 \le j \le$ *fuzzy_bp.size;* the array *ProArr*is created when the global tree is created.

VI.RESULT

The performance of the proposed algorithm UP growth plus-Mine. UP Growth is the state-of-the-art algorithm employing the pattern-growth approach and

FTNT is a new proposed algorithm. So compare UP growth plus-Mine with the algorithms FUF-Growth, UP-Mine and Fuzzy based UP growth plus on both types of datasets: Sparse transaction datasets and dense transaction datasets. All algorithms were written in java programming language. The configuration of the testing platform is as follows: Windows 7 32bit operating system, 4G Memory, Intel(R) Dual-Core CPU @ 2.60 GHz.

Table-1 Dataset Characteristics

Dataset	*|D|*	T	|I|	Type
T10I4d100k	300,000	33.8	1000	sparse
Mushroom	8,124	23	119	dense

Table 1 shows the characteristics of 4 datasets used in our experiments. '|D|' represents the total number of transactions; '|I|' represents the total number of distinct items; 'I' represents the mean length of all transaction itemsets; 'SD' represents the degree of sparsely or density. The synthetic dataset *T10I4d100k*came from the IBM Data Generator and the datasets and *mushroom* were obtained from FIMI Repository. These four datasets originally do not provide probability values for each item of each.

Table 2. Comparison algorithm of using*T10I4d100k* Vs No tree created.

Algorithms	**No of Tree**					
	0.04	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
UF-Growth +	369	115	39	17	8	4
UP Growth	410	232	101	52	43	21

FIG.1. COMPARISON OF DIFFERENT THRESHOLD VALUE WITH TREE CREATION

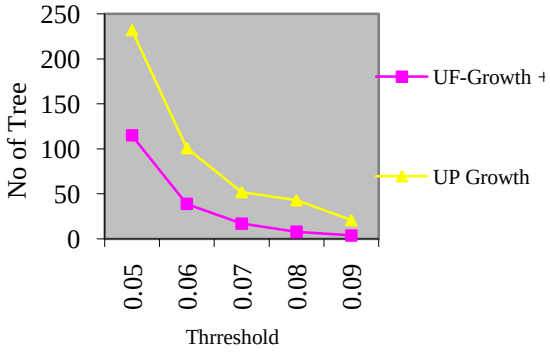


Fig 1 and Table 1 Show the total number of tree nodes generated by UF-Growth and UP, respectively, on the synthetic datasets.

Table 2. Comparison of different algorithm using *mushroom*

algorithms	No of Tree					
	0.04	**0.05**	**0.06**	**0.07**	**0.08**	**0.09**
UF-Growth +	443	345	234	189	132	112
UP Growth	546	434	367	264	212	187

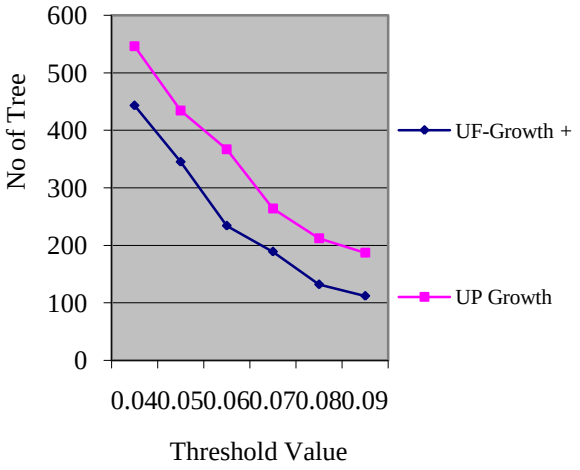


Fig. 2. Comparison of different threshold value with tree creation

Fig 2 and Table 2 Show the total number of tree nodes generated by UF-Growth and UP, respectively, on the synthetic datasets.

VII.CONCLUSION

The proposed two efficient algorithms named UP-Growth and UP-Growth+ for mining high utility itemsets from transaction databases. A data structure named UP-Tree was used for maintaining the information of high utility itemsets. PHUIs can be efficiently constructed from UP-Tree with only two original database scans. Moreover, we developed several strategies to reduce the overestimated utility and improve the performance of utility mining. In the experiment, both real and synthetic data sets were used for performance evaluation. Results show that the strategies efficiently improved performance by reducing both the search space and the no of candidates. Moreover, the proposed algorithms, UP-Growth+, outperform the state-of-the-art algorithms substantially when databases contain lots of long transactions or a low minimum utility threshold is used.

REFERENCES

[1] R.Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in Proc. ACM SIGMOD Int. Conf. Manage. Data, 1993, pp. 207–216.

[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 Trans. Knowl. Data Eng., vol. 21, no. 12, pp. 1708– 1721, Dec. 2009.

[3] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "ExAnte: A preprocessing method for frequent-pattern mining," IEEE Intell. Syst., vol. 20, no. 3, pp. 25–31, May/Jun. 2005.

[4] C. Bucila, J. Gehrke, D. Kifer, and W. M. White, "Dualminer: A dual-pruning algorithm for itemsetswith constraints," DataMining Knowl. Discovery, vol. 7, no. 3, pp. 241–272, 2003.

[5] R. Chan, Q. Yang, and Y. Shen, "Mining high utility itemsets," in Proc. Int. Conf. Data Mining, 2003, pp. 19–26.

[6] S. Dawar and V. Goyal, "UP-Hist tree: An efficient data structure for mining high utility patterns from transaction databases," in Proc. 19th Int. Database Eng. Appl. Symp., 2015, pp. 56–61.

[7] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility cooccurrence pruning," in Proc. 21st Int. Symp. Found. Intell. Syst., 2014, pp. 83–92.

[8] S. Krishnamoorthy, "Pruning strategies for mining high utility itemsets," Expert Syst. Appl., vol. 42, no. 5, pp. 2371–2381, 2015.

[9] Y.-C. Li, J.-S.Yeh, and C.-C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," Data Knowl.Eng., vol. 64, no. 1, pp. 198–217, 2008.

[10] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in Proc. ACMConf. Inf. Knowl.Manage., 2012, pp. 55–64.

[11] Y. Shen, Q. Yang, and Z. Zhang, "Objective-oriented utility-based association mining," in Proc. IEEE Int. Conf. Data Mining, 2002, pp. 426–433.

[12] U. Yun, H. Ryang, and K. H. Ryu, "High utility itemsetmining with techniques for reducing overestimated utilities and pruning candidates," Expert Syst. Appl., vol. 41, no. 8, pp. 3861–3878, 2014.