[1] Sanjay D. Bhanderi
[2] Haresh M. Rathod
[3] Prashant D. Maheta
[4] Rushi J. Trivedi
[5] Ravi J. Khimani

ITWUI - An Interactive Transaction Weighted Utility Item Tree for High Utility Itemset Mining

Journal of Electrical Systems

Abstract:- Currently in the competitive marketing era, high utility itemset (HUI) mining is the most useful and most advanced research area which inherits from association rule mining (ARM). HUI is useful for designing the strategies for the big retailer mall, e-commerce portal like amazon, alibaba, ebay, flipkart, etc.. Utility in a high utility itemset is a profit which is calculated from the product of quantity and price of each product from the transaction table. In this paper, we propose a matrix called transaction positioning matrix. Transaction positioning matrix is a tabular data structure which is derived from a transaction database. Based on this matrix, we have generated a tree for each item from the transaction named as interactive transaction weighted utility item (ITWUI) tree. ITWUI tree is created for each potential item present in the transaction database. Then finally, a high utility itemset is mined by processing the ITWUI tree in parallel.

Keywords:- High Utility Itemset,Weighted utility,Parallel Computing,Transaction Positioning Matrix

I. INTRODUCTION

The fast improvement of database strategies encourages the capacity and use of massive/gigantic information from business companies, governments, and research organizations. Collecting important knowledge from a massive database evolved many research related topics. Among them one is high utility itemset mining which we are concerned about.

As of late, many algorithms have been developed and proposed for high utility itemset mining. However, these algorithms suffer from many limitations like excessive use of main memory, long execution time, and may or may not have interactive properties. To overcome these limitations we propose a new algorithm for mining high utility itemset. Our contributions in the paper are as follows:

1. A matrix structure called Transaction Positioning (TP) matrix is designed which stores the positions of the next item with current Transaction Utility (TU) of the item.
2. A tree structure called Interactive Transaction Weighted Utility (ITWU) Tree is designed from TP Matrix structure. This tree structure stores the TWU value of the item in a header table.
3. An Efficient algorithm called ITWU-Miner is developed. This algorithm has the speciality of mining itemset from ITWU Tree using parallel processing concept.

TID\ITEM	A	B	C	D	E	TU
T1	0	3	6	1	4	87
T2	3	0	10	0	9	81
T3	7	0	4	0	0	47
T4	6	1	0	1	0	61
T5	0	0	8	0	3	36
T6	0	2	12	1	0	78
T7	9	0	0	0	7	73
T8	2	2	0	0	6	56

(a)

Item	A	B	C	D	E
Utility	5	11	3	2	4

(b)

Fig 1 : Example of (a) Transactional Database (b) Utility Database

[1,2,3,4,5] Assistant Professor, Computer Engineering Department, Government Engineering College, Rajkot, Gujarat, India.
Copyright © JES 2024 on-line : journal.esrgroups.org

In the next section we have discussed background work with preliminaries and problem definition, major challenges and issues faced during mining and we propose an algorithm with construction of matrix and tree. At last, the mining algorithm is discussed and future work is included

II. BACKGROUND

A. *Preliminaries*

Let's consider a finite set of items I = {i_1, i_2, . . ., i_m} and a set of n transactions database D = {T_1, T_2, . . ., T_n}. unique transaction id TID assigned to each transaction T_d ($1 \le d \le n$) in D. A set of k distinct items {i_1, i_2, . . ., i_k} is called itemset X, where X is subset of I. i.e. X $\subseteq$ I and $1 \le k \le m$. Here m is the total number of items, length of the itemset X is k so it can be referred as k-itemset. In utility mining, iu(i_p, T_d) called internal utility i.e. in transaction T_d each item i_p ($1 \le p \le m$) is associated with a quantity, and eu(i_p) is called external utility as each item i_p has a unit profit.[2]

Definition 1. In transaction T_d, Utility of an item is denoted as u(i_p; T_d) for item i_p which is defined : eu(i_p) x iu(i_p; T_d).[2]

Definition 2. In transaction T_d, u(X; T_d) is Utility of an itemset X which defined as [2]

$$u(X,T_d) = \sum_{i_p \in X \wedge X \subseteq T_d} U(i_p,T_d) \quad [3]$$

Definition 3. u(X) is the Utility of an itemset X in D is which is defined as [2]

$$u(X) = \sum_{X \subseteq T_d \wedge T_d \in D} U(X,T_d) \quad [3]$$

Definition 4. An itemset is to be considered a high utility itemset if utility of the itemset is greater than a user-specified minimum utility threshold and it is denoted by min_util. if utility is less than a user specified minimum utility the , it is called a low-utility itemset [2].

B. *Major Challenges and Issues*

There are two step processes for calculation of mining high utility itemset. The first step all potential itemsets (candidate itemsets) have been identified. Then high utility itemsets have been identified from the set of potential Itemsets in the second phase.. The fundamental issues for this two steps processes are::

1. Generation of candidate itemsets are too large, so search space requirement is also high which adheres to two problems 1) in order to store candidate itemset during mining process excessive memory required [4]; (2) to generate large candidate itemsets a lot of computation time is needed. due to it performance of the algorithm will be degraded[4].
2. It is extremely challenging to prune the search space viably and without missing mining all the high utility itemsets.
3. Essential requirement of all nanosecond or millisecond data is incremental dataset because of the expansion of nanosecond or millisecond data periodically.
4. When minimum utility support is changed in an interactive algorithm no need to run the algorithm from beginning.
5. The other essential issue is that all high utility mining algorithms used two passes of the database. to develop algorithm for single database pass is a big challenge.

C. *Related Work*

High utility itemset mining algorithm presented from transactional database [2]. In the technique they presented, it creates UP-Tree which requires two database scans and it is depending on the length of candidate itemset fully. algorithm generates a big number of itemset i.e. potential high utility itemsets (PHUIs). However the algorithm addresses the issue of memory usage by generating candidate itemsets level wise.

Unil Yun,Heungmo Ryang, Keun Ho Ryu presented a High Utility mining algorithm named MU-Growth. In the algorithm they have presented two techniques to prune the candidate itemsets[3]. They also introduce the tree based data structure MIQ- tree (Maximum Item Quantity) which stores database information with single pass for high utility itemset mining. Initially the tree is constructed from a transactional database with items and its

associated quantities and then based on TWU descending order the tree is rebuilt. Then, MU-Growth generated candidate itemsets from the recreated tree. and it identified actual high utility itemsets from it.

Three novel tree structures $IHUP_L$(Incremental HUP Lexicographic Tree),$IHUP_{TF}$ (IHUP Transaction frequency), $IHUP_{TWU}$ (IHUP Transaction-Weighted Utilization Tree) presented by Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, Young-Koo Lee, Ho-Jin Choi et al [5].. They work on the concept of "Build once Mine many " property which means no need to create a tree once it is created even though the threshold value has been changed but algorithms have to perform all the steps from the first step. so it can work for any threshold value to mine high utility itemsets after tree construction. $IHUP_L$-Tree arranges items in lexicographic order in the first database scan then creates a branch by inserting it into the tree.compact $IHUP_{TF}$-Tree is created from $IHUP_L$-Tree by using descending order of transaction frequency and then based on descending order of twu value $IHUP_{TWU}$-Tree is created. At the end, based on bubble sort, using a path adjusting method restructuring operation is performed and the candidate itemsets are identified and form it mining high utility itemset performed.

Ming-Yen Lin,Tzer-Fu Tu,Sue-Chen Hsueh et al proposed a two phase- the maximal phase and utility phase-UMMI(Utility Mining using Maximal Itemset property) algorithm to find high utility itemsets[6]. To computational complexity and search space, in the maximal phase they only searched maximal high TWU itemset in place of high TWU itemsets. In the utility phase the MLexTree structure is created to effectively identify high utility items. In order to store transaction utilities of itemsets in ascending order HTP (High TWU Pattern) tree is created in the maximal phase. then recursively, from minimum TWU value to the maximum TWU value from the mining HTP tree and constructing a conditional tree of the MTWU itemsets. MLexTree (Maximal Lexicographic Tree) constructed from MTWU In Utility Phase. The MLexTree joins the similar prefixed items's itemsets and then all MTWU itemsets store the itemset in lexicographic order.. Then the database is updated with scanning the database, and it will select those nodes whose utility is not less than min_util by mining MLexTree.

Interactive and incremental single-pass mining algorithm for mining weighted frequent pattern proposed by Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, Young-Koo Lee, Ho-Jin Choi et al.[8]. They presented two novel tree data structures: Incremental weighted frequent pattern tree based on weight ascending order(IWFPTFD) and Incremental weighted frequent pattern tree based on descending order(IWFPTFD). algorithm guaranteed that to speed up the prefix tree, in any branch non-candidate itemset not appear before candidate itemset. it also ensures the interactive and incremental weighted frequent pattern itemset mining with single database scan.

III.PROPOSED ALGORITHM

A. Construction of Transaction Positioning Matrix

Construction of Transaction Positioning Matrix performed in two phases. In phase one, two I/O scans of the database are required, For each item Transaction Frequency(TF) and TWU is calculated during the first scan of the database and using bubble sort TWU is sorted. Then based on TWU value the transaction positioning matrix is generated. As shown in fig. 3, there are three columns in the matrix: location which is to identify the item , index that stores TWU value along with item name and transactional array store the pointer for the next item in particular transaction with TU.

In the proposed work, for each item, we have first calculated TWU and using the bubble sort, sorted it in ascending order.

Item	TWU	TF
A	318	5
B	282	4
C	329	5
D	226	3
E	333	5

Sorting →

Item	TWU	TF
D	226	3
B	282	4
A	318	5
C	329	5
E	333	5

Fig 2 : Example of (a) Transactional Database (b) Utility Database

Transaction Positioning matrix is generated after sorting in ascending order. First, according to the transaction 's TWU value, sort the item. Then transactions are inserted one by one based on the ascending order of TWU value of item. For example transaction T1 is inserted as D,B,C and E as per the ascending order of the TWU value of the respective item. As n total number of items and m is the maximum transaction frequency so size of the matrix is n x m. To insert the item in the matrix, after the sorting, see the first item as in our case in transaction

T1, it is D, start with insert in the row of D, see the next item after D, it is B, search the free position in the matrix that is (2,1) and store it along with the its TU value of B. same as B will search the free position in the matrix for next item C, that is (4,1), and store its poison along with its TU value. This process is for the last item of the transaction, as there will not next item available, then stores (Φ,Φ) along with the TU, that is in our case first position of item E. similarly transactional positioning matrix is generated for all transactions in the database.

Loc	Index	Transactional Array				
		1	2	3	4	5
1	(D,226)	(2,1,87)	(2,2,61)	(2,3,78)		
2	(B,282)	(4,1,87)	(3,3,61)	(4,5,78)	(3,5,56)	
3	(A,318)	(4,2,81)	(4,3,47)	(Φ,Φ,61)	(5,4,73)	(5,5,56)
4	(C,328)	(5,1,87)	(5,2,81)	(Φ,Φ,47)	(5,3,36)	(Φ,Φ,78)
5	(E,333)	(Φ,Φ,87)	(Φ,Φ,81)	(Φ,Φ,36)	(Φ,Φ,73)	(Φ,Φ,56)

Fig.3 : Transaction Positioning Matrix.

B. Construction of ITWU Tree

ITWUI tree and header table are constructed from TP Matrix which lead to candidate itemsets. Header table stores the information like item name and TWU of respective item. ITWUI tree is constructed for each item which is independent of each other and can be implemented using the parallel processing concept. Mining can be done independently on each tree which makes this algorithm efficient from other algorithms. To construct an ITWUI tree for a particular item we have to start from the first column and make the tree according to its position defined in the Matrix. In this tree we have to include the TWU and TF with the item name. In our case suppose we want to make ITWUI tree for D then we have to start from (1,1) position which has value like (2,1,87) so we have to traverse to next item at location (2,1) and in similar way we have to create tree until (Φ,Φ) appears as shown in fig. After that we have to go to (1,2) and traverse through the full path, in a similar way we have to draw the tree till the next pointer is present..

C. ITWUI-Miner Algorithm

ITWUI-Miner algorithm identifies the candidate itemset from ITWUI tree during its generation. One more database scan needed to search high utility itemset from candidate itemsets. While generation of the item tree is completed it directly finds a candidate itemset based on TWU value from the header table. If the min_utility threshold is less than the TWU values of header table' items then candidate itemset is considered for that item. For the items the minimum utility is greater than the TWU value can be pruned. Database is scanned and candidate itemset's utility is calculated and minimum utility is less than that utility then that particular candidate itemset is considered high utility itemset. In fig consider for item 'D' then from header table we can directly prune item 'A' and 'E' for min_util=129.75, so candidate itemset are 'DBC'(165,2),'DB'(226,3) and 'D'(226,3).Considering all other items, we get candidate itemset like BC (165,2), B(282,4), A (318,5), AE (210,3), CE (204,3), C(328,5) and E(333,5).From this candidate itemset we got A:135,AE:158,CE:164 and BCD:149 as high utility itemset.

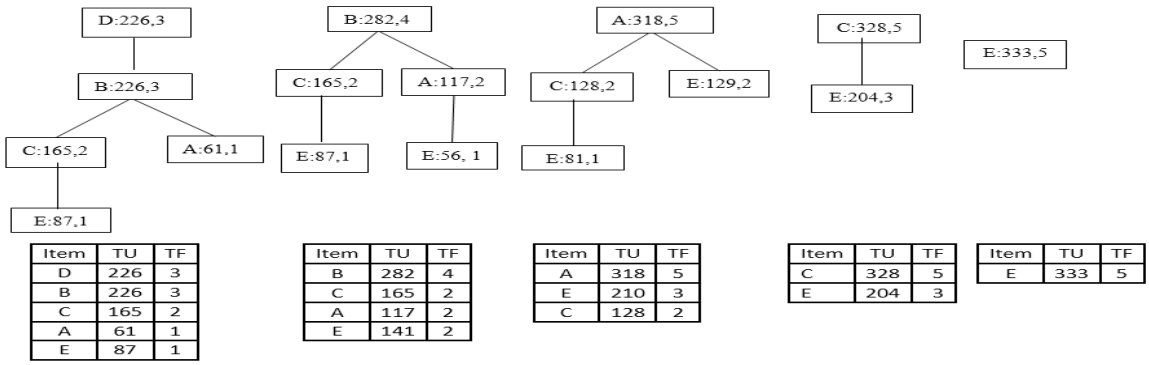

Fig.4 :High Utility Itemset Mining from ITWUI tree

IV. Result Analysis

A. Dataset used

In the Proposed algorithm, during the implementation phase we have selected the Chain Store Dataset as our main dataset as it contains more transactions and more distinct items so that we can work on a large number of items. Chain Store Dataset is downloaded from NU-MineBench 2.0 which contains two files. The first file is named as "real_data_aa.txt" and it contains 11,12,949 transactions and each transaction has a list of item ids which are bought by the customer. The other file named "product_price.txt" contains 46,086 distinct items available at the store with its profit per item and 7.26 items is the average transaction length.

B. Result and Analysis

We have analyzed proposed algorithms with different three algorithms, Two-Phase algorithm, HUI-Miner and IHUP algorithm. Experiments were performed on a Macbook computer system equipped with Apple M1 chip having 8-core CPU,8-core GPU and 16-core neural engine, running 64-bit Sonara operating system.

For different minimum thresholds, execution time is calculated and results are noted as shown in fig 4.

Min_util	ITWUI	Two-phase	HUI-Miner	IHUP
200000	49.593	1090.485	467.139	223.693
400000	22.603	345.439	119.043	77.673
600000	11.872	176.872	50.622	73.274
800000	10.857	110.744	32.153	47.159
1000000	9.623	78.371	22.714	33.54
1200000	8.393	59.946	14.43	28.299
1400000	8.128	48.857	11.154	25.662
1600000	7.441	39.451	8.658	20.467
1800000	7.551	38.751	7.831	19.235
2000000	7.285	34.664	7.02	17.02
2200000	7.083	30.467	6.973	15.538
2400000	7.035	28.549	7.036	14.071
2550000	6.864	28.44	6.879	13.51
2700000	6.957	28.377	6.723	13.994

Fig.4 : Execution time Comparison

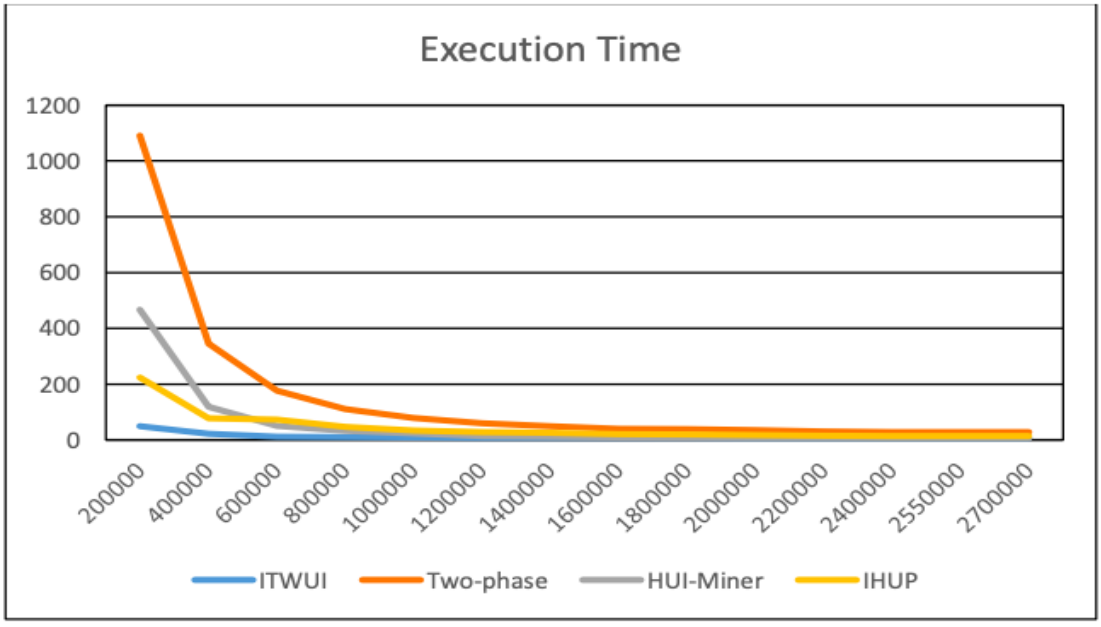

Fig .5 : Execution time comparison chart

As shown in fig. 4 and fig 5, as the value of min_ulit has increased, the size of the branch in ITWUI tree decreases and due to it the execution time also decreases.

V. CONCLUSION

In the proposed work we have presented a novel algorithm interactive transaction weighted utility item (ITWUI) tree algorithm for high utility mining. Result shows the effectiveness of the algorithm in the application of high utility mining.

In recent years, for voluminous data, big data and cloud computing have emerged as a research topic. so our plan is to develop the parallel and distributed ITWUI algorithm for a huge dataset.

REFERENCES

[1] Alva Erwin, Raj P. Gopalan, and N.R.Achutan, "Efficient Mining of High Utility Itemsets from Large Datasets", Springer-verlag Berlin Heidelberg,pp.554-561.

[2] Vincent Tseng,Bai-En Shie,Cheng-Wei Wu and Philips S. Yu,Fellow,IEEE, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases." IEEE Transactions on Knowledge and Data Engineering, vol 25 No, 8,August 2013,pp.1772-1786

[3] Unil Yun, Heungmo Ryang, Keun Ho Ryu, "High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates", Expert Systems with Applications(Elsevier), vol 41,2014,pp.3861-3878.

[4] Mengchi Liu,Junfeng Qu, "Mining High Utility Itemsets without Candidate Generation" CIKM'12,USA,ACM 2012,pp.55-64.

[5] C.F.Ahmed,S.K.Tanbeer,B.-S.Jeong,Y.-K.Lee, "Efficient Tree Structures for High utility for High Utility Pattern Mining in Incremental Databases" IEEE Transactions on Knowledge and Data Engineering,Vol.21,No.12,December 2009,pp.1708-1721.

[6] M.-Y.Lin,T.-F.Tu,S,-C.Hsueh, "High utility pattern mining using the maximal itemset property and lexographic tree structures" Informational Sciences 215(2012),Elsevier,pp.1-14.

[7] C.-W.Lin,G.-C.Lan,T.-P.Hong,"An Incremental mining algorithm for high utility itemsets" Expert Systems with applications 39 (2012) pg.7173-7180.

[8] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, Young-Koo Lee a,Ho-Jin Choi(2012) "Single-pass incremental and interactive mining for weighted frequent patterns" Expert Systems with Applications 39,ELSEVIER 2012, pp.7976–7994.

[9] Ahmed CF,Tanbeer SK,Jeong B-S, Lee Y-K (2011) , "HUC-Prune: An Efficient Candidate Pruning Technique to mine high utility patterns" Appl Intell, Springer,2009, PP: 181–198.

[10] Y.Liu, W.K. Liao and A. Choudhary,"A two phase algorithm for fast discovery of high utility itemset." Cheng, D. and Liu. H. PAKDD, LNCS-2005,PP: 689-695

[11] Alva Erwin, Raj P. Gopalan, and N.R.Achutan,"CTU-mine:An Efficient High Utility Itemset Mining Algorithm using the Pattern Growth Approach" Seventh International Conference on Computer and Information Technology,Aizu Wakmatsu,Japan,2007.

[12] Philipee Fournier-Viger,Cheng-Wei Wu,Souleymane Zida,Vincent S.Tseng, "FHM:Faster High Utility itemset mining using Estimated Utility Co-occurrence Pruning" ISMIS,Springer,LNAI-2014,pp.83-92.

[13] Wei Song,Yu Liu,Jinhong Li,"Mining high Utility Itemsets by Dynamically Pruning the Tree Structure" Appl Intell,Springer Science,2013,pp29-43.