An Enhanced HUI Miner Algorithm to Retrieve Optimum Number of High Utility Itemsets

Dr. S.Vijayarani[1] Ms. C.Sivamathi[2] Ms. N.Suhashini[3]
[1]Assistant Professor [2]Research Scholar
[1,2,3]Department of Computer Science Engineering
[1,2,3]Bharathiar University Coimbatore

Abstract— Utility Mining is a rising research domain in the data mining and it discovers high utility itemsets in the data sets. The utility can be any user defined choice and to assign this utility value is based on the attributes of the dataset. For example in a transaction database it can be a price of purchased item, profit of item or quantity of items purchased. An itemset is known as high utility itemsets if its utility value is greater than minimum utility threshold. Utility mining is used in many applications like inventory control, retail store or super market management and marketing. Most of the existing algorithms generate candidate set to calculate high utility itemsets. Candidate item generation is more difficult and tedious task. In order to avoid this, high utility itemsets can be retrieved without candidate generation. An algorithm named as HUI Miner algorithm retrieves high utility items without candidate generation method. In this work, HUI Miner algorithm is enhanced with a utility function and set of effective pruning strategies. In this work utility threshold is calculated automatically by using utility function. Performance of enhanced HUI Miner algorithm is compared with the existing HUI miner algorithm. Different sizes of datasets are used for experimentation. The average size of the transaction is 10. Results proved that the enhanced algorithm reduces the upper bounds of high utility itemset. Hence it saves time and memory.

Key words: Utility Mining, High Utility Itemsets, Pruning, HUI Miner, Enhanced HUI Miner

I. INTRODUCTION

Data Mining is defined as finding hidden information or extracting meaningful information from large amount of data. It is also called exploratory data analysis, data driven discovery and deductive learning. Data mining is the non-trivial extraction of implicit, previously unknown and potentially useful information from the data [1]. Frequent pattern mining is a popular technique in data mining, which helps to find frequent patterns in transaction database. The goal of frequent itemset mining is to find frequent itemsets. Many popular algorithms have been proposed for mining frequent patterns [1] [2]. They are, Apriori, FPGrowth, FIN, Éclat, etc. These algorithms take an input as a transaction database and a parameter “minSup” called the minimum support threshold. These algorithms then return all set of items (itemsets) that appears in minsup transaction.

In reality a retail business may be interested in identifying its most valuable customers i.e. customers who contribute a major portion of the profits to the business. These are the customers who may buy full priced items or high margin items which may be absent from a large number of transactions because most customers do not buy these items frequently [3] [4]. The practical usefulness of the frequent itemset mining is limited by the significance of the discovered itemsets [4]. There may be some itemsets which are not frequent but still they fetch more profit than some of the frequent itemsets.

Consider a retail store. Assume that the profit of 25kg rice is 500 INR and the profit of a milk is 3 INR. In a transaction database, milk occurs in 100 transactions and 25kg rice occurs in 3 transactions. The total profit of milk is 300 INR and the total profit of 25kg rice is 1500 INR. As per frequent itemset mining milk has high frequency. But the total profit of 25kg rice is much greater than a milk. Hence, traditional frequent itemset mining cannot discover the most profitable itemsets. This is because frequent itemset mining does not consider the profit (i.e. utility) of an item, which is also highly important in decision making [4] [5]. As per utility mining by considering profit 25kg rice has high utility. Hence there is need for new algorithms to discover high utility itemsets.

Retrieving high utility itemsets from a database is called as Utility Mining [6] [7] [8]. An itemset is known as high utility itemset if its utility is greater than user specified minimum utility threshold. Here utility can be interestingness, importance or profitability of an item to user [9] [10].

A. *Applications of Utility Mining:*

Following are some of the applications of utility mining. [15][16] [17].

- Website click stream analysis
- Business promotion in chain hypermarkets
- cross-marketing in retail stores
- online e-commerce management
- Mobile commerce environment planning
- Important patterns in biomedical applications.

B. *Terms in Utility mining:*

Transaction Database**:** Let I be a set of items. A transaction database is a set of transactions D = {T1, T2, ...,Tn} such that for each transaction T_c , $T_c \in$ I and T_c has a unique identifier c called its Tid. Each item i∈ I is associated with a positive number eu(i), called its external utility (e.g. unit profit). For each transaction T_c such that i∈T_c , a positive number iu(i, T_c) is called the internal utility of i (e.g. purchase quantity).

Example 1. Consider the Table1. This database contains five transactions (T1, T2, T3,T4 and T5). Transaction T2 indicates that items a, c, e and g appear in this transaction with an internal utility of respectively 2, 6, 2 and 5. Table 2 indicates that the external utility of these items are respectively 5, 1, 3 and 1.

Utility (u (I, T)) / Utility of Itemset in Transaction(u(X, T):[9][10] Utility is the product of iu (I,T) and eu (i) defined as u(i, T_c) = eu(i) × iu(i, T_c).The utility of itemset X in transaction T is the sum of the utilities of all item in T, where $u(i,T) = \sum i \in X \Lambda X \subseteq T^{u(i,T)}$.

Example 2. The utility of item a in T2 is u(a, T2) = 5 × 2 = 10. The utility of the itemset {a, c} in T2 is u ({a, c}, T2) = u (a, T2) +u(c, T2)) = 5×2 + 1×6 = 16

Utility of Itemset in Database (X): The utility of itemset X is the sum of the utilities of X in all transaction in database[10], where $u(X) = \sum T \in DB \Lambda X \subseteq T^{u(X,T)}$.

Example 3. The utility of the itemset {a, c} is u({a, c}) = u(a)+u(c) = u(a, T1)+ u(a, T2) + u(a, T3) + u(c, T1) + u(c, T2) + u(c, T3) = 5 + 10 + 5 + 1 + 6 + 1 = 28.

Transaction Utility (tu (i.T)): The utility of transaction T is the sum of the utilities of all the items in T[11], where $tu(T) = \sum i \in T^{u(i,T)}$.

Example 5. Fig. 2 (left) shows the TU of transactions T1, T2, T3, T4, T5 from our running example.

Transaction Weighted Utility (twu)[10[11]: The twu of itemsets X in database, is the sum of the utilities of the all transaction containing X in database [13] [14], where $twu(X) = \sum T \in DB \Lambda X \subseteq T^{tu(T)}$.

Example 6. Fig. 2 (center) shows the TWU of single items a, b,c, d, e, f, g. Consider item a. TW U(A) = T U(T1) + T U(T2) + T U(T3) = 8 + 27 + 30 = 65

Tid	Transaction	Count
T1	{a,c,d}	{1,1,1}
T2	{a,c,e,g}	{2,6,2,5}
T3	{a,b,c,d,e,f}	{1,2,1,6,1,5}
T4	{b,c,d,e}	{4,3,3,1}
T5	{b,c,e,g}	{2,2,1,2}

Table 1: An Example Transaction Database

Item	a	b	c	d	e	f	g
Profit	5	2	1	2	3	1	1

Table 2: Utility Table

TID	TU
T1	8
T2	27
T3	30
T4	20
T5	11

Table 3: Transaction Utility Table

ITEM	TWU
A	65
B	61
C	96
D	58
E	88
F	30
G	38

Table 4: Transaction Weighted Utility Table

II. Utility Mining Algorithms

The algorithms on high-utility itemset mining can be classified into two different paradigms, tree-based algorithms [18] and vertical mining algorithms [19].

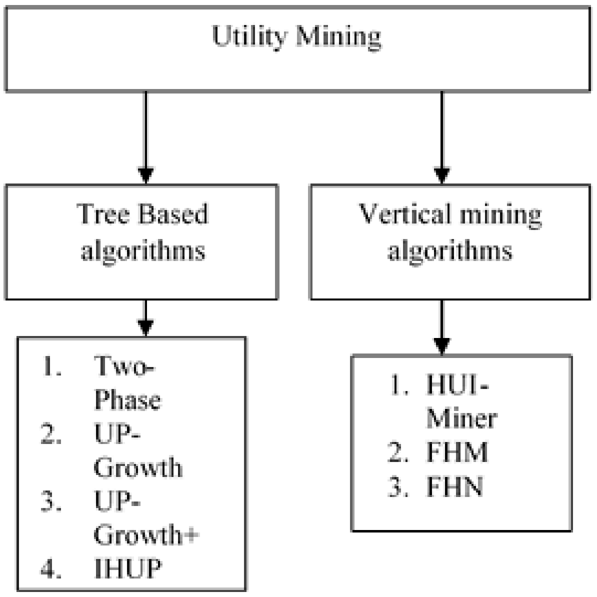

Fig. 1: Utility mining classification

A. *Tree Based Algorithm:*

The tree based algorithms mainly work in two phases [6]. In the first phase, they find candidate high-utility itemsets, which are then verified in the second phase. The advantage of tree-based approaches is that the tree data structure is a compressed representation of the complete transaction database and hence allows mining candidate high-utility itemsets quickly. However, the verification time taken by these algorithms increases with the number of candidates generated. Some of the algorithms which belong to this category are: Two-Phase, UP-Growth, UP-Growth+, CHUD and DAHU and IHUI.

B. *Vertical Mining algorithm:*

The vertical mining algorithms use an Inverted-list like data structure for its working. Liu et al. [19] proposed the utility-list data structure which is based on Inverted-list and an algorithm which finds k-length high-utility itemset by intersecting the utility-lists of k − 1 itemsets. Viger et al. [20] proposed a strategy to improve the performance of HUI-Miner by reducing the number of intersection/join operations. These vertical mining algorithms first generate all the singleton high-utility itemsets and then proceed to generation of pairs, triplets and so on. However, solutions based on vertical mining approach are simple and have shown to perform better as compared to tree-based approaches [19]. But, the joint operation cost is generally higher for small-size itemsets as compared to join operation cost for large-size itemsets. It is due to the size of lists associated with small itemsets being more than size of lists associated with large itemsets. On the other hand, it is efficient to perform join for itemsets with small size Inverted-list.

C. *HUI-Miner Algorithm:*

This algorithm discovers high utility itemset without candidate generation. It creates a vertical structure named Utility-List (UL) for each item and then finds high utility items from it. Algorithm 1 shows the pseudo-code of HUI-Miner. Here, Utility list contains itemsets, transaction utility and utility of the items; Internal utility is the quantity associated with iutils (internal utility) in the transaction T.

Remaining utility (rutils) of itemset in transaction is the sum of the utilities of remaining items in transaction. For each utility-list X in ULs (Utility List), if the sum of all the iutils in X exceeds minimum utility threshold, then item considered as high utility item. According to the algorithm only when the sum of all the iutils and rutils(remaining utility) in X exceeds minimum utilitywas processed further. When the initial utility-lists are constructed from a database, they are sorted in ascending orderbased on transaction-weighted utilityvalue. Therefore, all the utility-lists in ULs are ordered as the initial utility-lists are. To explore the search space, the algorithm intersects X and each utility-list Y after X in ULs. Suppose X is the utility-list of itemset Px and Y that of itemset Py, and then construct (P.UL, X, Y) to construct the utility-list of itemset Pxy. Finally, the set of utility-lists of all the 1-extensions of itemset Px is recursively processed.

III. Proposed Algorithm

The problems with HUI miner algorithms are: It generates a huge set of HUIs. Hence its mining performance is degraded consequently. Moreover this huge number of HUIs forms a challenging problem to the mining performance and higher processing time it consumes. It also generates huge number of candidate itemsets, then higher processing time it consumes. The proposed algorithm overcomes this limitation. This algorithm mine high utility itemsets by using effective pruning strategy. In this proposed algorithm a formula for setting the threshold (min_utility) value using the transaction database information was introduced. .

$$\boxed{\text{Min\_utility} = \sum \text{au} \,/\, \text{Number of items}}$$

Here, au ➔ actual utility of the item

$$\boxed{\text{au} = \sum \text{items (external utility * occurrence)} \in T_d}$$

A. *Pruning unpromising item strategy:*

The proposed strategy is mainly applied to remove unpromising items in transactions, and then improve the upper bounds of utility values of itemsets for mining. The strategy is still based on the transaction-weighted utilization (TWU) model proposed by Liu et al. [12].

Algorithm 2: Proposed algorithm

```
Input: Transaction database (DB), min_utility;
Output: High utility itemset;
1)  Scan DB of transactions   D;
2)  Determine transaction utility (tu) of   in D;
3)  for each item (X);
4)  Transaction-weighted-utility (twu) = ∑ tu  D;
5)  Actual Utility (au) = ∑  item(eu*occurrence)   D;
6)  Returntwu,au;
7)  min_utility= ∑ (au) / number of items;
8)  if (twu>= min_utility);
9)  if( au >= min_utility); //pruning strategy
10) Utility list   X;
11) else
12) item=   -  X   D;
13) end;
14) Repeat the step 06 until the end of the D;
15) Scan DB again of transactions   D;
    // D >= min_utility,au >= min_utility;
16) Sorting the DB in ascending order;
17) if SUM (X.iutils)+SUM (X.rutils) >= min _utility;
18) CONSTRUCT (D.UL,X,eutil);
19) HUI-miner(X,D.ULs,minutility);
20) End;
```

IV. Experimental Results

A. *Experiments:*

To evaluate the performance of proposed algorithm, experiments are conducted on various number of transactions. Here the proposed algorithm is compared with the HUI miner algorithm. In this section, performance results are reported and discussed.

B. *Experimental setup:*

Four different sizes of datasets were used in the experiments. Databases 1, 2, 3 and 4 were generated by Synthetic Data Generation Code. The databases do not provide item utility (external utility) and item count for each transaction (internal utility). Here internal utilities for items are generated randomly ranging from 1 to 10 by transaction utility values Generation code. Table V shows the statistical information about these databases, including the size on disk, the number of transactions, the number of distinct items, the average number of items in a transaction, and the maximal number of items in the longest transaction(s).

C. *Time Comparison:*

The running time of the Enhanced and Existing algorithms on all databases is depicted in Fig. 3. Running time was recorded by the "Timestamp" command, and it contains starting time, and ending time of the algorithm. For almost all databases, proposed algorithm performs the best and faster than the existing algorithm.

D. *Memory Comparison:*

Fig. 4 shows the memory consumption of the algorithms on all databases, the proposed algorithm always consumes less memory than the HUI miner algorithms. The reason is that this algorithm has to consume a very large amount of memory to store candidate high utility itemsets during their mining processes, while proposed does not. Generally, the memory consumption of these algorithms is proportional to the number of candidate itemsets they generate.

E. *Pruned Itemset Comparison:*

The processing order of items significantly influences the performance of a high utility itemset mining algorithm. The transaction-weighted-utility-ascending order leads to the best performance. The reason is that the processing order of items is capable of reducing the number of sets of utility-lists for a mining task. To comprehend the reason in depth, one can consult the related work in [12, 13]. Figure 3 shows the different number of pruning rates.

Database	Size(kb)	No. of Transactions	No. of Items	Avg No. of items	Max no. of items
1	10	500	20	10	18
2	20	1000	20	10	19
3	33	1500	19	10	17
4	40	2000	20	10	20

Table 5: Statistical Information about Database

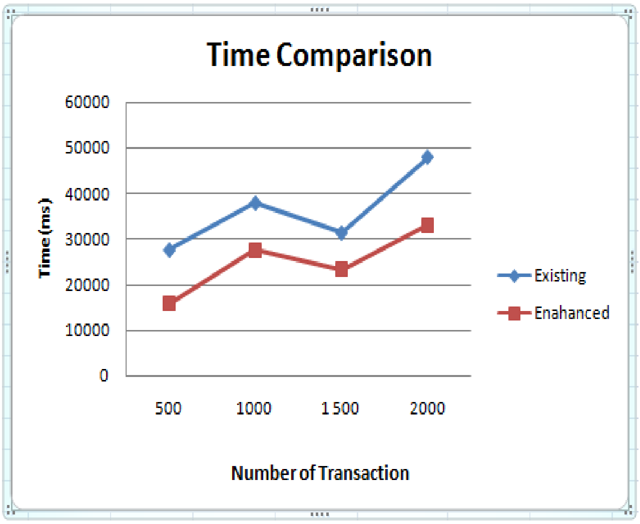

Fig. 2: Comparison of execution time in ms

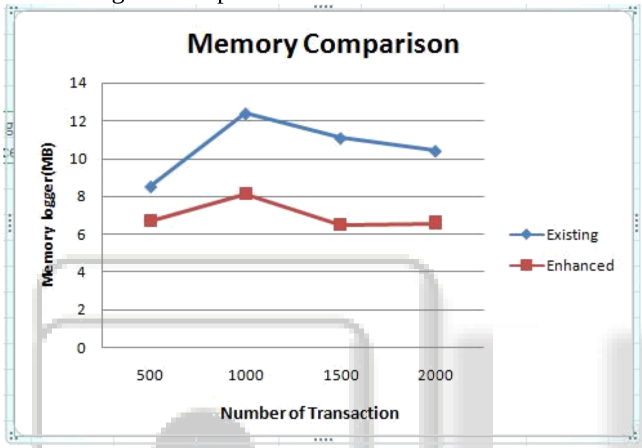

Fig. 3: Comparison of memory space occupied in mb

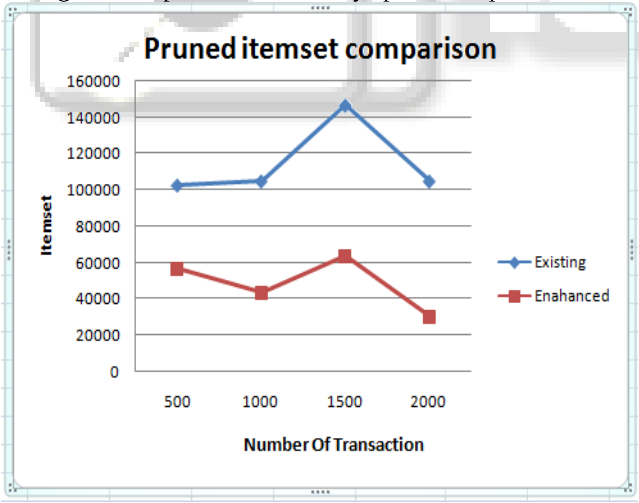

Fig. 4: Comparison of number of pruned itemsets

V. CONCLUSION AND FUTURE WORK

In this paper a HUI miner algorithm and enhanced HUI miner algorithm were implemented and their results are compared. This enhanced algorithm implements pruning strategy to reduce the number of high utility itemsets. The algorithm was implemented for four different sizes of datasets. Performance factors like time, memory space and pruning rates of HUI-Miner are compared with enhanced algorithm. Results show that the pruning strategy reduces the search space and that the enhanced algorithm works faster than HUI-Miner.

In future, new algorithms are to be developed for mining high utility items. Real time datasets can be collected and it will be used for experimentation. The work can be extended to handle data streams and negative utility values also.

REFERENCES

[1] R. Agrawal et al., "Mining association rules between sets of items in large databases", in proceedings of the ACM SIGMOD International Conference on Management of data, page no: 207-216.
[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules", in proceedings of 20th Int'l Conf. Very Large Data Bases (VLDB), page no: 487-499.
[3] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation", in Proceedings of the ACM-SIGMOD Int'l Conf. on Management of Data, pp. 1-12.
[4] R. Chan, Q. Yang, and Y. Shen, "Mining high utility Itemsets," The Third IEEE International Conference on Data Mining, pp. 19-26, 2003..
[5] Y. Liu, W. Liao and A. Choudhary, "A fast high utility itemsets mining algorithm", in Proceedings of the Utility-Based Data Mining Workshop.
[6] S.Shankar, T.P.Purusothoman, S. Jayanthi,N.Babu, "A fast agorithm for mining high utility itemsets" , in Proceedings of IEEE International Advance Computing Conference (IACC 2009), Patiala, India, page no:1459-1464.
[7] A. A. Bhosale , S. V. Patil, P. M. Tare, P. S. Kadam," Review Paper - High Utility Itemsets Mining on Incremental Transactions using UP-Growth and UP-Growth+ Algorithm", in proceedings of International Journal on Recent and Innovation Trends in Computing and Communication, Volume 2, page no: 3366 – 3368.
[8] Maya Joshi , Mansi Patel, " A Survey on High Utility Itemset Mining Using Transaction Databases", in proceedings of (IJCSIT) International Journal of Computer Science and Information Technologies, Volume 5, page no: 7407-7410.
[9] SadakMurali and KollaMorarjee," A Survey on Efficient Algorithm for Mining High Utility or Mining High Utility Itemsets", in proceedings of IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 1, Issue 5.
[10] Guo-Cheng Lan , Tzung-Pei Hong, and Vincent S. Tseng," An Efficient Gradual Pruning Technique For Utility Mining", in proceedings of International Journal of Innovative Computing, Information and Control, Volume 8, page no: 5165–5178.
[11] M. J. Zaki, "Scalable algorithms for association mining", in proceedings of IEEE Transactions on Knowledge and Data Engineering, page no:372–390.
[12] G. Liu et al.," Efficient mining of frequent patterns using ascending frequency ordered prefix-tree", in proceedings of Data Mining and Knowledge Discovery, page no:249–274.
[13] G. Liu, H. Lu, J. X. Yu, W. Wang, and X. Xiao, " An efficient implementation of pattern growth approach", in proceedings of IEEE Int'l Conf. Data Mining Workshop Frequent Itemset Mining Implementations, 2003.
[14] Teng, W. G., Chen, M. S., and Yu, P. S, "A RegressionBased Temporal Pattern Mining Scheme for

Data Streams", in proceedings of the 29th International Conference on Very Large Data Bases, page no: 93—104, September 2003.

[15] Teng, W. G., Chen, M. S., and Yu, "Resource-Aware Mining with Variable Granularities in Data Streams", SDM 2004.

[16] Yu-Chiang Li, Jieh-Shan Yeh, Chin-Chen Chang, "Isolated items discarding strategy for discovering high utility itemsets" in proceedings of Data & Knowledge Engineering 64 (2008) page no:198–217.

[17] BacLe,Huy Nguyen, Tung Anh Cao, Bay Vo,"A Novel Algorithm for Mining High Utility Itemsets" in proceedings of First Asian Conference on Intelligent Information and Database Systems.

[18] Vincent S Tseng, Cheng-Wei Wu, Bai-En Shie, and Philip S Yu, "UPGrowth: an efficient algorithm for high utility itemset mining", in proceedings of ACM SIGKDD, page no:253–262.

[19] Mengchi Liu and JunfengQu, "Mining high utility itemsets without candidate generation", in Proceedings of the 21st ACM international conference on Information and knowledge management, page no: 55–64.

[20] Philippe Fournier et al., "Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning", in proceedings of Foundations of intelligent systems, page no: 83–92.

[21] Yao Li, ZH Zhang, WB Chen, and Fan Min, "Mining high utility itemsets with discount strategies", in proceedings of ComputSci, page no: 6297–6307.