Study of Algorithms for Mining High Utility Itemsets

Radhika Chandwadkar[1], Manisha Thombare[2]

[1]*Asst. Professor, Dept of Computer Engineering, MVPS KBTCOE, Nashik, Maharashtra, India*
[2]*Asst. Professor, Dept. of Computer Engineering, MVPS KBTCOE, Nashik, Maharashtra, India*

Abstract – *Data mining techniques are applied for finding meaningful information and patterns from the large database. The traditional frequent itemset mining (FIM) algorithm generate large number of frequent itemset considering only the occurrence aspect of itemset It does not take into consideration the utility aspect of quantity and profit of item purchased. Hence an extension to FIM, High utility itemsets (HUIs) mining is emerging in information mining, which considers finding all itemsets having a utility meeting a user-specified minimum utility threshold. High-utility itemset mining (HUIM), aims to find a complete set of itemsets having high utilities in a given dataset. High average-utility itemset mining (HAUIM) is a variation of HUIM. HAUIM provides an alternative measurement named the average-utility to discover the itemsets by taking into consideration both of the utility values and lengths of itemsets. Efficient algorithms named TKU (mining Top-K Utility itemset) and TKO (mining Top-K utility itemset in One phase) are used in HUIM. A pattern growth approach is specified for efficiently mining of HAUIs. This paper studies the different algorithms for mining of high utility itemset.*

Key Words: Data mining ,Frequent Itemset mining, High Utility Itemsets, UP Tree, High average utility itemset mining.

1. INTRODUCTION

Frequent itemset mining is one of the major area in data mining. It is used to discover the frequently occurring itemset from the database that is above a user defined frequency threshold.[1] In Frequent itemset mining it is very difficult for user to find proper threshold value. In today's life users are interested to sell the itemset which gives more profit. Frequent itemset mining only finds the frequently occurring itemsets but it does not consider the quantity and number of item purchased. It also loses the useful information of profit gaining itemset having less selling frequency. Hence, frequent itemset mining cannot satisfy the demand of user who wants to search the profit gaining itemset.

To overcome these limitations high utility itemset mining has been proposed.[1] It considers both the profit and number of items purchased and help to improve the business strategy. In utility mining, each item is associated with a utility (e.g. unit profit) and an occurrence count in each transaction (e.g. quantity). The utility of an itemset represents its importance, which can be measured in terms of weight, value, quantity or other information depending on the user specification. An itemset is called high utility itemset (HUI) if its utility is not less than a user-specified minimum utility threshold.HUI mining is essential in many applications such as streaming analysis, market analysis, mobile computing and biomedicine [4]. But efficient mining of HUIs from databases is not an easy task because the downward closure property used in FIM does not hold for the utility of itemsets. In other words, pruning search space for HUI mining is difficult because a superset of a low utility itemset can be high utility itemset.

However it is difficult for users to choose an appropriate minimum utility threshold in practice. Depending on the threshold, the output size can be very small or very large. Besides, the choice of the threshold greatly influences the performance of the algorithms. If threshold value is set large then there are chances that no high utility itemset are found and if threshold value is set too small, too many high utility itemset are generated and hence it is very difficult to understand the result.

For such limitations of HUIM, the concept of high average-utility mining (HAUIM) was introduced [11]. The average-utility of an itemset is derived by dividing its utility to the number of its items. An itemset is considered as a high average-utility itemset (HAUI) if its average-utility value is no less than a given minimum utility threshold (minUtil). HAUIM is important for several application domains, such as, business applications, medical data analysis, streaming data analysis etc.

2. RELATED WORK

Although many studies have taken the task of HUI mining, it is difficult for users to choose an appropriate minimum utility threshold in practice. To set the proper value of threshold, user need to try various values of threshold by guessing and re-executing the algorithm repeatedly till it generate proper threshold value. This is a very tedious and time consuming process. Hence to precisely control the output size and discover the itemsets with the highest utilities without setting the thresholds, a promising solution is to redefine the task of mining HUIs as mining top-k high utility itemsets (top-k HUIs). Vincent S. Tseng, Cheng-wei Wu, Philippe Fournier-Viger and Philip S. Yu[4] proposed a novel framework for discovering top-k high utility itemsets mining, where k value indicates desired number of HUIs to be mined. Two types of efficient algorithms named TKU (mining Top-K Utility itemsets) and TKO (mining Top-K utility itemsets in one phase) are proposed for mining the itemsets in which there is no need to specify threshold value. TKU algorithm is used for mining potential Top-k high utility itemsets. TKO is one phase algorithm; it uses list based structure named utility-list to maintain the information of Top-k high utility itemset.

Another strategy using the concept of transaction-weighted utilization (TWU) model [5] was introduced to facilitate the performance of the mining task. In this model, an itemset is called high transaction-weighted utilization itemset (HTWUI) if its TWU is no less than min_util, where the TWU of an itemset represents an upper bound on its utility. Therefore, a HUI must be a HTWUI and all the HUIs must be included in the complete set of HTWUIs. A classical TWU model-based algorithm consists of two phases. In the first phase, called phase I, the complete set of HTWUIs are found. In the second phase, called phase II, all HUIs are obtained by calculating the exact utilities of HTWUIs with one database scan.

Later the focus on enhancing the HUIM was on the use of data structures used in the process. Shuning Xing, Fangai Liu, Jiwei Wang, Lin Pang and Zhenguo Xu by introducing a Fast Utility Tree [6] (FU-Tree) proposed an UP-Tree process that gives better scalability for mining high utility itemsets.

The research further focused on improving the efficiency of the algorithm for mining the high utility itemsets. Serin Lee, Jong Soo Park suggested a new algorithm, TKUL-Miner [7], to mine top-k high utility itemsets efficiently. It uses new utility-list structure for maintaining necessary information at each and every node on the search tree for mining the itemsets. Authors proposed efficient algorithm to raise the border minimum utility threshold rapidly. Also, for calculating smaller overestimated utilities, two additional strategies are suggested to prune unpromising itemsets effectively.

Y. Liu, W. Liao, and A. Choudhary proposed two phase algorithm [8] to overcome the limitation of utility mining and mine high utility itemsets from the database. In first phase algorithm defines transaction weighted utilization, and discover the transaction weighted utilization model and this model support transaction-weighted downward closure property. In the last phase one additional database scan is carried out to filter out the overestimated itemsets. Another fast high utility itemset mining algorithm was suggested by Ameena Aiman, Raafiya Gulmeher[3].They proposed the FHM algorithm that had a novel pruning procedure named EUCP (Estimated Utility Co-occurrence Pruning) that can prune itemsets without performing joins

3. HIGH UTILITY ITEMSET MINING (HUIM)

One of the concept of high utility itemset mining, is to let the users specify k, i.e., the number of desired itemsets, instead of specifying the minimum utility threshold. Setting k is more intuitive than setting the threshold because k represents the number of itemsets that the users want to find whereas choosing the threshold depends primarily on database characteristics, which are often unknown to users. Top-k high utility itemset mining finds out desired number of k high utility itemset, where k value is taken from user.

Two efficient algorithms named TKU (mining TopK Utility itemsets) and TKO (mining Top-K utility itemsets in One phase) are proposed for mining the complete set of top-k HUIs in databases without the need to specify the min_util threshold. The TKU algorithm adopts a compact tree-based structure named UP-Tree [10] to maintain the information of transactions and utilities of itemsets. TKU inherits useful properties from the TWU model and consists of two phases. In phase I, potential top-k high utility itemsets (PKHUIs) are generated. In phase II, top-k HUIs are identified from the set of PKHUIs discovered in phase I. On the other hand, the TKO algorithm uses a list-based structure named utility-list [9] to store the utility information of itemsets in the database. It uses vertical data representation techniques to discover top-k HUIs in only one phase.

3.1 TKU Algorithm

TKU (mining Top-k Utility itemsets) for discovering top-k HUIs without specifying min_util.TKU is an extension of UPGrowth [10], a tree-based algorithm for mining HUIs. TKU adopts the UP-Tree structure of UP-Growth to maintain the information of transactions and top-k HUIs. TKU is executed in three steps:

(1) Construction of UP Tree.
A UP-Tree can be constructed by scanning the original database twice. In the first scan, the transaction utility of each transaction and TWU of each item are computed. During the second database scan, transactions are reorganized and then inserted into the UP-Tree.

(2) Generation of potential top-k high utility itemsets (PKHUIs) from the UP-Tree.

The TKU algorithm uses an internal variable named border minimum utility threshold (denoted as min_utilBorder) which is initially set to 0 and raised dynamically after a sufficient number of itemsets with higher utilities has been captured during the generation of PKHUIs.

(3) Identification of top-k HUIs from the set of PKHUIs.

After identifying PKHUIs, TKU calculates the utility of PKHUIs by scanning the original database once, to identify the top-k HUIs. At this stage only the candidate itemset X is considered, if its estimated utility value reached after phase I is not less than min_utilBorder. Thus the top K high utility itemsets are identified.

3.2 TKO Algorithm

The second algorithm that is under discussion is TKO (mining Top-k utility itemsets in One phase). It can discover top-kHUIs in only one phase. It utilizes the basic search procedure of HUI-Miner and its utility-list structure [9]. Whenever an itemset is generated by TKO, its utility is calculated by its utility-list without scanning the original database.. The utility-lists of items are called initial utility-lists, which can be constructed by scanning the database twice In the first database scan, the TWU and utility values of items are calculated. During the second database scan, items in each transaction are sorted in order of TWU values and the utility-list of each item is constructed. TKO initially sets the min_utilBorder threshold to 0 and initializes a min-heap structure TopK-CI-List for maintaining the current top-k HUIs during the search. The algorithm then scans database twice to build the initial utility-lists. Then, TKO explores the search space of top-k HUI using a procedure named Top K-HUI-Search. It is the combination of a novel strategy named RUC (Raising threshold by Utility of Candidates) with the HUI-Miner search procedure [10]. During the search, TKO updates the list of current top-k HUIs in TopK-CI-List and gradually raises the min_utilBorder threshold by the information of TopK-CI-List. When the algorithm terminates, the TopKCI-List captures the complete set of top-k HUIs in the database.

4. PROBLEM DEFINITION

Let I = {I1, I2, I3…….Ip} be the set of items used in database and D = {T1, T2, T3……Tm} is a set of transaction used in transaction database. Where each transaction T_d in transaction database have a unique identifier d, called Tid. The quantity and profit of item I_k(1 <=k ”<= p) is denoted by $q(I_k)$ and $p(I_k)$ respectively. An itemset Y is a set of all distinct items which is a subset of I. Table 1 shows example of transaction database and the unit profit of item in transaction

Table -1. An Example of Transaction Database

TID	Transaction
T1	(A,1) (B,1) (C,4) (D,1)
T2	(B,1) (D,3)
T3	(A,2) (D,1)
T4	(C,1)
T5	(A,1) (B,2) (D,1) (E,3)
T6	(A,1) (B,1) (C,1) (D,1) (E,1)
T7	(B,2) (C,3) (E,1)
T8	(D,1) (E,2)
T9	(A,7) (C,1) (D,1)
T10	(B,1) (C,1) (D,1) (E,1)

Table -2. Unit profit Item

A	B	C	D	E
3	10	1	6	5

The utility of item I_k in transaction T_d is denoted as $u(I_k, T_d)$ and defined as:

$u(I_k, T_d) = q(I_k, T_d) \times p(I_k)$

The utility of itemset in a transaction T_d is denoted as $u(Y, T_d)$ and defined as the summation of utility of all the items present in the transaction.
The utility of an itemset Y in a database D is denoted as u(Y) and defined as the summation of utility of the itemset in all the transactions.
The transaction utility of a transaction T_d is denoted as TU (T_d) and defined as the summation of utility of all items present in the transaction. Transaction utility of above transactions is shown in Table 3.

Table -3. Transaction utility of transaction

TID	Transaction	TU
T1	(A,1) (B,1) (C,4) (D,1)	23
T2	(B,1) (D,3)	28
T3	(A,2) (D,1)	12
T4	(C,1)	1
T5	(A,1) (B,2) (D,1) (E,3)	44
T6	(A,1) (B,1) (C,1) (D,1) (E,1)	25
T7	(B,2) (C,3) (E,1)	28
T8	(D,1) (E,2)	16
T9	(A,7) (C,1) (D,1)	28
T10	(B,1) (C,1) (D,1) (E,1)	22

The Transaction weighted utility (TWU) of an itemset Y is denoted as TWU(Y) and is defined as the summation of TU of the transactions in which the itemset is present.

Table -4. Transaction weighted utility of items

Item	A	B	C	D	E
TWU	132	170	127	208	135

Transaction-weighted utility of itemset supports antimonotone property. Transaction-weighted utility is calculated in Table 4. The Transaction-weighted utility are then arrange in descending order according to their transaction utility as shown in Table 5. It indicates that TWU of any superset of Y cannot be greater than Y and this property is called Transaction- weighted Downward Closure property.

Table 5. Rearranging the item with respect to TWU

Item	TWU
D	208
B	170
E	135
A	132
C	127

The Minimum utility of an item in transaction T_d is defined as the occurrence of the minimum utility value of the item in the database.

Table 6. Items and MIUs

Item	A	B	C	D	E
MIU	3	10	1	6	5

The Maximum utility of an item in transaction T_d is defined as the occurrence of the maximum utility value of the item in the database.

Table 7. Items and MAUs

Item	A	B	C	D	E
MAU	21	20	4	18	15

In first scans, it finds out transaction utility of transaction and it also computes TWU of each and every item and in second scan it reorganizes the transaction and constructs the Up Tree. The rearrangement of item is done according to their TWU as shown in the above Table 5.

Next step is performed in three steps, in the very first step calculate MIU item. The MIU item is calculated in Table 6.In the second step calculate MAU of item. The MAU item is calculated in Table 7 .After that Pre-Evaluation Matrix (PEM) is generated. If the k th value in pre evaluation matrix is higher than calculated the minimum utility border value then minimum utility border is set to kth highest value of pre evaluation matrix. The pre-evaluation matrix is shown in Table 8

Table -8. Pre EvaluationMatrix

Item	B	C	D	E
A	49	33	66	26
B		59	102	90
C			31	20
D				59
E				

In the final step, it generates top-k high utility itemsets. TKU uses UP Tree where as TKO uses utility list and generate the top- k HUIs in one step

5. HIGH AVERAGE UTILITY ITEMSET MINING (HAUIM)

The result of HUIM has itemsets generated with a long length. However, as the length of the itemset increases, its utility tends to be larger since the utility of an itemset is the sum of the utility of each item that it contains. Therefore, HUIM mainly suffers from generating a large number of itemsets with long lengths. In addition, because of the nature of the utility measurement in HUIM, most of the discovered HUIs may contain items with low utilities. To address these limitations, the concept of high average-utility mining (HAUIM) was introduced with a more fair measurement named average-utility [11]. The average-utility of an itemset is derived by dividing its utility to the number of its items. An itemset is considered as a high average-utility itemset (HAUI) if its average-utility value is no less than a given minimum utility threshold (minUtil).

A typical HAUIM approach aims to find a complete set of HAUIs based on a given minUtil threshold. This process is computationally complex due to anti-monotonic characteristic of average-utilities of itemsets. The first proposed algorithm to mine HAUIs is the Two-phase high average-utility pattern mining (TPAU) algorithm [11]. But the HAUIM algorithms need long execution times and large amounts of memory to perform their mining tasks, especially when the database size is large or the minimum utility threshold is low. Hence to enhance the efficiency of solving the problem of mining HAUIs, efficient strategies have been developed, [3]such as :

(1) Introducing more effective upper-bounds and pruning strategies for early pruning unpromising itemsets from the search space

(2) Proposing efficient data structures for reducing the memory consumption and the cost of database scans in addition to avoid the costly join operations

(3) Developing an effective mining method to discover the complete and correct set of HAUIs by utilizing all the strategies mentioned above together

In the algorithm, the anti-monotone property is used to decrease the number of itemsets to be scanned level by level. There are two phases in the algorithm. In phase 1, the average-utility upper bound is used to overestimate the itemsets. The average-utility upper bound is an overestimated utility value instead of actual utility value. The average-utility upper bound can ensure the anti-monotone property. Thus, each subset of an itemset with high average-utility upper bound must be high; each superset of an itemset with low average-utility upper bound must be low. It can thus prune many low average-utility upper bound itemsets level by level and decrease the time to scan a database.

In phase 2, we need to scan the database once to check the result of phase 1 is actually high or not. The algorithm first finds all the candidate average utility 1-itemsets C1. The 1-itemsets whose average-utility upper bound is larger than or equal to minimum average-utility threshold are put in the set of candidate average-utility 1-itemset C1. Candidate average-utility 2-itemsetsC2 are formed fromC1. The algorithm then checks all the candidate average-utility 2-itemsetsC2 by comparing the average-utility upper bound with the minimum average-utility threshold. The itemsets which do not exceed the minimum average-utility threshold are removed from the candidate

2-itemsets. The same procedure is repeated until all the itemsets have been found. Then we calculate the actual average-utility value of each candidate average-utility itemset. If the itemset is larger than or equal to the minimum average utility threshold, put it in the set of high average-utility itemsets.

Consider the above example with alternate representation of the Database as shown in Table 9.

Table 9: Alternate representation of Database

TID	A	B	C	D	E
T1	1	1	4	1	0
T2	0	1	0	3	0
T3	2	0	0	1	0
T4	0	0	1	0	0
T5	1	2	0	1	3
T6	1	1	1	1	1
T7	0	2	3	0	1
T8	0	0	0	1	2
T9	7	0	1	1	0
T10	0	1	1	1	1

The utility value of each item occurring in each transaction in Table 9 is calculated. Take item B in transaction 7 as an example. The quantity of item B in transaction 7 is 2, and its profit is 10. The utility value of B is thus calculated as 2*10 which is 20. The utility values of all the items in each transaction are shown in Table 10.

The utility values of the items in each transaction are compared and the maximal utility value in the transaction is found. The maximal utility value in each transaction is shown in Table 10

Table 10: Utility values and Maximal utility

TID	A	B	C	D	E	Maximal Utility
T1	3	10	4	6	0	10
T2	0	10	0	18	0	18
T3	6	0	0	6	0	6
T4	0	0	1	0	0	1
T5	3	20	0	6	15	20
T6	3	10	1	6	5	10
T7	0	20	3	0	5	20
T8	0	0	0	6	10	10
T9	21	0	1	6	0	21
T10	0	10	1	6	5	10

The average-utility upper bound of 1-itemsets is calculated. Take item A as an example. It appears in transactions 1, 3, 5, 6 and 9. The average-utility upper bound of A is thus the total amount of the maximal utility values of these transactions. It is calculated as 10 + 6 + 20 + 10 + 21, which is 67. The upper bound values of all the items are shown in Table 11.

Table 11:The candidate average-utility 1-itemsets,C1

Candidate 1-Itemset	Avg. utility upper bound
A	67
B	88
C	72
D	105
E	70

Check whether the average-utility upper bound of 1-itemsets is larger than or equal to user-defined minimum average-utility threshold k, which is 45.4. In this example, the average-utility upper bound of 1-itemsets exceeds the minimum average-utility threshold k. All the items are recorded as candidate average-utility 1-itemsets,C1.

The candidate average-utility 2-itemsets (C2) are then generated from C1. They are {AB}, {AC}, {AD}, {AE}, {BC}, {BD}, {BE}, {CD}, {CE}, {DE}. The average-utility upper bound of each 2-itemset is calculated. Take the itemset {AB} as an example. It appears in transactions 1, 5 and 6. The average-utility upper bound of {AB} is thus the total amount of the maximal utility values of these transactions as 10 + 20 + 10, which is 40. The average-utility upper bound of each 2-itemset is thus checked against the user-defined minimum average-utility threshold k. In this example, the itemsets {AB}, {AC}, {AE} and {CE} do not exceed k. These itemsets are thus removed from C2. The remaining candidate average-utility 2-itemsets are shown in Table 12.

Table 12:The candidate average-utility 2-itemsets,C2

Candidate 2-Itemset	Avg. utility upper bound
AD	67
BC	50
BD	68
BE	60
CD	51
DE	50

C3 is then generated from C2 Since the average-utility upper bounds of both the two candidate 3-itemsets are less than k, they are removed from C3 and C3 becomes null.

The actual average-utility value au's of each candidate average-utility itemset is calculated. Take the itemset {AD}as an example. The actual utility values of items A and D in transaction 1 are 3 and 6, respectively. Since the itemset {AD} contains 2 items, its actual average-utility value in transaction 1 is calculated as (3 + 6)/2, which is 4.5. The itemset {AD} appears in transactions 1, 3, 5, 6 and 9. The actual average-utility value of {AD} is thus the total amount of actual average-utility values of these transactions. The value is calculated as (9 + 12 + 9 + 9 + 27)/2, which is 33. The actual average-utility value of each candidate average-utility itemset is shown in Table

13.

Table 13: Actual Average utility values of average candidate itemset.

Candidate itemset	Actual average utility
A	36
B	80
C	11
D	60
E	40
AD	33
BC	29.5
BD	51
BE	45
CD	15.5
DE	29.5

The actual average utility value of each candidate average utility itemset is then compared with the user defined minimum average-utility threshold k. In this example the actual average –utility values of itemsets {B}, {D} and {BD} are larger than or equal to k. They are thus put into the set of high average-utility itemsets, H, as shown in Table 14.

Table 14: High average utility itemsets.

High Average Utility Itemsets	Average utility
B	80
D	60
BD	51

Thus the high average utility itemsets can be mined from the transactional database

6. CONCLUSION

High-utility itemset mining (HUIM), which is an extension of well-known frequent itemset mining (FIM), takes into account utilities (such as, unit quantities and unit profits) of the itemsets. However HUIM leads to the generation of huge number of itemsets with long lengths. To address this problem and extract more meaningful results, the concept of high average-utility itemset mining (HAUIM) was introduced.It is observed that HAUIM gives better results as compared to HUIM. The further work can be directed to improve the efficiency of high average utility mining algorithms.

REFERENCES

[1] Snehal D. Ambulkar , Dr. Prashant N. Chatur "Efficient Algorithms for mining High Utility Itemset", International Conference on Recent Trends in Electrical, Electronics and Computing Technologies,2017.

[2] Ameena Aiman, Raafiya Gulmeher,- "Efficient Algorithms for Mining Top-K High Utility Itemsets", International Journal of Computer Sciences and Engineering, Vol.-6, Issue-7, July 2018.

[3] Irfan Yildirim and Mete Celik, - "An Efficient Tree-Based Algorithm for Mining High Average-Utility Itemset",IEEE Access,Volume 7, 2019

[4] Vincent S. Tseng, Cheng-wei Wu, Philippe Fournier-Viger and Philip S. Yu, "Efficient Algorithms For Mining Top-K High Utility Itemsets", In IEEE Transactions on Knowledge and Data Engineering, vol.28 no.1,2015

[5] Y. Liu, W. Liao, and A. Choudhary, ―A Fast High Utility Itemsets Mining Alg orithm, ‖ in Proc. of the Utility-Based Data Mining Workshop, pp. 90- 99, 2005

[6] Shuning Xing, Fangai Liu, Jiwei Wang, Lin Pang, Zhenguo Xu, "Utility Pattern Mining Algorithm Bases On Improved Utility Pattern Tree", In 8th international Symposium On Computational Intelligence and Design, pp.258 – 261,2015

[7] Serin Lee, Jong Soo Park, "Top-K High Utility Itemset Mining Based On Utility List Structures", In Proceedings of IEEE International Conference on Data Mining (ICDM), Maebashi, pp. 101 - 108, 2016.

[8] Y. Liu, W. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets", In Proceedings of the 9th PacificAsia Conference on Knowledge Discovery and Data Mining, Springer, Vol. 3518, pp. 689-695, 2005

[9] M. Liu and J. Qu, ―Mining High Utility Itemsets without Candidate Gene ration,‖ in Proc. of ACM Int'l Conf. on Information and Knowledge Management, pp. 55 - 64, 2012

[10] V. S. Tseng, C. Wu, B. Shie, and P. S. Yu, ―UP-Growth: An Efficient Alg orithm for High Utility Itemset Mining , ‖ in Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, pp. 253–262, 2010

[11] T.-P. Hong, C. H. Lee, and S. L. Wang, ''Effective utility mining with the measure of average utility,'' Expert Syst. With Appl., vol. 38, no. 7,pp. 8259–8265, 2011. doi:10.1016/j.eswa.2011.01.006