© 2018 IJSRSET | Volume 4 | Issue 2 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
National Conference on Advanced Research Trends in Information and Technologies (NCARTICT-2018), Department of IT, L. D. College of Engineering, Ahmedabad, Gujarat, India In association with International Journal of Scientific Research in Science, Engineering and Technology

FEA-HUIM: Fast and Efficient Algorithm for High Utility Item-Set Mining Using Novel Data Structure and Pruning Strategy

Suresh B. Patel[1], Mahendra N. Patel[2], Dr. S. M. Shah[3]

[1]PG Scholar, Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India
[2]PG Scholar, Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India
[3]Computer Engineering Department, Government Engineering College, Modasa, Gujarat, India

ABSTRACT

The aim is to recognize the item sets from transaction databases that direct the high profit of the business. It identifies groups of items that are brought together that earn a high profit. It can help the owner to earn more by promoting the sales of high utility items, so High Utility mining has attracted significant attention from the researchers. A number of algorithms have been designed to mine high-utility item-sets using various approaches and various data structures. However, it is necessary to improve the existing methods in terms of execution time and memory consumption. All previous high utility item-set mining algorithms like two-phase, HUI-Miner, FHM, mHUI-Miner scan the database multiple times. From the observation that we identified the performance of the algorithms can be improved by reducing the database scanning frequency and cost. In previous algorithms like HUI-Miner and mHUI-Miner, performs a time-consuming utility lists join operation on item-sets. In this research we propose a novel data structure Item Utility Matrix with Index vector and efficient procedure to join the utility list. We also propose a transaction aggregation to reduce the size of utility list. Our proposed algorithm outperforms the previous methods in execution time required.

Keywords: Data Mining, High Utility Item-set, Transaction Weighted Utility, Item Utility Matrix, Index Vector.

I. INTRODUCTION

In recent days, there is rapid growth in producing data in the world. The same conventional methods and processing power of assessing and examining the data do not follow this rapid growth. Due to this limitation, a large amount of data is still kept without used. Data mining is a research area that tries to overcome these problems and proposes some solutions for the extraction of important and useful information from this hugeset of data. The process for extracting useful information from large amount of data is known as Data Mining. In other words, we can say that data mining is the procedure of mining knowledge from data.

The rapid growth of database methods facilitates the store and use of large data from corporate sector, government offices, and scientific organizations. How to find out important and useful information from various databases has received significant attention, which results in the sharp rise of related research areas. Among this area, the high-utility item set mining problem is one of the most important, and it is derived from the well-known problem frequent item-set mining problem[1].

Frequent Item-set Mining (FIM) is a famous data mining task. For a transaction database, FIM consists of finding out frequent item-sets, i.e., set of items (Item-sets) appearing frequently in the transaction database[2][3], FIM is crucial to many applications. A conventional application of FIM is market-basket analysis. In this context, frequent item-sets are showing and then used by retail store managers to co-promote frequently purchased item-sets[2]. Though much work has been done on FIM, a fundamental limitation of FIM is, it assumes that in each transaction only item can appear or not regardless of number of quantity and all items have the same importance (weight, unit profit or value). These

two things do not hold in the real world. For example, consider a database of customer's transactions. It is casual that a customer will buy one or more unit of the same item (e.g., a customer may buy several pouches of milk), and all the products do not have the same profit (e.g., selling a shampoo earn more profit than selling a juice-bottle). Conventional FIM algorithms do not consider purchase quantities and profit of items. Thus, FIM algorithm would not consider this useful information and only find frequent item-sets, rather than finding those earning a more profit. As a result, many monotonous frequent item-sets generating a low profit may be discovered, and many rare item-sets yielding a high profit may be missed. To address this type of issue, the problem of High-Utility Item-set Mining (HUIM) has been defined [4][5][6] as "Derived the item sets from the transaction database yields the high profit" As opposite to FIM[2][3], HUIM considers the item quantity and each item profit(e.g., unit profit). The objective of HUIM is to find out the item-sets having a high-utility (a high importance, such as a high profit), that is High-Utility Item-sets. The High-utility Item-set mining has been developed as an significant research area of data mining in current years and has inspired several other important data mining tasks such as high-utility sequential pattern mining[4].

II. PROBLEM BACKGROUND

A. Preliminary

Let I = {i1,i2,i3,…in} be a set of single items. A transaction database DB that consists of a transaction table and a utility table. The transaction table contains a set of transactions {T1,T2,T3,….,Tm}. Where Tid is the unique transaction identifier for each transaction. Each transaction is a subset of I and counts value is associated with each item in the transaction. The utility table stores all the utility values for each item i in I.

Table 1. Transaction Database

Tid	**Transactions**				
T1	c 2	b 1	e 1	-	-
T2	a 3	e 2	g 1	b 4	-
T3	a 1	b 2	c 3	d 4	e 5
T4	f 3	g 1	-	-	-
T5	b 1	a 1	d 1	-	-

Table 2. Utility table

Item	a	b	C	d	e	f	g
Profit	5	1	3	4	2	1	2

Definition 1: Transaction Database.
Let I be an item-set (symbols). A transaction set in transaction database DB = {T1, T2, T3,…Tm} such that for all individual transaction Ti , Ti ∈ I and Ti transaction has a unique identifier i called its Transaction id. The Profit value p (i) in Utility Table associated with i ∈ I is known as an external utility. For each transaction Ti in transaction table such that i ∈ T, a positive number q (i, Ti) is known as the internal utility of i (e.g. In Transaction Ti it is the purchase quantity of item i).

Example 1. Consider the database in Table 1 & 2. This database has five transactions like T1, T2, T3, T4, and T5. In T2 Transaction items a, e, g, b exist with an internal utility 3, 2, 1 and 4 respectively. The external utility of these items is 5, 2, 2 and 1 respectively.

Definition 2: Utility of an Item in a Transaction.
In the transaction Ti the utility of an item i is $u(i, Ti) = q(i, Ti) \times p(i)$

Example 2. The utility of item a in T2 is $u (a, T2) = 5 \times 3 = 15$.

Definition 3: Utility of an Item-set in a Transaction.
The utility of an item-set X (a group of items $X \subseteq I$) in a transaction Ti is denoted as u(X, Ti) and defined as $u (X, Ti) = \sum_{i \in x} u(i, Ti)$

Example 3. The utility of the item-set {a, c} in T2 is $u ((a, e), T3) = u (a, T3) + u (e, T3) = 1 \times 5 + 5 \times 2 = 10$.

Definition 4: Utility of an Item-set in a Database.
The utility of an item-set X is denoted as u(X) and defined as $u(X) = \sum_{Ti \in g(x)} u(X, Ti)$, where g(X) is the set of transactions containing X.

Example 4. The utility of the item-set {b, e} in database is
$u(b, e) = u((b,e),T1) + u((b,e),T2) + u((b,e),T3)$
$= u(b,T1) + u(e,T1) + u(b,T2) + u(e,T2) + u(b,T3) + u(e,T3)$
$= 1 + 2 + 4 + 4 + 2 + 10$
$= 23$

Definition 5: Problem Definition.
The problem of high-utility item-set mining is to discover all high-utility item-sets. An item-set X is a high-utility item-set if its utility u(X) is no less than a user-specified minimum utility threshold MinUtil given by the user. Otherwise, X is a low-utility item-set.
High Utility itemset = $X \subseteq I$ where $u(X) \geq$ MinUtil

Example 5. itemset X = (b,e) and MinUtil=15. u(b,e) is 23 greater than MinUtil so X is high utility itemset.

B. Challenges in HUIM
The problem of HUIM is widely accepted as, tougher than the problem of FIM. In FIM, the downward-closure-property states that the frequency (support) of an item-set is anti-monotonic [2], Means, supersets of an infrequent item-set are infrequent and subsets of a frequent item-set are frequent. This property is called the Apriori property. It is very powerful to trim (prune) the search space. But in High-Utility-Item-set-Mining, the utility of an item-set is neither monotonic nor anti-monotonic. That is, a High Utility Item-set may have a superset or a subset having a lesser, equal or more utility [7] [14] [15] [16]. Thus, methods that have been used in FIM to trim the search space based on the downward-closure-property of the support cannot be directly applied in High-Utility-Item-set-Mining, for trim the search space.

For example, consider the following transaction database and utility table

Table 3. Transaction Database

Transaction	a	b	c	d
T1	3	0	2	4
T2	0	4	1	0
T3	4	1	3	1
T4	1	1	0	1
T5	0	6	2	0

Table 4. Utility Table

Item	a	b	c	D
Profit	5	7	2	1

Consider User Specific threshold MinUtil = 45. The utility of item set X = {a, c, d} is 50 which is more than the MinUtil so X is the high utility item set. Another itemset Y={c}. utility of Y is 18 less than MinUtil so Y is not high utility itemset event Y is subset of X. now consider itemset P={a,b,d} Utility of P is 41 ≤ MinUtil and subset of P is Q={b} utility of Q is 84 >= MinUtil so P is not a high utility itemset even it is superset of high utility itemset Q . so more challenges task in the problem of HUIM is the prune the search space.
In the problem of HUIM uses a measure called the Transaction-Weighted-Utility (TWU), which is an upper bound of transaction utility of item-set and it is an anti-monotonic.

Definition 6: Transaction Utility
The transaction utility of a transaction Ti is the summation of the utilities of every item in transaction Ti that is $TU(Ti) = \sum i \in Ti\ u(i, Ti)$. In another word, the transaction utility of a transaction is the total profit made by that transaction.

Example 6. The Transaction utility of transaction T1 is
TU (T1) = u (c, T1) + u (b, T1) + u (e, T1)
= 6 + 1 + 2 = 9

Definition 7: Transaction Weighted Utility of an Item-set.
Let an item-set X. The Transaction-Weighted-Utility (TWU) of X is the sum of the transaction utilities TU of transactions Ti containing X and is denoted as TWU(X). Formally, $TWU(X) = \sum_{x \in Ti} TU(Ti)$. The TWU signifies the total profit made by the transactions holding the item-set X.

Example 7. The TWU(b) = TU(T1) + TU(T2) + TU(T3) + TU(T5)
= 9 + 25 + 42 + 10
= 86

The Transaction-Weighted-Utility of an item-set is the overestimation of the actual utility of an item-set, $TWU(X) \geq u(X)$ utility. TWU is anti-monotonic, i.e. $TWU(X) \geq TWU(Y)$ if $X \subset Y$, means that if the TWU of item-set X is smaller than the user specify threshold, there is no need to consider all the supersets of X, because the TWU of the supersets of X are definitely to be smaller as well.

III. RELATED WORK

Much research has been done in the area of -High Utility Item-set Mining which is described here. In previous research, a variety of algorithms for finding high utility item-set like Two-Phase [4], UP-Growth [16], HUI-

Miner [15], FHM [6], mHUI-Miner [17] etc from transaction database have been proposed.

In 2005, Ying Liu proposed a two-phase algorithm [4] that find out the high utility item-set in two phase. Phase-1 calculates the TWU transaction weighted utility of each item and maintains a Transaction-weighted-Downward-Closure Property. Thus, only the group of high transaction weighted utility item-sets are appended into the candidate set at each level during the level-wise search. Phase I sometimes overestimate some low utility item-sets, but it never underestimates any item-sets. In phase II, overestimated item-sets are filtered out for that one more time database scan is performed. This algorithm demands multiple times databases scan and generates a large number of candidate-sets because of a level-wise method.

In 2010, Vincent S. Tseng proposed a UP-Growth [16] that reduce the number of candidate item-set UP-Growth uses four strategies, Discarding-Global-Unpromising items (DGU), Decreasing-Global-Node utilities (DGN), Discarding-Local-Unpromising items (DLU), and Decreasing-Local -Node utilities (DLN). Also, it defines a tree data structure, called UP-Tree, with two times database scans and conducts mining high utility item-sets. It calculates the TWU of every item by first time scanning the database and then removes the item having low TWU than MinUtil from each transaction and transaction are arranged TWU descending order. Then the transactions are appended to the UP-Tree. Also applied DGU and DGN are for reducing overestimated utilities in the same stage. After that, high utility item-sets are discovered from the UP –Tree with applying DLU and DLN. The proposed method performs in three parts: (1) Construction of Tree "UP- Tree" (2) Discover potential high utility item-sets from the "UP-Tree" by UP-Growth (3) Identification of actual high utility item-sets from the set of potential high utility item-sets.

Mengchi Liu proposed an HUI-Miner (High Utility Itemset Miner) [15] that discovers the high utility item-set without generating candidate item-set. They proposed a new data structure called utility-list which stores the utility of item-sets and also stores the heuristics information for the decision of pruning. Initially, it creates utility list for 1-itemset. Then, HUI-Miner algorithm constructs utility list for k-itemset recursively by the pairing of utility lists of k-1 item-set. For mining high utility item-set, each utility list for an item-set contains transaction id for all transaction containing the item-set, utility of the item-set in the transaction and the remaining utility value.

In 2014, Philippe Fournier-Viger proposed an FHM[6] that extends the Hui-Miner Algorithm. This is Depth First Search Algorithm. It uses the utility-lists to calculate the actual utility of itemsets. This algorithm proposed a novel data structure named EUCP (Estimated Utility Co-occurrence Pruning) to minimize the number of joins operations of utility list. Estimated Utility Co-Occurrence Structure (EUCS) maintains the transaction weighted utility (TWU) of all 2-itemsets. It is built during the initial database scans. EUCS defined as triangular matrix or hash-map. The memory usage of the EUCS structure is small. FHM is faster than HUI -Miner.

Recently in 2017, Alex Yuxuan Peng proposed a mHUI-Miner [17] used a prefix tree structure to avoid construction of unnecessary utility-lists. A pleasant property of prefix tree structure is that a path in the tree corresponds to a database transaction. mHUI-Miner creates a local prefix tree and extends an itemset by joining the utility list. It maintains the utility information in the utility list.

To improve the performance of the mHUI-Miner algorithm by reducing the database scanning cost and frequency. It also improves the performance by proposing the efficient algorithm for joining the Utility-List.

IV. PROPOSED WORK: FEA-HUIM

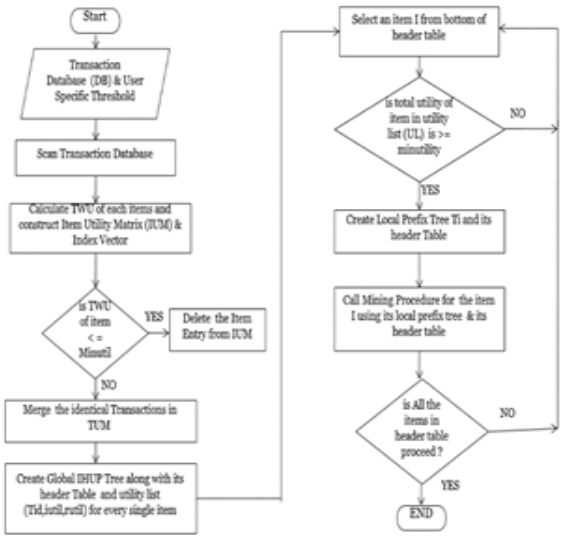


Figure 1. Proposed Flowchart of FEA-HUIM

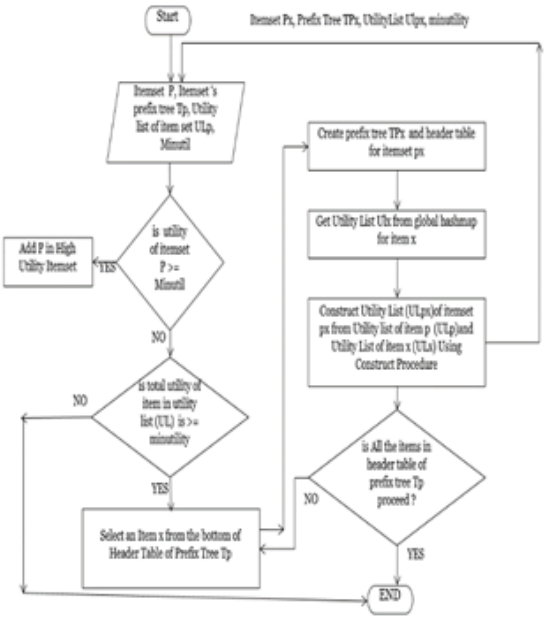


Figure 2. Proposed Flowchart of Mining Procedure

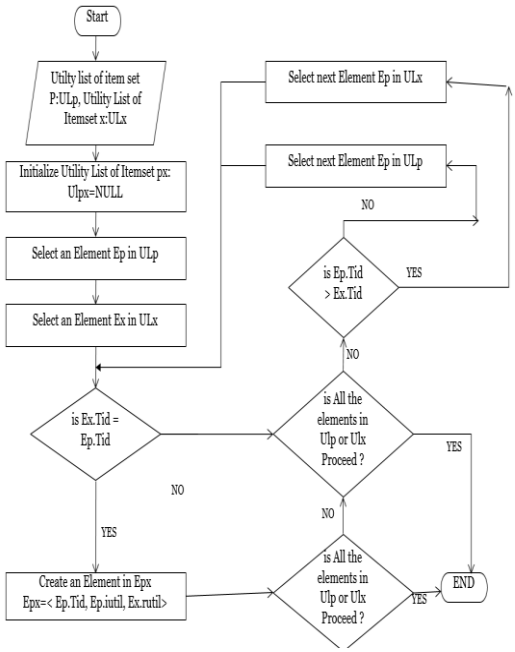


Figure 3. Proposed Flowchart of Construct Procedure

In this section, our proposed algorithm FEA-HUIM is presented. It takes the transaction database and MinUtil: user-specific threshold. The algorithm first scans the database and calculates TWU of each individual item. Simultaneously stores the database into our novel data structure IUM: Item Utility Matrix and creates index vector. The index vector stores the item column number from the IUM as TWU descending order. The IUM performs the aggregation of the similar transaction. Then creates a global prefix tree. From the global prefix tree, the Mining algorithm discovers the High Utility Item-set. The mining procedure also uses the construction procedure for joining the utility list of item-sets. We also propose the efficient construct algorithm. Our proposed flow chart as below in Figure 1 to 3.

Now we discuss our proposed system with an example, Consider the transaction database and utility table mentioned in table 1 and table 2 respectively. Then we scan the database and create our novel data structure IUM and index vector as mentioned below.in IUM first row specifies the items and second-row store the TWU of each item. Create a utility list of each item. The utility list maintains the utility information. Utility list structure is as <Tid, iutil,rutil>. The iutil field store the utility of item-set from the transaction Tid, rutil field store the utility of remaining items from the transaction.

Table 5. Item Utility Matrix

a	b	c	d	e
77	86	51	52	76
-	1	6	-	2
15	4	-	-	4
5	2	9	16	10
-	-	-	-	-
5	1	-	4	-

Table 6. Index Vector

2	1	5	4	3

Table 7. Utility List of 1-itemset

{B}			{A}			{E}			{D}			{C}		
Tid	*Iutil*	*Rutil*	*Tid*	*Iutil*	*rutil*	*Tid*	*Iutil*	*rutil*	*Tid*	*Iutil*	*rutil*	*Tid*	*Iutil*	*rutil*
T1	1	0	T2	15	4	T1	2	1	T3	16	17	T1	6	3
T2	4	0	T3	5	2	T2	4	19	T5	5	6	T3	9	33
T3	2	0	T5	5	1	T3	10	7						
T5	1	0												

In Item Utility Matrix (IUM) merge the similar transaction as performing the sum of item utility. Then read the transaction and insert into the global prefix tree as per index vector order. i.e first read the item specify by column number in the index vector first position. Then select the next item specified by the column number in the index vector second position as so on. In our example for transaction T1 inset the items b-e-c in order. Same way inserts all transaction in the global tree. Also, maintain the header table as below figure.

Item	TWU
b	86
a	77
e	76
d	52
c	51

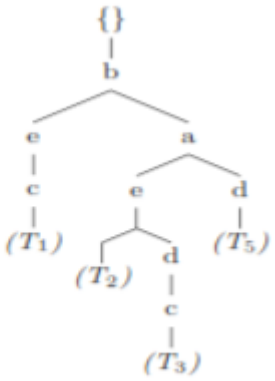
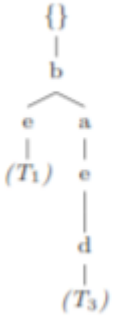


Figure 4. Global Tree

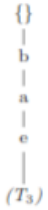
In the next step select the item from the bottom of the header table i.e c and create a local prefix tree and its header table then recursively add the item in the current item set if its total utility is larger than the MinUtil threshold. i.e after c fetch the item d from the bottom of the header table and create a local prefix tree, header table, utility list by joining the utility list of item-set c and utility list of item d.

Similarly, perform the same procedure for all the remaining items from the header table of global prefix tree T. Finally discover all the high utility item-sets are {cdea} utility is 40, {cdeab} Utility is 42 and {eab} Utility is 40.



Item	TWU
b	86
a	77
e	76
d	52

Figure 5. local prefix tree and header table of 'c'



Item	TWU
b	86
a	77
e	76
d	52

ULcd

Tid	Iutil	Rutil
T3	25	17

Figure 6. local prefix tree, header table and Utility List of 'cd'

V. COMPARISON AND DISCUSSION

In the previous research like Two-Phase, FHM, mHUI-Miner scan the database two times. Database scanning cost is high so it consumes must time for database scanning. Two-Phase, Up-Growth algorithm generates a number of candidate itemset then find out the actual high utility item set so it degrades the performance of the algorithms. In HUI-Miner, candidate item set is not generated but it performs the costly join operation of utility list. Recently, mHUI-Miner has been proposed that resolve the problem of candidate set generation and minimize the number of the join operation. For the problem of High Utility Item-set Mining mHUI- Miner outperforms the previous. Still, in mHUI-Miner database scan multiple times so it is time-consuming while in our proposed algorithm it scans the database only once. In mHUI-Miner to generate global tree perform the sorting operation on items for each transaction to insert. While in our proposed method it is just read the item as per order mentioned by index vector. It also observed that in mHUI-Miner construct procedure each entry in the one utility list is compared with all entry of another utility list so the complexity of construct procedure in the mHUI-Miner is the O (mn) where m & n is the number of transaction entries in the utility list of item-set x and item-set y. But due to transaction aggregation the size of utility list can be reduced, so the complexity of our proposed construct algorithm is at most O (m + n). Roughly the overall time complexity of mHUI-Miner algorithm is $O(2T_{db}I^2mn)$ while over proposed algorithm is $O(T_{db}I^2(m+n))$ where T_{db} is the database scanning cost, I is the number of items, m & n are the number of transaction entries in utility lists. From the above fact that the proposed algorithm is faster than the mHUI-Miner.

The proposed algorithm yet to be implemented in java platform and will perform the experiment on Retails, Chainstore, Accident, Mushroom datasets for various utility threshold. It will prove that our proposed algorithm is better in terms of execution time than the previous mHUI-Miner.

VI. CONCLUSION

We find out that most of HUIM algorithm consume the time for database scanning, unnecessary generating candidate set. Some algorithms consume memory and

time for construction of un-necessary utility list and joining this utility lists. Performance of the High Utility Item-set mining algorithm can be improved by, reducing database scanning cost, frequency, and efficient Pruning Strategy. We propose the algorithm that scans the database only once. Also, propose the transaction aggregation that reduces the size of utility list and fast construction operation for joining the utility list.

Roughly the overall time complexity of mHUI-Miner algorithm is $O(2T_{db}I^2mn)$ while over proposed algorithm is $O(T_{db}I^2 (m+n))$ where T_{db} is the database scanning cost, I is the number of items, m & n are the number of transaction entries in utility lists. From the said theoretical analysis we conclude that our proposed algorithm outperforms the previous algorithm with respect to execution time.

VII. REFERENCES

[1]. J. F. R. Ceglar, Aaron, "Association mining," in ACM Computing Surveys (CSUR), 2006, pp. 5–5.

[2]. S. R. Agrawal R, "Fast algorithms for mining association rules in large databases," in Proceedings of the 20th international conference on very large databases, 1994, pp. 487–499.

[3]. & M. Han, J., Pei, J., Yin, Y., "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data Min. Knowl. Discov., vol. 8, no. 1, pp. 53–87, 2004.

[4]. Y. Liu, W. Liao, and A. Choudhary, "A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 253–262.

[5]. M. Liu and J. Qu, "Mining High Utility Itemsets without Candidate Generation Categories and Subject Descriptors," in Proceedings of the 21st ACM international conference on Information and knowledge management, 2012, pp. 55–64.

[6]. P. Fournier-Viger, C. W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," Springer, Cham. pp. 83–92, 2014.

[7]. V. S. Tseng, C. W. Wu, P. Fournier-Viger, and P. S. Yu, "Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases," IEEE Trans. Knowl. Data Eng., vol. 28, no. 1, pp. 54–67, 2016.

[8]. J. B. Ahmed CF, Tanbeer SK, "Mining high utility web access sequences in dynamic web log data," in Proceedings of the international conference on software engineering artificial intelligence networking and parallel/distributed computing, IEEE, London, UK, 2010, pp. 76–81.

[9]. T. V. Liu Y, Cheng C, "Mining differential top-k co-expression patterns from time course comparative gene expression datasets," BMC Bioinform, vol. 14. p. 230, 2013.

[10]. Q. Li, "Data Mining Association Analysis Algorithm." Harbin: Harbin Engineering University, 2010.

[11]. Y. Q. Lin, "A Review of Association Rules Mining Algorithm[J]," Softw. Guid., vol. 11, pp. 27–29, 2012.

[12]. Z. W. Chi, X., & Fang, "Review of association rule mining algorithm in data mining," in In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, 2011, pp. 512–516.

[13]. Y. D. (2001) Pei J, Han J, Lu H, Nishio S, Tang S, "H-Mine: hyper-structure mining of frequent patterns in large databases.," in In: Proceedings of the 2001 IEEE international conference on data mining, IEEE, San Jose, CA, 2001, pp. 441–448.

[14]. C. F. Ahmed, S. K. Tanbeer, B. Jeong, and Y. Lee, "Mining High Utility Patterns in Incremental Databases," IEEE Trans. Knowl. DATA Eng., vol. 21, no. 12, pp. 656–663, 2009.

[15]. M. Liu and J. Qu, "Mining High Utility Itemsets without Candidate Generation Categories and Subject Descriptors," in Proceedings of the 21st ACM international conference on Information and knowledge management, 2012, pp. 55–64.

[16]. V. Tseng, C. Wu, B. Shie, and P. Yu, "UP-Growth: an efficient algorithm for high utility itemset mining," in Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 253–262.

[17]. P. R. Alex Yuxuan Peng, Yun Sing Koh, "mHUIMiner: A Fast High Utility Itemset Mining Algorithm for Sparse Datasets," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2017, pp. 196–207.