Mining high-utility itemset using weighted matrix-based Apriori algorithm

Muhammad Haris*, Abdus Salam†, Zia Ur Rahman‡

Abstract

Data mining is the process of discovering interesting and useful patterns and relationships in large sets of data. A recently published weighted matrix-based Apriori algorithm introduced new dimensions in the field of mining frequent patterns. The algorithm scans the transaction database and generates a 0-1 transaction matrix for getting the weighted support and confidence. The items and transactions are weighted to reflect the importance of the transaction database. The algorithm uses only minimum support as an interesting measure, often low utility items are taken as frequent itemsets, which is not beneficial for users. The field of mining high utility itemsets consider interesting utility factors like profit, cost, etc. while finding frequent patterns. This paper proposes a modified algorithm for mining high-utility itemset using a weighted matrix-based Apriori algorithm. Transaction database is transformed into a quantity transaction database. Transaction utility is calculated by multiplying each item's internal utility with its external utility available in that transaction. The weight support and utility of items are calculated to discover high-utility frequent itemsets. The exploratory outcomes show that the modified algorithm Mining high-utility itemset using a weighted matrix-based Apriori algorithm achieves good performance in the generation of high utility frequent itemsets.

Keywords: Data mining, Utility Mining, Frequent itemset, Weight matrix

Introduction

Association Rule Mining (ARM) is an important field of data mining, discovers interesting associations and relationships between large datasets (Badhon, B., et al., 2021). ARM discovers how many times an item set occurs in a transaction database. A common example is market basket analysis. Market basket analysis is a basic procedure used by large retailers to find associations among the purchased items. It permits retailers to discover relationships among the items that customers often buy together. Frequent itemset mining is the first step of ARM. The occurrence of an itemset in the database is higher than or equal to some minimum threshold then it is known as frequent itemset. Once all the

* Department of Computing and Technology Abasyn University, Peshawar, Pakistan, hariskhan8181@gmail.com

† Department of Computing and Technology Abasyn University, Peshawar, Pakistan, dr.salam@abasyn.edu.pk

‡ Department of Computing and Technology Abasyn University, Peshawar, Pakistan, ziacsepk@gmail.com

frequent itemsets are generated, one can proceed by iterating over them, one by one, enumerating through all the possible association rules.

Two main approaches are used to generate Frequent Itemsets (FIs). The first approach finds FIs by generating and pruning candidate itemsets, and the second approach produces FIs without generating candidate itemset. The Apriori algorithm is a well-known method representing the first approach (Sharma, A., & Tripathi, K., 2021) and the FP-Growth technique represents the second approach (Shawkat, M., et al., 2021). ARM and its corresponding algorithms work by considering only frequent itemsets and they do not consider the interesting utility factors like profit, cost, etc.

The problem of High Utility Itemset Mining (HUIM) was formulated to tackle this problem, which is an important technique for knowledge discovery (Chan et al., 2003). Several HUIM algorithms have been suggested (Yun et al., 2020), (Nguyen, L. T et al., 2020), (Hong, T. P. et al., 2020). The purpose of HUIM is to calculate all item sets that have a higher utility that exceed the user-specified minimum utility threshold (for example, all sets of goods purchased together that earn higher profits). HUIM takes input a database of transactions with weight and quantity. It analyzes the itemsets purchased by the customers that earn enough revenue for the retailers. The deficiency in HUIM is that, it finds out the interesting itemset with high utility without considering the frequency of items within the data, therefore, often that items may be generated as high utility itemsets which have a low frequency (Nguyen, L. T et al., 2020).

In this paper, we proposed a modified weighted matrix-based Apriori algorithm for finding high utility frequent itemsets (HUFIs). The paper is structured as follows. Preliminaries shortly introduce the idea of ARM, FIs and a few basic terms related to HUIM. Then the related work is discussed. Following the relevant work, the details about Mining high-utility itemset using a weighted matrix-based Apriori algorithm are discussed. In the end, experiment results are analyzed to measure the performance of the designed algorithm.

Preliminaries

In this part, initial, we talk about basic ideas related to the ARM, FI, and HUIM in brief. Then we investigate some related works.

ARM

ARM is a significant research area in the domain of data mining (Badhon, B., et al., 2021). The main goal of ARM is to discover the strong association rules between large amount of itemsets. The rules of

the association are described as A→B, where A is forerunner and B is later. It means if A appears, then B also expects to appear.

Frequent Itemset (FIs)

An item or itemsets is said to be frequent itemset , if it is equal or greater to user defined minimum support, otherwise, it is called rare itemsets.

Utility Mining (UM)

Utility mining is a new trusted scope in the field of data mining. The aim of UM is to determine itemsets that earn higher profits (high value).

Basic Terminologies Used in Utility Mining

Internal Utility (X)

X represent the count of each item present in the transaction. In Table 1 transaction T_3 has (A, 5), (B, 3), and (C, 4) are the internal utilities of the itemsets.

External Utility (Y)

Y is the marginal profit of each item available in the transaction. Table 2 (A,8), (B, 80), and (C, 30) denote its profit associated with it.

Table 1: Quantity transaction database

Tid	A	B	C
1	10	1	5
2	0	4	3
3	5	3	4
4	0	6	0

Table 2: External utility values

Items	External utility
A	8
B	80
C	30

The item or itemset Utility (U_t)

The U_t of an item or itemset is the multiplication of its X and Y. It can be represented as $U_t = X * Y$. Eg., U_t (A, T_1) = 10 * 8 = 80, U_t (B, T_1) = 1 * 80 = 80, U_t (C, T_1) = 5 * 30 = 150.

Utility of an item or itemset for the Database (U_tD)

The Utility of an item or itemset for the database is the multiplication of its quantity and its corresponding external utility. For example, $U_tD(A) = 15 * 8 = 120$, $U_tD(B) = 14 * 80 = 1120$ and $U_tD(C) = 12 * 30 = 360$.

Transaction Utility (TU_t)

Transaction utility is calculated by multiplying each item's X with its Y available in that transaction. Eg., $TU_t(T1) = 10 * 8 + 1 * 80 + 5 * 30 = 310$.

High Utility Itemset (HUIs)

If the utility of itemset exceeds the user-defined minimum utility threshold, then the itemset is called a HUIs.

Related work

In general, two different approaches are used to generate Association rule mining. First, the algorithm would have a candidate itemset generation and second, the algorithm would not have a candidate itemset generation. A well-known Apriori algorithm is the first algorithm developed for mining FIs (Rakesh Agrawal & Ramakrishnan Srikant, 1994). There are some inherent deficiencies of this algorithm. The algorithm needs to scan the entire database for finding FIs repetitively. If the minimum support value is kept low on large itemsets then it needs very high computations. To improve the efficiency of this algorithm, many types of algorithms have been proposed. In (Park, J. S et al., 1995) Direct Hashing and Pruning algorithm was proposed to improve the traditional Apriori algorithm. Unwanted itemsets can be efficiently removed. Hash table and bit vector are implemented to reduce the candidate itemsets. But this algorithm works is small datasets otherwise it takes more memory and time.

In (Yu, W. et al., 2008) a new Apriori-gen operation, which creates the candidate 2-itemsets called Reduced Apriori Algorithm with Tag (RAAT) was utilized to further upgrade the Apriori algorithm by reducing pruning operation. In (Chang, R. et al., 2011) a novel algorithm formed on the inadequacy of Apriori called an APRIORI-IMPROVE algorithm was proposed. The algorithm does not create candidate sets. The performance study shows that APRIORI-IMPROVE is efficient than the Apriori algorithm. In (Yang, Q. et al*.,* 2018) a Matrix Based Apriori Algorithm used to upgrade the performance of the Aprori. In this method, the candidate itemset generation was generated with a transactional Boolean matrix. Initially, it takes the logic of classical Apriori idea to find 1-frequent itemset and then it uses a Boolean matrix to find 2-itemsets, 3-itemsets with the transactional weight, and Boolean matrix. So it is highly efficient than the original approach.

In (Sun, L. N., 2020) a novel weighted matrix based Apriori algorithm for transaction database was recently being published to improve the performance of the Apriori algorithm. Initially, it scans the

dataset to construct a 0-1 transaction matrix. This matrix is used to discover association rules as well as significance to the user. It has shown remarkable improvements as compared with the original approach. The basic limitation of these algorithms is that it neglects the paramount information about the occurrence of the frequency of items in every transaction and the interesting U factors like profit, cost, etc.

HUIM was proposed in (Chan et al., 2003) to mine high utility itemsets in a transaction database. Recently, a variety of works have been done to mine HUIs. It has been applied to different valuable applications which is very useful for human life. In previous studies, numerous approaches were proposed on the utility mining problem. In (Yao, H. et al., 2004) a fundamental approach of utility mining was proposed, it gives a basic idea of utility bound property, support bound and, mathematical model of utility mining. This model was taken as a basement for many research works in utility mining. In (Liu, Y. et al., 2010) the same mathematical model of utility mining was converted into two different algorithms, i.e. utility mining and utility_H algorithms. Here, they used various utility constraints and different pruning strategies. In (Shankar, S. et al., 2009) an enhanced version of the utility mining algorithm was proposed, i.e. Fast Utility mining (FUM) algorithm. The original utility mining algorithm generates lots of candidate itemsets and it does not handle the redundant itemsets. That problem was efficiently handled by fast utility mining (FUM) algorithm.
In utility mining, a tree-like structure was used for extracting high-value itemsets with different cutting strategies for candidate itemsets. Utility pattern tree (UP – tree) like is a tree-like data arrangement that can be used to keep the details of high value itemsets. By using UP-tree, a new algorithm was proposed called Utility Pattern Growth (UP - Growth) algorithm (Tseng, V. S. et al., 2010) It uses some concepts of FP-tree. UP-Growth is better than all other previous algorithms. By continuing the research, a new algorithm was proposed with updated pruning strategies, i.e. UP-Growth+ algorithm (Tseng, V. S. et al., 2013). Another approach to keep the details of high value itemsets called utility list structure. It contains information of high utility and pruning status of that item. This list structure helps to propose an algorithm called HUI – Miner (Liu, M. et al., 2012). In this approach, sorting strategies used in TWU, so it outperforms other algorithms. To lessen the quantity of candidate itemsets FHM has been proposed (Fournier-Viger, P. et al., 2014).

Duong et al. proposed new algorithm ULB-Miner which was faster than FHM that uses an efficient method. It creates a list in segments for the utility to lessen the cost of utility list creating (Duong,

Q. H. et al., 2018). In (Nguyen, L. T. et al., 2019) proposed an method named modified efficient HUIM (MEFIM) that depends on a new tight dataset format which can efficiently find out the desired itemsets. They then upgraded the MEFIM algorithm and introduced iMEFIM algorithm (Vo, B. 2020), which is more efficient than MEFIM in deals of processing time and storage of memory. In (Vo, B. 2020) CoHUI-Miner was proposed to decrease the requirement of space usage, and improve the mining process performance for correlated CoHUIs.

Mining high-utility itemset using weighted matrix based Apriori algorithm

This part will address the idea of improving the Apriori algorithm, its application in discovery high utility FIs. The procedure is discussed below.

Weighted matrix based Apriori algorithm

Transaction database T is given, which contains m number of transactions and n numbers of items. Transaction Matrix (TM) Z can be constructed, written as follows:

$$Z = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

where: $a_{ij} = \begin{cases} 1 & a_{ij} \in T_i \\ 0 & a_{ij} \notin T_i \end{cases}, i = 1, 2, \dots, m, \ j = 1, 2, \dots, n;$

The rows of the TM Z are equivalent to a transaction, and columns are equivalent to items involved in the database T. For explanation, $I = \{I_1, I_2, \dots, I_n\}$, is represented as total itemsets in database T, where n is the items quantity, The $P(I_j)$ and $W(I_j)$ is the probability and weight of the item I_j respectively. The probability and weight are calculated as shown in Equations 1 and 2.

$$P(I_j) = l/m \qquad (1)$$

$$W(I_j) = 1/P(I_j) \qquad (2)$$

Where, l stands for occurrences of item quantity in the set of transactions, namely, in the above matrix the number of 1 in the j column and m indicates to the number of transactions records. In the transaction data set T_k stands for the k_{th} record, the average weight of total items contained is represented by $Wt(T_k)$, where j = 1, 2, ⋯ , n. The final equation is shown in Eq. (3).

$$Wt(T_k) = \sum_{j=1}^{n\&I_j \in T_k} W(I_j) \,/\, |T_k| \qquad (3)$$

The weight support of the item is denoted as W_s as shown in equation (4). Equation (4) represents the weight percentage of transactions that hold the item in total transaction weights. As indicated by the constructed matrix Z, it is not difficult to understand that the transaction that holds the item, as given in equation (4). Where S stands for any item in the database T.

$$W_s = \sum_{k=1}^{m\&\, S \in T_k} Wt(T_k) \,/\, \sum_{k=1}^{m} Wt\,(T_k) \qquad (4)$$

So, by scanning the database T the weight matrix can create the 0–1 TM. Weight is assigned to items and transactions and then total weight support of items, consequently receiving the frequent itemsets (Sun, L. N., 2020).

Improvement in weighted matrix based Apriori algorithm

The following additional steps are incorporated in the proposed weighted matrix based Apriori algorithm to accomplish this task:

1. Transform the transaction database (T) into a quantity transaction database (T_Q). The quantity transaction database (T_Q) will contain the additional information of quantity sold and value of each item respectively.
2. Next, we find the U of every item for discovering the HUIs. Finding the utility U of each item I_j in every transaction T_c will be calculated using Equation 5,
   i. $U(I_j, T_c) = X \times Y \qquad (5)$

   where X represents internal utility, which signifies the count of every item present in the dataset transection and Y represents external utility which is the marginal profit of each item.
3. Next, we compute the sum of items utility present in all transactions, shown in Eq. 6.

$$U(F, T_c) = \sum I_j \in F\, U(I_j, T_c) \qquad (6)$$

where $F \subseteq I$ is an itemset. If k items are included in itemset called k-itemset. The U in itemset F in a transaction T_c, is indicaated as in the above equation if $F \subseteq T_c$.

Improved algorithm framework

An improved weighted matrix based Apriori algorithm for finding HUFI comprise of the following steps.

1. Scan the database (T_Q) and construct the TM.
2. Item weight $W(I_j)$ and transaction weight $Wt(T_k)$ are calculated.
3. Calculate the utility (U) of each item (I_j) in every transaction (T_c) using Eq. 5.
4. Then calculate the sum of item's utility.
5. As indicated by the constructed matrix Z, a candidate set C_1 is achieved.
6. The weight support (W_s) and utility (U) of every item in C_1 are calculated to find out the high-utility frequent-1 itemsets L_1 with the minimum support and minimum utility.
7. To discover L_k, C_k a set of candidates with k length is created using the join step with L_{k-1}, and then the item's weight support and utility (U) in C_k is calculated while no new FIs L_k is found.

With regards to clear articulation, the suggested algorithm is illustrated in form of the flow chart in Fig. 1.

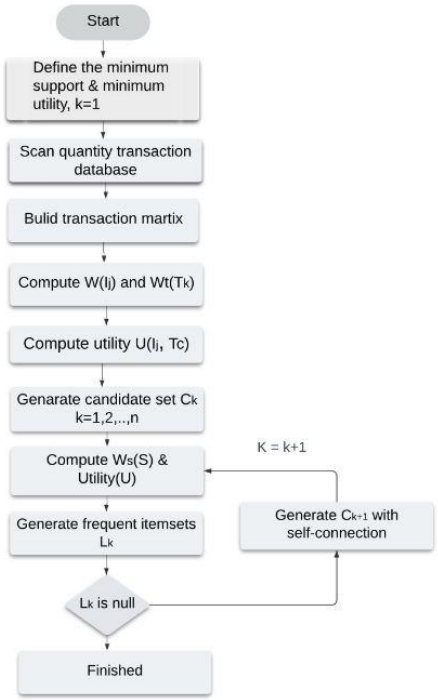

Fig. 1: Flow chart of proposed algorithm

Arithmetic example

An arithmetic example of our proposed algorithm is discussed in this section. Table 3 is an example of a quantity transaction database T_Q

containing four transactions and five items. Where I is the set of items, and its internal utilities (purchase quantities) are also given in Table 3. The external utility of these items (e.g., unit profits) are given in Table 4 where we assume the minimum support is 0.30 and the minimum utility is 20.

Table 3: Quantity transaction database

Tid	A	B	C	D	E
1	2	0	5	3	0
2	0	4	3	0	3
3	5	3	4	0	6
4	0	6	0	0	4

Table 4: External utility values

Items	External utility
A	4
B	5
C	8
D	10
E	5

1. The transaction database T_Q is scanned to construct transaction matrix (TM) Z with four rows and five columns, and written as follows:

$$Z = \begin{bmatrix} 2 & 0 & 5 & 3 & 0 \\ 0 & 4 & 3 & 0 & 3 \\ 5 & 3 & 4 & 0 & 6 \\ 0 & 6 & 0 & 0 & 4 \end{bmatrix}$$

   The row of the TM Z is equivalent to a transaction, and columns are equivalent to items involved in the transaction database T_Q.
2. According to Eq. 1 we can get the probability $P(I_j)$ of items appearing in the T_Q. which is shown in table 5.

Table 5: Probability of items $P(I_j)$ in T_Q

I_j	$P(I_j)$
A	0.50
B	0.75
C	0.75
D	0.25
E	0.75

3. Similarly, according to Eq. 2, we can get the $W(I_j)$, as displayed in table 6.

Table 6: Weights of items $W(I_j)$

I_j	$W(I_j)$
A	2.00
B	1.33
C	1.33
D	4.00
E	1.33

4. According to equation 3, we calculate the average weight of transactions. For instance, in the first record of the transaction database T_Q (1) contain A, C, D items. Submission of the weights of these items are shown in Table 7.

Table 7: Transaction weights

Tid	$Wt(T_k)$
1	2.44
2	1.33
3	1.50
4	1.33

5. According to Eq. 4, we can get the weight support Ws of items, as an example take item A, from the TM we can see that the occurrence of item A is in the first transaction and the third transaction, so Ws(A) = 0.597. Utilizing a similar strategy to compute Ws of every item in candidate set C1, the outcome is displayed in Table 8.

Table 8: The Ws for candidate 1-itemset

C_1	Ws(S)
A	0.597
B	0.630
C	0.798
D	0.370
E	0.630

6. According to Eq.5, we can get the utility U of each item I_j in every transaction T_c, where X represents internal utility and Y represents External utility. Then calculate the sum of items utility, whose results are given in table 9.

Table 9: Utility of items in every transaction

Items	T1	T2	T3	T4	Total Utility
A	8	0	20	0	28
B	0	20	15	30	65
C	40	24	32	0	96
D	30	0	0	0	30
E	0	15	30	20	65

L1 = {A, B, C, D, E} can obtained by comparison with the minimum support 0.30 and minimum utility 20, Using the join step on L_1

the candidate set C_2 = {AB, AC, AD, AE, BC, BD, BE, CD, CE, DE} can be achieved. According to the TM using Eq. 4 and Eq. 6, we can compute the Ws and utility of candidate itemset C_2, as displayed in Table 10.

Table 10: The Ws and utility of C_2

C_2	Ws(S)	Utility(U)
AB	0.227	35
AC	0.597	100
AD	0.370	38
AE	0.227	50
BC	0.429	91
BD	0.00	0
BE	0.603	130
CD	0.320	70
CE	0.429	101
DE	0.00	0

Comparing C_2 with minimum support and minimum utility, we get L_2 = {AC, AD, BC, BE, CD. CE}. The mechanism will be repeated till the algorithm cannot locate new HUFIs.

Experimental results and analysis

Experiments were carried out to evaluate the performance of the proposed Mining high-utility itemset using weighted matrix based Apriori algorithm. They were conducted on a computer equipped with a 4rd-generation Intel(R) Core(TM) i7-4600U 2.70 GHz processor, having 8 GB RAM, operated by Windows 7 Professional 64-bit operating system. We compared the performance of the proposed algorithm with Weighted Matrix based Apriori algorithm. Both algorithms were designed and implemented using the programming language Python 3.8, with the help of Jupyter notebook.

Weighted matrix-based Apriori algorithm uses only minimum support as interestingness measure, while in our proposed algorithm both values minimum support and minimum utility are used as interestingness measures. where we assume the minimum support is 0.30 and the minimum utility is 20. The experiment compares the Weighted Matrix based Apriori algorithm and our proposed algorithm by applying five number of transactions and ten items in the implementation, see Tables 11 and 12.

Table 11: Quantity transaction database Table

Tid	A	B	C	D	E	F	G	H	I	J
1	2	0	0	0	0	1	5	0	0	3
2	0	0	3	0	0	0	0	2	0	0
3	1	4	0	2	5	2	0	1	4	0
4	2	0	2	0	0	2	0	0	0	2
5	1	0	0	2	0	1	0	0	4	0

Table 12: External utility

Items	External utility	Items	External utility
A	2	F	3
B	6	G	7
C	7	H	8
D	2	I	3
E	9	J	7

In results, it can be seen from Table 13 that even those items are also included in 1-itemset which are frequent but has very low utility, e.g. A, D in Table 14, 2-itemset AD, DF are frequent items while it is not frequent itemsets in the results of our proposed algorithm. Another important observation is that when minimum support and minimum utility is slightly increased, there is a significant drop in the number of high utility-frequent itemsets. It is observed that Weighted Matrix based Apriori algorithm has consumed slightly more memory than our proposed algorithm.

Table 13: Comparison results for 1-itemset

Items	Weighted Matrix based Apriori algorithm Ws(s)	Frequent itemset	Proposed Support Ws(s)	Proposed Utility	High-Utility Frequent itemset
A	0.788	Yes	0.788	12	No
B	0.247	No	0.247	24	No
C	0.381	Yes	0.381	35	Yes
D	0.407	Yes	0.407	8	No
E	0.247	No	0.247	45	No
F	0.788	Yes	0.788	18	No
G	0.212	No	0.212	35	No
H	0.381	Yes	0.381	24	Yes
I	0.407	Yes	0.407	24	Yes
J	0.381	Yes	0.381	35	Yes

Table 14: Comparison results for 2-itemset

Items	Weighted Matrix based Apriori algorithm Ws(s)	Frequent itemset	Proposed Support Ws(s)	Proposed Utility	High-Utility Frequent itemset
AD	0.407	Yes	0.407	12	No
AF	0.788	Yes	0.788	30	Yes
AI	0.407	Yes	0.407	28	Yes
AJ	0.381	Yes	0.381	43	Yes
CH	0.381	Yes	0.381	59	Yes
DF	0.407	Yes	0.407	17	No
DI	0.407	Yes	0.407	32	Yes
FI	0.407	Yes	0.407	33	Yes
FJ	0.381	Yes	0.381	44	Yes

Conclusions and Future Work

In this paper, we proposed a modified weighted matrix based Apriori algorithm for finding HUIs. The database T is transformed into a quantity transaction database. The database contains the additional information of quantity sold and the value of each item respectively. Transaction utility is then calculated by multiplying each item's internal utility with its external utility available in that transaction. The Ws and U of every item in C_1 is calculated to find out the HUIs L_1 with the minimum support and minimum utility. The modified algorithm Mining high-utility itemset using weighted matrix based Apriori algorithm achieves excellent performance in generation of valid HUFIs. However, the proposed method is not suitable to be used on large transaction datasets. This is due to the joining step performed during generation of each candidate itemset C_k. In the future, we will continue working to improve the proposed algorithm for large transaction databases.

References

Badhon, B., Kabir, M. M. J., Xu, S., & Kabir, M. (2021). A survey on association rule mining based on evolutionary algorithms. *International Journal of Computers and Applications*, 43(8), 775-785.

Sharma, A., & Tripathi, K. (2021). Hybrid Version of Apriori Using MapReduce. In *Mobile Radio Communications and 5G Networks* (pp. 585-592). Springer, Singapore.

Shawkat, M., Badawi, M., El-ghamrawy, S., Arnous, R., & El-desoky, A. (2021). An optimized FP-growth algorithm for discovery of association rules. *The Journal of Supercomputing*, 1-28.

Chan, R., Yang, Q., & Shen, Y. D. (2003, November). *Mining high utility itemsets*. In Third IEEE international conference on data mining (pp. 19-19). IEEE Computer Society.

Yun, U., Nam, H., Kim, J., Kim, H., Baek, Y., Lee, J., ... & Pedrycz, W. (2020). *Efficient transaction deleting approach of pre-large based high utility pattern mining in dynamic databases*. Future Generation Computer Systems, 103, 58-78.

Vo, B., Nguyen, L. T., Bui, N., Nguyen, T. D., Huynh, V. N., & Hong, T. P. (2020). *An efficient method for mining closed potential high-utility itemsets*. IEEE Access, 8, 31813-31822.

Mai, T., Nguyen, L. T., Vo, B., Yun, U., & Hong, T. P. (2020). *Efficient algorithm for mining non-redundant high-utility association rules*. Sensors, 20(4), 1078.

Sun, L. N. (2020). *An improved apriori algorithm based on support weight matrix for data mining in transaction database*. Journal of Ambient Intelligence and Humanized Computing, 11(2), 495-501.

Yu, W., Wang, X., Wang, F., Wang, E., & Chen, B. (2008, November). *Notice of Retraction: The research of improved apriori algorithm for mining association rules.* In 2008 11th IEEE International Conference on Communication Technology (pp. 513-516). IEEE.

Park, J. S., Chen, M. S., & Yu, P. S. (1995). *An effective hash-based algorithm for mining association rules*. Acm sigmod record, 24(2), 175-186.

Chang, R., & Liu, Z. (2011, July). *An improved apriori algorithm*. In Proceedings of 2011 International Conference on Electronics and Optoelectronics (Vol. 1, pp. V1-476). IEEE.

Yang, Q., Fu, Q., Wang, C., & Yang, J. (2018, June). *A matrix-based Apriori algorithm improvement.* In 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC) (pp. 824-828). IEEE.

Huang, Y., Lin, Q., & Li, Y. (2018, May). *Apriori-BM algorithm for mining association rules based on bit set matrix.* In 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC) (pp. 2580-2584). IEEE.

Yao, H., Hamilton, H. J., & Butz, C. J. (2004, April). *A foundational approach to mining itemset utilities from databases*. In

Proceedings of the 2004 SIAM International Conference on Data Mining (pp. 482-486). Society for Industrial and Applied Mathematics.

Liu, Y., Li, J., Liao, W. K., Choudhary, A., & Shi, Y. (2010). *High utility itemsets mining*. International Journal of Information Technology & Decision Making, 9(06), 905-934.

Shankar, S., Babu, N., Purusothaman, T., & Jayanthi, S. (2009, March). A fast algorithm for mining high utility itemsets. In 2009 IEEE International Advance Computing Conference (pp. 1459-1464). IEEE.

Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010, July). *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 (pp. 253-262).

Tseng, V. S., Shie, B. E., Wu, C. W., & Yu, P. S. (2013). Fellow,". *Efficient Algorithms for Mining High Utility Itemsets from Transactional Databases*", IEEE Transaction on knowledge and data engineering, 25(8).

Liu, M., & Qu, J. (2012, October). *Mining high utility itemsets without candidate generation*. In Proceedings of the 21st ACM international conference on Information and knowledge management (pp. 55-64).

Fournier-Viger, P., Wu, C. W., Zida, S., & Tseng, V. S. (2014, June). *FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning*. In International symposium on methodologies for intelligent systems (pp. 83-92). Springer, Cham.

Duong, Q. H., Fournier-Viger, P., Ramampiaro, H., Nørvåg, K., & Dam, T. L. (2018). Efficient high utility itemset mining using buffered utility-lists. Applied Intelligence, 48(7), 1859-1877.

Nguyen, L. T., Nguyen, P., Nguyen, T. D., Vo, B., Fournier-Viger, P., & Tseng, V. S. (2019). Mining high-utility itemsets in dynamic profit databases. Knowledge-Based Systems, 175, 130-144.

Vo, B., Nguyen, L. V., Vu, V. V., Lam, M. T., Duong, T. T., Manh, L. T., ... & Hong, T. P. (2020). Mining correlated high utility itemsets in one phase. IEEE Access, 8, 90465-90477.

Vo, B., Nguyen, L. T., Nguyen, T. D., Fournier-Viger, P., & Yun, U. (2020). A multi-core approach to efficiently mining high-utility itemsets in dynamic profit databases. IEEE Access, 8, 85890-85899.