International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Fast and Enhanced Approach for High Average Utility Itemset Mining with Lossy Items

J. Wisely Joe[1], S. P. Syed Ibrahim[2]**

Submitted: 21/03/2023 **Revised**: 26/05/2023 **Accepted**: 10/06/2023

Abstract: Existing utility mining algorithms consider only frequency, productivity and quantity of every item purchased for utility calculation. No attention has been given to the length of transactions. Uncovering high average utility itemsets from the transaction dataset solves the above issue. In some circumstances, things may appear in reality with a negative unit value. For example, a retail store may trade items at a loss to energize the trade of similar products. To resolve the issue, we propose an effective mechanism named appropriate average high utility itemset mining with negative utilities (AAHUIM-NU) which creates high quality decision makers. To solve the issue of data set repository and numerous sweeps of the data set, proposed algorithm involves number of lists for caching the relevant data which possesses only less storage. It utilizes a minimized transaction utility, decreased maximal utility, and generalized tight upper bounds to find out the pruning threshold and to minimize the running time and memory. Exploratory outcomes on datasets demonstrate that AAHUIM-NU is productive in terms of processing time, memory usage, and versatility. It performs well on thick datasets which has an excessive number of long transactions. The experimental results are recorded in tables and given in this paper.

Keywords: *High average-utility itemsets Negative utility, Premature pruning strategies, Tighter upper bound, Appropriate maximal utility*

1. Introduction

The intention of data mining is to list the promising information from databases that satisfy the requirements of different applications of data owners. Very popular frequent itemset mining (FIM) [1] [2] [3] algorithms for extracting association rules and recurrent itemsets in databases are apriori and frequent pattern-growth(FP-growth). These algorithms are very convenient association mining algorithms but they only analyse patterns based on the frequency of entries in a database. The components like weight and cost of item, importance or profitability of mined patterns can also be considered for prime decision making. To achieve this, high utility itemset mining (HUIM) was introduced to choose only the itemsets having calculated utility more than the threshold. The fundamental objective of the HUIM algorithm [4] is to track down high utility itemsets(HUI) by taking into account of the above said client inclinations. There were numerous algorithms proposed to extricate HUI by utilizing the above model. Specialists proposed incremental high utility pattern mining(IHUP) [5], transaction weighted utility mining (TWU-Mining) [6], isolated items discarding strategy(IIDS)[7], utility pattern growth(UP-Growth) [8], UP-Growth+ [9] algorithms. They create countless candidate sets which prompts performance degradation. Liu et al. proposed a one-stage HUI-Miner [10] algorithm, which focusses on the generation of a list structure named utility list to represent the database without any loss and can productively prune the majority of the search space. Though HUIM can disclose productive and more meaningful facts than the result of FIM, it has a key block. HUIM algorithms ignore the number of entries present in the transactions. The average utility measure in HUIM algorithms increase the utility measure by a factor proportional to the length of an itemset, has been introduced by Hong et al. to fairly compute the utility of itemsets. With the utilities and item count added together, the average utility of an itemset is determined. TPAU algorithm [11] first introduced the average utility upper bound(auub) concept and it is based on level-wise threshold increasing methodology in two phases. From that point forward, a few strategies like PBAU [12], PAI [13], HAUPGrowth [14], HAU Itemset-Tree [15], HAU Itemset-Miner [16], EHAUPM [17], MHAI [18], FHAUIM [19], TUB-HAUPM [20], and dHAUIM [21], have been emerged with effective factors to improve the results. The high average utility itemset mining(HAUIM) approaches suggested in the aforementioned publications, however, anticipate that the products have exclusively beneficial

wiselybritto@gmail.com[1], syedibrahim.sp@vit.ac.in[2]
[1] Research Scholar, SCOPE, VIT Chennai Campus Tamil Nadu, India.
[2] Professor. SCOPE.VIT Chennai Campus, Tamil Nadu, India
**Corresponding Author Email: syedibrahim.sp@vit.ac.in*

external utility. On the other hand, as stated in HUINIVMine[22], GHUM[23], FHN[24], EHIN[25], MHAUIPNU [26] , TS-HOUN[27], EHNL [28], EFIM [29], HUPNU[30], and CHN[31], databases may also include negative external utilities in many application scenarios. A chain of supermarkets might sell items with less profit or even no profit or at a loss to encourage the sale of other related items or simply to tempt customers to their retail location [22-25]. When the data sets additionally contain negative utilities, existing high average utility itemset mining HAUIM calculations might deal with the issue of fragmented detection of high average utility itemsets(HAUIs) because the upper-bound underrates the candidate itemsets. The significant difficulties we face while planning for an algorithm to deal with negative items are developing an algorithm for discovering HAUIs with negative utility items, setting a tighter average bound for pruning unpromising items and design a state-of-the art algorithm.

Appropriate Average High Utility Itemset Mining with Negative Utilities is the proposed approach in this article for efficiently discovering HAUIs with both positive and negative utility items. A novel tight bound model All of the necessary definitions are spelled out in great detail. Let I be a fixed collection of n unique things. D is a database with a collection of transactions in it. T is a transaction which contains purchased items and their quantity from a vender. Every transaction has a distinct identifier (TID) and is present in Table 1. All of the items in list (I) have the same or different integer profit as shown in Table 2. There are 6 transactions with transaction ids T1, T2…. T6. The items present in transaction T1 is {A, B, C, F} and their utilities are (1,1,1,4). The interestingness of those items may be plus or minus. The items C and F are lossy items while others have positive profit. Utility of transaction T6 is not included in any calculation as it has only negative items. All the above definitions are utilized in our AAHUIM-NU algorithm. An itemset X is supposed to be an HAUI iff its support count (SC) is greater than support threshold (ST) and its aauub is higher than the utility threshold (UT). HAUI ← {x | aauub(x) ≥ UT U SC(x) ≥ ST}. Number of times the item is present in the database is stored in S-Count List and it is shown Table 3. We introduce the concept of a Present List (P-List) as follows: p(X) = {Set of transactions chosen from database | ∀i ∈ X, T}, where D is database, T is transaction, X is an itemset and i is an item. We can see that a support set is a subset of all the transactions in which X appears. A sample P-List is given in Table 4. Maximal utility (mu)of transaction Td ∈ D is mu (Td) = max {u (i j) | i, j in Td}. Itemset's average utility upper bound is defined as auub(X) = Sum (mu (Td)) | X⊆Td. X is considered a high average utility upper bound called appropriate average utility upper bound(aauub) is proposed to reduce the dimensions for extracting appropriate average high utility itemsets(AAHUIs) with beneficial and undesirable items. To lessen the amount of database and memory consumption, a list data structure was created with two early pruning strategies to contain only the necessary information. Two early pruning strategies are provided to enhance the proposed algorithm's performance by reducing candidate set formation. The first trimming approach is based on the user given minimum threshold. The utility of the items in the transaction and the frequency of entries in the transaction are used to prune transactions. Experiments on various datasets are carried out to demonstrate the efficacy of the proposed AAHUIM-NU algorithm. The rest of this article is structured out as follows. The following section covers over the associated work. The next part explains the fundamental concepts of the HAUIM problem. In the following section, we'll go over the proposed algorithm. Experimental analysis is presented before the closing part. Conclusions and future work are presented in the concluding section.

2. Preliminaries And Problem Definition

itemset (HAUUBI) if it's auub is greater than user given UT [auub(X) ≥ UT]. The k-HAUUBI is a HAUUBI with k components.

The transaction utility list (TU-L) of T is then made up of m sorted elements, each of which represents the utility value of each item in T. The following is how the TU-L is defined and displayed in Table 5. TU-L (T) = {u_i | for any $1 \le i \le j \le m$, $u_i \ge u_j$}, where u_i denotes the utility of a unique item in T. We take T2 as an example from Table 1. Here, the items B, D and E have corresponding utilities 1,2 and 6 [u(B) = 1, u(D) = 2, u(E) = 6]. Thus, TU-L(T2) = {6,2,1}. TU-List after first level pruning is given in Table 6.

A percentage of the total transaction utility(TTU) is used to represent the user-specified minimum utility limit, whereas the utility threshold is defined as UT = δ ×TTU. We introduce the concept of a critical support count (csc) which is used for first level pruning of unpromising items. Critical support count is an integer value calculated from mu and database maximal utility list(D-MUL), given in equation (1):

$$\sum_{k=1}^{csc} mu_k \ge UT \text{ and } \sum_{k=1}^{csc-1} mu_k < UT \quad (1)$$

Consider a transaction of length m, T ∈. Then, for a k-itemset X such that X ⊆ T, the appropriate maximal utility (AMU) is stated as in equation (2):

$$AMU_k\ (T) = \sum_{j=1}^{k} u_j \quad (2)$$

where $1 \leq k \leq m$, and uj is the jth entry in TU-L (T). The aauub of k-itemset X is written as in equation (3):

$$aauub(X) = \sum \left(\frac{AMU_k(T)}{k} \right) \mid (X \subseteq T \wedge T \in DB) \quad (3)$$

Table 1. A transactional database

Item / Tid	A	B	C	D	E	F	TU	MU	RTU	RMU
T1	1	1	1			2	-3	7	8	7
T2		1		1	1		9	6	9	6
T3	1	1	1			1	1	7	8	7
T4			3	2		1	-9	4	0	4
T5	1		2				1	7	7	7
T6			2				-14	-6	-	-

Table 2. A profit table PT

Item	A	B	C	D	E	F
Profit	7	1	-3	2	6	-4

Table 3. S-Count List

Item	A	B	D	E
SC	3	3	2	1

Table 4. P-List after pruning

Itemset	Present in
p(a)	{T1, T3, T5}
p(b)	{T1,
p(d)	{T2, T4}

Table 5. Initial TU- List

Tid	TU-List
T1	{7,1}
T2	{6.2.1}
T3	{7,1}
T4	{4}
T5	{7}

Table 6. TU-List after pruning

Tid	TU- List
T1	{7,1}
T2	{2,1}
T3	{7,1}

Set of aauub of productive 1-itemsets are {(A,21),(B,20),(D,10)}. The upper bound generated by this aauub is tight when compared with previous algorithms. Let us take 1-appropriate average high utility upper bound itemset(1-aahuubi) = {A, B, D}. Extensions of element A are, {AB, AD, BD}. aauub(AB) = ((7+1) +(7+1))/2 = 8. If existing algorithms are used for calculation, auub (AB) = (7+7)/2 = 7. The following is the definition of X's itemset maximal utility list (I-MUL): I-MUL(X) = $\{mu_i \mid$ for any $i \leq j$, $mu_i \geq mu_j\}$, where mu is the maximal utility of transaction T, and $T \in p(X)$. Let D be a transactional database with n rows. Then the database maximal utility list is stated as: D-MUL(D) = $\{mu_i \mid$ for any $1 \leq i \leq j \leq n$, $mu_i \geq mu_j\}$, where mu_i is transaction's maximal utility T, and $T \in D$. The list of maximal utilities of transactions in which itemset X occurs as a subset, resulting from I-MUL and D-MUL, is called the maximal utility list of itemset X. All members are arranged in descending order of maximal utility. The empty set can be viewed as the maximal utility list D-MUL () of the database , that is, D-MUL (∅). For example, for the database in Table 1, p(AB) = {T1, T3}, support (AB) = 2, I-MUL(AB) = {7, 7} and D-MUL(D) = {7,7,7,6,4}.

The search space tree is the main strategy to represent the combinations of items to be processed further. The items are arranged by ≻ order of sorted items. Rearranging of itemsets is accomplished by growing aauub as this minimizes the quantity of candidate sets to be managed. The pruning procedure is a significant part for mining HAUIs productively. The principal pruning system depends on the utility given in the data set. The other pruning methodologies are based on the determined utility threshold (UT), counted support count (SC) which is an integer value holds number of transactions where X happens as a subset and appropriate average high utility upper bound.

Strategy 1- Pruning the negative itemsets

Strategy 2 -Pruning the search space by using Support Count

Strategy 3 -Pruning the search space by deleting transactions

Strategy 4 -Pruning an itemset's negative extensions

Strategy 5 -Pruning the search space by aauub

3. METHOD

Steps 1-3 scan the transaction database once. Then, as needed, the pruning procedures described in the previous article are employed. Items are filtered by deleting negative utility items from database and also the entire transaction if it has only negative items (steps 5-6). Construct the present list(P-List) of all distinct items and their presence in various transactions. This list is further used in all stages of algorithms to avoid unnecessary scanning of database (step 7). Then calculate the transaction utility TU of all items (step 8). Simultaneously identify maximum utility and store in mu list. Step 10 deals with the construction of TU-List for each transaction in database. This list avoids repeated scanning of database for utility calculations or further processing. In step 11, the algorithm stores the support of each item in a list called S-Count List. Sort the maximal utilities of 1-items in descending order and store it in D-MUL. Calculate aauub of X where X is an item in database with the help of calculated AMU(step 13).

Algorithm 1 AAHUIM-NU

Input: Transaction database ***DB***, Minimum Utility Threshold UT, Profit Table

Output: All AAHUIs

```
1:LI{}=ϕ
2:Read (Item i, Profit p) from PT
3:      LI{}={ LI{} U i , p<0}
4:Read Transactions from DB
5:      Remove Transaction if it has all negative utility items, ∀i∈LI
6:      Remove Negative Utility Items
7:      Generate P-List
8:      Calculate Transaction Utility TU
9:      Find Maximum Utility of Transaction mu
10:     Construct TU-L for each transaction T
11:     Construct S-Count List for item i
12:     List mu in "<" order (D-MUL)
13:     Calculate aauub(X)      //X is an unique item in database
14:Calculate TTU
15:Calculate Utility Threshold UT
16:Calculate Critical Support Count csc
17:AAHUUBI(i ,csc)
```

Steps 14-16 deal with the calculation of total transaction utility, UT and csc values. Now execute AAHUUBI function for finding promising appropriate average high utility itemsets by other pruning methods. The function initially creates extension of all possible candidate sets by exploring the candidates in a depth-first manner that includes x where k>1. For every item present in the extension, apply pruning strategy 2 and 5 by comparing the support count of item and csc values. For the qualified itemsets aahuub and UT are compared. If the itemset clears both the tests, it will be added to the consolidated list of promising AAHUIs. For itemsets prefixed by other AAHUUBIs, the same traversal method will be applied.

4. Results And Discussion

4.1. Experiments and Evaluation

We conducted numerous tests to evaluate the performance of the AAHUIM-NU algorithm. This experiment utilises seven genuine datasets from an open-source software and data mining library called SPMF[32]. The dataset's basic properties are shown in Table 7. Our proposed AAHUIM-NU algorithm is a novel approach for mining appropriate average high utility itemsets with negative utility. We considered FHN algorithm [24] and EHIN algorithm [25] for comparison studies. We examined the working efficiency of the above algorithms in terms of space and runtime for different minimal utility thresholds. For pruning unpromising elements and selecting potential HAUIs these algorithms use auub(average utility upper bound), gauub(generalised average utility upper bound), and aauub(appropriate average utility upper bound). We set a minimum utility threshold before evaluating the algorithm's performance. Required changes are implemented to all datasets in order to make them suitable as inputs to our proposed algorithm.

Table 7. Characteristics of the datasets

DATA SETS	#TRAN	#ITEMS	ATS	#PIs	#NIs	Density	TYPE
T251200D10K	10000	200	25	134	66	12.5	DENSE
CHESS	3196	76	37	51	25	48.68421	DENSE
CONNECT	67557	130	43	87	43	33.07692	DENSE

RETAIL	88162	16,470	10.3	11035	5435	0.062538	SPARSE
MUSHROOM	8124	119	23	80	39	19.32773	DENSE
ACCIDENT	340183	468	33.8	280	188	7.222222	DENSE
KOSARAK	990002	41270	8.1	27651	13619	0.019627	SPARSE

4.2. Runtime analysis on datasets

This section assesses the AAHUIM-NU and comparison algorithm's runtime performance across all datasets. In dense datasets the AAHUIM-NU method's runtime is much shorter than those of the FHN and EHIN algorithms, as demonstrated in Figure 1. The algorithm AAHUIM-NU is superior to FHN and EHIN, as shown and it can attain a level of about 20 times greater on dense datasets and about 2 times greater on sparse datasets. Because dense datasets have well-defined scanning methods and trimming procedures that are fully utilised. Analysis shows that AAHUIM-NU not only surpasses in terms of time efficiency, but also evolves more smoothly when the threshold decreases. The FHN algorithm overlooks the itemsets that do not appear in the dataset because it searches the search space of itemsets by merging smaller itemsets rather than scanning the entire dataset. The suggested AAHUIM-NU algorithm performs better since it contains strong pruning techniques.

4.3. Memory usage on datasets

The AAHUIM-NU technique utilises substantially less memory than the FHN and EHIN algorithms, as shown in the Figure 2. The memory usage of the FHN algorithm increases swiftly as the threshold value decreases, but the memory usage of the EHIN method increases gradually like the proposed method. Among these, the FHN method uses more memory since FHN saves all utility lists in memory. The suggested approach is ineffective for highly sparse datasets like kosarak and retail because they are extremely sparse and create unnecessary intermediate sets. The AAHUIM-NU algorithm outperforms FHN and EHIN as demonstrated in Figure 2. by storing only the necessary data in simple structures and never stores the complete dataset in memory after scan 1.

4.4. Comparison of Candidates generated

The candidates generated by three alternative upper bounds auub, gauub, and aauub on our AAHUIM-NU method are assessed below to determine efficiency and the findings are shown in Figure 3. for various parameter settings. The AAHUIM-NU algorithm incorporates effective pruning algorithms for trimming negative itemsets and negative transactions as an upgraded variation. A unique pruning method that eliminates all single item transactions with positive items since they are ineffective in generating viable candidates through the join operation. As a result, the pruning procedures used in our algorithm help to prune the search area and lower the number of unpromising patterns.

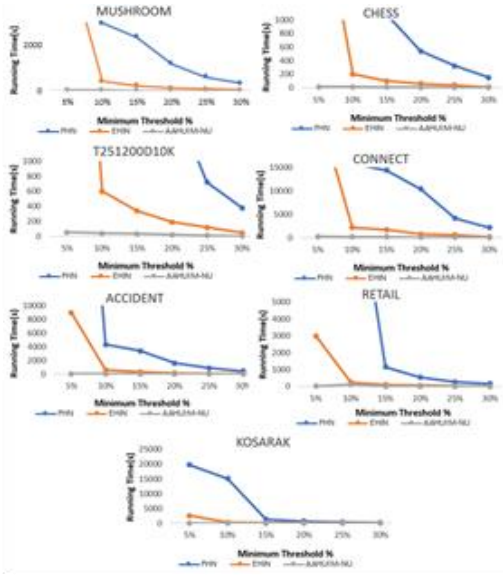

Fig 1. Runtime Analysis for different minimum

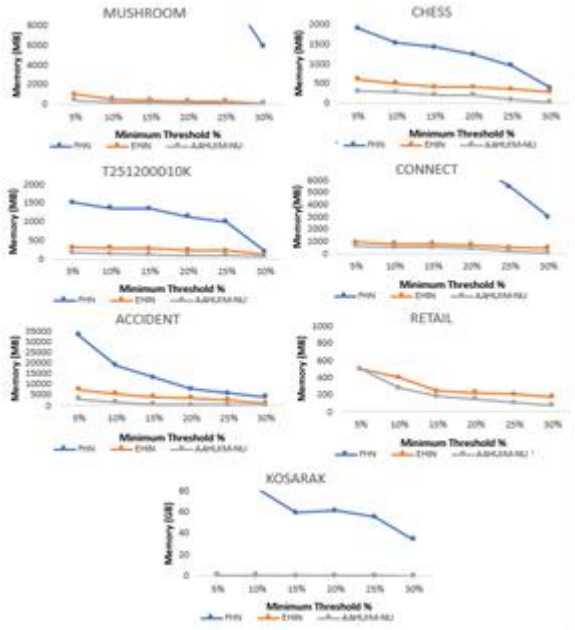

Fig 2. Memory usage of different datasets for thresholds different minimum utility thresholds

4.5. Effect of the number of negative items

The impact of the number of negative items in records on algorithm performance is investigated in this experiment. To verify our algorithm performance, the running times and candidate sets generated for every dataset are compared by increasing the set of non-profitable things from 25% to 50% of all unique items. The minimum utility criterion for each trial is set to 10%. Studies displayed in Figure 4. showed that when the exposure to negative items in a dataset grows, the mining algorithms take shorter time to complete, generate fewer candidate sets, and visit fewer nodes.

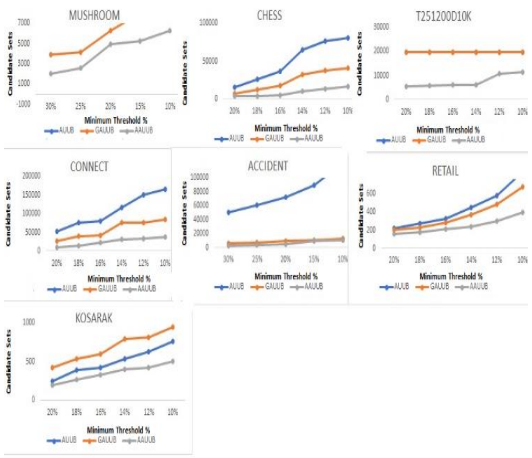

Fig 3. The number of candidate sets under changing minimum utility thresholds

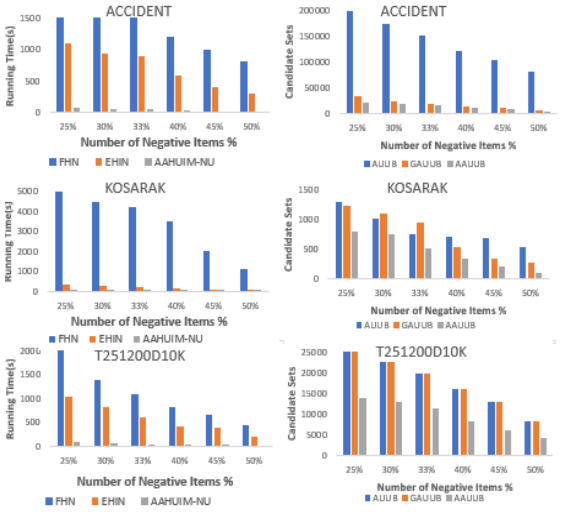

Fig 4. The impact of the quantity of negative elements on algorithm performance

Furthermore, when the number of negative items in the given records rises, the performance of the proposed AAHUIM-NU algorithm improves, including both terms of throughput and total variety of elements. The auub bound performs better and the search space size is significantly reduced. When compared to the other two constraints, the auub bound produces a significant number of candidate-sets in dense datasets with a large number of transactions. The pruning algorithms described in this work are quite effective at reducing running time and lowering memory use.

5. Conclusion

This paper proposes a novel methodology for handling both positively and negatively valued items when mining the productive high average utility itemsets. The AAHUIM-NU algorithm is a depth-first HAUIM approach with a single phase. The technique employs several lists to maintain only the data needed for future use thus reducing the number of scans. All of the proposed five early pruning approaches have an impact on running time and candidate generation either directly or indirectly. The dataset size will be lowered when the unpromising elements are trimmed and processing time and memory usage are reduced. The performance of the AAHUIM-NU algorithm is much superior than that of the comparative algorithms, and the algorithm performs particularly well in dense datasets, according to experimental results. Due to the algorithm's deprived performance on sparse datasets, future research could focus on ways to improve the algorithm's runtime on sparse datasets. In incremental datasets and huge data, we can also use the mining of AAHUIs with negative elements.

Acknowledgements

Authors are declaring thanks to School of Computer Science and Engineering (SCOPE), VIT Chennai Campus, Tamil Nadu, India for their support in completion of this research work.

Conflicts of interest

The authors declare no conflicts of interest

References

[1] R. Agrawal and R. S&ant, "Fast Algorithms for Mining Association Rules .," In Proc. Int'l Conf.Very Large Data Bases, pp. 487–499, 1994.

[2] P. Naresh and R. Suguna, "IPOC: An efficient approach for dynamic association rule generation using incremental data with updating supports," Indonesian Journal of Electrical Engineering and Computer Science, vol. 24, no. 2, pp. 1084–1090, Nov. 2021, doi: 10.11591/ijeecs.v24.i2.pp1084-1090.

[3] A. G. Shaaban, M. H. Khafagy, M. A. Elmasry, H. El-Beih, and M. H. Ibrahim, "Knowledge discovery in manufacturing datasets using data mining techniques to improve business performance," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 3, pp. 1736–1746, Jun. 2022, doi: 10.11591/ijeecs.v26.i3.pp1736-1746.

[4] T. B. Ho, D. Cheung, H. Liu, Y. Liu, W.-K. Liao, and A. Choudhary, "A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets," 2005.

[5] C. F. Ahmed, S. K. Tanbeer, B. S. Jeong, and Y. K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," IEEE Trans Knowl Data Eng, vol. 21, no. 12, pp. 1708–1721, Dec. 2009, doi: 10.1109/TKDE.2009.46.

[6] B. Vo, H. Nguyen, T. B. Ho, and B. Le, "Parallel method for mining high utility itemsets from vertically partitioned distributed databases," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2009, pp. 251–260. doi: 10.1007/978-3-642-04595-0_31.

[7] Y. C. Li, J. S. Yeh, and C. C. Chang, "Isolated items discarding strategy for discovering high utility itemsets," Data Knowl Eng, vol. 64, no. 1, pp. 198–217, Jan. 2008, doi: 10.1016/j.datak.2007.06.009.

[8] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-Growth: An efficient algorithm for high utility itemset mining," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 253–262. doi: 10.1145/1835804.1835839.

[9] V. S. Tseng, B. E. Shie, C. W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," IEEE Trans Knowl Data Eng, vol. 25, no. 8, pp. 1772–1786, Aug. 2013, doi: 10.1109/TKDE.2012.59.

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

[11] T. P. Hong, C. H. Lee, and S. L. Wang, "Effective utility mining with the measure of average utility," Expert Syst Appl, vol. 38, no. 7, pp. 8259–8265, Jul. 2011, doi: 10.1016/j.eswa.2011.01.006.

[12] G. C. Lan, T. P. Hong, and V. S. Tseng, "An efficient projection-based indexing approach for mining high utility itemsets," Knowl Inf Syst, vol. 38, no. 1, pp. 85–107, 2014, doi: 10.1007/s10115-012-0492-y.

[13] Rajendra, K. ., Subramanian, S. ., Karthik, N. ., Naveenkumar, K. ., & Ganesan, S. . (2023). Grey Wolf Optimizer and Cuckoo Search Algorithm for Electric Power System State Estimation with Load Uncertainty and False Data. International Journal on Recent and Innovation Trends in Computing and Communication, 11(2s), 59–67. https://doi.org/10.17762/ijritcc.v11i2s.6029

[14] G. C. Lan, T. P. Hong, and V. S. Tseng, "Efficiently mining high average-utility itemsets with an improved upper-bound strategy," Int J Inf Technol Decis Mak, vol. 11, no. 5, pp. 1009–1030, Sep. 2012, doi: 10.1142/S0219622012500307.

[15] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "Efficiently Mining High Average Utility Itemsets with a Tree Structure," Proc. IIDS, Springer, pp. 131–139,

2010.

[16] T. Lu, B. Vo, H. T. Nguyen, and T.-P. Hong, "A New Method for Mining High Average Utility Itemsets," in LNCS, 2014, pp. 33–42.

[17] J. C. W. Lin, T. Li, P. Fournier-Viger, T. P. Hong, J. Zhan, and M. Voznak, "An efficient algorithm to mine high average-utility itemsets," Advanced Engineering Informatics, vol. 30, no. 2, pp. 233–243, Apr. 2016, doi: 10.1016/j.aei.2016.04.002.

[18] J. C. W. Lin, S. Ren, P. Fournier-Viger, and T. P. Hong, "EHAUPM: Efficient High Average-Utility Pattern Mining with Tighter Upper Bounds," IEEE Access, vol. 5, pp. 12927–12940, Jun. 2017, doi: 10.1109/ACCESS.2017.2717438.

[19] U. Yun and D. Kim, "Mining of high average-utility itemsets using novel list structure and pruning strategy," Future Generation Computer Systems, vol. 68, pp. 346–360, Mar. 2017, doi: 10.1016/j.future.2016.10.027.

[20] J. C. W. Lin, S. Ren, P. Fournier-Viger, T. P. Hong, J. H. Su, and B. Vo, "A fast algorithm for mining high average-utility itemsets," Applied Intelligence, vol. 47, no. 2, pp. 331–346, Sep. 2017, doi: 10.1007/s10489-017-0896-1.

[21] Prof. Amruta Bijwar. (2016). Design and Analysis of High Speed Low Power Hybrid Adder Using Transmission Gates. International Journal of New Practices in Management and Engineering, 5(03), 07 - 12. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/46

[22] J. M. T. Wu, J. C. W. Lin, M. Pirouz, and P. Fournier-Viger, "TUB-HAUPM: Tighter Upper Bound for Mining High Average-Utility Patterns," IEEE Access, vol. 6, pp. 18655–18669, 2018, doi: 10.1109/ACCESS.2018.2820740.

[23] T. Truong, H. Duong, B. Le, and P. Fournier-Viger, "Efficient Vertical Mining of High Average-Utility Itemsets Based on Novel Upper-Bounds," IEEE Trans Knowl Data Eng, vol. 31, no. 2, pp. 301–314, Feb. 2019, doi: 10.1109/TKDE.2018.2833478.

[24] C. J. Chu, V. S. Tseng, and T. Liang, "An efficient algorithm for mining high utility itemsets with negative item values in large databases," Appl Math Comput, vol. 215, no. 2, pp. 767–778, Sep. 2009, doi: 10.1016/j.amc.2009.05.066.

[25] S. Krishnamoorthy, "Efficiently mining high utility itemsets with negative unit profits," Knowl Based Syst, vol. 145, pp. 1–14, Apr. 2018, doi: 10.1016/j.knosys.2017.12.035.

[26] J. C. W. Lin, P. Fournier-Viger, and W. Gan, "FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits," Knowl Based Syst, vol. 111, pp. 283–298, Nov. 2016, doi: 10.1016/j.knosys.2016.08.022.

[27] K. Singh, H. K. Shakya, A. Singh, and B. Biswas, "Mining of high-utility itemsets with negative utility," Expert Syst, vol. 35, no. 6, Dec. 2018, doi: 10.1111/exsy.12296.

[28] I. Yildirim and M. Celik, "Mining High-Average Utility Itemsets with Positive and Negative External Utilities," New Gener Comput, vol. 38, no. 1, pp. 153–186, Mar. 2020, doi: 10.1007/s00354-019-00078-8.

[29] G. C. Lan, T. P. Hong, J. P. Huang, and V. S. Tseng, "On-shelf utility mining with negative item values," Expert Syst Appl, vol. 41, no. 7, pp. 3450–3459, Jun. 2014, doi: 10.1016/j.eswa.2013.10.049.

[30] K. Singh, A. Kumar, S. S. Singh, H. K. Shakya, and B. Biswas, "EHNL: An efficient algorithm for mining high utility itemsets with negative utility value and length constraints," Inf Sci (N Y), vol. 484, pp. 44–70, May 2019, doi: 10.1016/j.ins.2019.01.056.

[31] Prof. Madhuri Zambre. (2016). Automatic Vehicle Over speed Controlling System using Microcontroller Unit and ARCAD. International Journal of New Practices in Management and Engineering, 5(04), 01 - 05. Retrieved from http://ijnpme.org/index.php/IJNPME/article/view/47

[32] Souleymane Zida, Philippe Fournier-Viger, Jerry Chun-Wei Lin, Cheng-Wei Wu, and Vincent S. Tseng, "EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining," LNAI, vol. 9413, pp. 530–546, 2015, doi: 10.1007/978-3-319-27060-9.

[33] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and V. S. Tseng, "Mining high-utility itemsets with both positive and negative unit profits from uncertain databases," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer Verlag, 2017, pp. 434–446. doi: 10.1007/978-3-319-57454-7_34.

[34] K. Singh, S. S. Singh, A. Kumar, H. K. Shakya, and B. Biswas, "CHN: an efficient algorithm for mining closed high utility itemsets with negative utility," IEEE Trans Knowl Data Eng, 2018, doi: 10.1109/TKDE.2018.2882421.

[35] P. Fournier-Viger, A. Gomariz, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: A Java Open-Source Pattern Mining Library," 2014.

.