A Review of High Utility Itemset Mining for Transactional Database

Eduardus Hardika Sandy Atmaja[✉1,2][0000-0003-1739-0116] and Kavita Sonawane[1][0000-0003-0865-6760]

[1] St. Francis Institute of Technology, Mumbai, India
eduardus@student.sfit.ac.in, kavitasonawane@sfit.ac.in
[2] Sanata Dharma University, Yogyakarta, Indonesia

Abstract. High utility itemset mining (HUIM) is an expansion of frequent itemset mining (FIM). Both of them are techniques to find interesting patterns from the database. The interesting patterns found by FIM is based on frequently appeared items. This approach is not that efficient to identify the desired patterns, as it considers only existence or non existence of items in database and ignores utility. Whereas the patterns are more meaningful for the user if the utility is considered. The utility can be quantity, profit, cost, risk or other factors based on user interest. HUIM is another approach to find interesting patterns by considering utility of items along with the frequency. It uses minimum utility threshold to determine if an itemset is high utility itemset (HUI) or not. There are several challenges to implement utility from traditional pattern mining to HUIM. Lately, there are many research contributions that proposed different algorithms to solve these issues. This review work explores various HUIM techniques with detailed analysis of different strategies like Apriori, Tree based, Utility Lists based, and Hybrid. These strategies are used to implement various HUIM techniques in order to achieve the effectiveness in pattern mining. The observations and analytical findings based on this detailed review done with respect to various parameters can be recommended and used for further research in the pattern mining.

Keywords: Frequent itemset · High utility itemset · Apriori · Tree based · Utility List based · Hybrid

1 Introduction

FIM is a technique to find interesting patterns by finding items which frequently appear together in the transaction [1]. It has been experimented by many researchers [2-6]. These research are the basic techniques to solve FIM problem. The general problem is solved by finding interesting patterns using frequency of items in transaction. Minimum threshold of support is applied to decide whether or not an itemset is frequent. For every frequent itemsets that has confidence higher than minimum threshold of confidence becomes association rule. For example, a rule: Pencil, Eraser → Pen [Support = 10%, confidence = 90%]. It means that 10% customers bought pencil,

eraser and pen together in one transaction. The value 90% for confidence interprets that customers who bought pencil and eraser together also bought pen. This information can be used by the manager to set up marketing strategies by putting the items together in the display to increase sales. FIM only considers frequency to discover interesting patterns, but in reality there are many factors that affect meaningfulness of the patterns. Utility is an important factor that can make the generated patterns more meaningful. It can be quantity, profit, cost, risk or other factors based on user interest. FIM does not consider these factors, as a result there are many items with high utility cannot be detected. HUIM is then introduced to solve the problem. It finds interesting patterns not only by considering frequency but also the utility. Minimum threshold of utility is applied to determine whether an itemset is HUI or not. With this approach, the undiscovered items which has high utility can be found.

The main challenge being faced by researchers in HUIM is that how to discover interesting patterns effectively and efficiently. Effective means that the interesting patterns truly represent the real life conditions, for example, patterns should have high profit and correlation when they are appearing together in the transaction. Efficient means that the algorithms should produce interesting patterns by consuming less time and memory. The main efficiency problem is that the number of item combinations may be huge. To solve that problem, many research have been proposed such as Apriori based [7, 8], tree based [9-17], utility list based [18-23, 26-31, 33, 34, 36-38] and hybrid based algorithm [24, 25]. Moreover, there are several HUIM variations such as high average utility itemset mining (HAUIM) [16, 35], HUIM in incremental databases [17, 32], HUIM in sequential database (HUSPM) [18-20, 37, 38], HUIM in regular occurrence [34], Close HUIM [29, 31, 33], and correlated HUIM [28, 30]. These kinds of HUIM are introduced to make the results more meaningful.

In the last phase of pattern mining, once the HUIs are generated, post processing steps can be applied to generate association rules leading to final validation of identified patterns. To evaluate performances of various strategies and techniques researcher have used different interestingness measures. These measures have a significant role to decide whether a rule is interesting or not. It also can be used to rank the rules [1, 28, 30, 39]. With this interestingness measures we can get patterns based on the user's potential interest. Along with this, performance of HUIM systems can be evaluated in terms of memory consumption and execution time. However, with the existing challenges HUIM is found to be very interesting area to explore. With this objective in mind this review work tried to explore major HUIM algorithms along with their algorithmic details and performance comparisons to mine HUI.

2 Literature Review

The objective of this review is to explore, analyze and compare various research contributions in HUIM. This paper presents the detailed study and analysis of different strategies with respect to working principle, its experimentation and the performance observations for each algorithm. All the major findings are summarized along with advantages and disadvantages into six different sections (Point 2.1-2.6) as follows.

2.1 Apriori Based Algorithm

Apriori [2] is an algorithm that introduces downward closure property on FIM. It means that if an itemset is frequent then every subset must be frequent and if an itemset is infrequent then every superset is infrequent. The pseudocode of Apriori is given in Algorithm 1. The function in line 3 has an important role to generate candidate k-itemsets by pairing between itemsets from (k-1)-itemsets then the unnecessary candidates are removed by line 9. The main advantage of Apriori is that it can generate all significant patterns but it needs a lot of time to generate candidate itemset. Apriori also needs database scan every iteration, it makes the algorithm inefficient. The candidate generation also consumes much memory to save the candidates.

Algorithm 1: Apriori

Input : database D, minimum support threshold $minSup$
Output : frequent itemsets F_k

```
1  F1={i|i ∈ I ∧ i.count ≥ minSup}//frequent 1-itemsets
2  for(k=2; Fk-1≠∅; k++){
3    Ck=apriori_gen(Fk-1);//generate candidate k-itemsets
4    for each transaction t ∈ D {
5       Ct=subset(Ck, t);//identify all candidates
6       for each candidate c ∈Ct
7           c.count++;//increment support count
8    }
9    Fk={c ∈ Ck|c.count ≥ minSup}//frequent k-itemsets
10 }
11 return ∪Fk;//return all frequent itemsets
```

UMining and **UMining_H** [7] work similar to the origin Apriori to mine HUI but there are three steps modified viz. first in the pruning step, utility upper bound in Eq. 1 is used instead of calculating actual utility which is similar process to line 4-7 in Algorithm 1. Second, it uses utility values instead of frequent values. Third, in the generating step, since it does not assure downward closure property, the line 3 in Algorithm 1 cannot be adapted. The k-itemsets are created by combining (k-1)-itemsets with list items I scanned from database. UMining_H uses same framework as UMining but it uses Eq. 2 instead of Eq. 1 to calculate the upper bound. It also may prune some HUIs, although it produces correct HUIs. Both algorithms do not apply downward closure property so it produces a large number of candidate itemset.

$$b(S^k) = \frac{\sum S^{k-1} \in C^{k-1} u(S^{k-1})}{|C^{k-1}|-1} \quad (1)$$

$$b'(S^k) = \frac{s_{min}}{|C^{k-1}|-1} \times \sum_{S^{k-1} \in C^{k-1}} \frac{u(S^{k-1})}{s(S^{k-1})} \quad (2)$$

where $S^k$is an itemset of k, $C^k$is candidate itemset of k, s is support, and $|C^{k-1}|$ is cardinality of C^{k-1}.

Two Phase [8] has similar concept to downward closure property in Apriori namely transaction weighted downward closure (TWDC) property. It means that whenever an itemset is not HUI, every its superset is not HUI. This concept reduces the search space because all supersets from low transaction weighted utilization (TWU) itemsets can be cut down. Although in the first phase there is a TWU pruning to save execution time but in the second phase it needs more time to extra database scan.

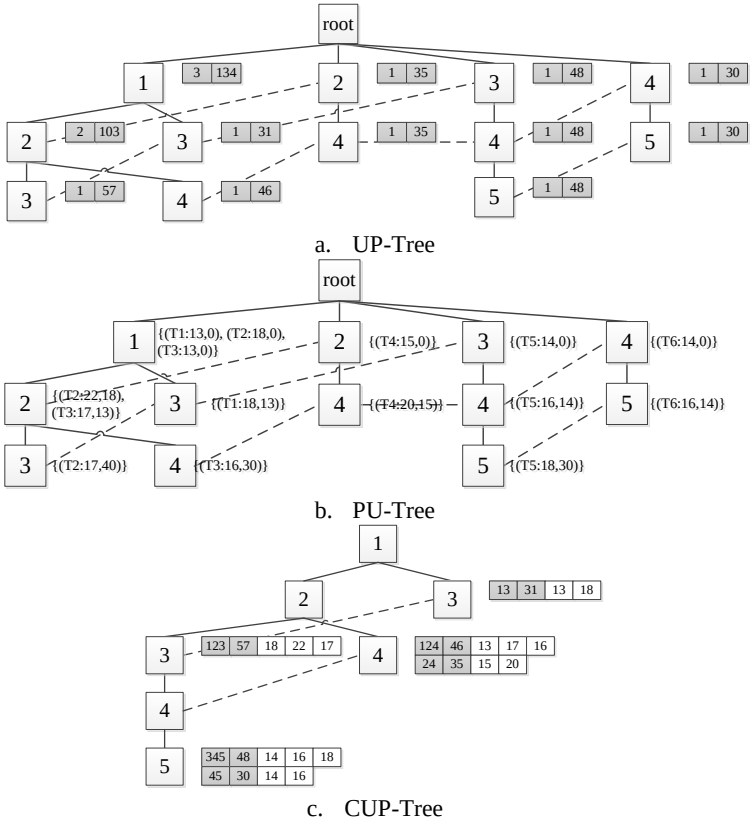

Fig. 1. Tree Representations

2.2 Tree Based Algorithm

To reduce search space in Apriori based algorithm, tree structure algorithm based on pattern growth approach is proposed [12, 13]. **UP-Growth** [12] and **UP-Growth+** [13] are extension of FP-Growth [4] in FIM. It uses a compressed tree namely UP-Tree to store crucial information about the utility. Fig. 1a shows UP-Tree representation. There are four proposed strategies called discarding global unpromising items (DGU), discarding global node utilities (DGN), discarding local unpromising items (DLU) and decreasing local node utilities (DLN). These strategies are used to reduce candidate itemset. UP-Growth+ added two new strategies called discarding

local unpromising items and their estimated node utilities (DNU) and decreasing local node utilities for the nodes of local UP-Tree by estimated utilities of descendant nodes (DNN).

Table 1. Tree based algorithms comparison

Algorithm	Key Principle	Parameters: execution time, memory usage	
		Advantages	Disadvantages
UP-Growth	An extension of FP-Growth to mine HUI with compressed tree structure called UP-Tree and four new strategies: DGU, DGN, DLU, and DLN.	Faster algorithm: The four strategies support in reducing the overestimation of utilities and candidates.	- Dataset with distinct items, the tree grows bigger. - Leads to high memory consumption. - High traversal and mining time.
UP-Growth+	An extension of UP-Growth with two new strategies: DNU and DNN.	Two additional strategies are effectively helping to reduce overestimation utilities and candidates and performs better than UP-Growth.	The disadvantage is same as UP-Growth because it has same tree structure.
CHUI-Mine	A pattern growth approach by dynamically pruning and mining the tree during the tree construction.	- It reduces both: the number of candidates and the search space. - The dynamic pruning helps to reduce the memory consumption. - Concurrent mining makes it faster.	If the tree structure grows widely and deeply, in some cases it makes the algorithm slower.
MIP	A pattern growth approach with PUtree and PUN-List data structures.	- PUN-list avoids costly and repeated utility computation. - PUN-list generates candidates efficiently.	In sparse dataset, it consumes more memory because the PUtree is grown bigger.
CTU-Mine	An extension of CT-PRO to mine HUI with compact tree structure namely CTU-Tree.	It can mine complete HUI and performs good in dense datasets.	It overestimates the potential HUI, then it increases the memory consumption and computational process.
CTU-PRO	A variant of CTU-Mine with more compact tree structure namely CUP-Tree.	- It has compact tree structure that can reduce the database size. - It also efficient in sparse and relatively dense datasets.	- It needs more time to do the local mining due to several global tree scanning. - It needs more memory consumption during local mining.
CTU-PROL	A parallel version of CTU-PRO by using paralel projection to the transactions.	- It uses parallel projection effectively handles very large database. - Concurrent mining makes it faster.	- It is slower for high utility threshold. - Parallel Processing of data leads more memory consumption.

CHUI-Mine [14] is an algorithm that can dynamically prune the tree during tree construction by reducing count of items. It uses tree structure similar to UP-Growth. When the count of an item is zero, then nodes having that item and its descendants can be removed. The pruned nodes are moved to the buffer and ready to mine using pattern growth approach in concurrent time without waiting for the tree insertion process finished. **MIP** [15] is also based on pattern growth approach. PUN-list is the new data structure. PUN-list which is node in PUtree contains list of transactions (using vertical data format). MIP works in PUtree with depth first search approach to explore and mine HUI. Fig. 1b shows PUtree representation.

CTU-Mine [9] is an extension of CT-PRO [5] in FIM. Compressed Transaction Utility Tree (CTU-Tree) is the proposed data structure which is very compact and efficient because it maps transactions into an ascending sequence of integers without generating more branch. CTU-Tree is similar to CUP-Tree used by CTU-PRO but it sorts the items into TWU ascending order. It also does not have link node for the prefix item and it can be mined using top-down approach. **CTU-PRO** [10] and **CTU-PROL** [11] are variant of CTU-Mine. CTU-PRO has new structure namely Compressed Utility Pattern Tree (CUP-Tree). Fig. 1c shows CUP-Tree representation. CUP-Tree has link node to the prefix item so it can be mined quickly using bottom-up approach. CTU-PROL is similar to CTU-PRO, but it mines LocalCUP-Tree separately in concurrent time. CTU-PROL divides transactions into several parallel projections and generates its LocalCUP-Tree from GlobalCUP-Tree. Table 1 shows tree based algorithms comparison.

2.3 Utility List Based Algorithm

Utility list based is proposed to improve performance of tree based algorithms. **HUI-Miner** [21] is the former of utility list inspired by ECLAT [3] in FIM. The utility list is built to produce k-itemset similar to Apriori but there is no candidate generation. It uses enumeration tree to extend the search space. HUI-miner prunes the search space by applying minimum utility threshold. **FHM** [22] proposed a novel structure namely EUCS (Estimated Utility Co-Occurrence Structure). It is a matrix in triangular shaped that consists of items with its co-occurrence with other items. With EUCS, candidate itemset can be easily found without joining any item from utility list. EUCP (Estimated Utility Co-occurrence Pruning) is the proposed strategy to prune candidate itemset from EUCS. **IMHUP** [23] is proposed to improve HUI-Miner and FHM. Indexed utility list (IU-List) is proposed to maintain the database efficiently. Two new techniques called reducing upper-bound utilities in IU-Lists (RUI) by decreasing upper-bound utilities in IU-List and combining HUI without creating IU-List (CHI) are applied.

HMiner [26] has two new data structures called compact utility list (CUL) and virtual hyperlink. It is used to store and determine duplicate transactions. The algorithm also uses three strategies to accelerate the mining process, viz. initial TWU computation and 1-itemset CUL generation, search tree exploration, and k-itemset CUL construction. It also uses several pruning properties such as U-Prune and EUCS to decrease the search space and mine HUI efficiently.

ULB-Miner [27] has a new structure called utility list buffer. It is used for storing the potential HUI by temporarily inserting the data in the data segment. StartPos and EndPos are two pointers that express start and end index of data segments. It helps to access the data quickly. Whenever an itemset is not needed anymore, it can be replaced by other candidate to maintain efficient memory management.

SPHUI-Miner [36] has a new data format called high utility-reduced transaction pattern list (HUI-RTPL) which consumes small memory. It has two data structures, selective database projection utility list (SPU-List) to reduce the scanning process and maintain information of the database and Tail-Count list which helps to prune the search space efficiently. It has two new upper bounds (tup and pu) that also help to reduce the search space effectively. Table 2 shows utility list based algorithms comparison.

Table 2. Utility list based algorithm comparison

Algorithm	Key Principle	Parameters: execution time, memory usage	
		Advantages	Disadvantages
HUI-Miner	The first utility list based algorithm.	It reduces execution time by avoiding candidate generation and utility calculation.	It need more time to join k-itemset among utility lists.
FHM	An extension of HUI-Miner with a new pruning mechanism namely EUCP.	EUCP helps to reduce execution time by reducing join operations.	The performance decreases for dense dataset.
IMHUP	Indexed utility list-(IU-List) based algorithm with two strategies namely RUI and CHI.	- It reduces execution time by reducing join operations and search space. - It reduces memory usage by reducing utility list construction.	The upper bound utility should be tighten.
HMiner	A utility list algorithm with two new data structures namely CUL and virtual hyperlink.	The new data structures reduce memory usage.	It consumes more memory for creating CUL for every itemset in sparse dataset.
ULB Miner	Another utility list algorithm that uses utility list buffer to reuse memory whenever possible.	- It reduces execution time by accessing and mining the data structure quickly. - The utility list buffer consumes less memory.	More distinct items leads to high memory usage.
SPHUI-Miner	A projection utility list based with a new data format namely HUI-RTPL and two new data structures namely SPU-List and Tail-Count.	- The data structure reduces memory usage. - It reduces execution time by reducing database scan and search space.	The memory consumption increases when the minimum utility is decreased.

2.4 Hybrid Based Algorithm

It is possible to combine tree and utility list based algorithm [24, 25]. **mHUIMiner** [24] is an algorithm that combines HUI-Miner and IHUP-tree [17]. IHUP-tree is used to avoid expanding items that do not appear in database. It makes the mining faster. The tree does not contain utility information, it is used only to escort the mining and extension process. It needs low memory consumption, because there is no calculation and storing of utility in the tree. The concept of utility list from HUI-Miner is used to maintain information about utility of the items. These two strategies work together in the mining process, the tree helps to traverse and expand the search space and the utility list is used to prune the candidate based on minimum utility. These two strategies make the algorithm more efficient, but it is weak on dense dataset.

The other hybrid based algorithm is **UFH** [25] that combines UP-Growth+ and FHM. UP-Growth+ is used to construct and mine the tree. FHM is called after UP-Growth+ builds the conditional pattern base. The utility list is built based on conditional pattern base. Then FHM is called to mine the utility list. It means that FHM works with this utility list in local tree to mine HUI. The hybrid framework performs better because in UP-Growth+ there is a transaction merging process to reduce the memory consumption, it also provides actual utility calculation to prune the tree efficiently. Then, FHM mines the utility list efficiently because the size of utility list is reduced by UP-Growth+.

2.5 Other Variations

The other HUIM problem is that it may produce long itemsets which gets an inconsistent predicted profit opposing the actual value. This condition is happen if the count of distinct items is huge and the transactions contain many items. To overcome this challenge HAUIM [16, 35] is proposed. It considers the average utility (consider both length and utility) of itemsets to decrease itemsets with unreasonable estimated profit. The earlier concept of HUIM assumes that transactional databases are static especially the utilities. In real-life, the utilities may change over time. For example, mask is cheaper before a pandemic, the price is increased because of high demand. This issue may produce inaccurate results on real datasets. Moreover, transactional databases also can be manipulated such as additions, deletions, and modifications. This kinds of problem can be solved by HUIM in incremental databases [17, 32]. Sequence dataset is the other problem in HUIM. Different from the usual dataset, sequence dataset maintains the order of the item that cannot be reordered. For example, DNA sequence cannot be reordered because it represents the important information about someone DNA. To overcome the problem, HUSPM which maintains the important sequence of items is proposed [18-20, 37, 38]. Regular occurrence of items in HUIM [34] is also interesting to investigate. It can be used to investigate the occurrence behavior of itemsets with their utility values. For example, in the retail dataset, we can explore regular purchases items which have high profit. Close and maximal HUI are compact representation of HUIM [29, 31, 33]. It helps to decrease the number of candidate itemset. Close HUI means that if an itemset is HUI then its supersets

do not have the same frequency. Maximal HUI means that if an itemset is HUI then its supersets are not HUI. It can prune redundant patterns efficiently.

2.6 Interestingness Measures

Interestingness measure is very essential in data mining. It can be used in HUIM either for pruning or ranking the patterns based on user specific preferences [39]. A HUI may has low correlation to each items since there is only utility threshold calculation to prune the candidates. Some of the HUIM algorithms use interestingness measures to prune the candidate during the mining process [28, 30]. In [28], all confidence and bond measures are used and in [30], Kulczynski measure is used to prune weakly correlated candidate itemsets. There are another interestingness measures such as X^2, lift, jaccard, cosine, and max confidence [1]. Interestingness measure can be used to rank the generated patterns by sorting the measure value either in ascending or descending order. The ranking represents patterns from the most interesting to the less interesting. This may provide more meaningful patterns because it has high correlation that represents the real condition.

2.7 Overall Observations and Analytical Key Findings

Based on review of high utility itemset algorithms above, the overall observations and analytical key findings can be described as follows:

i. Utility threshold plays significant role in HUIM algorithms as selection of this threshold directly impacts the search space, memory utilization and the processing time.
ii. Changes in utility threshold values are based on the transactional databases as per the buying selling properties and strategies being applied the real time retail store transactions.
iii. Many researchers have contributed to handle these dynamics mentioned in point ii. This leads to trade off among various parameters while trying address the problems associated with the changing pattern in the utilities.
iv. So it still remains a challenge to handle such dynamics and come to the completed and generalized solution for HUIM.
v. Observations based on contribution from various researchers indicating that the positive finding of various algorithms can be combined. The process of determining the threshold can be automated and can be generalized to gain interesting patterns leading to high profit.
vi. This should be experimented with the new real time application areas where the utilities are changing dynamically and also huge transactions are being generated.
vii. This study leads to the suggestion that, there is need of changing not only the algorithm but also the strategy of applying the various key solutions in distributed manner to handle different problems with dedicated key solutions in parallel fashion in order to improve the overall performance.

3 Conclusion

HUIM is found to be effective over FIM as it considers the utility of every item and not just the frequency. This leads to benefits in terms desired pattern generation along with the effective association rules. This mining approach is investigated by many researchers using several data structures such as array, tree, utility list and hybrid based. Each data structures has its own advantages and disadvantages or constraints with respect to the processing and handling of different data sets. Based on the review of literature in this work, we found that most of the research done had aimed to improve the existing algorithms and also to address the issues so the HUI can be mined efficiently. Various challenges identified in this area are running time, memory consumption, generation of desired patterns and association rules. We can delineate that, there are basically three strategies named apriori, tree based, utility list based and hybrid approaches. Most of the papers have discussed about how to create pruning techniques to reduce the search space. Some algorithms applied more than one pruning mechanisms to eliminate irrelevant candidate. The pruning techniques greatly help the main algorithm to reduce the candidate itemset. We can state that, a qualified pruning technique is also necessary to efficiently produce the HUIs

Overall study of this research survey is opening the door towards new research directions along with the existing techniques 1) using efficient data structure such as tree and/or utility list based, 2) using efficient mining and pruning strategies, 3) working on utility thresholding and its impact in order to improve, 4) application of parallel programming/processing strategies, 5) effective use of interestingness measures either to prune or rank the pattern that may help in final decision making or recommendations.

References

1. Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques 3rd edition. Morgan Kaufmann, Waltham (2012)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: 20th Int. Conf. on Very Large Data Bases, pp. 487-499. Morgan Kaufmann, San Francisco (1994)
3. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. on Knowl. and Data Eng. **12**(3), 372–390 (2000). doi:10.1109/69.846291
4. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: 2000 ACM SIGMOD Int. Conf. on Management of Data, pp 1-12. Association for Computing Machinery, New York (2000). doi:10.1145/335191.335372
5. Sucahyo, Y.G., Gopalan, R.P.: CT-PRO: a bottom-up non recursive frequent itemset mining algorithm using compressed fp-tree data structure. In: IEEE ICDM Workshop on Frequent Itemset Mining Implementations. (2004)
6. Aryabarzana, N., Bidgoli, B.M., Teshnehlab, M.: negFIN: an efficient algorithm for fast mining frequent itemsets. Expert Systems with Applications. **105**, 129-143 (2018). doi:10.1016/j.eswa.2018.03.041
7. Yao, H., Hamilton, H.J.: Mining itemset utilities from transaction databases. Data & Knowl. Eng. **59**(3), 603-626 (2006). doi:10.1016/j.datak.2005.10.004

8. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: 9th Pacific-Asia Conf. on Knowl. Discovery and Data Mining, pp. 689-695. Springer, Berlin (2005). doi:10.1007/11430919_79
9. Erwin, A., Gopalan, R.P., Achuthan, N.R.: CTU-Mine: an efficient high utility itemset mining algorithm using the pattern growth approach. In: 7th IEEE Int. Conf. on Computer and Information Tech., pp. 71-76. IEEE, Fukushima (2007). doi:10.1109/CIT.2007.120
10. Erwin, A., Gopalan, R.P., Achuthan, N.R.: A bottom-up projection based algorithm for mining high utility itemsets. In: 2nd Int. Workshop on Integrating Artificial Intelligence and Data Mining, pp. 3-11. Australian Computer Society, Australia (2007)
11. Erwin, A., Gopalan, R.P., Achuthan, N.R.: Efficient mining of high utility itemsets from large datasets. In: 12th Pacific-Asia Conf. on Knowl. Discovery and Data Mining, pp. 554-561. Springer, Berlin (2008). doi:10.1007/978-3-540-68125-0_50
12. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: UP-Growth: an efficient algorithm for high utility itemset mining. In: 16th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining, pp. 253-262. Association for Computing Machinery, New York (2010). doi:10.1145/1835804.1835839
13. Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. on Knowl. and Data Eng. **25**(8), 1772-1786 (2013). doi:10.1109/TKDE.2012.59
14. Song, W., Liu, Y., Li, J.: Mining high utility itemsets by dynamically pruning the tree structure. Appl. Intell. **40**, 29-43 (2014). doi:10.1007/s10489-013-0443-7
15. Deng, Z.H.: An efficient structure for fast mining high utility itemset. Appl. Intell. **48**, 3161–3177 (2018). doi:10.1007/s10489-017-1130-x
16. Yildirim, I., Celik, M.: An efficient tree-based algorithm for mining high average-utility itemset. IEEE Access. **7**, 144245-144263 (2019). doi:10.1109/ACCESS.2019.2945840
17. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Lee, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. on Knowl. and Data Eng. **21**(12), 1708-1721 (2009). doi:10.1109/TKDE.2009.46
18. Yin, J., Zheng, Z., Cao, L.: USpan: an efficient algorithm for mining high utility sequential patterns. In: 18th ACM SIGKDD Int. Conf. on Knowl. Discovery and Data Mining, pp. 660-668. Association for Computing Machinery, New York (2012). doi:10.1145/2339530.2339636
19. Gan, W., Lin, J.C.W., Zhang, J., Chao, H.C., Fujita, H., Yu, P.S.: ProUM: projection-based utility mining on sequence data. Information Sciences Informatics and Computer Science Intelligent Systems Application Journal. **513**, 222-240 (2020). doi:10.1016/j.ins.2019.10.033
20. Gan, W., Lin, J.C.W., Zhang, J., Viger, P.F., Chao, H.C., Yu, P.S.: Fast utility mining on sequence data. IEEE Trans. on Cybernetics. **51**(2), 487-500 (2020). doi:10.1109/TCYB.2020.2970176
21. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: 21st ACM Int. Conf. on Information and Knowl. Management, pp. 55-64. Association for Computing Machinery, New York (2012). doi:10.1145/2396761.2396773
22. Viger, P.F., Wu, C.W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: 21st Int. Symp. on Methodologies for Intelligent Systems, pp. 83-92. Springer, Cham (2014). doi:10.1007/978-3-319-08326-1_9
23. Ryang, H., Yun, U.: Indexed list-based high utility pattern mining with utility upper-bound reduction and pattern combination techniques. Knowl. and Information Systems an Int. Journal. **51**, 627-659 (2017). doi:10.1007/s10115-016-0989-x

24. Peng, A.Y., Koh, Y.S., Riddle, P.: mHUIMiner: a fast high utility itemset mining algorithm for sparse datasets. In: 21st Pacific-Asia Conf. on Knowl. Discovery and Data Mining, pp. 196-207. Springer, Cham (2017). doi:10.1007/978-3-319-57529-2_16
25. Dawar, S., Goyal, V., Bera, D.: A hybrid framework for mining high-utility itemsets in a sparse transaction database. Appl. Intell. **47**, 809-827 (2017). doi:10.1007/s10489-017-0932-1
26. Krishnamoorthy, S.: HMiner: efficiently mining high utility itemsets. Expert Systems with Applications. **90**, 168-183 (2017). doi:10.1016/j.eswa.2017.08.028
27. Duong, Q.H., Viger, P.F., Ramampiaro, H., Norvag, K., Dam, T.L.: Efficient high utility itemset mining using buffered utility-lists. Appl. Intell. **48**, 1859-1877 (2018). doi:10.1007/s10489-017-1057-2
28. Viger, P.F., Zhang, Y., Lin, J.C.W., Dinh, D.T., Le, H.B.: Mining correlated high-utility itemsets using various measures. Logic Journal of Interest Group in Pure and Appl. Logics (IGPL). **28**(1), 19-32 (2018). doi:10.1093/jigpal/jzz068
29. Wu, C.W., Viger, P.F., Gu, J.Y., Tseng, V.S.: Mining compact high utility itemsets without candidate generation. In: High-Utility Pattern Mining: Theory, Algorithms and Applications, pp. 279-302. Springer: Cham (2019). doi:10.1007/978-3-030-04921-8_11
30. Vo, B., Nguyen, L.V., Vu, V.V., Lam, M.T.H., Duong, T.T.M., Manh, L.T., Nguyen, T.T. T., Nguyen, L.T.T., Hong, T.P.: Mining correlated high utility itemsets in one phase. IEEE Access. **8**, 90465-90477 (2020). doi:10.1109/ACCESS.2020.2994059
31. Wei, T., Wang, B., Zhang, Y., Hu, K., Yao, Y., Liu, H.: FCHUIM: efficient frequent and closed high-utility itemsets mining. IEEE Access. **8**, 109928-109939 (2020). doi:10.1109/ACCESS.2020.3001975
32. Vo, B., Nguyen, L.T.T., Nguyen, T.D.D., Viger, P.F., Yun, U.: A multi-core approach to efficiently mining high-utility itemsets in dynamic profit databases. IEEE Access. **8**, 85890-85899 (2020). doi:10.1109/ACCESS.2020.2992729
33. Vo, B., Nguyen, L.T.T., Bui, N., Nguyen, T.D.D., Huynh, V.N., Hong, T.P.: An efficient method for mining closed potential high-utility itemsets. IEEE Access. **8**, 31813-31822 (2020). doi:10.1109/ACCESS.2020.2974104
34. Amphawan, K., Lenca, P., Jitpattanakul, A., Surarerks, A.: Mining high utility itemsets with regular occurrence. Journal of ICT Research and Applications. **10**(2), 153-176 (2016). doi:10.5614/itbj.ict.res.appl.2016.10.2.5
35. Wu, J.M.T., Lin, J.C.W., Pirouz, M., Viger, P.F.: TUB-HAUPM: tighter upper bound for mining high average-utility patterns. IEEE Access. **6**, 18655-18669 (2018). doi:10.1109/ACCESS.2018.2820740
36. Bai, A., Deshpande, P.S., Dhabu, M.: Selective database projections based approach for mining high-utility itemsets. IEEE Access. **6**, 14389-14409 (2018). doi:10.1109/ACCESS.2017.2788083
37. Lin, J.C.W., Li, Y., Viger, P.F., Djenouri, Y., Zhang, J.: Efficient chain structure for high-utility sequential pattern mining. IEEE Access. **8**, 40714-40722 (2020). doi:10.1109/ACCESS.2020.2976662
38. Viger, P.F., Li, J., Lin, J.C.W., Chi, T.T., Kiran, R.U.: Mining cost-effective patterns in event logs. Knowl. Based Systems. **191**, 1-25 (2020). doi:10.1016/j.knosys.2019.105241
39. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: A Survey. Association for Computing Machinery (ACM) Computing Surveys. **38**(3), 9 (2006). doi:10.1145/1132960.1132963