Various Research Opportunities in High Utility Itemset Mining

Sandeep Dalal, Vandna Dahiya

Abstract: ***Pattern mining is a technique, which discovers interesting, hidden, unpredicted and useful patterns of data from the database. Most of the research work in pattern mining has been focused on the traditional way of Frequent Itemset Mining (FIM) and Association Rule Mining (ARM) for pattern-discovery. Patterns in frequent itemset mining are based on the occurrence frequency of items. Although frequent pattern mining is useful, the assumption that 'frequent patterns are interesting,' doesn't hold for numerous applications. High Utility Itemset Mining (UIM) overcomes this limitation of frequent itemset mining. The aim of HUIM is to find the patterns based on a utility function where the utility can be measured in terms of revenue, profit, weight, frequency, interestingness or time spent on some webpage, etc. Mining patterns with high utility can be seen as a generalization of FIM where the transaction database is the input and every item is having a utility factor representing its importance and might have non-binary quantities in the transactions. This paper surveys various recent advances and research opportunities in the field of high utility itemset mining.***

Keywords: Itemset Mining, High Utility, Frequent Itemset, Data Mining, Candidate Pruning

I. INTRODUCTION

Data mining is the process of discovering hidden and useful information from huge databases. Various data mining algorithms have been proposed to analyze the data depending upon the type of knowledge to be mined (Aggarwal, 2015; Han & Kamber, 2011). Patterns mining algorithms are designed to extract interesting, useful, unexpected and unpredicted patterns from data (Fournier et al, 2017; Viger et al, 2017). The patterns can be of various types such as sequential patterns, itemsets, outliers, graph structures, trends etc., each providing a different knowledge to the user. It is an unsupervised learning where no prior class, category or label type is required.

Frequent itemset mining is the key area in pattern mining, which discovers the itemsets that occur frequently in the database (Agrawal & Srikant, 1994). The information is very useful in various spheres such as market basket analysis, web analysis, click stream analysis, software bug detection etc. It finds the itemsets with the occurrence count more than the minimum threshold specified by the user. For example, if a customer buys a mobile phone, he may also buy a screen cover and phone cover. If another customer buys milk, he may also buy curd and bread. Such kind of patterns can be found on mining large set of transactions.

Based on the buying history of a customer, recommendations could be provided to customers. In past, the researchers mainly focused on the conventional way of frequent itemset mining and Association Rule Mining (ARM) for pattern-discovery where patterns were discovered based only on the occurrence frequency of items. Such patterns are beneficial but are not convenient for every domain. For example, FIM may generate the frequent pattern for itemset {milk, butter}, as they are the most common items of a grocery store but might leave the itemset {champagne, nuts}, which is less common but having more profit. So, there is the need to consider other profit-generating factors also apart from the occurrence frequency of the item. Utility mining addresses this issue with a utility factor, which is associated with every item (Fournier et al, 2014; Lin, J.C.W. et al, 2011). The utility factor composed of quantity of the item and some measure of interestingness such as weight, profit, side effect or other preference of user. The utility mining is therefore a task of discovering the set of items occurring together in transaction database and yields a high profit. There are many application areas where the technique for high utility mining is employed such as online shopping, recommendation systems, cross marketing, biological gene analysis, mobile commerce etc. HUIM can be reflected as generalization of FIM, as if the weights have unit values, it will degenerate to FIM. The generalized model can be used for various tasks such as to discover all itemsets with high yield of profit, to find the set of most visited webpages or to find frequent patterns in the traditional way.

HUIM is complex than FIM because the utility of an itemset does not satisfy monotonic and anti-monotonic properties (Yun, U. et al, 2014; Viger et al, 2015). A subset of high utility itemset may or may not be HUI. This paper presents a review of high utility itemset mining algorithms and various extensions to the problem of HUIM along with the future prospects.

The rest of the paper is organized as follows: Section 2 introduces the problem of high utility itemset mining with the mathematical preliminaries. A survey of popular HUIM algorithms has been described in section 3. The possible extensions of HUIM have been presented in section 4. Section 5 presents various research opportunities in HUIM. Lastly, conclusion is drawn in section 6.

Revised Manuscript Received on November 30, 2019.

Sandeep Dalal, Assistant Professor, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India.

Vandna Dahiya, Research Scholar, Department of Computer Science and Applications, Maharshi Dayanand University, Rohtak, India.

II. PROBLEM DEFINITION

A. Motivation

The goal of high utility itemset mining is to discover all the itemsets with a utility, which is no less than a user-specified value called as minimum utility threshold and may be denoted as *minutil*. Transactional database can serve as a basis for decision-making strategies as important market and user trends can be found out in these transactions. Lets take the example of online advertising. By checking the browsing history/records of a client, the Ad agency could display the ads the user is interested in. This will be more relevant to the user and profitable for the Ad agency. But there will be too many candidate ads with different prices. The problem is now to display the set of Ads, which are more profitable for the user. Humbly displaying the top-k expensive Ads will not be much useful, as only few users will be interested in them. Whereas, if there are Ads with less profit but a huge number of interested users, the summed revenue will be more than the top-k money-making Ads. FIM cannot solve this problem as it only considers the unit profit and frequency of the item. HUIM algorithms can be embraced to find the set of most optimal Ads to be displayed.

B. Notations

The standard key terms used in HUIM are defined in this section. Consider a transactional database D, composed of set of transactions, T = {T_1, T_2, T_3…T_n}. Each transaction $T_r \in D$ has a unique identifier T_{id}. Let I ={I_1, I_2, I_3…I_m} be the set of distinct items. An itemset X is a set of items {I_{i1}, I_{i2}…I_{iz}} where Z denotes the length of X. The common definitions used in HUIM are presented here. Table 1 and table 2 are used for the references.

Definition 1 (Internal Utility and External Utility): Each item I is associated with a positive number called as external utility $P(I_j)$, corresponds to the unit profit and internal utility $Q(I_j, T_r)$ based on number of occurrence of the item. For example, External Utility of item A is 3 and its Internal Utility in T_1 is 1.

Definition 2 Utility of an item I_i in a transaction T_j is defined as: $u(I_i, T_j)$, product of profit of item with its quantity in a transaction. For example, utility of A in T_3, $u(A, T_5)$ is 6.

Definition 3 Utility of an itemset X in a transaction T_j is defined as: $u(X, T_j) = \sum I_i \in X (I_i, T_j)$. For example, utility of itemset $u(\{BE\}, T_4) = 2*6 +1*4 = 16$.

Definition 4 Utility of a transaction T_j is defined as: $TU(T_j)$ and computed as $u(T_j, T_j)$. For example, transactional utility $TU(T_4) = u(T_4, T_4)$ i.e. 12+10+4+10 = 32.

Definition 5 Utility of an itemset in database D is defined as: $U(X) = \sum_{Tj \in D \wedge X \subseteq Tj}(X, T_j)$. For example, utility for itemset {C, D} is U({C, D}, T_1) + U({C, D}, T_3) is 29 + 39 = 68

Definition 6 High Utility Itemset HUI, an itemset X is called as high utility itemset (HUI) iff the utility of an itemset X is no less than a minimum utility threshold specified by the user, *minutil*.

So, the problem statement is to find the high utility itemsets from the database. Various algorithms have been proposed so far to mine the HUIs, which are being discussed in the next section.

Table I: Transaction Database

Tid	Transactions
T_1	(A, 1), (C, 1), (D, 3)
T_2	(A, 2), (C, 6), (E, 2), (F, 4)
T_3	(A, 1), (B, 2), (C, 3), (D, 3), (E, 1)
T_4	(B, 2), (C, 2), (E, 1), (F, 2)
T_5	(A, 2), (F, 5)

Table II: External Utilities of Items

Item Name	A	B	C	D	E	F
External Utility	3	6	5	8	4	5

III. OVERVIEW OF HUIM ALGORITHMS

Chan, Yand and Shen first presented the idea of HUIM in 2003 and Yao et. al. gave the model in 2004. After that, many algorithms have been proposed which differ in the type of data structure and strategy they use. They can be grouped into two categories based on the number of phases they have. The algorithms of group A first generate the possible candidates in phase one and then calculate their utilities in second phase by recursive calls to explore the tree structure. The group B algorithms directly calculate the utilities of itemsets without candidate generation by using various pruning strategies, data structures and other techniques.

A. Group A- The Two Phase Algorithms

The first category of algorithms is referred as two-phase algorithms as they usually have two phases. In phase 1, candidate-sets are generated by overestimating the utilities of itemsets using TWU-model. An itemset I is kept in memory if TWU(I) ≥ *minutil* and its supersets can be explored. Else if TWU(I) < *minutil*, the itemset is discarded. In phase 2, database is scanned to count the exact utilities of the candidates of phase 1. The low-utility itemsets are filtered out and high utility itemsets are reverted to the user. The two-phase algorithms are complete algorithms as they generate all high utility itemsets from the search space. But these algorithms generate too many candidate sets and the database is also required to scan multiple times to weed out the low-utility itemsets. This consumes more memory and computation time. So, the methodology of two-phase algorithms is inefficient. Various optimal strategies have been designed to prune more number of candidates in the search space by decreasing the TWU upper bound. But to overcome the generate-and-test approach, one-phase algorithms have been designed where there is no need to generate the potential candidates of HUI.

Some of these algorithms are discussed in this section.

- **Two-Phase Algorithm**

The two-phase algorithm adapts the Apriori algorithm (Agrawal & Srikant, 1994) (Agrawal & Srikant, 1994; Liu Y et al, 2005). It explores the search space in breadth-first manner, where, the single items are considered first. These single-itemsets are then used to generate 2-itemsets, then 3-itemsets and so on until the largest itemset is found. Then, the second phase of the algorithm starts where exact utility of each itemset is calculated and itemsets with utility ≥ *minutil* are returned to the user and rest are discarded. The transaction database is required in standard horizontal format. Minimum utility threshold is also given as input. The two-phase algorithm suffers from the problem of level-wise candidate generation and test approach. It combines the itemsets without looking into the database. So, there may be some patterns generated by it, which do not exist at all in the database and expends the time in their processing. Also, database has to be scanned repeatedly to calculate the utilities. These two factors consume a lot of time. Also, with the breadth-first approach, memory requirements are also high, as at any moment, the algorithm may need to keep all k-itemsets in the memory in the worst case. Various researchers have modified this second phase by storing the itemsets in better data structure like hash-tree to reduce the memory cost. Pattern growth algorithms tackle some of the downsides of two-phase algorithms (Fournier et al, 2013; Ahmed C. F. et al, 2009; Hong T.P. et al, 2014). Database is scanned in the phase 1 to generate the itemsets, which actually exists in the database. Also, the cost of repeated scans of phase 2 is lessened by using compact-representations such as projected database, which is reduced version of original database. Most of the algorithms here use depth-first approach where less number of itemsets are prerequisite to be kept in memory during the search. Some of the pattern-based two-phase algorithms are discussed here.

- **IHUP**

Incremental High Utility Pattern mining (IHUP) algorithm was formerly proposed for incremental and interactive HUIM. On changing the inputs, i.e. when there is any update in a database or the minimum utility value is changed, IHUP can use the former results of mining and avoids unnecessary re-computations. It uses an FP-tree like structure called as IHUP-TWU-tree, where restructuring is not required. It is a very compact structure with all the useful information in it. IHUP takes advantage of the fact that in some cases of updates, there may be the transactions with common items. So, it exploits the path overlapping or prefix-sharing way and a very suitable algorithm for interactive and incremental mining.

- **UP-Growth**

Utility-Pattern Growth (UP-Growth) uses a compact tree-structure named as UP-tree. First, the utility values are counted for single items. Using the order of these utilities, FP-tree is constructed like a prefix tree. For every node, UP-tree maintains the transaction information. Two novel strategies were proposed - Discarding Local Unpromising Items (DLU) to discard the low-utility items from the path of UP-tree and Decreasing Local Node Utilities (DLN), where minimum item utilities of descendant nodes are decreased while creating local-UP-trees. This algorithm works well in case of low minimum utility and when the length of transactions is very long. Further improvement has been done in this approach by using histograms for item-quantities for the nodes, called as UP-Hist algorithm.

The two-phase algorithms are complete algorithms as they generate all high utility itemsets from the search space. But these algorithms generate too many candidate sets and the database is also required to scan multiple times to weed out the low-utility itemsets (Fournier et al, 2011) This consumes more memory and computation time. Thus, the methodology of two-phase algorithms is inefficient. Various optimal strategies have been designed to prune more number of candidates in the search space by decreasing the TWU upper bound. Still, the accumulated values of TWU are required in node utilities, which also result in generating huge number of candidates. To overcome the level-wise generate-and-test approach, one-phase algorithms have been designed where there is no need to generate the potential candidates of HUI.

B. Group B- One Phase Algorithm

The algorithms in this group calculate the utilities of itemsets directly and do not generate the candidate sets. So, there is no need to store the candidate sets in the memory as an itemset is identified as high or low utility itemset immediately. Many novel concepts of upper bounds were introduced in one-phase algorithms such as *remaining utility*, *local-utility*, *sub-tree utility* etc. HUI-Miner was the first design in one-phase algorithms (Liu & Qu, 2012; Fournier et al, 2018). Many optimal versions have also been designed for this algorithm such as HUP-Miner, HUI-Miner*, mHUIMiner etc. Other one-phase algorithms are D^2HUP, FHM, EFIM etc. The brief overview of some of the imperative algorithms is presented here.

The algorithms of one-phase can be divided further into two categories-utility-list based and pattern-growth based.

- **Utility-List Based Algorithms**

The algorithms here use the vertical representation of database where, a list of items is maintained indicating the transactions having them. This is unlike from the conventional-horizontal representation where the entries are composed of transactions and their items. Utility-information of the itemsets is stored in utility-list data structure, which is a vertical data structure and inspired from tid-list data structure of frequent itemset mining. This data structure is very significant as the utility of an itemset can be obtained directly from it. It is also used to prune the search-space.

HUI-Miner

HUI-Miner was the first algorithm where there was no need to generate the candidate sets (Liu & Qu, 2012; Fournier et al, 2018). It can discover HUIs directly. So, it outperforms the two-phase algorithms. Utility information is stored in utility-list, which is a vertical data structure and is very advantageous as the utility of an itemset can be obtained directly from it without scanning the database. A single scan of database is requisite to create the utility-lists of all itemsets taking single items. Then join operation on utility-ist is implemented to get the utility-lists of larger itemsets. The join operation is costly and thus the algorithm is not efficient for larger datasets.

HUP-Miner

HUP-Miner is an extension to HUI-Miner (Krishnamoorthy S, 2015). Two pruning strategies were added to it- the first is based on partitioning of database and the second is look-ahead pruning. HUP-Miner routines the number of partitions internally based on the input value k supplied by the user. The value of k concludes the running time and memory usage for the algorithm. The optimal value of k should be found experientially for a particular dataset. Apart from the overhead of external calculation of k, HUP-Miner is faster than HUI-Miner.

FHM

Fast High Utility Itemset Mining (FHM) algorithm is an improvement over HUI-Miner (Fournier 2014). A novel approach of Estimated Utility Co-occurrence Pruning (EUCP) was proposed, with less memory overhead and was used with utility-list data structure during mining of HUIs. The algorithm uses depth-first approach and creates a utility-list for every itemset visited in the search space. There are less join operations (up to 95% less than HUI-Miner) in this approach. First the EUC structure is created using a single scan of database. The longer itemsets are gained by executing join-operations on the utility-lists of smaller itemsets. Due to less number of costly join operations, this algorithm is nearly six times speedier than HUI-Miner.

The algorithms with utility-lists data structure are easy to implement and more efficient than two-phase algorithms. There are some drawbacks of these algorithms. First, they may explore some itemsets, which never appear in the database as they get them from join operations and not by database scanning. So, time is wasted in their computation. Second, the memory consumption may be high as utility-list is maintained for each visited-itemset and in the worst case it may have the tuple for all the transactions. The join operation is also costly. To overcome some of these limitations, the optimal versions of the algorithm HUI-Miner and FHM have been proposed such as ULB-Miner (Duong et al, 2017), HUI-Miner* (Fournier et al, 2018), mHUI-Miner (Peng AX. et al, 2017) etc. A buffer called as utility-list buffer (ULB) is used in ULB-Miner where the memory can be reused to store the utility-lists, which then improved the runtime and memory usage. In HUI-Miner*, an improved version of utility-list called as utility-list* is used to speed up the algorithm. Another HUI-Miner based algorithm is mHUI-Miner where the itemset development process is guided using a tree structure and needless creation of utility list is avoided. It thus avoids the consideration of itemsets that do not exist in the database.

- **Pattern-Growth One-phase**

The limitations of utility-list based algorithms have been addressed in pattern-growth one-phase algorithms. The search space is explored by database scanning and only existing patterns or itemsets are explored further.

D^2HUP

D^2HUP was the first algorithm of this category (Liu J et al, 2012). It also generates HUIs without candidate generation. The algorithm uses depth-first search. A novel data structure called as Chain of Accurate Utility-lists (CAUL) was proposed. It counts the itemsets as prefix addition of another itemsets. It filters out the irrelevant items during the budding HUIs from sparse data. This approach takes less memory than tree structures. This algorithm is more efficient than Up-Growth and Two-phase.

EFIM

Efficient High Utility Itemset Mining (EFIM) performs depth-first search and uses horizontal representation of database, which reduces the memory usage (Fournier et al, 2015). Efficient techniques for database projection and merging of similar transactions were proposed in this algorithm that condenses the size of database and decreases the cost of database scans. Each itemset is processed in linear time and space. Further, a reusable array-based utility counting technique was proposed called as Fast Utility Counting (FUC) to compute new upper bounds - local utility and sub-tree utility. These novel upper bounds further reduce the search space. In experiments, EFIM is found to be approximately two to three times faster and consumes upto 8 times less memory than all the above algorithms such as HUI-Miner, UP-Growth, FHM.

C. Comparison of HUIM Algorithms

A brief overview of some standard HUIM algorithms has been presented in the above section. The table below provides an assessment of these algorithms in terms of their characteristics.

Table III: Comparisons of Various HUIM Algorithms

Sr. No	Algorithm	Year	Phase	Search-Type	Data Base	Approach	Performance
1	Two-Phase	2005	Two	Breadth-first	Horizontal	Apriori based, overestimate the utilities in phase 1, then filters in phase 2.	Inefficient, generates many candidates in phase 1.
2	IHUP	2009	Two	Depth-first	Horizontal (prefix-tree)	'Build once, Mine many' property for incremental HUIM.	Efficient in memory usage and running time, but huge number of recursive calls due to tree structure.
3	UP-Growth	2010	Two	Depth-first	Horizontal (prefix-tree)	Based on FP-Growth, it constructs UP-tree to store the itemsets.	Uses DLN, DGN, DLU, DGU (local and global) decreasing and discarding strategies for pruning, Fewer candidates. Efficient for dense databases but huge number of recursive calls due to tree structure.
4	HUI-Miner	2012	One	Depth-first	Vertical (utility-list)	Avoids generation and test approach of candidates.	Inefficient join operation and not scalable. Worst for sparse datasets as no good pruning strategy.
5	FHM	2014	One	Depth-first	Vertical (utility-list)	Extends HUI-Miner, precompute the TWUs and less join operations	95% less join-operations and 6 times faster than HUI-Miner
6	HUP-Miner	2015	One	Depth-first	Vertical (partitioned utility-list)	Extends HUI-Miner, limits the search space using various pruning and partition strategies	2-3 times faster than HUI-Miner
7	EFIM	2015	One	Depth-first	Horizontal (merging)	Linear time search is possible with array based utility counting technique	Low memory consumption, because of projected database and merging approach
8	D^2HUP	2016	One	Depth-first	Vertical (hyper linked list-chain of accurate utility list)	Maintains CAUL- chain of accurate utility list	40 times faster than HUI-Miner
9	mHUI-Miner	2017	One	Depth-first	Vertical (utility-list)	Global tree for transaction information and utility list for different items.	Outperforms others for sparse datasets and comparable for dense datasets.
10	ULB-Miner	2017	One	Depth-first	Vertical (buffered)	Memory-re-utilization approach and estimated utility co-occurrence structure	Using buffer, 10 times faster than other utility-list based algorithms and consumes less space.
11	UP-Hist	2018	Two	Depth-first	Horizontal (histogram)	Uses histogram to store the utility information of nodes	Efficient than UP-Growth

IV. VARIOUS EXTENSIONS TO HUIM

In this section, various extensions possible to high utility itemset mining has been discussed for further possible research opportunities.

- **Top-K High Utility Itemsets**

In HUIM, there is a prerequisite to externally specify the value of minimum utility threshold. It directly influences the performance of algorithm along with the number of patterns mined. If the utility threshold were set too low, many irrelevant itemsets would be there with increased memory consumption and running time. If the value is set too high, there would be very less itemsets and the important information would get lost. To tackle this problem, the parameter minimum-utility is superseded by the parameter k and the algorithm is operated as to find the top-k high utility itemsets from the database (Fournier et al, 2016).

- **HUIM with Negative Utilities**

Occasionally, items are sold with low or negative profits to attract customers to the stores. In such cases, utility of an item become negative and the traditional HUIM algorithms cannot mine them because they will not satisfy the TWU property (Chu C et al, 2009). Algorithms can be developed with novel upper bounds that can mine the itemsets with negative utilities.

- **HUIM with Discount Strategies**

Various discount strategies can be combined in marketing (Bansal R et al, 2014). For example, An item may be tagged as a discounted item in various ways such as percentage discount, buy one get one, buy two get 70% on another two etc. In such cases, there is a need of additional information to be stored with the items.

- **HUIM with Length Constraints**

Often, users are interested in smaller set of itemsets, as longer itemsets are often rare. Length constraints on itemsets can be applied as further extension to HUIM (Duong et al, 2016).

- **HUIM with Correlations**

Itemsets mined with the HUIM algorithms are often not correlated (Fourier et al, 2015). For example, any thing buying with an expensive item would be high utility itemset. But such itemset is not much useful to promote the less expensive item. Users are interested in correlated items to make business strategies.

- **Periodic HUIM**

Periodic high utility mining is a concept where, mining is done for group of items that are bought together cyclically. For example, a customer may buy same set of kitchen or household items weekly or monthly. Marketing strategies can be developed for such customers for example, offering discounts, rewards or points to attract more sales (Duong et al, 2016).

- **HUIM for Dynamic Databases**

Most of the algorithms assume the database as static. When, there is any update in the database, results cannot be updated and there is the need to run the algorithm from scratch. There is need to develop the incremental and interactive algorithms for dynamic databases (Fourier et al, 2015; Ryang H & Yum U, 2016).

V. RESEARCH PROSPECTS

Several algorithms have been proposed by various researchers in the field of HUIM for mining various types of itemsets. Still, numerous research opportunities are there in this field. Some of them are discussed here:

- **Novel Applications**

The pattern mining algorithms can be applied in various domains such as social network analysis, graph analysis, community algorithms, Internet of Things, Big Data. Several novel applications can be expected by using HUIM methods in these emerging areas.

- **Complex Data and Patterns**

Most of the HUIM algorithms are for transactional data. The algorithms can be enhanced to work on complex and dynamic data like graphs, spatial data, time-series, complex sequence data etc. The pattern mining algorithms can extend their support in mining more complex and beneficial patterns such as closed patterns, maximal patterns, sub graphs etc.

- **Scalability**

Scalability is one of the core issues to be deal with to meet the new data challenges. Most of the HUIM algorithms have been developed for small databases. Scalability of these algorithms is one of the core aspects for future research.

- **Work Partitioning and Load Balancing**

One of the challenges in parallel processing is how to partition the jobs so that they can be executed concurrently on parallel nodes. The ideal scenario is to distribute the workload equally. Precise methods are needed to guesstimate the resource requirement for each process to uniformly partition the work in parallel processing. Also, dynamic load balancing techniques are desirable to re-distribute the work to further optimize the processing.

Privacy

Various algorithms have been developed for privacy preserving but they are not used in the field of high utility itemset mining. Privacy is the chief concern and of paramount importance in data mining especially when most of the data used for HUIM is personalized to the user.

- **Enhanced Algorithms**

The performance of the algorithms can be enhanced in terms of time and memory usage for larger databases.

Various opportunities are there in the field of parallel and distributed frameworks to raise the speed and scalability of the algorithm. Also, work can be done in GPU and multi-core environments.

VI. CONCLUSION

High utility itemset mining is an emerging area of research. The utility information is of great interest for various decision-making domains like medical, business, security, banks, retail etc. This paper has presented a survey of various popular algorithms in the field of utility mining, which can be very helpful for developing the more efficient and optimize methods. Various extensions to the problem of HUIM have been discussed along with several research opportunities. HUIM can be used in various greener domains to create novel applications. The future work can incorporate soft computing, parallel computing and other frameworks to enhance the performance of the algorithms on larger, distributed, dynamic and complex data sets.

REFERENCES

Aggarwal, (2015). C.C.: Data mining: the textbook. Springer, Heidelberg.

1. Agrawal, R., Srikant, R.(1994): Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, pp. 487–499. Morgan Kaufmann
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.-S., Lee, Y.-K. (2009): Efficient Tree Structures for High-utility Pattern Mining in Incremental Databases. IEEE Trans. Knowl. Data Eng. 21(12), 1708–1721
3. Bansal, R., Dawar, S. and Goyal, V. (2015): An efficient algorithm for mining high-utility itemsets with discount notion. In: Proc. Intern. Conf. on Big Data Analytics, 84–98. Springer
4. Chan, R., Yang, Q., Shen, Y. (2003): Mining High Utility Itemsets. In: Proc. of 3rd IEEE Int'l Conf. on Data Mining, pp. 19–26. IEEE
5. Chu, C., Tseng, V.S. Liang, T. (2009): An Efficient Algorithm for Mining High Utility Itemsets with Negative Item Values in large databases. Applied Mathematics and Computation 215(2), 767–778
6. Dalal Sandeep, Dahiya Vandna, (2018): Review of High Utility Itemset Mining Algorithms for Big Data, In: Journal of Advanced Research in Dynamical and Control Systems- JARDCS, 10(4), pp: 274-283
7. Duong, Q.H., Fournier-Viger, P., Ramampiaro, H., Norvag, K. Dam, T.-L. (2017):Efficient High Utility Itemset Mining using Buffered Utility-Lists. Applied Intelligence 48(7), 1859–1877
8. Fournier-Viger, P., Lin, J.C.-W., Kiran, R. U., Koh, Y. S., Thomas, R. (2017): A Survey of Sequential Pattern Mining. Data Science and Pattern Recognition 1(1), 54–77
9. Fournier-Viger, P., Lin, J.C.-W., Vo, B, Chi, T.T., Zhang, J., Le, H. B. (2017): A Survey of Itemset Mining. WIREs Data Mining and Knowledge Discovery, e1207 doi: 10.1002/widm.1207
10. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S. (2014): FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Proc. 21st Inter. Symp. Methodologies for Intelligent Systems, pp. 83–92. Springer
11. Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S. (2014): FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Proc. 21st Inter. Symp. Methodologies for Intelligent Systems, pp. 83–92. Springer
12. Fournier-Viger, P., Lin, J.C.-W., Duong, Q.-H., Dam, T.-L. (2016): FHM+: Faster High-Utility Itemset Mining using Length Upper-Bound Reduction. In: Proc. 29th Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 115–127. Springer
13. Fournier-Viger, P., Lin, J.C.-W., Duong, Q.H., Dam, T.L. (2016): PHM: Mining Periodic High-Utility Itemsets. In: Proc. 16th Industrial Conf. on Data Mining, pp. 64–79. Springer
14. Fournier-Viger, P., Lin, J.C.-W., Gueniche, T., Barhate, P. (2015): Efficient Incremental High Utility Itemset Mining. In: Proc. 5th ASE International Conf. on Big Data. ASE
15. Goyal, V., & Dawar, S. (2015). Up-hist tree: An efficient data structure for mining high utility patterns from transaction databases. In Proceedings of the 19th inter- national database engineering & applications symposium (pp. 56–61). ACM.
16. Han, J., Pei, J., Kamber, M. (2011): Data mining: concepts and techniques. Elsevier, Amsterdam
17. Krishnamoorthy, S. (2015): Pruning Strategies for Mining High Utility Itemets, In: Expert Systems with Applications, 42950, 2371-2381
18. Lan, G.-C., Hong, T.P., Tseng, V.S. (2014): An efficient projection-based indexing approach for mining high utility itemsets. Knowl. and Inform. Syst. 38(1), 85–107
19. Lin, J.C.-W., Hong, T.P., Lu, W.H. (2011): An effective tree structure for mining high utility itemsets. Expert Systems with Applications 38(6), 7419–24
20. Lin, Y.C., Wu, C.W., Tseng, V.S. (2015): Mining high utility itemsets in big data. In: Proc. Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp. 649–661. Springer
21. Liu, M., Qu,. J. (2012): Mining high utility itemsets without candidate generation. In: Proc. 21st ACM Intern. Conf. Information and knowledge management, pp. 55–64. ACM
22. Liu, Y., Liao, W.K. and Choudhary, A.N. (2005): A two-phase algorithm for fast discovery of high utility itemsets. In: Proc. 9th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp. 689–695. Springer
23. Liu, J., Wang, K., Fung, B. (2012): Direct discovery of high utility itemsets without candidate generation, In: Proc. 12th IEEE Intern. Conf. Data Mining, pp. 984–989. IEEE
24. Peng, A.X., Koh, Y.S., Riddle, P. (2017): mHUIMiner: A Fast High Utility Itemset Mining Algorithm for Sparse Datasets. In: Pacific-Asia Conf. on Knowledge Discovery and Data Mining, pp. 196– 207
25. Philippe Fournier-Viger, Jerry Chun-Wei Lin, Tin Truong Chi, Roger Nkambou. (2019): A Survey of High Utility Itemset Mining, In: High Utility Pattern Mining, pp. 1-46, Springer
26. Qu, J.-F., Liu, M., Fournier-Viger, P. (2018): Efficient algorithms for high utility itemset mining without candidate generation . In: Fournier-Viger et al. (eds). High-Utility Pattern Mining: Theory, Algorithms and Applications, to appear. Springer
27. Ryang, H., Yun, U. (2016): High utility pattern mining over data streams with sliding window technique. Expert Systems with Applications 57, 214–231
28. Tseng, V., Wu, C., Fournier-Viger, P., Yu, P.S. (2016): Efficient Algorithms for Mining Top-K High Utility Itemsets. IEEE Trans. Knowl. Data Eng. 28(1), 54–67
29. Tseng, V.S., Shie, B.-E., Wu, C.-W., Yu., P. S. (2013): Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. 25(8), 1772–1786
30. Wu, C.-W., Fournier-Viger, P., Yu., P.S., Tseng, V.S. (2011): Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets. Proc. 11th IEEE Intern. Conf. on Data Mining, pp. 824– 833. IEEE
31. Yao, H., Hamilton, H.J., Geng, L. (2006): A Unified Framework for Utility-based Measures for Mining Itemsets. In: Proc. of ACM SIGKDD Workshop on Utility-Based Data Mining, pp. 28-37. ACM
32. Yun, U., Ryang, H., Ryu, K.H. (2014): High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates. Expert Syst. Appl. 41(8), 3861–3878
33. Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.W., Tseng, V.S. (2015): EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. In: Proc. 14th Mexican Intern. Conf. Artificial Intelligence, pp. 530–546. Springer
34. Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.W., Tseng, V.S. (2015): EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. In: Proc. 14th Mexican Intern. Conf. Artificial Intelligence, pp. 530–546. Springer
35. Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J.C.-W., Tseng, V.S. (2015): Efficient Mining of High Utility Sequential Rules. In: Proc. 11th Intern. Conf. on Machine Learning and Data Mining, pp. 157–171. Springer