Received February 25, 2020, accepted March 9, 2020, date of publication March 19, 2020, date of current version March 30, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2981962*

# A Survey of Key Technologies for High Utility Patterns Mining

**CHUNYAN ZHANG, MENG HAN, RUI SUN, SHIYU DU, AND MINGYAO SHEN**
School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China

Corresponding author: Meng Han (2003051@nun.edu.cn)

This work was supported in part by first-class discipline construction of Ningxia University (Electronic Science and Technology Discipline) under Grant NXYKXY2017A07, in part by the National Nature Science Foundation of China under Grant 61563001, and in part by the (Computer Application Technology) Key Discipline Project of the Ningxia Hui Autonomous Region under Grant PY1902.

**ABSTRACT** Recently, high utility pattern mining (HUPM) is one of the most important research issues in data mining. Because it can consider the non-binary frequency values of items in a transaction and the different profit values of each item. It has been widely used. First of all, this paper briefly describes the related concepts, formulas and examples of application for HUPM. Secondly, the key technologies for HUMP are introduced in detail, and they are divided into main methods including Apriori-based, tree-based, projection-based, list-based, data format-based, and index-based and so on. The paper further compares data sets, uses, advantages and disadvantages of algorithms, laid the foundation for the next research direction. Then, this article outlines the high utility derivative patterns, including high average utility pattern, high utility sequential pattern, and high utility compact pattern and so on. Because static data is difficult to meet the actual needs, this paper summarizes the efficient use of HUPMs' methods over data streams, mainly based on incremental methods, based on the sliding window model methods, based on the time decay model methods and based on the landmark model methods and so on.

**INDEX TERMS** Survey, pattern mining, high utility pattern, data streams, incremental databases.

## I. INTRODUCTION

Frequent itemsets mining (FIM) is one of the core tasks in data mining. FIM mines the itemsets that often appear together in the transaction database, and assumes that all items have the same importance (unit profit, price, etc.). However, an item may only appear once or zero times in a transaction. For example, in a department store, diamond sales may be much lower than ballpoint pen sales, but the former has a much larger profit margin than the latter, which may therefore make more sense. Traditional FIM will discard this information, potentially mining many low-margin frequent itemsets.

High utility mining (HUIM) is an important area of FIM. HUIM considers the number and profit of items and itemsets to measure the "usefulness" of them. If the total utility of the itemsets in the database is not less than the user-specified minimum utility threshold (*minutil*), it is called high utility itemsets (HUIs). Otherwise, it is called low utility itemsets. For example, in the background of market basket analysis, it involves finding all itemsets that produce a profit. Meanwhile, they are at least equal to a certain minimum utility value. The goal of HUIM is to identify itemsets or items that are meaningful to the users. Therefore, researchers have proposed many methods for mining HUIs in order to quickly take appropriate measures to meet the needs of users.

In the early period of HUIM, the researchers propose Two-Phase [1] based on Apriori, which is one of the classical algorithms. The algorithm proposes the attributes of transaction weighted utility (TWU) and transaction weighted downward closure (TWDC). In phase I, the TWDC is used to find the high transaction weighted utility itemsets (HTWUIs); in phase II, an additional database scan is needed to identify the actual HTWUIs. The algorithm effectively reduces the number of candidates and accurately obtains a complete set of HUIs. However, the algorithm leads to excessive execution time and memory usage due to generation and test methods, and requires a large number of database scans. For this reason, other algorithms are proposed to overcome the shortcomings of the Two-Phase. As one of them, IHUP [2] uses the ihp-tree data structure to generate HTWUIs for mining HUIs in the incremental database [2]. However, it also generates a large

The associate editor coordinating the review of this manuscript and approving it for publication was Hao Ji.

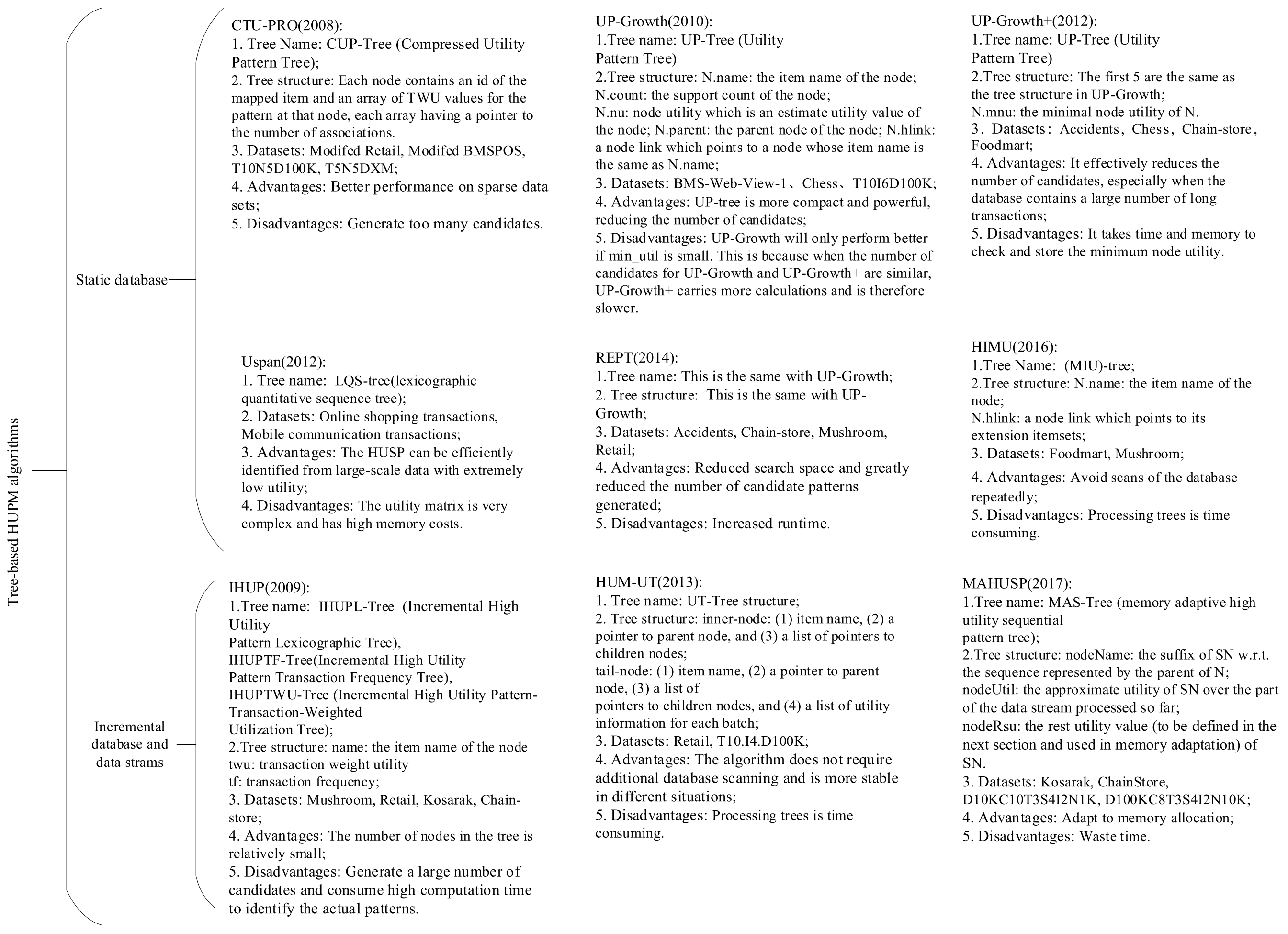
number of candidates. With the continuous advancement of technology, HUP-growth [9] mines HUIs without candidate generation. It uses a two-phase model and the HUP-tree structure to maintain 1-HTWUIs, thus speeding up the mining process. In real-world applications, databases are often large and dynamic because the contents of the database change frequently regardless of the number of transactions inserted or deleted [53]. PRE-HUI-INS [17] is based on the pre-large concept and preserves the HTWUIs to avoid database redoing until the cumulative total utility of the inserted transactions reaches a safe limit. PRE-HUI-DEL [53] is used to pre-delete the pre-large concept to update the HUIs, thereby speeding up the processing time of updating information. PIHUP [18] can handle dynamically added transactions with an additional threshold, requiring only one scan of the database. Therefore, it is more suitable for processing dynamic data. HUPPP [79] extends the HUPM problem to high utility partial-cycle pattern problem, taking into account not only the sequence of events and the length of the period, but also the number of events and personal profits. The algorithm uses a two-phase periodic utility upper limit pattern to avoid the information loss during mining. For example, this algorithm can discover itemsets that customers regularly purchase and generate high profits. It considers the relative order of transactions, so it tends to find patterns that are stable in terms of utility throughout the database.

Because the two-phase algorithms generate a large number of candidates in phase I, and cause scalability problems in phase II, it is less efficient. Although much effort has been made to reduce the number of candidates generated by phase I, the challenge still exists when the original data contains many long transactions or where *minutil* is low. Therefore, the proposal of one-phase algorithms effectively alleviates this problem.

HUI-Miner [12] is the first one-phase algorithm to discover HUIs. It proposes a new structure, utility list [12]. The utility list stores not only utility information about the itemsets, but also heuristic information about whether the itemsets should be pruned. Unlike previous algorithms, HUI-Miner does not generate candidate HUIs. Although HUI-Miner is very effective, mining HUIs is still computationally expensive. Because it has to perform expensive join operations for each pattern generated by its search process. Therefore, FHM [13] proposes a new pruning strategy, EUCP (Estimated Utility Co-occurrence Pruning), which can prune the itemsets without performing the connection, considering the co-occurrence between the 2-itemsets and improving the pruning performance. CHUI-Mine [57] proposes a new structure called EU-List (Extended Utility List) to maintain utility information for itemsets in a transaction, which allows efficient calculation of in-memory itemsets without scanning the original database. It is the first compact algorithm to find a complete set of closed high utility itemsets from the database without generating candidates. EFIM [50] relies on two new upper bounds: revision sub-tree utility and local utility to prune the search space more efficiently. MHUI [65] is used to mine HUIs with multiple *minutils*. This algorithm introduces the concept of minimal suffix utility and proposes a generalized pruning strategy for mining of HUIs efficiently. kHMC [71] finds Top-k HUIs, using real item utilities (RIU), co-occurrence utility descending order (CUD) and coverage of coverage (COV). These three strategies increase their internal *minutil* and reduce the search space effectively. FOSHU [76] considers the spot time period of the item and the item with positive/negative elements. The algorithm uses a utility list and the depth-first search while mining HUIs at all times. HUPNU [52] relies on probabilistic-utility lists with positive and negative margins to mine HUIs directly in uncertain databases without generating and testing candidates.

In recent years, the key technologies [34] for mining HUIM have been extensively studied. UDHUP-Apriori [14] speeds up the mining process by mining the latest high utility patterns in a horizontal way and recursively deriving HUIs using methods similar to Apriori. HUI-list-INS [15] is an incremental algorithm for inserting transactions in a dynamic environment, which reduces the amount of computation without generating a candidate list. It also uses an enumeration tree and 2-itemsets to speed up the calculation. PHUS [60] proposes the maximum utility metric to simplify the utility evaluation of subsequences in a set of quantitative sequences, and also adopts an effective sequence-utility upper-bound (SUUB) model to avoid missing information in mining. The algorithm also designs an efficient projection-based pruning strategy to produce a more accurate sequence-utility upper subsequence. CHUM [32] uses an EU-List structure, which is a vertical representation of the database. The vertical data structure is capable of computing closures of the itemsets and efficiently generating candidates.

From the development of HUPM to the present, there are many reviews about the mining of high utility patterns, such as [34], [86], [87] and so on. Survey [86] focuses on the types of HUIs, specifically the compact representation of itemsets, including closed HUIs, maximum itemsets, generators of HUIs and minimum HUIs; Top-k HUIs; high utility itemset mining with average utility metrics; high utility itemset for negative utility and other novel pattern mining. Survey [34] comprehensively reviews utility-oriented pattern mining, describing various problems related to mining-based utility patterns and methods to solve these problems. This paper proposes the most common and advanced classification methods for mining different types of high utility patterns, and provides a comprehensive review of advanced topics in existing high utility pattern mining techniques, and discusses their advantages and disadvantages. Finally, it introduces some famous UPM open source software packages. Survey [87] studies different algorithms of HUPM, their workflows and their limitations. This article provides an overview of comparative studies of various algorithms that are used to improve the mining efficiency of HUIs. This paper describes in detail the algorithms of Two-Phase, U_Mining, FUM, CTU-PROL, IHUP, HUI_Miner, UP_Growth, UP_Growth$n$+, FHM and so on. There are papers presenting algorithms,



Tree-based HUPM algorithms

Static database

CTU-PRO(2008):
1. Tree Name: CUP-Tree (Compressed Utility Pattern Tree);
2. Tree structure: Each node contains an id of the mapped item and an array of TWU values for the pattern at that node, each array having a pointer to the number of associations.
3. Datasets: Modifed Retail, Modifed BMSPOS, T10N5D100K, T5N5DXM;
4. Advantages: Better performance on sparse data sets;
5. Disadvantages: Generate too many candidates.

UP-Growth(2010):
1.Tree name: UP-Tree (Utility Pattern Tree)
2.Tree structure: N.name: the item name of the node; N.count: the support count of the node;
N.nu: node utility which is an estimate utility value of the node; N.parent: the parent node of the node; N.hlink: a node link which points to a node whose item name is the same as N.name;
3. Datasets: BMS-Web-View-1、Chess、T10I6D100K;
4. Advantages: UP-tree is more compact and powerful, reducing the number of candidates;
5. Disadvantages: UP-Growth will only perform better if min_util is small. This is because when the number of candidates for UP-Growth and UP-Growth+ are similar, UP-Growth+ carries more calculations and is therefore slower.

UP-Growth+(2012):
1.Tree name: UP-Tree (Utility Pattern Tree)
2.Tree structure: The first 5 are the same as the tree structure in UP-Growth;
N.mnu: the minimal node utility of N.
3. Datasets: Accidents, Chess, Chain-store, Foodmart;
4. Advantages: It effectively reduces the number of candidates, especially when the database contains a large number of long transactions;
5. Disadvantages: It takes time and memory to check and store the minimum node utility.

Uspan(2012):
1. Tree name: LQS-tree(lexicographic quantitative sequence tree);
2. Datasets: Online shopping transactions, Mobile communication transactions;
3. Advantages: The HUSP can be efficiently identified from large-scale data with extremely low utility;
4. Disadvantages: The utility matrix is very complex and has high memory costs.

REPT(2014):
1.Tree name: This is the same with UP-Growth;
2. Tree structure: This is the same with UP-Growth;
3. Datasets: Accidents, Chain-store, Mushroom, Retail;
4. Advantages: Reduced search space and greatly reduced the number of candidate patterns generated;
5. Disadvantages: Increased runtime.

HIMU(2016):
1.Tree Name: (MIU)-tree;
2.Tree structure: N.name: the item name of the node;
N.hlink: a node link which points to its extension itemsets;
3. Datasets: Foodmart, Mushroom;
4. Advantages: Avoid scans of the database repeatedly;
5. Disadvantages: Processing trees is time consuming.

Incremental database and data strams

IHUP(2009):
1.Tree name: IHUPL-Tree (Incremental High Utility Pattern Lexicographic Tree),
IHUPTF-Tree(Incremental High Utility Pattern Transaction Frequency Tree),
IHUPTWU-Tree (Incremental High Utility Pattern-Transaction-Weighted Utilization Tree);
2.Tree structure: name: the item name of the node
twu: transaction weight utility
tf: transaction frequency;
3. Datasets: Mushroom, Retail, Kosarak, Chain-store;
4. Advantages: The number of nodes in the tree is relatively small;
5. Disadvantages: Generate a large number of candidates and consume high computation time to identify the actual patterns.

HUM-UT(2013):
1. Tree name: UT-Tree structure;
2. Tree structure: inner-node: (1) item name, (2) a pointer to parent node, and (3) a list of pointers to children nodes;
tail-node: (1) item name, (2) a pointer to parent node, (3) a list of pointers to children nodes, and (4) a list of utility information for each batch;
3. Datasets: Retail, T10.I4.D100K;
4. Advantages: The algorithm does not require additional database scanning and is more stable in different situations;
5. Disadvantages: Processing trees is time consuming.

MAHUSP(2017):
1.Tree name: MAS-Tree (memory adaptive high utility sequential pattern tree);
2.Tree structure: nodeName: the suffix of SN w.r.t. the sequence represented by the parent of N;
nodeUtil: the approximate utility of SN over the part of the data stream processed so far;
nodeRsu: the rest utility value (to be defined in the next section and used in memory adaptation) of SN.
3. Datasets: Kosarak, ChainStore, D10KC10T3S4I2N1K, D100KC8T3S4I2N10K;
4. Advantages: Adapt to memory allocation;
5. Disadvantages: Waste time.

**FIGURE 1.** Tree-based HUPM algorithms.

year of publication, authors, and deficiencies of the proposed algorithm.

Although these reviews are mainly from the perspective of mining types of HUIs, typical algorithms in HUPM, and key technologies of HUPM, we will further classify these key technologies. The difference lies in:

1) In the Apriori-based methods of the third chapter, the types of mining HUIs are classified into the classification criteria, namely, the algorithms of mining the complete set, the algorithms of mining the average itemsets, the algorithms of mining the closed itemsets, and the algorithm of mining the potential itemsets.

2) In the tree-based methods of the third chapter, the algorithm is classified into two categories, namely, the algorithms for mining static data and the algorithms for mining dynamic data (incremental databases and data streams), and each tree structure is described in detail in Figure 1. For example, the name of the tree, the components of the tree, and the like are listed.

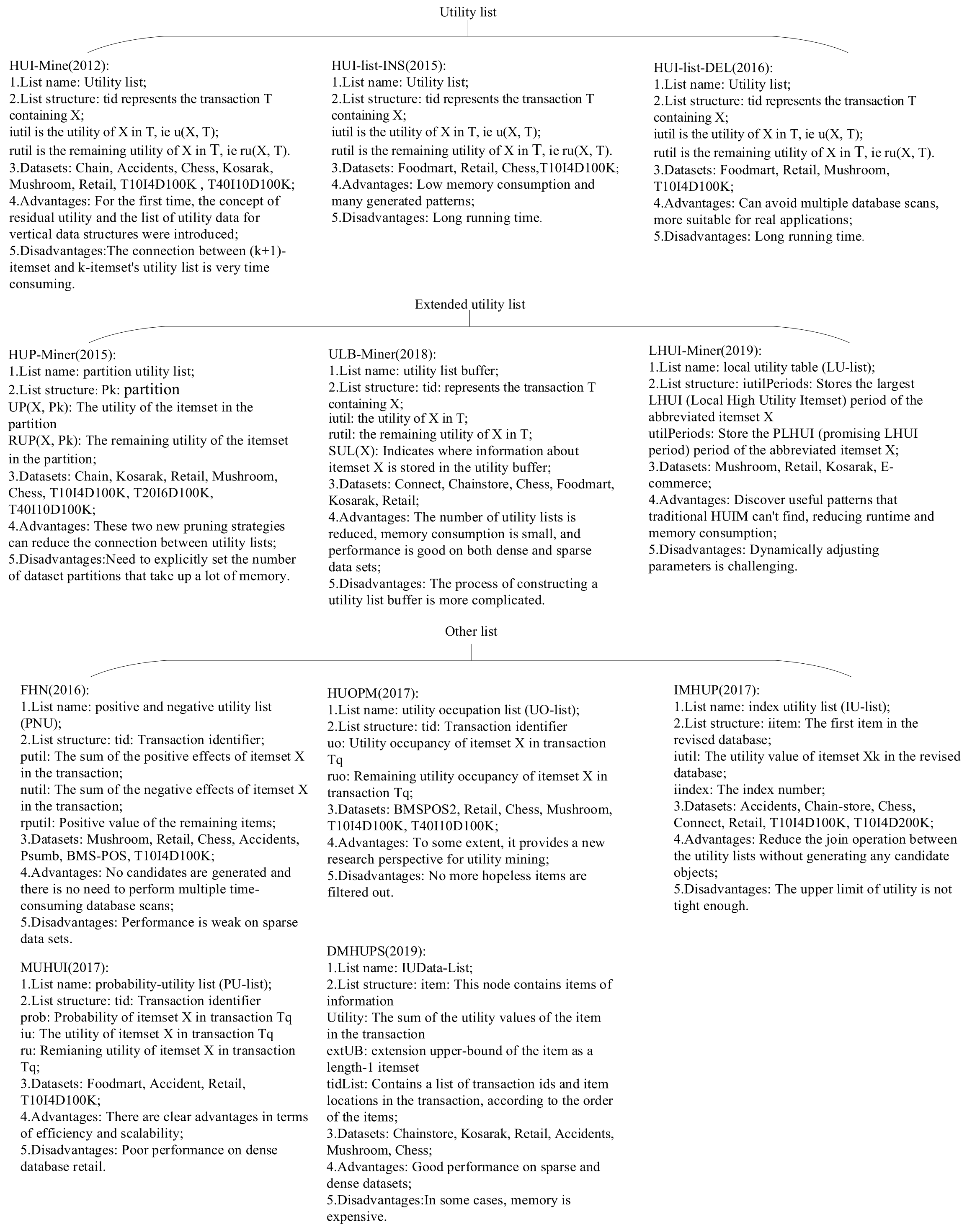
3) In the projection-based methods of the third chapter, the mapping methods used are the classification standard, which are the algorithms using parallel projection, the algorithms using prefix projection, and the algorithms using a combination of projection and transaction merging.

4) In the list-based methods of the third chapter, the algorithms are divided into three categories, namely, the algorithms using a utility list structure, the algorithms using an extended utility list structure, the algorithms using other list structures, and each list structure is performed in Figure 2. The detailed description, for example, lists the names of the lists, the components of the lists, and the like.

5) In the data-format-based methods of the third chapter, the data format used by the algorithms is a classification standard, which are the algorithms using horizontal representation, the algorithms using vertical representation, and the algorithms using an index.

This paper briefly describes the key technologies for HUMP in detail, outlines the high utility derivative patterns, and summarizes the efficient use of HUPMs' methods over data streams. Therefore, the main contributions of this paper are:

1) This paper provides a comprehensive and systematic summary of developments in the field, according to the knowledge level introduces the concept of HUPM, formulas



Utility list

HUI-Mine(2012):
1.List name: Utility list;
2.List structure: tid represents the transaction T containing X;
iutil is the utility of X in T, ie u(X, T);
rutil is the remaining utility of X in T, ie ru(X, T).
3.Datasets: Chain, Accidents, Chess, Kosarak, Mushroom, Retail, T10I4D100K , T40I10D100K;
4.Advantages: For the first time, the concept of residual utility and the list of utility data for vertical data structures were introduced;
5.Disadvantages:The connection between (k+1)-itemset and k-itemset's utility list is very time consuming.

HUI-list-INS(2015):
1.List name: Utility list;
2.List structure: tid represents the transaction T containing X;
iutil is the utility of X in T, ie u(X, T);
rutil is the remaining utility of X in T, ie ru(X, T).
3.Datasets: Foodmart, Retail, Chess,T10I4D100K;
4.Advantages: Low memory consumption and many generated patterns;
5.Disadvantages: Long running time.

HUI-list-DEL(2016):
1.List name: Utility list;
2.List structure: tid represents the transaction T containing X;
iutil is the utility of X in T, ie u(X, T);
rutil is the remaining utility of X in T, ie ru(X, T).
3.Datasets: Foodmart, Retail, Mushroom, T10I4D100K;
4.Advantages: Can avoid multiple database scans, more suitable for real applications;
5.Disadvantages: Long running time.

Extended utility list

HUP-Miner(2015):
1.List name: partition utility list;
2.List structure: Pk: partition
UP(X, Pk): The utility of the itemset in the partition
RUP(X, Pk): The remaining utility of the itemset in the partition;
3.Datasets: Chain, Kosarak, Retail, Mushroom, Chess, T10I4D100K, T20I6D100K, T40I10D100K;
4.Advantages: These two new pruning strategies can reduce the connection between utility lists;
5.Disadvantages:Need to explicitly set the number of dataset partitions that take up a lot of memory.

ULB-Miner(2018):
1.List name: utility list buffer;
2.List structure: tid: represents the transaction T containing X;
iutil: the utility of X in T;
rutil: the remaining utility of X in T;
SUL(X): Indicates where information about itemset X is stored in the utility buffer;
3.Datasets: Connect, Chainstore, Chess, Foodmart, Kosarak, Retail;
4.Advantages: The number of utility lists is reduced, memory consumption is small, and performance is good on both dense and sparse data sets;
5.Disadvantages: The process of constructing a utility list buffer is more complicated.

LHUI-Miner(2019):
1.List name: local utility table (LU-list);
2.List structure: iutilPeriods: Stores the largest LHUI (Local High Utility Itemset) period of the abbreviated itemset X
utilPeriods: Store the PLHUI (promising LHUI period) period of the abbreviated itemset X;
3.Datasets: Mushroom, Retail, Kosarak, E-commerce;
4.Advantages: Discover useful patterns that traditional HUIM can't find, reducing runtime and memory consumption;
5.Disadvantages: Dynamically adjusting parameters is challenging.

Other list

FHN(2016):
1.List name: positive and negative utility list (PNU);
2.List structure: tid: Transaction identifier;
putil: The sum of the positive effects of itemset X in the transaction;
nutil: The sum of the negative effects of itemset X in the transaction;
rputil: Positive value of the remaining items;
3.Datasets: Mushroom, Retail, Chess, Accidents, Psumb, BMS-POS, T10I4D100K;
4.Advantages: No candidates are generated and there is no need to perform multiple time-consuming database scans;
5.Disadvantages: Performance is weak on sparse data sets.

MUHUI(2017):
1.List name: probability-utility list (PU-list);
2.List structure: tid: Transaction identifier
prob: Probability of itemset X in transaction Tq
iu: The utility of itemset X in transaction Tq
ru: Remianing utility of itemset X in transaction Tq;
3.Datasets: Foodmart, Accident, Retail, T10I4D100K;
4.Advantages: There are clear advantages in terms of efficiency and scalability;
5.Disadvantages: Poor performance on dense database retail.

HUOPM(2017):
1.List name: utility occupation list (UO-list);
2.List structure: tid: Transaction identifier
uo: Utility occupancy of itemset X in transaction Tq
ruo: Remaining utility occupancy of itemset X in transaction Tq;
3.Datasets: BMSPOS2, Retail, Chess, Mushroom, T10I4D100K, T40I10D100K;
4.Advantages: To some extent, it provides a new research perspective for utility mining;
5.Disadvantages: No more hopeless items are filtered out.

DMHUPS(2019):
1.List name: IUData-List;
2.List structure: item: This node contains items of information
Utility: The sum of the utility values of the item in the transaction
extUB: extension upper-bound of the item as a length-1 itemset
tidList: Contains a list of transaction ids and item locations in the transaction, according to the order of the items;
3.Datasets: Chainstore, Kosarak, Retail, Accidents, Mushroom, Chess;
4.Advantages: Good performance on sparse and dense datasets;
5.Disadvantages:In some cases, memory is expensive.

IMHUP(2017):
1.List name: index utility list (IU-list);
2.List structure: iitem: The first item in the revised database;
iutil: The utility value of itemset Xk in the revised database;
iindex: The index number;
3.Datasets: Accidents, Chain-store, Chess, Connect, Retail, T10I4D100K, T10I4D200K;
4.Advantages: Reduce the join operation between the utility lists without generating any candidate objects;
5.Disadvantages: The upper limit of utility is not tight enough.

**FIGURE 2.** List-based HUPM algorithms.

and application examples. And it compares important concepts.

2) This paper has a deeper understanding and introduction of key technologies. And from different algorithms, the Apriori-based, tree-based, list-based, projection-based, vertical/horizontal-based and index-based methods are described in detail from different perspectives.

3) Further summarizes the high utility derivative patterns, and outlines the characteristics and advantages of derivative patterns compared to traditional patterns.

4) This paper briefly analyzes the methods of high utility patterns in incremental databases and on data streams. It is mainly described from three aspects, namely based on sliding window model methods, time decay

model-based methods and landmark window model-based methods.

The remainder of this paper is organized as follows: Chapter II describes related concepts, such as transactional utility and transaction-weighted utility, which are used in the algorithms of the following chapters. Chapter III details extended analysis of HUPM, chapter IV describes high utility derivative patterns. Chapter V describes methods for HUPM over data streams. Finally, chapter VI identifies the next research directions related to HUPM and chapter VII summarizes the full text.

## II. BASIC CONCEPTS

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items, and $D = \{T_1, T_2, \ldots, T_n\}$ is a transaction database. The items in each transaction $T_i$ are a subset of $I$. The number of items $i_p$ $(1 \leq p \leq m)$ in the transaction $T_q$ $(1 \leq q \leq n)$ is represented by $o(i_p, T_q)$. The external utility $s(i_p)$ is the unit value of the item $i_p$ in the utility table (for example, per unit profit). The utility of the item $i_p$ in transaction $T_q$, represented by $u(i_p, T_q)$, is defined as $o(i_p, T_q) \times s(i_p)$. Let itemset $X$ be a subset of $I$. The utility of $X$ in transaction $T_q$, represented by $u(X, T_q)$, is defined as:

$$u(X, T_q) = \sum\nolimits_{i_p \in X} u(i_p, T_q) \tag{1}$$

The utility of itemset $X$ in the database, represented by $u(X)$, is defined as:

$$u(X) = \sum\nolimits_{T_q \in D \wedge X \subseteq T_q} u(X, T_q) \tag{2}$$

If the itemset's utility is not less than the *minutil*, the itemset is called a high utility itemset. Otherwise, it is called a low utility itemset. The task of HUIM is to find all high utility itemsets. Since the utility does not have an anti-monotonic property, the concept of transaction utility [1] (TU) and transaction weighted utility [1] (TWU) is used to prune the search space of HUIs. The transaction utility of a transaction, expressed as $tu(T_q)$, is the sum of the utilities of all items in $T_q$:

$$tu(T_q) = \sum_{i_p \in T_q} u(i_p, T_q) \tag{3}$$

The transaction weighted utility of itemset $X$, expressed as the sum of the transaction utility of $TWU(X)$ for all transactions containing $X$:

$$TWU(X) = \sum_{T_q \in D \wedge X \subseteq T_q} tu(T_q) \tag{4}$$

Suppose there is a small transaction database, as shown in Table 1. Table 2 shows the profit (external utility) for each item. The value of each row in Table 1 represents the number of each item purchased in a particular transaction (i.e., local transaction utility). The last column shows the transaction utility of each transaction. In transaction $T_2$, three $A$, one $C$ are purchased, and a transaction utility of 60\$ is generated. The utility of item $A$ is $u(A, T_2) = 3 \times 10 = 30\$$. The

**TABLE 1.** Transaction database.

| Tid | A | B | C | D | Transaction Utility ($) |
|---|---|---|---|---|---|
| $T_1$ | 1 | 1 | 0 | 1 | 70 |
| $T_2$ | 3 | 0 | 1 | 0 | 60 |
| $T_3$ | 0 | 1 | 1 | 1 | 90 |
| $T_4$ | 1 | 2 | 0 | 1 | 90 |
| $T_5$ | 1 | 1 | 1 | 0 | 60 |

**TABLE 2.** Utility table.

| Item | Profit ($) |
|---|---|
| A | 10 |
| B | 20 |
| C | 30 |
| D | 40 |

utility of the item $A$ in the entire database is $u(A) = 1 \times 10 + 3 \times 10 + 1 \times 10 + 1 \times 10 = 60\$$. In transactions $T_1$ and $T_4$, the itemset $AD$ occurs two times. In transaction $T_1$, $u(AD, T_1) = 1 \times 10 + 40 = 50\$$; in transaction $T_4$, $u(AD, T_4) = 1 \times 10 + 40 = 50\$$; and in the entire database, $u(AD) = 50 + 50 = 100\$$. The $TWU$ of item $C$ is the sum of the transactions $T_2$, $T_3$ and $T_5$, $TWU(B) = tu(T_2) + tu(T_3) + tu(T_5) = 60 + 90 + 60 = 210\$$, and the $TWU$ of the itemset $BC$ is the sum of the transactions $T_3$ and $T_5$, $TWU(BC) = tu(T_3) + tu(T_5) = 90 + 60 = 150\$$.

## III. KEY TECHNOLOGIES FOR HIGH UTILITY PATTERNS MINING

In recent years, researchers have proposed a large number of algorithms to mine HUIs from data. This chapter selects some classic and representative HUPM algorithms, and classifies the key technologies used in the algorithm according to different mining principles and data structures. Table 3 is a summary table of the HUPM algorithms listed in chapter III according to the release year. Specifically, for ease of discussion, this chapter divides these tasks into the following categories:

- Apriori-based methods;
- Tree-based methods;
- Projection-based methods;
- List-based methods;
- Vertical/horizontal data and index-based methods.

### A. APRIORI-BASED METHODS

The researchers propose a well-known downward closure property, also known as the Apriori [38] property, which specifies that all non-empty subsets of a frequent itemset must be frequent, while any supersets of an infrequent itemset can't be frequent.

In addition to processing data, HUPM can also be used for unstructured data, including text, images, and video. Taking the image as an example, the analysis is performed as shown in Table 4, which is a graphical representation of the entities in different application fields.

**TABLE 3. Algorithms summary table by release year.**

| Year | Algorithms |
|---|---|
| 2005 | Two-phase[1] |
| 2008 | CTU-PRO[31] |
| 2009 | IHUP[2] |
| 2010 | FUP[16], UP-growth[11], UtilityLevel[58] |
| 2011 | CHUD[30], MHAI[9] |
| 2012 | USpan[59], HUI-Miner[12] |
| 2013 | UP-growth+[10], HUM-UT[33] |
| 2014 | REPT[29], PHUS[60], FHM[13] |
| 2015 | EFIM[50], HUI-list-INS[15], HUP-Miner[46], CHUI-Miner[57], HUI-MMU[77], HUI-MMUTID[77] |
| 2016 | PHUI-UP[67], HIMU[20], MAUGrowth[44], HUI-list-DEL[54], CHUM[32], HAUI-Miner[72], FHN[75], IMHUP[49], IHUI-Mine[47] |
| 2017 | MAHUSP[26], MUHUI[74], HUPNU[52], SPHUITP[63] |
| 2018 | CHN[69], EHIN[78], SPHUI-Miner[70], ULB-Miner[64], HUOPM[55], CRUSP[66], MHUI[65] |
| 2019 | dHAUIM[83], MEFIM[84], EHNL[85], LHUI-Miner[39], PHUI[39], NPHUI-Miner[39], DMHUPS[56], ISR-MOEA[73] |

**TABLE 4. Graphical representation of entities in different application area.**

| Application | Graphics | Vertex | Side |
|---|---|---|---|
| Web mining | Web browsing mode | Web page | hyperlink between pages |
| computing chemistry | Chemical compound structure | atom or ion | bond between atoms or ions |
| network computing | computer network | computer and server | interconnection between machines |
| semantic Web | collection of XML documents | XML element | father-son contact between elements |
| bioinformatics | protein structure | amino acid | contact residue |

In the mining process, the graph can be transformed into a similar transaction form, using an existing algorithm such as Apriori. In this case, the edge and the corresponding vertex combination are mapped to an item. The width of the transaction is determined by the number of sides of the graph. However, this method works only if each edge of the graph has a unique combination of vertices and edges.

The Apriori-like algorithms for mining high utility subgraphs consists of the following steps.

1) Candidate generation: combine efficient $(k-1)$-sub-graph pairs to obtain candidate $k$-sub-graphs.

2) Candidate pruning: discard all candidate $k$-subgraphs of the inefficient $(k-1)$-sub-graph.

3) Support count: count the number of graphs for each candidate.

4) Candidate delete: discard all candidate subgraphs that are smaller than *minutil*.

In order to apply Apriori's downward closure property to the utility problem, based on the two-phase model, the Two-Phase [1] was designed, and two attributes of TWDC and TWU were introduced to discover HUIs. In the first phase, the algorithm uses Apriori similar level candidate generation and testing strategies to find all itemsets with TWUs that are not less than the *minutil*. Then, in the second phase, the algorithm scans the database to find the actual utility value of the itemset found in the first phase. The TWU attribute not only limits the search space, but also covers all HUIs.

With the continuous research of relevant researchers, other HUPM can be explored based on the Apriori method, such as high average utility patterns and high closed utility patterns. FUP [16] uses the downward closure property to search the high average utility itemsets step by step. With this attribute, the number of candidate sets generated at each level is greatly reduced and the search space is also reduced. PHUI-UP [67] uses a horizontal method to mine potential high utility itemsets (PHUIs) based on Apriori-like methods and designed upper limit models. In an uncertain database, the algorithm can be used by researchers as one of the most advanced algorithms in future work. With the continuous advancement of HUPM research, Apriori-based algorithms are also used to mine closed itemsets. CHUD [30] is based on a two-phase algorithm that extracts the set of possible high utility closed itemsets in the first phase and calculates the actual utility information of the set in the second phase.

In summary, all early HUPM methods have improved Apriori-based algorithms. Apriori uses step-by-step candidate generation and testing methods. The advantage is that using the Apriori method can delete a large number of candidates, improve the efficiency of mining useful patterns, and also have good performance in restoring all HUIs, such as the algorithms Two-Phase [1] and CHUD [30]; also can reduce the time to reprocess the entire update database, such as algorithm FUP [16]. However, Apriori-based algorithms also have many shortcomings, such as multiple database scans, such as the algorithms Two-Phase [1] and PHUI-UP [67]; due to the generation of a large number of candidate sets in Phase I, a large amount of memory is consumed, such as the algorithms CHUD [30] and FUP [16]. The specific algorithms are shown in Table 5.

## B. TREE-BASED METHODS

Although the Apriori methods can effectively mine HUIs, but there are problems such as generating a large number of candidates, repeatedly scanning the database, and running slowly and so on. In order to avoid these shortcomings, tree-based HUIM algorithms are proposed. These tree-based algorithms consist of three steps: 1) building the tree; 2) generating candidate HUIs from the tree using algorithms; 3) identifying HUIs from the candidate set.

Tree-based methods are widely used in static databases. UP-Growth [11] and UP-Growth+ [10] can reduce the number of candidates for extraction by an improved overestimation method, they need two database scans to build their own tree structure, named UP-Tree. CTU-PRO [31] mines HUIs by traversing the compressed utility pattern (cup) tree from bottom to top. The TWU concept is used to prune the search space in CTU-PRO, but it avoids rescanning the database to determine the actual utility of a high transaction-weighted utility set. The algorithm adapts to larger data sets by building parallel subdivisions that can be independently

**TABLE 5.** **Apriori-based HUPM algorithms.**

| Algorithms Name | Type of HUIs | Datasets | Advantages | Disadvantages |
|---|---|---|---|---|
| Two-Phase (2005) | complete itemsets of HUIs | T20I6D1000K, Chain-store | Cut a large number of candidates. | The candidate level is searched after the test, and multiple database scans are required. |
| FUP (2009) | high average utility itemsets | Chain-store | Reduce the time to reprocess the entire update database. | Generate a large number of candidates. |
| CHUD (2011) | closed high utility itemsets | Mushroom, Foodmart, BMSWebView1, T10I8D200K | It can reduce a large number of undesired HUIs. In addition, the combinations mentioned in this paper are superior to current state-of-the-art algorithms when all HUIs can be recovered. | The algorithm consumes a lot of memory and consumes more runtime in transaction intersections. When there are a large number of candidate high utility closed itemsets, the algorithm will degrade performance. |
| PHUI-UP (2016) | potential high utility itemsets | T10I4D100K, Foodmart, Accident, Retail | It is an improvement for uncertain databases. | There is a problem of repeatedly scanning the database. |

complete itemsets of HUIs: In database $D$, let $Xi$ ($1 \leq i \leq n$) be the itemset. If $u(Xi) \geq minutil$, the set of $Xi$ is the complete itemset of high utility itemsets.
high average utillity itemsets: The sum of the utility of an itemset in the transactions in which it occurs is divided by the number of items it contains.
closed high utility itemsets: If the itemset $X$ is a high utility itemset and there is no itemset $Y$ such that $X \subset Y$ and $support(X)=support(Y)$, then the itemset $X$ is called a closed high utility itemset.
potential high utility itemsets: If the potential probability of $X$ is greater than or equal to the minimum potential probability, the itemset $X$ is represented as a high potential itemset (HPI). If $X$ satisfies both the HUI and the HPI, then $X$ is a potential high utility itemset (PHUI).

mined on disk. Recently, many novel tree structures have been developed to improve the performance of the excavating HUPM. USpan [59] constructs a lexicographic quantitative sequence tree (LQS-tree) structure using a sequence weighted utility (SWU) and a sequence weighted downward closure (SWDC) attribute. Trim the sequence without the desired one and extract the complete HUSP. dHAUIM [83] uses a new structure called the IDUL prefix tree to quickly calculate the average utility and the utility ceiling of the itemset using a recursive process to maintain a high utility average itemset. Based on the high utility average patterns, the researchers proposed the MAUGrowth [44], which applied MAUTree to mine the high-average utility rare patterns from the database. The algorithm takes into account the length of the pattern in order to effectively reduce the dependence of the pattern on its own length in order to mine rare patterns that are more meaningful than the patterns mined by previous algorithms. In order to predict the exact number of patterns mined by thresholds and accurately control the mining results, the top-k mining is proposed. REPT [29] is an efficient algorithm to mine the top-k HUIs with a greatly reduced number of candidates. Reduce the search space by effectively increasing the minimum threshold when building a global tree through three strategies. The traditional HUIM algorithms only allow the user to specify a *minutil* to evaluate the utility of all patterns. But in real life, each item in the database is different, so using a single *minutil* makes it difficult to measure the utility of an item or itemset fairly. For example, in a retail store, if the itemset {necklace} earns more than $1,000 per week, it may consider as the HUI; and the itemset {bread, milk} eaens more than $100 per week, it may consider as the HUI. Using traditional HUIM algorithms, if the *minutil* is set to higher, the user will miss useful patterns with lower utility, and if the *minutil* is set to lower, the number of HUIs will be large. Therefore, using a single *minutil* to evaluate the utility of an item is insufficient because it does not take into account the importance of each item. HIMU [20] uses multiple *minutils* to mine HUIs. It proposes multiple item utility set-enumeration (MIU-tree), and global and conditional downward closure (GDC and CDC) properties of HUIs in MIU-tree. This algorithm is more flexible and more realistic than using a single *minutil*.

With the advent of the big data era, researchers have used tree-based methods to solve problems in incremental databases and data streams. IHUP [2] is one of the most advanced algorithms for HUPM. It constructs its own tree structure, called IHUP-Tree, with a single channel, and finds all HUIs based on traditional overestimation methods. However, it generates a large number of candidates. MAHUSP [26] designs an efficient tree structure, MAS-Tree, for storing potential HUSP on the data streams. The algorithm can not only effectively discover the HUSP on the data streams, but also adapt to the memory allocation in the sacrifice of the discovered the quality. From the constant discovery of researchers, existing algorithms require multiple database scans to mine HUIs, which hinders their efficiency. HUM-UT [33] is used to find HUIs from transactional data streams and proposes a new data structure, UT-Tree (utility on the tail tree). The structure is created by a database scan, and utility information is only stored on the tail node to maintain utility information for the transaction item set to avoid multiple database scans.

In summary, the advantages of tree-based algorithms are that the performance of larger data sets containing dense data sets and sparse data of long patterns are better, such as the algorithm CTU-PRO [31]; effectively reduce the quantity of candidates, avoiding repeated scans of databases, such as the algorithms UP-Growth [11], UP-Growth+ [10], REPT [29], HIMU [20] and HUM-UT [33]; the number of tree nodes in the tree structures used is small, and adapt to memory allocation, such as the algorithm IHUP [2], MAHUSP [26]. Although these tree structures are usually compact, they may not be minimal and still take up a lot of storage space. The mining performance of these methods is closely related to the number of conditional trees constructed throughout the mining process and the cost of building/traversing each condition tree. Therefore, one of the performance bottlenecks of these algorithms is to generate a large number of conditional trees with high time and space costs. The disadvantages are that a large number of candidates are generated, such as the algorithms CTU-PRO [31] and IHUP [2]; it takes time and memory to check and store the minimum node utility, such as the algorithm UP-Growth [11]; the process of processing the tree structure is consuming time, such as the algorithms REPT [29], HIMU [20], HUM-UT [33] and MAHUSP [26]. The specific algorithm is shown in Figure 1.

### C. PROJECTION-BASED METHODS

In order to overcome the shortcomings of the tree-based approach, researchers have proposed some projection-based methods to improve the mining performance, which have been widely used in data mining. The general idea is to recursively project the processed database to some smaller mapping sub-databases. Then in each mapping sub-database, grow the itemset or sub-sequence fragment [34].

When the main memory is not large enough to handle large data sets, researchers use a parallel projection scheme to use disk storage. CTU-PROL [31] creates a subdivision for a data set that is too large to be saved in main memory, using parallel projections that can then be independently mined. The inverse monotonic nature of TWU is used to trim the search space subdivided in CTU-PROL.

Prefix-based projection method, the utility upper bound can be effectively improved and the mining process can be optimized. PHUS [60] extends the PrefixSpan [61] and uses a projection-based pruning strategy to achieve a compact upper bound on sequence utility. The concept of maximum utility metric and sequence-utility upper-bound (SUUB) model is proposed. Therefore, it can avoid considering too many candidates and use the SUUB model to improve the performance of mining HUSP.

In actual research, researchers not only use mapping methods to mine HUIs, but also combine them with transaction merging to further improve the mining performance. EFIM [50] is an efficient algorithm based on one-phase projection. In order to reduce the cost of database scanning, EFIM further proposes database projection and transaction merging methods, namely high database projection (HDP) and high transaction merge (HTM). In order to process all HUIs in the dynamic unit profit database, the extended algorithm of EFIM, iMEFIM [84] is designed. It relies on database projection and another novel compact database format to efficiently discover the required itemsets. CHN [69] also applies this method and references a sub-tree-based pruning strategy, which reduces the pruning search space and speeds up the mining process. When studying a larger itemset, projection and merging will reduce the size of the database. EHNL [85] also applies the above methods, as well as negative utility and length constraints to mine HUIs. EHIN [78] uses dataset projection and merging techniques to reduce memory requirements and speed up the execution time of the mining process. Transaction consolidation is performed two times before and after mapping the dataset. These techniques reduce the search space.

Analyze the complexity of EFIM [50] and iMEFIM [84] algorithms from the time perspective. The time complexity of the EFIM [50] algorithm is $O(lnw)$, where $l$, $n$ and $w$ are the number of itemsets in the search space, the number of transactions in the database, and the average transaction time length. The iMEFIM [84] algorithm extends the EFIM algorithm to handle dynamic profit databases and reduces the cost of database scanning by using the P-set data structure. In addition, it takes a lot of time to attach and access elements inside the P-set. If the P-set is implemented using an array, it is $O(l)$ time. The value of $l$ is the total number of transactions contained in the P-set data structure, that is, $l = |\mathrm{P} - \mathrm{Set}|$.

In order to reduce the scanning time of the database more efficiently, the researchers propose the SPHUI-Miner [70], which is a new HUIs mining algorithm based on selective database projection. The selective projection of the database creates unique data instances and new mappings for data with smaller dimensions, enabling faster HUIs mining.

In summary, the projection-based algorithms have the advantages of avoiding rescanning the database and reducing scanning costs, such as the algorithms CTU-PROL [31], EFIM [50], CHN [69], and EHNL [85]; in projection-based pruning strategies, a more accurate sequence utility upper limit of the subsequence can be obtained, and therefore, the pruning effect and the execution efficiency are excellent, such as the algorithm PHUS [60]. The disadvantage is that a large number of redundant candidates are generated, such as the algorithms CTU-PROL [31], PHUS [60], CHN [69] and EHNL [85]. Specific algorithms are shown in Table 6.

### D. LIST-BASED METHODS

In HUPM, in addition to the tree-based approach, researchers have also explored another list-based approach. The mining steps are as follows: 1) perform a scan on the database to construct a utility list for each itemset; 2) scan the database again, modify the transaction in the utility list; 3) delete the itemset smaller than *minutil*, and reduce the search space. A list-based approach clearly maintains information about itemsets in a transaction, quickly calculates the utility of the itemset, and shortens the search space time.

HUI-Miner [12] uses a novel structure called utility-list to store utility information about the itemset and heuristic information for pruning search space. HUI-Miner effectively mines HUIs from the built-in utility list, avoiding expensive generation and utility calculations for a large number of candidate sets. According to Table 1 and Table 2, during the second database scan, the algorithm constructs the utility list for Table 7: itemset $\{AB\}$ and Table 8: itemset $\{BC\}$.

Each element in the utility list of itemset $X$ contains three fields: *tid*, *iutil*, and *rutil* [12]. *Tid* represents the transaction

**TABLE 6. Projection-based HUPM algorithms.**

| Algorithms Name | Specific methods | Datasets | Advantages | Disadvantages |
|---|---|---|---|---|
| CTU-PROL (2008) | parallel projection | Modifed Retail, Modifed BMSPOS, T10N5D100K, T5N5DXM | Avoid rescanning the database to determine the actual utility of high TWU itemsets. | A large number of candidates are generated, and the resources used by the algorithm may be high. |
| PHUS (2014) | prefix-based projection | S8T6I4N4KD200K | Good performance in both trimming effect and execution efficiency. | Generate a large number of candidates. |
| EFIM (2017) | projection and transaction merging | Accident, BMS, Chess, Connect, Foodmart, Mushroom | Less memory is consumed, and the complexity is roughly linear with the number of items in the search space. | Sometimes recursive projection is time consuming and takes up a lot of memory. |
| CHN (2018) | projection and transaction merging | Accidents, Chess, Mushroom, Pumsb, BMSPOS, Retail, kosarak, T40I10D100K, T10I4D100K | Good performance on dense and sparse data sets. | Using TWU, a large number of redundant candidates are generated. |
| SPHUI-Miner (2018) | projection and transaction merging | Webdocs, Accidents, Chess, Connect, Mushroom, Foodmart, Chainstore, Retail | Reduced database scan time and memory usage, enabling faster HUIs mining. | |
| EHNL (2019) | projection and transaction merging | Accidents, Chess, Mushroom, T40I10D100K | Dataset mapping and transactional consolidation techniques reduce scanning costs. | Generating a large number of candidates, mapping sometimes wastes time. |

**TABLE 7. Utility list for itemset {AB}.**

| Tid | Iutil | Rutil |
|---|---|---|
| 1 | 30 | 40 |
| 4 | 50 | 40 |
| 5 | 30 | 30 |

**TABLE 8. Utility list for itemset {BC}.**

| Tid | Iutil | Rutil |
|---|---|---|
| 3 | 50 | 40 |
| 5 | 50 | 0 |

$T$ containing $X$; *iutil* is the utility of $X$ in $T$, i.e. *iutil*$(X, T)$; *rutil* is the remaining utility of $X$ in $T$, i.e. *rutil*$(X, T)$.

Due to the introduction of utility-list, many algorithms use this structure to mine HUIs, thereby improving the mining performance. HUI-list-INS [15] inherits the HUI-Miner [12] and builds a utility list structure for mining HUIs in an incremental database for maintaining and updating discovered HUIs and for transaction insertion. HUI-list-DEL [54] is an algorithm for discovering HUIs by maintaining the built-in utility list structure of records deleted in the dynamic database. In this algorithm, new HUIs can be generated directly without the candidate generation and a large number of database scans.

With the continuous development of HUPM, the utility list can no longer meet the needs of the algorithm. The researchers have proposed a number of extension structures based on the utility list to further improve performance. The partition utility list data structure introduced by HUP-Miner [46] borrows the basic idea expressed by the tid-list [40]. It is also an extension of the utility list. The partition utility list of the item set *Rxy* is calculated by performing an intersection of the tid-lists of the itemsets *Rx* and *Ry*. This process is very similar to HUI-Miner [12]. The LA-Prune (based on forward pruning concept) strategy used by this algorithm provides a tighter utility ceiling for $k$-itemsets, so a large number of undesired $k$-items ($k \geq 2$) can be pruned, limiting the search space of HUIs. CHUI-Miner [57] uses a new structure of extended utility list (EU-List) to maintain the utility information of the itemsets in the transaction, which allows the original database not to be scanned and effectively computes in-memory itemsets utility and utility unit arrays. The algorithm uses a divide-and-conquer approach to mine the complete set of CHUIs in the database without generating candidates. CHUM [32] proposes a generic utility list (gutility-list) that stores utility information and heuristic information about search space pruning, which is different from the utility list proposed in HUI-Miner [12] because gutility-list can calculate closed itemsets quickly and instantly. ULB-Miner [64] uses the designed utility list buffer structure to efficiently store and retrieve utility lists and reuse the memory during the mining process. A linear time method is also used to construct a utility list segment in the utility list buffer. LHUI-Miner [39] aims to discover local high utility itemsets (LHUI), depending on the local utility table (LU-list). Since the utility of itemsets changes over time, it is desirable to find a point in time at which the utility of the itemsets change significantly (increases or decreases). Hence, this algorithm can find useful patterns, for example, the product set {schoolbag, pen and notebook} makes high profits during the back-to-school shopping season, and is not HUIs during other time periods (such as summer or autumn). Therefore, the extended PHUI [39] mines the peak high utility itemsets. It includes a time period in which the lookup itemset has a very high utility. In addition, since the collection of PHUIs can be large and some items in the PHUI do not contribute much to their peaks, NPHUI-Miner [39] is used to mine a set of non-redundant peak efficient itemsets. The complexity of the proposed algorithms are as follows. The time and space complexity required by LHUI-Miner to process each itemset is linear. The number of itemsets depends on how the parameters are set. The steps performed by the

PHUI-Miner algorithm are similar to the LHUI-Miner, so the complexity is similar. The complexity of the post-processing steps performed by NPHUI-Miner is $O(n \times 2n)$ in the worst case ($n$: $n$ individual terms). However, it can be seen from the experiments that the execution time of NPHUI-Miner and the execution time of PHUI-Miner are similar.

A major limitation of the utility list-based algorithm is that creating and maintaining a utility list is time consuming and can consume a lot of memory. The reason is that many utility lists are built, and the intersection/join operation of the utility list is costly. Therefore, researchers have developed a number of novel and special list structures to improve the mining performance. HAUI-Miner [72] algorithm based on the average utility list (AU-list) structure, which is used to store the information needed to discover the HAUI, thus a very large database compresses into a compressed structure to speed up the mining process. MHAI [9] proposes a new list structure, which is a high average utility set list (HAI-List). This structure can capture the necessary information compactly, which allows the algorithm to generate HAUI from a given transaction database without generating a candidate set. HUOPM [55] considers user preferences in terms of frequency, utility, and occupancy. The algorithm uses the utility occupation list (UO-list) and the frequency utility table (FU-table) to store the information needed about the database to mine the high utility occupied pattern (HUOP). The proposed method can effectively discover a complete set of HUIs without the candidate generation. For example, considering travel route suggestions for tourists visiting, eating, and spending time/money, HUOPM can successfully utilize the frequency and utility contribution rate of a particular travel route. DMHUPS [56] utilizes the data structure of the IUData-List, which stores information of a 1-itemset and their position in the transaction to efficiently obtain the initial database. In addition, the algorithm simultaneously calculates the utility of multiple promising candidates and the stricter upper limit of expansion, avoiding the generation of redundant items, and finding multiple efficient patterns. CRUSP [66] applies a removed-utilities list (RUL) and a removed-utility-positions list (RUPL). This list specifies the only items that need to be considered, as possible candidates for concatenation of sequential patterns in question, or any future sequential patterns that appear as descendants in the search tree. MUHUI [74] is based on a probability-utility list (PU-list) structure, which can directly mine PHUI in an uncertain database(Such as wireless sensor networks, RFID, GPS and WiFi systems) without generating candidates, and through the effective pruning strategy to avoid building a PU-list for many undesired itemsets, which greatly improves performance. Although traditional HUIM algorithms consider both profit and quantity of items, they usually assume that the profit value of items in the database is positive. However, in practical applications, the profit value of the goods in the store may be negative. For example, when a supermarket wants to attract customers' attention to a particular product, a common sales strategy is to offer free or unpackaged products along with other products. Therefore, researchers have proposed HUIM algorithms for negative utility. FHN [75] relies on a novel list structure, positive and negative utility list (PNU), while considering positive and negative unit profit. The structure is designed to maintain all the information about HUIs, allowing the algorithm to directly mine HUIs without generating candidates and without having to perform multiple time-consuming database scans. HUPNU [52] mines HUIs for positive and negative unit profits based on a probability-utility list with positive-and-negative profits (PU$\pm$-list). Further effective pruning strategies are proposed. When constructing the PU$\pm$-list, many early items that do not have pre-sets can be trimmed to reduce the search space. IMHUP [49] uses an index utility-list (IU-list) to discover HUIs more efficiently through newly proposed item join operations without any comparison.

In summary, the advantages of list-based algorithms are that HUI-Miner [12] first introduced the concept of remaining utility and the utility list of vertical data structures. Subsequently, many algorithms use utility list structures, such as HUI-list-INS [15] and HUI-list-DEL [54], which can reduce memory consumption and avoid multiple database scans. Expanded utility list structures and other list structures can reduce memory consumption and connection operations between utility lists, such as ULB-Miner [64], IMHUP [49], HUP-Miner [46] and MUHUI [74]. Some novel algorithms that consider both HUIs for a specific period of time can be used to mine patterns that cannot be found by traditional HUIM, thereby reducing runtime and memory consumption, such as LHUI-Miner [39]; and expanding the occupancy rate to evaluate the transaction database, to a certain extent, providing a new research perspective for utility mining, such as HUOPM [55]. Therefore, although most list-based methods can speed up mining and have a good performance on sparse and dense databases, disadvantages are that the connection between lists requires high costs, time-consuming time, and excessive memory usage and so on. For example, in HUI-Miner [12], the connection between the utility list of the $(k + 1)$-itemset and the utility list of the $k$-itemset is very time consuming, resulting in a long running time. Based on the expanded structure of the utility list or other list structure, there are problems such as complicated construction process. It is necessary to display the number of data set partitions that occupy a large amount of memory; the process of constructing the list is more complicated, and dynamically adjusting parameters is challenging, such as HUP-Miner [46], ULB-Miner [64] and LHUI-Miner [39]. The specific algorithms are shown in Figure 2.

## *E. DATA FORMAT-BASED METHODS*

In order to overcome the shortcomings of the above methods, researchers have recently proposed horizontal and vertical data structures and index structures to mine HUIs. On the one hand, it can speed up the progress of the data mining, on the other hand, it can improve the performance of mining.

The horizontal data structure is the most basic data structure. It is used in many algorithms. UtilityLevel [58] uses a horizontal candidate generation and the testing mechanism to mine the HUSP. Therefore, it generates a large number of candidate sequences and requires multiple database scans. SPHUITP [63] mines short-term high utility patterns in a horizontal manner. These patterns appear regularly, profitable and also efficient for use during periods of constraints.

Like the Eclat [40] mining frequent itemsets, HUI-Miner [12] proposes a table structure with a vertical data structure. The algorithm first checks all 1-extension sets by constructing a utility list, then, after identifying and outputting HUIs from the extended set, recursively processes the promising set of extensions one by one and discards the other sets of extensions. FHM [13] is an enhanced version of the HUI-Miner [12] that uses the same vertical data structure to speed up the mining process. CHUM [32] uses a database vertical representation to speed up the generation of the itemsets closure and calculates the execution time of its utility information without accessing the database, and efficiently generates an order retention generator. MHUI [65] is used to efficiently mine HUIs with multiple *minutils*. The proposed algorithm utilizes a vertical database representation to efficiently store itemsets information and introduces a new concept of suffix minimum utility (SMU) to efficiently mine HUIs.

ISR-MOEA [73] is a multi-objective evolutionary algorithm based on index set representation for mining diverse top-k HUIs. In the algorithm, it is recommended to use an index set individual representation scheme to quickly encode and decode the top-k pattern set. IHUI-Mine [47] uses the subsume index [48] to enumerate the required HUIs and trim the search space. HUI-MMU [77] is used to mine HUIs with multiple *minutils*. The improved HUI-MMUTID [77] is based on the TID indexing strategy, HUIMMUTID, to speed up the mining process.

In summary, data format-based algorithms have the advantages that algorithms based on horizontal or vertical representation can greatly reduce the number of patterns found, effectively identify HUIs in the database, and avoid ‘‘rare item problems’’, such as SPHUTTP [63], HUI-MMUTID [77]. An index-based novel algorithm can achieve multi-objective evolution, and explore a variety of top-k high utility patterns to further improve user satisfaction, such as ISR-MOEA [73]. Disadvantages are that it is necessary to scan the database multiple times to mine HUIs, and the memory consumption is large, such as UtilityLevel [58], SPHUTTP [63], CHUM [32], HUI-MMUTID [77]; the algorithm of mining HUIs with multiple minimum utility thresholds may be insensitive to the selection of the *minutil*, such as MHUI [65]. Specific algorithms are shown in Table 9.

## IV. HIGH UTILITY DERIVATIVE PATTERNS

With the increasing use of data mining, HUIM has become a key issue in recent decades. Traditional HUIM can no longer meet the needs of users. In order to return the representative HUIs to the user, some derived HUIs representations are proposed, such as high average utility pattern, high utility sequential pattern, and high utility compact pattern and so on.

### A. HIGH AVERAGE UTILITY PATTERNS

One of the main challenges of HUIM is that HUIM's search space is very large when the number of different items or the size of the database is too large. Another challenge is that the existing HUIM approaches ignore the fact that a longer itemset leads to a higher utility. A large itemset may have an unreasonable estimated profit, not its actual value. Therefore, the concept of high average utility itemset mining (HAUIM) is proposed. HAUIM introduces an average utility metric that finds more useful patterns by considering both the utility and length of the itemsets, so it is more appropriate in the real world.

**TABLE 9.** Data format-based HUPM algorithms.

| Algorithm Name | Data Format | Datasets | Advantage | Disadvantage |
|---|---|---|---|---|
| UtilityLevel (2010) | Horizontal representation | D100K.C8.T6.S6.I5.N10K, D200K.C10.T8.S8.I7.N10, BMS-WebView-1, BMS-WebView-2 | More direct and simpler. | A large number of candidate sequences are generated and multiple database scans are required. |
| SPHUTTP (2017) | Horizontal representation | Retail, Chess, Mushroom, T10I4D100K | Short-period constraints and utility metrics can greatly reduce the number of patterns found. | Need multiple database scans, which is a waste of time. |
| FHM (2015) | Vertical representation | Chain-store, BMS, Kosarak, Retail | It not only has the advantages of HUI-Miner, but also reduces the connection among the utility lists. | It consumes a bit more memory than HUI-Miner and performs poorly on dense data sets. |
| CHUM (2016) | Vertical representation | Mushroom, Retail, Foodmart, Chain-store, T15I5D100K, T10I4D100K | Good performance in dense, sparse and real data sets. | Take up a lot of memory. |
| MHUI (2018) | Vertical representation | Chain, Kosarak, Pumsb, Accidents、Connect, Chess, T10I4D100K, T40I10D100K | Good performance on medium length and dense benchmark data sets. | Not sensitive to the selection of the minimum utility threshold. |
| HUI-MMUTID (2015) | Index | Retail, T10I4D100K | It can effectively identify all HUIs in the database and avoid "rare item problems". | The memory consumed is large. |
| ISR-MOEA (2019) | Index | Chess, Mushroom, Connect, Accidents, Powerc, OnlineRetai, D300kN3k, D200kN3k, D300kN3k | Further improve user satisfaction. | The random initialization used cannot be overwritten at all. |

HAUI-Miner [72] is based on an efficient average-utility (AU) list structure that preserves only the information needed in the mining process, thereby compressing very large databases into compressed structures for more efficiently discovering high average-utility itemsets (HAUIs). FUP [16] is a two-phase average utility mining algorithm that can gradually maintain HAUIs as the database grows. EHAUPM [80] proposes two new and more rigorous upper bound models as an alternative to the traditional auub model using HAUI. The looser upper-bound model (lub) considers the maximum utility remaining in the transaction to reduce the upper limit of the utility of the itemset. The second rtub model ignores irrelevant items in the transaction to further tighten the utility. TUB-HAUPM [81] applied two new, more restrictive upper-bounds, greatly reducing the search space for mining HAUI. The maximum following utility upper-bound (mfuub) model is first designed to take into account the arithmetic mean of the current set of items and the following terms in the search path. The second is the revised top-k average utility upper-bound (krtmuub) model, which is an extension of the rtub model. It gets a tighter limit than the rtub model. IHAUPM [82] processes incremental databases with transaction insertions. An efficient HAUP tree structure is used to maintain the information needed for future mining progress. The itemset for each case can be properly maintained and processed. Compared to existing state-of-the-art algorithms that run in batch mode, design algorithms sometimes do not need to rescan the original database to update the discovered HAUI. dHAUIM [83] uses four tight average utility ceilings based on vertical database representation, three effective pruning strategies, and a new generic framework for comparing average utility bounds to efficiently mine HAUIs. MEMU [51] uses a plurality of minimum high average utility thresholds based on the average utility list to discover all HAUIs. Each item can be associated with a user-defined minimum high average utility threshold, which is more realistic than the original HAUIM task.

### B. HIGH UTILITY SEQUENTIAL PATTERNS

Another major challenge for HUIM is the possibility of discovering that a certain itemset has a higher utility and ignores its continuity. In practical applications, each user's historical transaction record is listed as a long-term transaction sequence, and a time-regular purchase pattern can be found from the time series data. Therefore, high utility sequential pattern (HUSP) refers to identifying patterns in the database that the sum of all their events exceeds the minimum utility.

UtilityLevel [58] uses the horizontal generation candidate generation method, while UtilitySpan [58] uses the pattern growth method. They are all used to mine HUSPs to extract more realistic utility information from the sequence database.

Since the upper limit of the algorithms before sequence utility is not compact enough, HuspExt [62] extracts HUSP based on accumulated rest of match (CRoM) and pruning techniques. CRUSP [66] uses RUL and RUPL, specifying the only items that need to be considered, as a possible candidate for the sequence pattern being considered, or any future sequential pattern that appears as a descendant in the search tree.

In HUSPM, some methods use a utility matrix to store a sequence database in memory. However, the utility matrix consumes a large amount of main memory. To solve this problem, the AHUS [45] and AHUS-P [45] introduce a pure array structure that reduces memory consumption compared to the utility matrix. AHUS-P uses shared memory to parallelize the mining process and identify HUSP based on the advantages of multi-core processor architecture.

With the continuous improvement of the methods, the researchers proposed algorithms over the data streams. MAHUSP [26] uses a memory adaptive mechanism to use the bounded portion of the memory and proposes a MAS-Tree tree structure to efficiently discover the HUSP on the data streams. MAHUSP guarantees that all HUSPs are found under certain circumstances.

### C. HIGH UTILITY COMPACT PATTERNS

The high utility compact patterns have high utility closed pattern and high utility maximum pattern. One limitation of HUIM is that it may present a large amount of redundant HUIs to the user, resulting in excessive memory consumption or lack of storage space or not operate. In order to achieve high efficiency of mining tasks and provide users with simple mining results, high utility closed itemset mining (CHUI) is used to mine compact and lossless HUIS, and high utility maximum pattern is used to mine the largest HUIs.

CHUD [30], by using a new structure called an array of utility units, allows efficient recovery of all HUIs and their utility to mine CHUI. This is the first algorithm to study the compact and lossless representation of HUIs. CHUI-Miner [57] directly calculates the utility of itemsets without generating candidates. This is the first job to solve the problem of mining CHUIs without the candidate generation. CHUM [32], which is scalable and efficient. This algorithm uses a vertical representation of the database to speed up the generation of the itemsets closure and calculate the execution time of its utility information without accessing the database. CHN [69] mines CHUI based on negative utility and the depth-first search, and uses the transaction consolidation and dataset projection techniques to reduce dataset scanning costs. In addition, two-phase extension techniques are used to examine closures and trim search space. CLS-Miner [68] utilizes a utility list structure and integrates three proposed pruning strategies and a fast pre-check method. This method is useful for mining CHUI issues.

The complexity of the CHUD [30], CHUI-Miner [57] and CLS-Miner [68] algorithms are briefly discussed here. CHUD [30] uses depth-first search to find closed high utility candidate sets. In the worst case, CHUD [30] dose not trim any itemsets, and accessse all itemsets in the search space, with the time complexity of approximately $O(2^{|I|})$ ($I$: the length of the list used in the algorithm). Similarly, the worst time complexity of CHUI-Miner [26] is also about $O(2^{|I|})$.

Similarly, in the worst case, CLS-Miner's [68] pruning strategy does not trim any itemsets, so the time complexity is also $O(2^{|I|})$. Although the worst-case time complexity of the three algorithms is roughly the same, CLS-Miner [68] has the advantage of processing fewer itemsets in the search space, and it combines a novel fast pre-check method to avoid the disadvantages of two-phase algorithms.

Existing CHUI mining algorithms assume that the database is static, making them very expensive in the case of incremental data because the entire data set must be processed for each new batch of transactions. IncCHUI [37] effectively mines CHUI from an incremental database. The algorithm proposes an incremental utility list structure that can be built and updated using only one database scan. In addition, an efficient hash-based approach is used to update or insert a new closed set.

For data streams, the CHUI may still be too large, so GUIDE [19] finds maximum HUIs from data streams with different models (i.e., landmarks, sliding windows, and time fading models). The algorithm is based on the maximum utility itemset (MUI-Tree) to maintain the basic information of the mining process. But restoring all HUIs from the maximum HUIs collection is very inefficient, as many subsets of maximum HUIs may be inefficient. In GUIDE [19], the effectiveness of the algorithm is analyzed from two aspects, namely the accuracy and pattern reduction ratio (abbreviated as *PRR*). Assume that HUI is a high utility itemset, *MaxHUI* is the maximum HUI, and *Actual maximal HUI* is the actual maximum HUI.

## V. METHODS FOR HIGH UTILITY PATTERNS OVER DATASTREAMS

Most of the algorithms proposed in the previous paper are applied to static data. With the rapid development of technologies such as Internet of Things, cloud computing and big data, data streams are widely used in many fields such as network, trade management, and medical data analysis. Compared to static data, data stream has some unique attributes, such as fast arrival rates, unknowns, unrestricted, and the inability to backtrack previous transactions. Therefore, for different purposes of use, there are four commonly used models in data streams: incremental methods, sliding window models, time decay models and landmark models [3]–[5].

### A. INCREMENTAL METHODS

In recent years, more and more data have been produced in various application fields, and the characteristics and quantity of data have been constantly changing with the passage of time. Because static methods must start their own mining operations from scratching every time they enter new data, they suffer from fatal computational overhead. Therefore, in order to better handle incremental data, the researchers proposed utility mining methods based on incremental data, which only process new input data without additional database scans and reflect them into previous processing without any errors.

FUP-HU [36] is an incremental and HUPM algorithm based on Apriori, which is the result of applying the concept of FUP (incremental frequent pattern mining method) to two-phase. When a new transaction is inserted into the original database, this algorithm divides the itemset into four parts based on whether they are high transaction weighted utility sets in the original database or in the new transaction. Each part is then processed to maintain the set of efficient items found in its own way. PRE-HUI [8] is a variant of FUP-HU that uses two types of thresholds, the upper and lower thresholds, and uses a new concept to predict changes in mode state on incremental data, called pre-large concept. The PRE-HUI algorithm guarantees better performance than the FUP-HU, but it is not an accurate method because the mode loss may be due to the way the lower threshold is set. HUIPRED [7] effectively performs pattern extraction using a pre-large method. In other words, the proposed method reduces the rescan of the entire database by using two thresholds. Meanwhile, the method efficiently processes the database by utilizing the proposed data structure. Therefore, it is more effective than the latest techniques.

Because Apriori-based algorithms generate a large number of candidates, with long running time and large memory consumption. Researchers have developed tree-based, list-based incremental HUPM algorithms. IHUP [2] builds its own tree structure in a single database scan and extracts candidates for high utility patterns. The algorithm then identifies the actual high utility patterns of the candidate through an additional database scan. Whenever new incremental data is entered, the method processes them, updates the previously constructed tree structure, and performs its own mining operations. IMHAUI [43] proposes a new tree structure of the incremental high-average utility itemset tree (IHAUI-Tree) to maintain the information of the incremental database so that the HAUI can be mined without multiple database scans. This algorithm uses a path adjustment method as one of the reconstruction techniques to maintain the compactness of IHAUI-Tree. LIHUP [41] constructs a global data structure through a single scan, reconstructs the data structure according to the optimal sort order, and updates the utility information in the reorganization step to effectively mine the HUIs from the incremental databases. IIHUM [42] mines the HUIs from incremental databases based on index lists without any candidate generation, and designs reorganization and pruning techniques to process incremental data more efficiently.

### B. METHODS BASED ON LANDMARK MODELS

In some applications, users may wish to process all data equally from past points in time (landmarks) and discover long-term patterns from the data streams. Landmark models are used for this purpose.

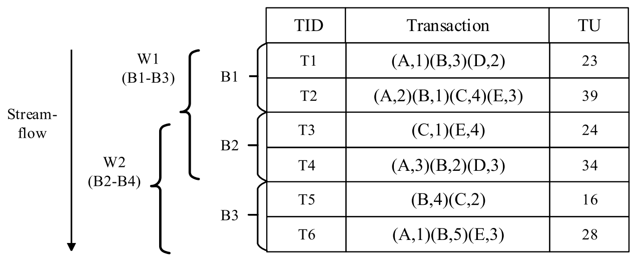
MAHUSP [26] is based on the landmark window mining HUIs. It is a new method based on the data streams incremental mining shell, which not only recognizes the nearest shell, but also recognizes the shell for a long time. The algorithm uses a novel and compact data structure, MAS-Tree, for storing potential shells on the data streams. At the same time, two effective memory self-use mechanisms are used to solve the problem that the existing memory is not enough to add a new potential shell to MAS-Tree. Therefore, the algorithm MAHUSP can efficiently find the shell on the data streams, with high recall and accuracy.

## C. METHODS BASED ON SLIDING WINDOW MODELS

In the sliding window model, the data stream is divided into batches. These batches consist of many transactions. The pattern uses a window containing a limited number of batches to maintain the latest batch in the data structure. The window size is specified by the user. When entering new data, the most previous data in the old batch will be excluded from the window, and the new data will be entered into the window as the latest batch. Next, solve the problem of mining HUPM in the sliding window on the data stream. Let data stream $DS = (T_1, T_2, \ldots, T_n)$ be a set of transactions, $I = \{i_1, i_2, \ldots, i_m\}$ is a group of items, and pattern $P$ is a group of items in $I$. The length of $P$ is expressed as $k$, $1 \leq k \leq m$, and the set of $k$-items indicates that the itemset is a pattern of length $k$. Each transaction $T_i$ (where $1 \leq i \leq n$) in the data stream $DS$ represents the $i$th arriving transaction. In the sliding window model, each window $W_k$ consists of a fixed number of equal size and non-overlapping batches.

Figure. 3 is an example of a data stream divided into three batches $\{B_1, B_2, B_3\}$, where each sliding window consists of two batches. In this example, there are two sliding windows, $W_1 = \{B_1, B_2\}$ and $W_2 = \{B_2, B_3\}$. Here, $W_1$ is the initial sliding window. In addition, $W_2$ is the result of sliding $W_1$ when the first window is full and the new batch is reached by removing the oldest batch and inserting a new batch. That is, in the sliding window model, the algorithm uses only a fixed number of recent batches in the current window.

Therefore, the mining algorithms based on this model can always keep the latest information in the window. THUIMine [6] is the first algorithm in the field to exploit HUPM on data streams in resource-constrained environments, but it has many drawbacks in terms of runtime and memory usage because it is a similar to Apriori's algorithm. The window in the GUIDE$_{SW}$ [19] is a time sensitive window



| TID | Transaction | TU |
|---|---|---|
| T1 | (A,1)(B,3)(D,2) | 23 |
| T2 | (A,2)(B,1)(C,4)(E,3) | 39 |
| T3 | (C,1)(E,4) | 24 |
| T4 | (A,3)(B,2)(D,3) | 34 |
| T5 | (B,4)(C,2) | 16 |
| T6 | (A,1)(B,5)(E,3) | 28 |

**FIGURE 3. Example of stream data.**

that is used to fix the size of the transaction. The algorithm mining this model not only has high utility values, but also maximizes the conceptual data flow, further improving accuracy and runtime. SHUGrow [35] uses the tree structure SHU-Tree, which has a node utility counter in the global tree. Each utility value of the counter is associated with each batch in the current window, i.e., if there are $n$ batches in the current window, the number of node utilities is $n$ in the counter.

Based on the Top-k patterns, Vert-top-k-DS [21] proposes a new data structure, iList, a batch of FIFOs that can quickly insert and delete batches from the window. T-HUDS [22] uses a compact data structure similar to FP-Tree, HUDS-Tree, which is used to dynamically maintain a compressed version of a transaction in a sliding window. Based on this structure, the algorithm looks up the Top-k HUIs in the data streams without specifying *minutil*.

Based on the sequential patterns, HUSP-Stream [23] uses the vertical representation item utility list (ItemUtilLists) and the outer utility tree to model the basic information of the shell in the current window, incrementally representing complete HUSP in the data streams. HUSP-UT [24] improves the above algorithm and uses a new data structure UT-Tree. This algorithm is superior to the most advanced shell flow algorithm. HUPMS [25] is based on an efficient flow tree. By capturing the important information of the data stream into a shell tree, using the pattern growth method to mine all the HUIs in the current window is very effective for incremental and interactive mining on the data stream.

## D. METHODS BASED ON TIME DECAY MODELS

The time decay model was designed to find the latest relevant information from the data streams by distinguishing the importance of recent transactions from old transactions. The model uses a user-specified attenuation factor f to reduce the importance of transactions over time. With this model, if the frequency of the old transaction item set is high, it can be considered in the recent mining process. Therefore, this model is more reliable and efficient than the sliding window model, because the information of the old transaction is not completely excluded from the mining process. This method can be applied to HUPM, and the utility of the transaction is reduced according to the arrival time of the transaction to mine the latest HUIs.

The frequent itemset mining method based on this model reduces the occurrence count of the itemset in the old transaction, that is, the frequency, by multiplying them by the attenuation factor. Let $S = \{T_1, T_2, \ldots, T_n\}$ be a data stream composed of multiple transactions, and $X$ is a set of items generated from $S$. When the frequency $T_k$ of $X$ in each transaction is expressed as $freq(T_k, X)$, based on the time decay model, the attenuation frequency of $X$ in $S$ is expressed as $dfreq(X)$ and is calculated as follows:

$$dfrep(X) = \sum_{k=1}^{n} frep(T_k, X) \times f^{n-k} \tag{5}$$

GENHUI [27], based on the time decay model, reduces the utility of the transaction based on the arrival time of the transaction in order to assign more weight to the most recent data than the old data. At the same time, its tree data structure is updated regularly to reflect the most recent utility information into the tree structure. Whenever an update process is performed, the algorithm prunes outdated nodes from the tree so that only information that has a high impact on the mining results is retained. HAUPM [28] mines the most recent HUIs by considering the time factor of a given data to discover important, recent pattern information. In order to facilitate the efficient mining process, the algorithm designs and uses a new data structure, damped average tree (DAT) and transaction utility list (TUL) to mine HAUP.

## VI. NEXT DIRECTION

Although the solution to the high utility patterns mining problems has been developed, the existing methods still have shortcomings, which provides researchers with the next research direction:

1) The high utility patterns store a large amount of valid information, but there are still many redundant patterns. The research and analysis of the compact high utility patterns for reducing the redundancy patterns, the author of the project team will next study the one-phase algorithm of the Top-k closed high utility patterns. The algorithm uses the improved uList structure, the real and remaining utility of the pattern are used to pruning the traversal space, and the content of the Top-k buffer area stored in the result sets are updated in real time, and the closed high utility patterns are generated.

2) High utility patterns consume a lot of memory and time due to the large number of candidate sets generated. In order to reduce these two aspects of analysis, the author's project team will next study the incremental algorithm based on the improved utility list. The algorithm applies the utility list buffer structure in the ULB-Miner [46] algorithm to greatly reduce the memory occupied by the generated candidate sets. This algorithm mines the itemsets from an incremental perspective, which is more suitable for practical applications.

3) Scalability is an important research issue in HUPM. Scalability is the processing of more data in a certain dimension. The following measures can be taken to address the scalability issues in HUPM: (1) collocation: by collocated data and code, reducing the necessary overhead for obtaining the required data; (2) caching: if the data and code cannot be concatenated, the data is cached to reduce the overhead associated with its use; (3) by dividing the processing code, concatenating the relevant partitions, and associating the relevant processing processes as much as possible, the processing time of a single work unit can be reduced.

4) HUPM is mainly applied to a single data stream at this stage, without multiple data streams. Multiple data streams can be processed simultaneously in a parallel manner. First, the program needs to be parallelized, that is, the work parts are allocated to different processing processes (threads). In theory, the execution speed of $n$ parallel processing may be $n$ times faster than that performed on a single processor.

5) With the widespread use of big data, cloud computing and other technologies in real life, in HUPM, static data mining can no longer meet people's needs. In HUPM in a big data environment, you can use the Spark tool to store data and implement a single-to-distribute transformation; you can use the Storm tool to process large data streams in real time; you can also use the Scrapy framework in Python to crawl website data, extracting structured data, and this is a very good web framework for grabbing data from the Internet. These tools and frameworks can be used to solve problems such as computation, storage resources, fault tolerance, and optimization of load balancing in HUPM.

## VII. SUMMARY

High utility patterns mining (HUPM) is a vital task in utility mining. So far, many techniques and methods have been proposed for HUIM's tasks. This paper introduces the basic concepts, examples and related concepts of HUPM, and expounds the key technologies of HUPM, from Apriori-based, tree-based, projection-based, data format-based, and list-based methods. The workflow, use, datasets, advantages and disadvantages of the algorithms are analyzed. It outlines the advantages and disadvantages of efficient use of derivative models compared to traditional high utility models. With the advent of the era of big data, static data can no longer meet the actual needs. Therefore, it is imperative to study the methods of HUPM on data streams. In this paper, incremental methods, based on the sliding window model methods, based on the time decay mode methods, based on the landmark models and other methods, the HUIs are extracted from the data streams. Finally, the next research direction is proposed. Most of the algorithms mentioned in this paper are still applied to static data, so the application of data flow is the focus of future work.

## REFERENCES

[1] Y. Liu, W. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Advances in Knowledge Discovery and Data Mining*, vol. 3518. 2005, pp. 689–695.

[2] 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.

[3] J. Cheng, Y. Ke, and W. Ng, "A survey on algorithms for mining frequent itemsets over data streams," *Knowl. Inf. Syst.*, vol. 16, no. 1, pp. 1–27, Jul. 2008.

[4] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," *ACM SIGMOD Rec.*, vol. 35, no. 1, pp. 14–19, Mar. 2006.

[5] C.-H. Lin, D.-Y. Chiu, Y.-H. Wu, and A. L. P. Chen, "Mining frequent itemsets from data streams with a time-sensitive sliding window," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2005, pp. 68–79.

[6] C.-J. Chu, V. S. Tseng, and T. Liang, "An efficient algorithm for mining temporal high utility itemsets from data streams," *J. Syst. Softw.*, vol. 81, no. 7, pp. 1105–1117, Jul. 2008.

[7] U. Yun, H. Nam, J. Kim, H. Kim, Y. Baek, J. Lee, E. Yoon, T. Truong, B. Vo, and W. Pedrycz, "Efficient transaction deleting approach of pre-large based high utility pattern mining in dynamic databases," *Future Gener. Comput. Syst.*, vol. 103, pp. 58–78, Feb. 2020.

[8] J. C.-W. Lin, T.-P. Hong, W. Gan, H.-Y. Chen, and S.-T. Li, "Incrementally updating the discovered sequential patterns based on pre-large concept," *Intell. Data Anal.*, vol. 19, no. 5, pp. 1071–1089, Sep. 2015.

[9] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7419–7424, Jun. 2011.

[10] 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.

[11] V. S. Tseng and C. W. Wu, "UP-growth: An efficient algorithm for high utility itemset mining," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Washington, DC, USA, 2010, pp. 253–262.

[12] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, HI, USA, 2012, pp. 55–64.

[13] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, "FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning," in *Proc. Int. Symp. Methodol. Intell. Syst.*, 2014, pp. 83–92.

[14] J. C.-W. Lin, W. Gan, T.-P. Hong, and V. S. Tseng, "Efficient algorithms for mining up-to-date high-utility patterns," *Adv. Eng. Informat.*, vol. 29, no. 3, pp. 648–661, Aug. 2015.

[15] J. C.-W. Lin, W. Gan, T.-P. Hong, and B. Zhang, "An incremental high-utility mining algorithm with transaction insertion," *Sci. World J.*, vol. 2015, pp. 1–15, Feb. 2015.

[16] T.-P. Hong, C.-H. Lee, and S.-L. Wang, "An incremental mining algorithm for high average-utility itemsets," in *Proc. 10th Int. Symp. Pervas. Syst., Algorithms, Netw.*, 2010, pp. 421–425.

[17] C.-W. Lin, T.-P. Hong, G.-C. Lan, J.-W. Wong, and W.-Y. Lin, "Incrementally mining high utility patterns based on pre-large concept," *Int. J. Speech Technol.*, vol. 40, no. 2, pp. 343–357, Mar. 2014.

[18] J. Lee, U. Yun, G. Lee, and E. Yoon, "Efficient incremental high utility pattern mining based on pre-large concept," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 111–123, Jun. 2018.

[19] B.-E. Shie, P. S. Yu, and V. S. Tseng, "Efficient algorithms for mining maximal high utility itemsets from data streams with different models," *Expert Syst. Appl.*, vol. 39, no. 17, pp. 12947–12960, Dec. 2012.

[20] W. Gan, J. C.-W. Lin, P. Fournier-Viger, and H.-C. Chao, "More efficient algorithms for mining high-utility itemsets with multiple minimum utility thresholds," in *Database and Expert Systems Applications* (Lecture Notes in Computer Science), vol. 9827. 2016, pp. 71–87.

[21] S. Dawar, V. Sharma, and V. Goyal, "Mining top-k high-utility itemsets from a data stream under sliding window model," *Int. J. Speech Technol.*, vol. 47, no. 4, pp. 1240–1255, Dec. 2017.

[22] M. Zihayat and A. An, "Mining top-k high utility patterns over data streams," *Inf. Sci.*, vol. 285, pp. 138–161, Nov. 2014.

[23] M. Zihayat, C.-W. Wu, A. An, and V. S. Tseng, "Mining high utility sequential patterns from evolving data streams," in *Proc. ASE BigData Socialinform.*, 2015, pp. 1–6.

[24] H. Tang, Y. Liu, and L. Wang, "A new algorithm of mining high utility sequential pattern in streaming data," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 1, pp. 342–350, 2019.

[25] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and H.-J. Choi, "Interactive mining of high utility patterns over data streams," *Expert Syst. Appl.*, vol. 39, no. 15, pp. 11979–11991, Nov. 2012.

[26] M. Zihayat, Y. Chen, and A. An, "Memory-adaptive high utility sequential pattern mining over data streams," *Mach. Learn.*, vol. 106, no. 6, pp. 799–836, Jun. 2017.

[27] D. Kim and U. Yun, "Mining high utility itemsets based on the time decaying model," *Intell. Data Anal.*, vol. 20, no. 5, pp. 1157–1180, Sep. 2016.

[28] U. Yun, D. Kim, E. Yoon, and H. Fujita, "Damped window based high average utility pattern mining over data streams," *Knowl.-Based Syst.*, vol. 144, pp. 188–205, Mar. 2018.

[29] H. Ryang and U. Yun, "Top-k high utility pattern mining with effective threshold raising strategies," *Knowl.-Based Syst.*, vol. 76, pp. 109–126, Mar. 2015.

[30] C. W. Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng, "Efficient mining of a concise and lossless representation of high utility itemsets," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 824–833.

[31] A. Erwin, R. P. Gopalan, and N. R. Achuthan, "Efficient mining of high utility itemsets from large datasets," in *Proc. 12th Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, 2008, pp. 554–561.

[32] J. Sahoo, A. K. Das, and A. Goswami, "An efficient fast algorithm for discovering closed+ high utility itemsets," *Int. J. Speech Technol.*, vol. 45, no. 1, pp. 44–74, Jul. 2016.

[33] L. Feng, L. Wang, and B. Jin, "UT-tree: Efficient mining of high utility itemsets from data streams," *Intell. Data Anal.*, vol. 17, no. 4, pp. 585–602, Jun. 2013.

[34] W. S. Gan, J. C.-W. Lin, and P. Fournier-Viger, "A survey of utility-oriented pattern mining," *J. OF Latex Class Files*, vol. 6, no. 1, 2018.

[35] H. Ryang and U. Yun, "High utility pattern mining over data streams with sliding window technique," *Expert Syst. Appl.*, vol. 57, pp. 214–231, Sep. 2016.

[36] C.-W. Lin, G.-C. Lan, and T.-P. Hong, "An incremental mining algorithm for high utility itemsets," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7173–7180, Jun. 2012.

[37] T.-L. Dam, H. Ramampiaro, K. Nørvåg, and Q.-H. Duong, "Towards efficiently mining closed high utility itemsets from incremental databases," *Knowl.-Based Syst.*, vol. 165, pp. 13–29, Feb. 2019.

[38] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993.

[39] P. Fournier-Viger, Y. Zhang, J. Chun-Wei Lin, H. Fujita, and Y. S. Koh, "Mining local and peak high utility itemsets," *Inf. Sci.*, vol. 481, pp. 344–367, May 2019.

[40] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Trans. Knowl. Data Eng.*, vol. 12, no. 3, pp. 372–390, May/Jun. 2000.

[41] U. Yun, H. Ryang, G. Lee, and H. Fujita, "An efficient algorithm for mining high utility patterns from incremental databases with one database scan," *Knowl.-Based Syst.*, vol. 124, pp. 188–206, May 2017.

[42] U. Yun, H. Nam, G. Lee, and E. Yoon, "Efficient approach for incremental high utility pattern mining with indexed list structure," *Future Gener. Comput. Syst.*, vol. 95, pp. 221–239, Jun. 2019.

[43] D. Kim and U. Yun, "Efficient algorithm for mining high average-utility itemsets in incremental transaction databases," *Int. J. Speech Technol.*, vol. 47, no. 1, pp. 114–131, Jul. 2017.

[44] D. Kim and U. Yun, "Efficient mining of high utility pattern with considering of rarity and length," *Int. J. Speech Technol.*, vol. 45, no. 1, pp. 152–173, Jul. 2016.

[45] B. Le, U. Huynh, and D.-T. Dinh, "A pure array structure and parallel strategy for high-utility sequential pattern mining," *Expert Syst. Appl.*, vol. 104, pp. 107–120, Aug. 2018.

[46] S. Krishnamoorthy, "Pruning strategies for mining high utility itemsets," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2371–2381, Apr. 2015.

[47] W. Song, Z. Zhang, and J. Li, "A high utility itemset mining algorithm based on subsume index," *Knowl. Inf. Syst.*, vol. 49, no. 1, pp. 315–340, Oct. 2016.

[48] W. Song, B. Yang, and Z. Xu, "Index-BitTableFI: An improved algorithm for mining frequent itemsets," *Knowl.-Based Syst.*, vol. 21, no. 6, pp. 507–513, Aug. 2008.

[49] H. Ryang and U. Yun, "Indexed list-based high utility pattern mining with utility upper-bound reduction and pattern combination techniques," *Knowl. Inf. Syst.*, vol. 51, no. 2, pp. 627–659, 2016.

[50] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "EFIM: A highly efficient algorithm for high-utility itemset mining," in *Proc. Mex. Int. Conf. Artif. Intell.*, 2015, pp. 530–546.

[51] J. C.-W. Lin, S. Ren, and P. Fournier-Viger, "MEMU: More efficient algorithm to mine high average-utility patterns with multiple minimum average-utility thresholds," *IEEE Access*, vol. 6, pp. 7593–7609, 2018.

[52] W. S. 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 *Advances in Knowledge Discovery and Data Mining* (Lecture Notes in Computer Science). 2017, pp. 434–446.

[53] C.-W. Lin, T.-P. Hong, G.-C. Lan, J.-W. Wong, and W.-Y. Lin, "Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases," *Adv. Eng. Informat.*, vol. 29, no. 1, pp. 16–27, Jan. 2015.

[54] J. C.-W. Lin, W. Gan, and T.-P. Hong, "A fast maintenance algorithm of the discovered high-utility itemsets with transaction deletion," *Intell. Data Anal.*, vol. 20, no. 4, pp. 891–913, Jun. 2016.

[55] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "HUOPM: High-utility occupancy pattern mining," *IEEE Trans. Cybern.*, vol. 50, no. 3, pp. 1195–1208, Mar. 2020.

[56] B. P. Jaysawal and J.-W. Huang, "DMHUPS: Discovering multiple high utility patterns simultaneously," *Knowl. Inf. Syst.*, vol. 59, no. 2, pp. 337–359, May 2019.

[57] C.-W. Wu, P. Fournier-Viger, J.-Y. Gu, and V. S. Tseng, "Mining closed+ high utility itemsets without candidate generation," in *Proc. Conf. Technol. Appl. Artif. Intell. (TAAI)*, Nov. 2015, pp. 187–194.

[58] C. F. Ahmed, "A novel approach for mining high-utility sequential patterns in sequence databases," *ETRI J.*, vol. 32, no. 5, pp. 676–686, Oct. 2010.

[59] J. Yin, Z. Zheng, and L. Cao, "USpan: An efficient algorithm for mining high utility sequential patterns," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 660–668.

[60] G.-C. Lan, T.-P. Hong, V. S. Tseng, and S.-L. Wang, "Applying the maximum utility measure in high utility sequential pattern mining," *Expert Syst. Appl.*, vol. 41, no. 11, pp. 5071–5081, Sep. 2014.

[61] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan,: Mining sequential patterns efficiently by prefix-projected pattern growth," in *Proc. ICDE*, Berlin, Germany, 2001, pp. 215–224.
[62] O. K. Alkan and P. Karagoz, "CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2645–2657, Oct. 2015.
[63] J. C.-W. Lin, J. Zhang, P. Fournier-Viger, T.-P. Hong, and J. Zhang, "A two-phase approach to mine short-period high-utility itemsets in transactional databases," *Adv. Eng. Informat.*, vol. 33, pp. 29–43, Aug. 2017.
[64] Q.-H. Duong, P. Fournier-Viger, H. Ramampiaro, K. Nørvåg, and T.-L. Dam, "Efficient high utility itemset mining using buffered utility-lists," *Int. J. Speech Technol.*, vol. 48, no. 7, pp. 1859–1877, Jul. 2018.
[65] S. Krishnamoorthy, "Efficient mining of high utility itemsets with multiple minimum utility thresholds," *Eng. Appl. Artif. Intell.*, vol. 69, pp. 112–126, Mar. 2018.
[66] S. Buffett, "Candidate list maintenance in high utility sequential pattern mining," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 644–652.
[67] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, "Efficient algorithms for mining high-utility itemsets in uncertain databases," *Knowl.-Based Syst.*, vol. 96, pp. 171–187, Mar. 2016.
[68] T.-L. Dam, K. Li, P. Fournier-Viger, and Q.-H. Duong, "CLS-miner: Efficient and effective closed high-utility itemset mining," *Frontiers Comput. Sci.*, vol. 13, no. 2, pp. 357–381, Apr. 2019.
[69] 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.*, early access, Nov. 20, 2018, doi: 10.1109/TKDE.2018.2882421.
[70] A. Bai, P. S. Deshpande, and M. Dhabu, "Selective database projections based approach for mining high-utility itemsets," *IEEE Access*, vol. 6, pp. 14389–14409, 2018.
[71] Q.-H. Duong, B. Liao, P. Fournier-Viger, and T.-L. Dam, "An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies," *Knowl.-Based Syst.*, vol. 104, pp. 106–122, Jul. 2016.
[72] 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," *Adv. Eng. Informat.*, vol. 30, no. 2, pp. 233–243, Apr. 2016.
[73] L. Zhang, S. Yang, X. Wu, F. Cheng, Y. Xie, and Z. Lin, "An indexed set representation based multi-objective evolutionary approach for mining diversified top-k high utility patterns," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 9–20, Jan. 2019.
[74] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, "Efficiently mining uncertain high-utility itemsets," *Soft Comput.*, vol. 21, no. 11, pp. 2801–2820, Jun. 2017.
[75] 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.
[76] P. Fournier-Viger and S. Zida, "FOSHU: Faster on-shelf high utility itemset mining—With or without negative unit profit," in *Proc. 30th Annu. ACM Symp. Appl. Comput. (SAC)*, 2015, pp. 857–864.
[77] J. C.-W. Lin, W. Gan, P. Fournier-Viger, and T.-P. Hong, "Mining high-utility itemsets with multiple minimum utility thresholds," in *Proc. 8th Int. Conf. Comput. Sci. Softw. Eng. (C3S2E)*, 2008, pp. 9–17.
[78] K. Singh, H. K. Shakya, and A. Singh, "Mining of high-utility itemsets with negative utility," *Expert Syst.*, vol. 35, no. 8, 2018, Art. no. e12296.
[79] T.-P. Hong, J.-H. Hsu, G.-C. Lan, K.-J. Yang, S.-L. Wang, and J. C.-W. Lin, "High utility partial periodic pattern mining," in *Proc. 4th Multidisciplinary Int. Social Netw. Conf. (MISNC)*, 2017, pp. 1–4.
[80] 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, 2017.
[81] 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.
[82] J. C.-W. Lin, S. Ren, P. Fournier-Viger, J.-S. Pan, and T.-P. Hong, "Efficiently updating the discovered high average-utility itemsets with transaction insertion," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 136–149, Jun. 2018.
[83] 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.
[84] L. T. T. Nguyen, P. Nguyen, T. D. D. Nguyen, B. Vo, P. Fournier-Viger, and V. S. Tseng, "Mining high-utility itemsets in dynamic profit databases," *Knowl.-Based Syst.*, vol. 175, pp. 130–144, Jul. 2019.
[85] 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.*, vol. 484, pp. 44–70, May 2019.
[86] P. Fournier-Viger, J. C.-W. Lin, T. Truong-Chi, and R. Nkambou, "A survey of high utility itemset mining," in *High-Utility Pattern Mining: Theory, Algorithms and Applications*. Springer, 2019, pp. 1–46.
[87] S. Patel and B. Madhushree, "A survey on discovering high utility itemset mining from transactional database," *Int. J. Adv. Eng. Res. Develop.*, vol. 2, no. 11, pp. 346–350, 2015.

**CHUNYAN ZHANG** was born in 1995. She is currently pursuing the M.S. degree with North Minzu University. Her research interest is data mining.

**MENG HAN** was born in 1982. She received the Ph.D. degree from Beijing Jiaotong University. She is currently an Associate Professor and a Master's Supervisor with North Minzu University. Her research interest is data mining.

**RUI SUN** was born in 1993. She is currently pursuing the M.S. degree with North Minzu University. Her research interest is data mining.

**SHIYU DU** was born in 1996. She is currently pursuing the M.S. degree with North Minzu University. Her research interest is data mining.

**MINGYAO SHEN** was born in 1994. He is currently pursuing the M.S. degree with North Minzu University. His research interest is data mining.