Applied Intelligence (2023) 53:27655–27703
https://doi.org/10.1007/s10489-023-04853-5

High utility itemsets mining from transactional databases: a survey

Rajiv Kumar[1] · Kuldeep Singh[2]

Accepted: 29 June 2023 / Published online: 16 September 2023

Abstract
Mining high utility itemsets are the basic task in the area of frequent itemset mining (FIM) that has various applications in diverse domains, including market basket analysis, web mining, cross-marketing, and e-commerce. In recent years, many efficient high utility itemsets mining (HUIM) algorithms are proposed to discover the high utility itemsets (HUIs). This survey presents a comprehensive summary of the current state-of-the-art HUIM approaches for transactional databases. This paper categorises the state-of-the-art approaches as level-wise, tree-based, utility-list-based, projection-based and miscellaneous. It provides the pros and cons of each category of mining approaches in detail. A taxonomy of the HUIM for transactional databases is presented. The survey also summarises and discusses approaches for other types of databases, including on-shelf, dynamic and uncertain. The paper explores the applications of HUIM in diverse domains and discusses the challenges and limitations of the approach. It presents an overview of 16 real-world which are utilized by various state-of-the-art HUIM approaches for transactional databases. Overall, this survey provides a valuable resource for researchers in the field of HUIM and offers insights into future directions for research and development in this area.

Keywords Data mining · High utility itemsets mining · Transactional datasets · Utility mining · Pattern mining · Sequential patterns

1 Introduction

In past decades, many approaches have been extensively studied to mine the interesting patterns such as FIM (Frequent Itemsets Mining) [1], and SPM (Sequential Pattern Mining) [2]. FIM extracts the patterns that have frequency or support values no less than the user-specified minimum threshold in a database. Traditional FIM algorithms consider each item equally in the database by assuming whether the item is appeared in the transaction or not. They assume the quantity is either 0 or 1 and utility of each item is always 1. But, in most of the applications, the frequent itemsets consist only a less fraction of the total profit, while non-frequent itemsets generate a larger portion of the profit. Hence, the retail businesses do not get significant benefits from these methods.

To overcome the limitations of FIM algorithms, various HUIM (High Utility Itemsets Mining) algorithms [3, 4] have been proposed that measure the usefulness (or interestingness) of the items. The objective of the HUIM is to mine the HUIs (High Utility Itemsets) which generates high profit. The utility is defined as the measurement of the usefulness of an itemset. Utility values of items in a transaction database consist of two parts: the item profit (external utility) and the quantity of the item in one transaction (internal utility). The utility of an itemset is defined as the external utility multiplied by the internal utility. HUI mining facilitates crucial business decisions to maximize revenue, minimize marketing expenditure, and reduce inventory.

The HUIM algorithms have various applications, for example, market basket analysis [5], web click-stream analysis [6], web mining [7], cross-marketing [8], gene regulation [9], mobile commerce environment [10], e-commerce [11], etc. However, it is a challenging and complex task to mine the HUIs because it does not hold the DCP (Downward Closure Property)[1]. It means that a super-set of low utility itemsets

✉ Kuldeep Singh
ksingh@cs.du.ac.in

Rajiv Kumar
rajiv.kumar@bennett.edu.in

1 School of Computer Science Engineering and Technology, Bennett University, Greater Noida, Uttar Pradesh, India

2 Department of Computer Science, University of Delhi, Delhi, India

[1] A k-itemset is HUIs only if all of its sub-itemsets are HUIs.

may be high utility itemsets. Hence, it is difficult to prune the search space of the mining process. A brute-force method can solve this problem that enumerates all itemsets from a database (i.e. use the principle of exhaustion). Obviously, this suffers from a combinatorial explosion, especially for databases containing many long transactions or for low minimum utility thresholds. Therefore, effectively pruning the search space and efficiently capturing all HUIs with no misses are crucial aspects of HUI mining.

To resolve this problem, Liu et al. [12] proposed Two-phase algorithm that presented TWU (Transaction Weighted Utilization) based anti-monotonic property[2]. Various algorithms are presented on the basis of TWU model [13]. However, these algorithms generate numerous candidates and need multiple database scans. To address these issues, tree-based HUIM algorithms [14, 15] have been proposed that avoid excessive candidates generation and test strategy to mine the HUIs. However, they still produce too many candidates and require more than one database scan. To overcome these issues, utility-list-based algorithms [16–18] have been proposed that avoid the excessive candidates generation and multiple database scans. However, they incur the problem of costly join operations and inefficient memory usage. To address these issues, projection-based algorithms [13, 19] have been presented to enhance the effectiveness of the mining process. To represent the concise and compact representation of HUIs, closed HUIM algorithms [20] are proposed to overcome the challenges of the conventional HUIM approaches. Closed HUIM approaches mine only those high utility itemsets that have no proper superset with the same support count. However, sometime they consume high memory.

This article presents a survey of HUIM algorithms from transactional databases. This survey emphasizes the classification of the existing literature, developing a perspective on the area, and evaluating trends. As shown in Fig. 1, there has been a rapid growth in the interest of HUIM in recent years in terms of the number of research papers published. Figure 1 shows the research articles published in recent years in the area of HUIM, including HUIM from on-shelf, sequential databases, uncertain databases, transactional databases, dynamic databases and many others. However, this survey presents a detailed analysis of HUIM approaches from transactional databases.

Many surveys [21–28] exist on HUIs mining. In literature, there are only three articles [22, 23, 28] that present a comprehensive review of HUIs mining algorithms. The major contribution of the existing state-of-the-art surveys are discussed as:

[2] If a set fails to pass a test, all the supersets will also fail the same test as well.

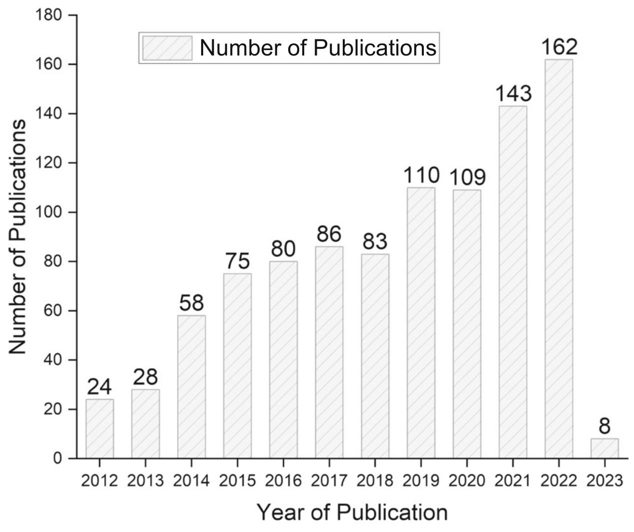

Fig. 1 Year-wise number of papers published for "High Utility Itemsets Mining". The year-wise number of publication data are obtained from Google Scholar. Note that the search phrase is defined as the subfield named with the exact phrase "utility pattern," and at least one of itemset\utility\pattern\sequential and utility pattern

- In 2019, Fournier-Viger et al. [21] presented a book for HUIM and the chapters of the book covers different ways to find HUIs as there are various type of data such as transactional databases, incremental, dynamic, on-shelf and sequential databases. One chapter mainly focuses on transactional database-based approaches and includes very few basic papers on transactional database-based HUIs mining. It extensively explores three approaches: Two-phase [12], pattern-growth [14] and FHM [17] approach. Our work does not adhere only three approaches, it covers and compare more than 60 approaches.
- The scope of the article [26] is very limited and only covers privacy-preserving-based utility mining. The study [27] covers the incremental (dynamic) database-based high utility itemsets mining. Similarly, [24] explains very limited about the HUI mining and its main concern is about the different types of databases such as voluminous, dynamic, and continuous data with high speeds and uncertainty. Later, Singh et al. [25] presented a survey on High average-utility itemsets mining. All these [24–27, 29] are not related to HUIM for transactional databases. We are not comparing these works with ours because they serve as inspiration for other data mining tasks.
- In 2019, Rahmati et al. [23] provide a systematic review of basic HUIs mining techniques. It categories the methods into four parts, Apriori-like, pattern-growth-based, utility-list-based and additional types of methods. It reviewed only 20 approaches.
- In 2019, Gan et al. [28] present a general, very comprehensive overview of the state-of-the-art methods.

It is very concise survey article that categorizes the HUIs mining algorithms into four groups, Apriori-based, tree-based, projection-based and new data format-based algorithms. It surveys HUIM and high-utility sequential pattern mining approaches. It discusses only 26 approaches related to HUIM.

- In early 2020, Zhang et al. [22] present a brief survey which classifies the algorithms into five categories, Apriori-based, tree-based, projection-based, list-based and vertical & horizontal data-based algorithms. It is very concise survey and only discussed the advantages, and disadvantages of the current state-of-the-art methods.

All the existing surveys consider less number of approaches compared to the last survey. Our survey is different from the existing surveys [22, 23, 28] in various aspects.

- Our survey specifically focuses on HUIM for transactional databases, however, existing surveys are general, cover HUIM, and other aspects of HUIM.
- The existing surveys included less number of approaches and they do not include approaches published in 2020 and later years, however, our work covers more approaches and included all methods to date.
- Our work presents a detailed overview along with the pros and cons of all the current HUIM approaches whereas the existing surveys are not as detailed.
- The existing surveys did not include a description of the databases which are utilized by HUIM approaches. However, our survey included a detailed description of the utilized databases which helps readers to find the relevant databases for experimental results.
- Most of the existing surveys do not describe the applications, open-source resources, and detailed future research directions. Our work presented applications, open-source resources, and detailed future research directions.

This survey only includes the transaction-based algorithms and gives a detailed overview of the existing algorithms. Our paper is comprehensively focused on the level-wise, tree-based, utility-list-based, projection-based HUIM algorithms in the transactional databases while the other survey papers do not extensively highlight the advancements in this area. Moreover, all the above-discussed the state-of-the-art works consider lesser algorithms than our work. Furthermore, we have reviewed considerably the other domains of HUIM such as incremental [27], negative utility [30], average-utility [25], multiple minimum utility thresholds [31], data stream [32], on-shelf [33], periodic and privacy-preserving [26] utility values while other survey papers were not reviewed considerably. Our feature representation is also completely different than the state-of-the-art works.

This work provides a comprehensive survey of HUIM algorithms from transactional databases that can serve as a recent advancement and opportunity in the field of data mining. The main contributions of this work are:

1. This paper presents a taxonomy of more than 60 state-of-the-art HUIM approaches for the transactional database that includes level-wise, tree-based, utility-list-based, projection-based, and miscellaneous approaches.
2. It presents the detailed comparison tables of more than 60 approaches in various parameters i.e. phases, number of database scans, utilized data structure, type of patterns mined, searching type, utilized pruning strategies, type of utility value, state-of-the-art algorithms, and the base approach. This survey discusses the pros and cons of each category of HUIM approaches in detail.
3. It also provides an in-depth summary and discussion of other existing HUIM approaches including incremental, negative utility, average utility, soft computing using HUIM, multiple minimum utility thresholds, data stream, on-shelf, periodic, and privacy-preserving utility.
4. The survey briefly discusses 16 real-world databases. It also discusses some open-source resources of HUIM. Furthermore, the article presents research opportunities and the future directions of the HUIM.

The rest of this paper is organized as follows: In Section 2, we introduce the preliminaries' definitions, properties, and applications. In Section 3, we categorize and describe basic HUIM algorithms along with their pros and cons. In Section 4, we discuss the other advance topics of HUIM algorithms. In Section 5, we present some open-source resources and transactional databases. In Section 6, we discuss the research opportunities and future directions of HUIM algorithms. Finally, we conclude the paper in Section 7.

2 Preliminaries and applications

This section provides the preliminaries, important definitions and applications of the high utility itemsets mining.

2.1 Preliminaries and definitions

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a finite set of n distinct items. Let the transactions $t_1, t_2, \ldots, t_m$ occur in the transactional database D, where each transaction $t_q \subseteq I$. An itemset X is a finite set of k items such that $X \subseteq I$, where k is the length of itemset denoted as k-itemset. Each item i in the transaction t_q

Table 1 Transaction database

tid	Transaction
t_1	$(b, 1), (c, 1), (d, 3)$
t_2	$(a, 2), (c, 4), (b, 1)$
t_3	$(a, 3), (b, 1), (d, 5), (e, 2)$
t_4	$(b, 2), (c, 1), (e, 3)$
t_5	$(a, 2)$
t_6	$(b, 1), (d, 2)$
t_7	$(a, 3), (b, 2), (d, 4), (e, 1)$
t_8	$(a, 3), (d, 1), (e, 3)$

Table 3 Transaction utility of the running example

tid	Transaction	Utility	tu
t_1	b, c, d	2, 1, 12	15
t_2	a, c, e	6, 4, 1	11
t_3	a, b, d, e	9, 2, 20, 2	33
t_4	b, c, e	4, 1, 3	8
t_5	a	6	6
t_6	b, d	2, 8	10
t_7	a, b, d, e	9, 4, 16, 1	30
t_8	a, d, e	9, 4, 3	16

has an internal utility denoted as $iu(i, t_q)$. Each item i in the transaction t_q is associated with the external utility denoted as $eu(i)$. The internal utility is simply the quantity of an item purchased during the visit of the retail-shop or super-market. Similarly, the external utility is price or profit of an item as depicted in Table 2. A minimum utility threshold is set according to the user preference, denoted as δ.

For example, the transaction database consists eight transactions $t_1, t_2, \ldots, t_8$ as shown in Table 1 that will be used as a running example. It consists five items a, b, c, d, e. The external utility of each item is shown in Table 2. For example, the transaction t_3 consists four items a, b, d, e and has internal utility 3, 1, 5, 2 respectively. The external utility of a, b, c, d, e are 3, 2, 1, 4, 1 respectively.

Definition 1 ***Internal Utility of an item*** *$(iu(i, t_q))$*

Each item $i \in I$ in a transaction t_q is associated with an internal utility, denoted as $iu(i, t_q)$.

For instance, the internal utility of itemset $\{d\}$ in t_3 is 5 as depicted in Table 1.

Definition 2 ***External Utility of an item*** *$(eu(i))$*

Each item $i \in I$ is associated with an external utility, denoted as $eu(i)$.

For instance, the external utility of itemset $\{d\}$ is 4 as shown in Table 2.

Definition 3 ***Utility of an item*** *$(u(i, t_q))$*

The utility of an item $i \in t_q$ is denoted as $u(i, t_q)$ and is defined as:

$$u(i, t_q) = iu(i, t_q) \times eu(i) \tag{1}$$

For instance, the utility of itemset $\{d\}$ in t_3 is computed as: $(d, t_3) = iu(d) \times eu(d) = 5 \times 4 = 20$. The utility value of all the items for the running example is depicted in 2nd column of Table 3.

Table 2 External utility value

Items	a	b	c	d	e
External Utility	3	2	1	4	1

Definition 4 ***Utility of an itemset in the transaction*** *$(u(X,t_q))$*

The utility of an itemset X in the transaction t_q, $X \subseteq t_q$, is denoted as $u(X, t_q)$ and is defined as:

$$u(X, t_q) = \sum_{i \in X \cap X \subseteq t_q} u(i, t_q) \tag{2}$$

For example, the utility of an itemset $\{a, d\}$ in t_3 is calculated as: $u(\{a, d\}, t_3) = (3 \times 3) + (5 \times 4) = 29$.

Definition 5 ***Utility of an itemset in the transactional database*** *$(u(X))$*

The utility of an itemset X in the transactional database D is denoted as $u(X)$ and is defined as:

$$u(X) = \sum_{X \subseteq t_q \in D} u(X, t_q) \tag{3}$$

For example, the utility of an itemset $\{a, d\}$ in the transactional database D is calculated as: $u(\{a, d\}) = u(\{a, d\}, t_3) + u(\{a, d\}, t_7) + u(\{a, d\}, t_8) = 29 + 25 + 13 = 67$.

Definition 6 ***Transaction utility*** *$(tu(t_q))$*

The transaction utility of a transaction t_q is denoted as $tu(t_q)$ and is defined as:

$$tu(t_q) = \sum_{i \in t_q} u(i, t_q) \tag{4}$$

For example, transaction utility of t_3 is calculated as: $tu(t_3) = u(a, t_3) + u(b, t_3) + u(d, t_3) + u(e, t_3) = 9 + 2 + 20 + 2 = 33$. The tu value of all the transactions is shown in the fourth column of Table 3.

Definition 7 ***Total utility of a transactional database*** *$(tu^{(D)})$*

The total utility of a transactional database D is denoted as $tu^{(D)}$ and is defined as:

$$tu^{(D)} = \sum_{t_q \in D} tu(t_q) \tag{5}$$

Table 4 High utility itemsets of the running example

Itemset	Utility	Support count	Itemset	Utility	Support count
$\{a\}$	39	5	$\{c, d\}$	13	1
$\{b\}$	14	5	$\{d, e\}$	46	3
$\{d\}$	60	5	$\{a, b, d\}$	60	2
$\{a, b\}$	24	2	$\{a, b, e\}$	27	2
$\{a, d\}$	67	3	$\{a, d, e\}$	83	3
$\{a, e\}$	40	4	$\{b, c, d\}$	15	1
$\{b, d\}$	66	4	$\{b, d, e\}$	45	2
$\{b, e\}$	16	3	$\{a, b, d, e\}$	63	2

For example, the total utility $tu^{(D)}$ of the transactional database D is calculated as: $tu^{(D)} = tu(t_1) + tu(t_2) + tu(t_3) + tu(t_4) + tu(t_5) + tu(t_6) + tu(t_7) + tu(t_8) = (15 + 11 + 33 + 8 + 6 + 10 + 30 + 16) = 129$.

Definition 8 ***High Utility Itemset (HUI)***
An itemset X is called to be high utility itemset (HUI) in a transactional database D if it satisfies the following condition:

$$HUI \longleftarrow \sum_{X \subseteq t_q \in D} u(X, t_q) \geq \delta \times tu^{(D)} \tag{6}$$

For example, the minimum utility threshold δ is set to 10%. In the running example, the itemset $\{d\}$ is an HUI in a transactional database D because the utility of itemset $\{a, d\}$ is $u(a, d) = 29$, as $(29 > \delta \times tu^{(D)} (= 10\% \times 129 = 12.9))$. On the other hand, the itemset $\{a, c\}$ is not an HUI because $u(\{a, c\}) = 10$ as $(10 < 12.9)$. All the HUIs of the running example are shown in Table 4.

Definition 9 ***Transaction weighted utility of an itemset (TWU(X))***
The transaction weighted utility (TWU) of an itemset X is denoted as $TWU(X)$ and is defined as:

$$TWU(X) = \sum_{X \subseteq t_q \in D} tu(t_q) \tag{7}$$

For example, TWU of itemset $\{d\}$ is calculated as: $TWU(d) = tu(t_1) + tu(t_3) + tu(t_6) + tu(t_7) + tu(t_8) = 15 + 33 + 10 + 30 + 16 = 104$. *Similarly,* $TWU(a, d) = tu(t_3) + tu(t_7) + tu(t_8) = 33 + 30 + 16 = 79$. *The TWU values of all the single length items for the running example are shown in* Table 5.

Problem statement: *Given a transaction database D and a user-specified minimum utility threshold δ, the objective of the HUIM is to discover the complete set of items that have utilities no less than the minimum utility count $(\delta \times tu^{(D)})$.*

It is a challenging task to prune the search space in the HUIM because the downward closure property of ARM [1, 34] does not hold for the utility measure of an item in the database. To address this issue, the TWDCP (Transaction Weighted Downward Closure Property) [12] is proposed that is based on the following definitions.

TWU based pruning strategy

The following properties related to TWU measure are used to prune the search space that are defined as:

Property 1 ***Overestimation***. *The TWU of an itemset X is higher than or equal to its utility [12] and is defined as: $TWU(X) \geq u(X)$*

For instance, $TWU(d)$ is 104 and $u(d)$ is 60. Similarly, $TWU(a, d)$ is 79 whereas $u(a, d)$ is 67.

Property 2 ***Anti-monotone***. *The TWU of an itemset follows the anti-monotone property [12] and is defined as: let X and Y be two itemsets and $X \subset Y$, then $TWU(X) \geq TWU(Y)$*

Property 3 ***Pruning***. *Let X be an itemset. If $TWU(X) \geq (\delta \times tu^{(D)})$, then the itemset X is a candidate high utility itemset, otherwise, X is a low utility itemset.*

Definition 10 *$HTWUI$* ***(High Transaction Weighted Utilization Itemsets*** *$(HTWU^{(D)}(X))$*
An itemset X is $HTWUI$ in a transactional database D if it satisfies the following condition:

$$HTWUI(X) = HTWU^{(D)}(X) \geq \delta \times tu^{(D)} \tag{8}$$

For example, the itemset $\{a, b\}$ is $HTWUI$ in the transactional database D as $TWU(\{a, b\}) = tu(t_3) + tu(t_7) = (11+13) = 24$ *as* $(24 > (10\% \times 129 = 12.9))$. *On the other hand, the itemset $\{a, c\}$ is not $HTWUI$ as* $TWU(\{a, c\}) = tu(t_2) = 10$ *as* $(10 < 12.9)$.

Table 5 TWU value of the items

Items	*a*	*b*	*c*	*d*	*e*
TWU	96	96	34	104	98

Property 4 *$TWDCP$ (**Transaction Weighted Downward Closure Property**). The $TWDCP$ indicates that if an itemset X is not an $HTWUI$, then itemset X and its all supersets are low utility itemsets.*

Using the TWDCP property of TWU, many HUIM algorithms [12, 15] are proposed that significantly prune the search space and eliminate the numerous unpromising candidates from the search space. These algorithms consist of two phases. During the first phase, they discover all HTWUIs from the transactional database, while the HUIs are obtained from the set of HTWUIs in the second phase. Although these algorithms mine all the HUIs, but they may generate a large number of candidates in phase one that degrade the overall performance of the mining process. To overcome this problem, many closed HUIM algorithms [20] are proposed that provide a concise and lossless representation of HUIs. These are referred as CHUIs (closed HUIs). For more information, the readers may refer to the closed itemsets in [35, 36].

Definition 11 ***Support count of an itemset** ($sc(X)$)*

The support count of an itemset X is the number of transactions containing X in D and is denoted as $sc(X)$. The support of itemset X is denoted as $s(X)$ and is defined as:

$$s(X) = \frac{sc(X)}{|D|} \tag{9}$$

The complete set of all the itemsets in D is denoted as L and is defined as:

$$L = \{X | X \subseteq I, sc(X) > 0\} \tag{10}$$

For instance, the support count of itemsets $\{d\}$ and $\{abd\}$ are, respectively, 5 and 2. The support of itemsets $\{d\}$ and $\{abd\}$ are $5/8 = 0.625 (> 0)$ and $2/8 = 0.25 (> 0)$ respectively.

Definition 12 ***CHUIs (Closed High Utility Itemsets)***

An itemset X is CHUIs if it is HUIs and there does not exist any HUIs itemset Y such that $X \subset Y$ and $sc(X) = sc(Y)$.

As compared to the HUIs shown in Table 4, *itemset $\{a, b\}$, $\{a, b, d\}$, $\{a, b, e\}$ and $\{b, d, e\}$ are not CHUIs because these itemsets are the subset of the itemset $\{a, b, d, e\}$ with the same support count which is 2. Similarly, itemset $\{a, d\}$ and $\{d, e\}$ are the subset of $\{a, d, e\}$ with the support count 3. And, itemset $\{c, d\}$ is the subset of $\{b, c, d\}$.* Table 6 *shows all the CHUIs for the running example.*

2.2 Applications of high utility itemsets mining

In this section, we briefly discuss various applications of HUIM.

Market basket analysis

The market basket analysis [5] is a data mining technique that is used to analyze the data from large databases such as purchase history, to know the product grouping and products that are often purchased together. It considers the data across their various stores and purchases from different customer groups at different times. It is useful to increase the sales and customer satisfaction. It provides more benefits to retailers and makes the shopping experience more valuable and productive for the customers. The HUIM approaches obtain the utility of each itemset that can be predefined based on the user preferences. Hence, HUIM approaches [5, 37] are quite useful in the market basket analysis to obtain the detailed information about the purchasing behavior of the customers.

Web click stream analysis

Web click stream analysis [38] provides a rich set of click streams data. It includes the visited pages, time spent on each page and detailed history of web pages. Detailed analysis is used to report user behavior on a specific website. The HUIM approaches can discover valuable items and information from the website click-streams [38].

Web mining

Web mining [39] is a data mining technique that automatically discovers useful knowledge from the world wide web and its usage patterns. It improves the power of web search engines by identifying the web pages and web documents. It is used to predict the user behavior. For example, Yahoo and Google are the powerful web search engine. The HUIM approaches discover the high utility itemsets from the

Table 6 Closed high utility itemsets of the running example

Itemset	Utility	Support count	Itemset	Utility	Support count
$\{a\}$	39	5	$\{b, e\}$	16	3
$\{b\}$	14	5	$\{a, d, e\}$	83	3
$\{d\}$	60	5	$\{b, c, d\}$	15	1
$\{a, e\}$	40	4	$\{a, b, d, e\}$	63	2
$\{b, d\}$	66	4			

web-log such as high utility traversal items [40] and high utility access items [39]. These approaches are quite useful to improve web services, web browsing, and web pages.

Cross marketing

Cross marketing [32, 41, 42] helps to identify the customer purchasing behavior, improves customer service, enhances sales, focuses on customer retention and reduces the cost of businesses. Gan et al. [5] design a novel method to extract the non-redundant correlated purchase behaviors by considering the utility and correlation factor that result in high recall value and accurate precision.

Gene regulation

The gene regulation [9] technique is used to acquire information about the sequence structure, gene interaction and genetic pathway. This information significantly complements the analysis of any data and improves its results. The HUIM approach [43] is used to obtain promising gene regulation patterns from large gene expression data. It is useful to discover new drugs for health care.

E-commerce

E-commerce [11] warehouse management significantly improves the efficiency of the warehouse operations and improves the customer service. The proposed method [44] solves the item storage assignment problem by using the top-k HUIM method and a heuristic algorithm. Similarly, mobile commerce environment [10, 15] is useful to discover the sequential purchasing patterns (or mobile sequential patterns) by the movement of the path with the purchasing transactions. It is also useful to analyze and manage the online shopping websites. Yun et al. [45] proposed a method that integrates the movement of the path with the sequential patterns to discover the mobile sequential patterns.

Although there are plenty of applications of HUIM in the field of data mining. But, due to space limitation, we have taken into consideration only a limited number of applications. In the next section, we provide an up-to-date and comprehensive survey of the high utility itemsets mining algorithms from transactional databases along with their pros and cons.

3 High utility itemsets mining algorithms

Traditional HUIM approaches for transactional databases are generally divided into the following groups: level-wise, tree-based, utility-list-based and projection-based. The level-wise approaches [12, 46] are based on candidates generation and test methods. They discover HUIs in two phases. During the first phase, they overestimate the utility of itemsets based on the TWU method. In phase 2, they identify all the HUIs. However, they generate numerous candidates and require multiple database scans. In order to resolve the problems of level-wise approaches, the tree-based algorithms [47–51] are proposed that mainly consist of three steps: (1) tree construction; (2) candidates generation from the constructed tree; (3) identification of the HUIs from the set of candidates. However, they generate a large number of candidates that eventually consume high memory and degrade the performance of the mining process. In order to resolve these issues, utility-list-based algorithms [16–18] are proposed that do not generate the candidates and avoid multiple database scans. They are more efficient with regard to the run-time as compared to the level-wise and tree-based algorithms. However, they need to perform costly join operations that lead to high memory consumption and complexity. In order to resolve these issues, projection-based algorithms [13, 19] are proposed to enhance the effectiveness of the mining process by reducing excessive candidates generation. Moreover, they are highly memory-efficient and scalable. However, they do not perform well on some benchmark databases due to high threshold value.

In the past decade, numerous HUIM algorithms are proposed to discover the HUIs from transactional databases. This paper is divided into five categories: (1) level-wise (2) tree-based (3) utility-list-based (4) projection-based; (5) miscellaneous. Figure 2 represents the taxonomy of the state-of-the-art HUIM approaches for transactional databases.

3.1 Level-wise high utility itemsets mining

Level-wise high utility itemsets mining algorithms follow the TWU property to mine the HUIs from the transaction database. They perform the level-wise search and/or depth-first search to discover the items present in the transactional database. The low utility itemsets are pruned by using pruning strategies such as TWU, upper-bound utility, etc. The objective of these algorithms is to find the usefulness of the items to extract the maximum profit for the businesses.

Chan et al. [52] proposed OOA (Objective-Oriented utility-based Association) approach that mines the top-k closed utility patterns. It uses the weaker but anti-monotone condition [1] to reduce the search space to derive all OOA rules efficiently. The main advantage is that the user-specified minimum utility is not required that makes the proposed algorithm very effective. But, there is still little overhead to obtain the intended results.

In 2005, Yao et al. [53] proposed MEU (Mining with Expected Utility) algorithm that considers both the internal and external utility of an itemset to find the HUIs. It uses the heuristic model to limit the search space and provides a theoretical foundation of efficient utility mining algorithms.

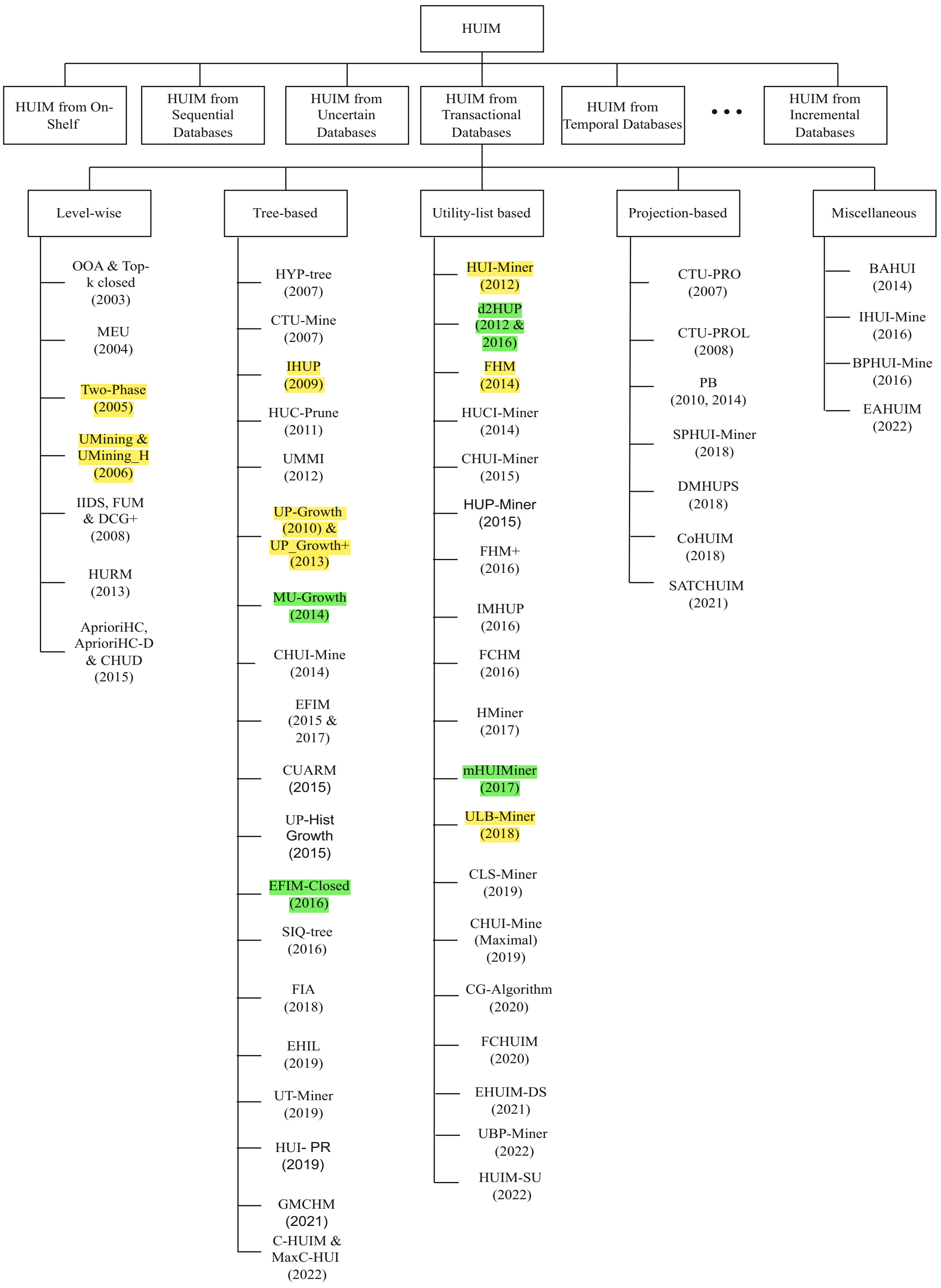

Fig. 2 Taxonomy of high utility itemsets mining algorithms for transactional databases

However, some high utility itemsets may miss, therefore, it cannot obtain the complete results. Moreover, MEU does not follow the DCP property of Apriori [34].

To address the limitations of MEU [52], Liu et al. [12, 54] proposed Two-phase algorithm. They proposed a TWU mining model by maintaining the TWDCP in phase one, while in phase two, only one extra database scan is required to prune the overestimated itemsets. However, it generates too many candidates and performs multiple database scans.

The main limitation of frequency mining association rules [34] is that they do not take into account the statistical correlation between itemsets. To overcome this limitation, Yao et al. [46] proposed two algorithms, namely UMining and UMining_H by incorporating pruning strategies, utility upper-bound and support upper-bound respectively. UMining is better than UMining_H because it discovers all HUIs. However, both algorithms suffer from excessive candidates generation.

Li et al. [8] proposed IIDS (Isolated Items Discarding Strategy) method to identify the HUIs. It can pertain to any existing level-wise methods and DCG (Direct Candidates Generation) [41] to decrease the number of candidates and hence, improve the mining performance. By applying IIDS to ShFSM and DCG, authors have implemented FUM (Fast Utility Method) and DCG+ respectively. But, it has also the same performance issues like Apriori. Moreover, these methods generate a large amount of candidates and requires multiple database scans.

HUIM mining methods lack to provide business insights so that customers can get maximum benefits [53]. To provide the business insights, Lee et al. [55] developed HURM (High Utility Rule Mining) method that gives the firms to quantitatively represent user' preferences for utility values. The algorithm provides a large amount of data from the various heterogeneous business environments and hence gives opportunities to firms to increase their business benefits. The performance of HURM is much better (best 133 percent, normal 120 percent and worst 113 percent) than that of HUIM methods in terms of business context.

Wu et al. [35] provide CHUD (Closed$^+$ High Utility itemset Discovery) algorithm to mine closed$^+$ HUIs. Later, this work is extended by Tseng et al. [20]. The authors proposed three algorithms namely, AprioriCH (Apriori-based algorithm for mining High utility Closed$^+$ itemsets), AprioriHC-D (AprioriHC algorithm with Discarding unpromising and isolated items) and CHUD that represent the compact and lossless representation of HUIs. Further, they proposed DAHU (Derive All High Utility Itemsets) that recovers all HUIs from the CHUI[3] (Closed High Utility Itemsets). CHUD is used to mine the closed itemsets. However, AprioriCH and AprioriHC-D have poor performance on dense databases as they suffer from excessive candidates generation, while CHUD suffers from high memory consumption.

[3] CHUI is a group of itemsets that have no supersets with the same support count in the database.

Summary

Level-wise HUIM approaches are proposed that allow a user to conveniently express his or her perspectives concerning the usefulness of itemsets as utility values and then find itemsets with utility values higher than the specified threshold. The challenge of level-wise HUIM is in restricting the size of the candidate set and simplifying the computation for calculating the utility. Level-wise HUIM algorithms generate a large amount of candidates. They perform multiple database scans to obtain the desired results that are same as the Apriori approach. Therefore, they take longer execution time and high memory consumption. These algorithms are not suitable for those databases that are large in size and have longer transactions. The detailed overview of level-wise HUIM algorithms is shown in Table 7. Furthermore, the pros and cons of all the level-wise HUIM algorithms are depicted in Table 8.

3.2 Tree-based high utility itemsets mining

In this section, the tree-based HUIM algorithms are discussed to address the limitations of level-wise approaches. The level-wise approaches [46, 53, 54] require two phases to find HUIs that cause numerous candidates generation and multiple database scans. To overcome these issues, tree-based HUIM algorithms [47–51] are proposed that are based on the pattern-growth approach [57] and compact tree structure. These algorithms significantly reduce the number of candidates with the overestimated methods and effectively reduce the search space to find HUIs. They just need two and/or three passes over databases and much faster than Apriori-like approaches. The objectives of these algorithms are to significantly decrease the number of candidates set and numerous database scans by proposing efficient pruning strategies and compact tree structures.

To address the problem of identifying the high utility itemsets, Hu et al. [58] proposed an efficient approximation by introducing the novel binary partition tree called HYP (High Yield Partition) tree. It contributes towards pre-defined utility, objective function or performance metric. It finds the segment of data by taking a combination of a few items and/or rules. However, when the HYP-tree becomes larger, it consumes a high amount of memory. Hence, it is not suitable for large databases.

To address the limitation of HYP-tree [58], Erwin et al. [47] proposed CTU-Mine (Compressed Transaction Utility Mine) algorithm that mines HUIs using pattern-growth approach. It introduces compact data representation, called

Table 7 An overview of key characteristics of level-wise high utility itemsets mining algorithms

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
OOApriori and top-K closed [52], Chan et al. (2003)	–	Multiple times	–	Top-K utility, Frequent closed patterns	–	Weaker but anti-monotonic condition	Positive and negative	OOApriori (Self) [52]	Apriori [34]
MEU [53], Yao et al. (2004)	–	Multiple times	Utility Table	Utility Itemsets	According to information stored in the transaction	Transaction utility, External utility	Positive only	–	–
Two-phase [12, 54], Liu et al. (2005)	Two	Two	Utility Table	HUIs	Level-wise search	TWU	Positive only	MEU [53]	Apriori [34]
UMining [46] and UMining_H [46], Yao et al. (2006)	–	Multiple times	–	HUIs	Level-wise search	Utility upper-bound and support upper-bound, Heuristic	Positive only	SIP [56]	–
IIDS, FUM and DCG+ [8], Li et al. (2008)	Two	Multiple times	Utility Table	CHUIs, HUIs	Level-wise search	Downward closure property	Positive only	Two-phase [12], DCG [41]	Apriori [34]
HURM [55], Lee et al. (2013)	Two	Multiple times	–	Antecedents, Consequent, High-utility rules (HURs)	Level-wise search	Antecedent pruning functions, Consequent pruning functions	Positive only	HUIM [46, 52]	–
AprioriHC, AprioriHC-D and CHUD (extended) [20], Tseng et al. (2015)	–	Multiple times	Utility unit array	CHUIs, HUIs	Depth-first search	TWU, DGU, IIDS	Positive only	Two-phase [12], UP-Growth [14]	Two-phase [12]

CTU-tree (Compressed Transaction Utility tree). This approach avoids candidates generation-and-test and best suits for dense databases and long patterns. But, CTU-tree is very complicated and store excessive information that leads to high memory consumption. Moreover, Two-phase algorithm [12, 54] is relatively faster than CTU-Mine when the threshold value is set high.

As we have seen in previous works [12, 46, 47, 58], they only work for the fixed databases and do not consider dymanic databases like insertion, deletion or modification of the transactions occur in the database. To address this issue, Ahmed et al. [32] proposed IHUP (Incremental High Utility Pattern) algorithm that is based on "build once mine many" property to mine the HUIs in the incremental database. It consists of three efficient tree structures, namely, IHUP$_L$-Tree (Incremental HUP Lexicographic Tree), IHUP$_{TF}$-Tree (IHUP Transaction Frequency Tree) and IHUP$_{TWU}$-Tree (IHUP Transaction Weighted Utilization Tree). The first tree structure, IHUP$_L$-Tree is arranged as per item's lexicographic order. The second tree structure, IHUP$_{TF}$-Tree obtains the compact size by arranging items as per the decreasing order of their transaction frequency. The final tree structure, IHUP$_{TWU}$-Tree is designed to decrease the mining time by considering the decreasing order of TWU value of items. The main advantage of IHUP is that the proposed tree structures are efficiently discover and achieve scalablity for increment and interactive HUP mining. However, the proposed algorithm produces large number of HTWUIs during the first phase.

Ahmed [48] proposed HUC-Prune (High Utility Candidates Prune) to avoid the candidates generation in level-wise fashion. It adopts the pattern-growth approach and uses the proposed HUC-tree (High Utility Candidates tree) structure. In the first database scan, HUC-Prune finds the patterns of single element candidates. During the second database scan, the HUC-tree is used to capture the TWU value of items in transactions. Finally, a third database scan finds all the HUIs from the candidate patterns. Hence, it requires maximum of

Table 8 Pros and cons of level-wise high utility itemsets mining algorithms

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
OOApriori top-K closed [52], Chan et al. (2003)	Based on the Apriori algorithm [1] with special mechanisms to handle the utility and generate the closed patterns.	German credit, Heart decease	User-specific minimum utility is not required.	There is small overhead to produce the desired results.	The utility constraint can be pushed deep into FP-tree algorithm or other frequent pattern mining algorithms.	Differentiate with existing research on the measure "Interestingness"
MEU [53], Yao et al. (2004)	The first theoretical model and strict definitions of HUIM.	…	MEU uses a heuristic to determine candidates and usually overestimates.	It cannot maintain the downward closure property of Apriori and the derived results are incomplete. The results are inappropriate when the utility threshold is low and the number of distinct items is large.	The future work could be designed the algorithm that might be compared with the other itemset mining algorithms.	It generalizes the previous work on the itemset share.
Two-phase [12, 54], Liu et al. (2005)	HUIs are discovered in two phased by using the TWDC property.	Grocery chain Store, T10.I6.DX000K, T20.I6.DX000K	It effectively reduces the search space by using TWDC property and captures the complete set of HUIs.	It generates too many candidates for HTWUIs and requires multiple database scans.	It can easily handle very large databases for which other existing algorithms are infeasible.	Same as Ariori.
Umining [46] and UMining_H [46], Yao et al. (2006)	The general HUIM model with several mathematical properties of the "utility measure".	Customer, Commercial, IBM synthetic	Efficiently handle utility constraints to express types of semantic significance. UMining may be prefer over UMining_H, because it guarantees the discovery of all HUIs.	It suffers from excessive candidate generations and poor scalability. The UMining_H may miss some HUIs. Unable to identify high utility rules from HUIs.	Pruning strategies can be incorporated into FP-growth etc. Extended from positive utility functions to more general utility functions	Provide a general framework for weighted itemset mining. UMining algorithm follows the basic framework of the Apriori algorithm with slight modification.
IIDS, FUM and DCG+ [8], Li et al. (2008)	By applying IIDS to ShFSM and DCG, two methods, FUM and DCG+, are implemented.	Chain-store, T10.I6.D1000k.N1000, T10.I6.D100k.N2000, T20.I6.D1000k.N1000	The performance of FUM and DCG+ are better than compared (ShFSM and DCG) algorithms. IIDS can be applied to any existing level-wise utility mining method to further reduce the number of redundant candidates.	It has the same performance issues as Apriori. They scan the database multiple times and suffer from the problem of the candidate generation-and-test scheme to find HUIs.	Extend IIDS to classification models.	IIDS can be applied to Apriori-like traditional mining.

Table 8 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
HURM [55], Lee et al. (2013)	The proposed algorithm mines the utility-based association rules that evaluates the association rules based on the specific need of business.	Market basket, supermarket transaction database	It provides the higher business benefits as compare to traditional HUIMs algorithms. The performance is (best 133%, normal 120%, worst 113%)	…	It might be used to process a massive amount of data collected from diverse business environments, providing firms with opportunities to enhance their business benefits.	Users can define their own preferences in utility functions and generate association rules that help them to accomplish their business objective.
AprioriHC, AprioriHC-D and CHUD (extended) [20], Wu et al. (2015)	Lossless and compact representation.	BMSWebView1, Foodmart, Mushroom and T10I8D200K	DAHU method recovers all HUIs from the set of CHUD without access to the original database. AprioriHC-D runs faster and produces fewer candidates than AprioriHC. CHUD reduces the number of itemsets up to 800 times.	CHUD uses more memory as compared to UP-Growth. AprioriCH and AprioriHC-D does not perform well on dense databases as they suffer from the generation of large number of candidates.	The authors performed only the integration of CHUI and HUI mining. There is wide scope to consider the other compact representations.	The combination of CHUD and DAHU provides a new way to obtain the complete set of HUIs.

three database scans to discover HUIs for a minimum utility threshold. The advantage of HUC-Prune is that it avoids excessive candidates and multiple database scans. It performs well with large dense databases, highly scalable, and memory efficient. However, the construction of tree is complex and large.

The previous level-wise and tree-based HUIM algorithms find the excessive promising itemsets in step one that cause the problem of mining congestion. To avoid this problem, Lin et al. [49] proposed UMMI (High Utility Mining using the Maximal Itemset property) algorithm that uses maximal itemset property. They developed HTP (High TWU Pattern) tree structure to find all the maximal high TWU itemsets. In step one, it significantly decreases the promising itemsets, while in step two, it utilizes the MLexTree (Maximal Lexicographic Tree) structure to find all HUIs. MLexTree structure is based on the LexTree (Lexicographic Tree) structure. UMMI is a highly efficient algorithm to find HUIs in very large databases. Moreover, it is linearly scalable in terms of the number of transactions. But, it needs additional memory to construct the HTP tree structure.

When there are large number of PHUIs (Potential HUIs) especially when the database consists of longer transactions or the low threshold is set, the previous algorithms do not perform well. To avoid this problem, Tseng at al. [14] proposed UP-Growth (Utility Pattern-Growth) to effectively discover the HUIs from the transaction databases. They proposed the compact UP-Tree (Utility Pattern Tree) structure to keep the information of HUIs. The proposed algorithm generates the HUIs in two database scans. It uses four pruning strategies, namely DGU (Discarding Global Unpromising items), DGN (Discarding Global Node utilities), DLU (Discarding Local Unpromising items) and DLN (Decreasing Local Node utilities). To extend the work of UP-Growth, UP-Growth+ [15] is proposed that utilizes the minimal utilities of each node in each path of the UP-tree. As compared to the UP-Growth, the UP-Growth+ significantly decreases the overestimated utilities of PHUIs and effectively reduces the number of candidates. The limitation of UP-Growth+ is that it consumes more time to recursively process all conditional prefix-trees to generate the candidates.

Yun et al. [59] proposed MU-Growth (Maximum Utility Growth) algorithm which efficiently reduces the candidates. They suggest MIQ-Tree (Maximum Item Quantity Tree) structure that holds the data information in a single pass. MIQ-tree can be restructured to reduce the overestimated utilities. It has following three steps: (1) During the first step, the initial tree is constructed by using the items and quantities associated with items in the transactions. Then, a tree is reconstructed based on the decreasing order of TWU values. (2) In the second step, MU-Growth algorithm performs the generation of candidates from the restructured tree.

(3) Finally, in the last step, actual HUIs are identified. The proposed algorithm performs efficiently on the databases that contain long transactions or the low minimum utility threshold value is set. MU-Growth performs well than the UP-Growth+ [15] with regard to all the threshold values. However, it suffers from the high execution time when the total number of transactions is increased in the database.

As the previously patter-growth approaches [14, 15, 47, 48, 59] generate excessive conditional trees that lead to high cost with regard to space and time. To address this challenge, Song et al. [50] design an efficient algorithm, called CHUI-Mine (Concurrent High Utility Itemsets Mine) which dynamically prunes the tree structure to discover the HUIs. They proposed CHUI-tree structure that is based on the pattern-growth approach and can be completed within two database scans. The main advantage of CHUI-Mine is that it can discover HTWUIs during the process of tree construction and avoids the generation of the whole tree structure. Hence, CHUI-Mine efficiently reduces memory consumption because the pruned trees are usually small. However, it does not consider the comparison with a faster tree-based algorithm IHUP [32] with respect to memory usage.

To avoid excessive candidates generation, HUI-Miner [16] and d2HUP [4] are proposed to mine HUIs in a single phase. Then, HUP-Miner [60] and FHM [17] are designed that are six times faster than that of HUI-Miner [16]. However, the computation task remains very expensive to find HUIs despite all these efforts. Zida et al. [61] developed EFIM (EFficient high utility Itemset Mining) algorithm that efficiently finds the HUIs in one phase only. It uses two upper bounds, namely sub-tee utility and local utility that effectually limit the search space. A novel array-based utility counting method, called FAC (Fast Utility Counting) is introduced that calculates these upper bounds in linear time and space. The authors proposed efficient database projection and transaction merging techniques to reduce database scans cost. EFIM is two to three times faster and eight times less memory consumption compared to the UP-Growth+ [15], HUI-Miner [16], d2HUP [4], HUP-Miner [60] and FHM [17] on various benchmark databases. This work is later extended in [51] that significantly improves the performance of EFIM [61] with regard to the run-time and memory usage. However, in some cases, the recursive projection takes considerable time and consumes high memory. Moreover, the transaction merging techniques do not scale well on sparse databases.

Shao et al. [62] proposed CUARM (Combined Utility Association Rules Mining) algorithm to mine actionable high utility association rules, named CUAR (Combined Utility Association Rules). The proposed CUAR structure generates patterns that are both strongly associated and have high utility. They also proposed AUG (Associated Utility Growth) that integrates the utility and association. The proposed algorithm is based on UG-Tree (Utility Growth Tree) structure that requires only two database scans. The advantage of CUARM is that it generates HUIs without the loss of representativeness (strong association). However, it does not consider the utilities which are lower than the underlying itemsets for the utility decrement itemsets.

The previous works [14, 15] are based on the UP-tree that generates a large number of candidates to compute the utility value to estimate the itemsets. To address this issue, Dawar et al. [63] proposed the efficient UP-Hist Growth algorithm and UP-Hist tree structure that discover the HUIs from a transaction database. UP-Hist tree keeps the histogram of item quantities with each node of the tree. The process of UP-Hist tree construction requires two database scans. The histograms compute better utility estimation to significantly limit the search space. However, histograms are required extra storage space to store the UP-Hist tree.

CHUD [20, 35] incurs highly computing cost to mine CHUIs. To resolve this challenge, Fournier-Viger et al. [64] proposed EFIM-Closed (EFficient high utility Itemset Mining Closed) algorithm that uses three strategies namely, CJU (Closure Jumping), FCC (Forward Closure Checking), and BCC (Backward Closure Checking) to efficiently mine CHUIs. EFIM-Closed uses two efficient techniques, namely HDP (High utility Database Projection) and HTM (High utility Transaction Merging) to reduce the cost of database scans. The proposed algorithm is up to 71 times faster and consumes up to 19 times less memory than that of CHUD [35]. However, EFIM-closed does not perform well in case of dense databases.

The previous algorithms [14, 15] require two database scans to construct the UP-tree. Ryang et al. [65] proposed SIQ-Tree (Sum of Item Quantities Tree) which scans the database in a single pass to capture the data information. The authors suggest restructuring method with two strategies, namely RPS (Reducing Path Support) and REPU (Reducing Estimated Path Utility using maximum item utility) that effectively decrease the number of candidate patterns with the reduced overestimation of utilities. A fast algorithm for HUIs is considered to enhance the mining performance. However, it needs a high amount of computing time as the item size increases.

The previous works [8, 14, 15, 32] require two phases to discover the HUIs which result of high cost in terms of candidates generation and utility calculation. To address these issues, Qu et al. [66] first present BIA (Basic Identification Algorithm) to find the HUIs. Second, they proposed a new candidate tree structure to store the candidates. A candidate tree is a modified prefix-tree [67] that shares a common path using the same prefix. Finally, FIA (Fast Identification Algorithm) is developed to fast identify the HUIs. The proposed algorithm performs well when the candidate tree completely fits in memory. However, it does not take into consideration the case when the candidate tree is too large to completely fit in memory.

To overcome the problem of mining lots of tiny itemsets, concise HUIM is required. To address this issue, Singh et al. [68] proposed EHIL (Efficient High utility Itemsets with Length constraints) algorithm that considers the length constraints to find HUIs. The minimum and maximum length constraints, respectively remove tiny itemsets and restrict the higher length of the itemsets, to restrict the length of HUIs. The proposed algorithm uses two database compaction techniques namely, database projection and transaction merging, to speed up the execution process. EHIL utilizes two pruning strategies, namely RLU (Revised Local Utility) and RSU (Revised Sub-tree Utility) that use depth-first search to reduce a large number of unpromising candidates. An efficient array-based utility counting technique is employed that computes the upper bounds to speed up the utility counting process. It computes the utility of itemsets without the need of original database scan. EHIL incorporates length constraints in the redefined sub-trees and TWU pruning strategies. EHIL outperforms FHM+ algorithm [69] with regard to memory consumption and run-time. The memory consumption is up to 28 times less than that of FHM+ algorithm, while the improvements of execution time are range from 5 percent to two orders of magnitude across various compared databases. However, the performance of the proposed algorithm depends on the user-defined threshold, minimum utility, the minimum and maximum length. Moreover, it does not perform well on dense databases.

The algorithms [15–17, 51] do not perform consistently across sparse and dense databases. To overcome this issue, Dawar et al. [70] proposed UT-Miner (Utility Tree Miner) that mines HUIs in one phase without generating any candidate. The proposed algorithm uses the lightweight method that constructs the projected database to explore the search space. They proposed UT (Utility Tree) structure to store the information compactly with each node of the tree in the transaction. UT-Miner is one of the top-performing efficient algorithm across dense and sparse databases.

To design efficient pruning strategies, Wu et al. [71] proposed HUI-PR (HUIM with Pruning Strategies) to efficiently find the HUIs. The proposed work is the extension of EFIM [61]. It proposes two pruning strategies, namely strict local utility and strict remaining utility to significantly limit the search space and effectively estimate the candidates. HUI-PR has following advantages: (1) estimates fewer candidates than that of d2HUP [4] and EFIM [61]. (2) generates fewer branches of search space than that of EFIM [61]. (3) significantly reduces the memory consumption on large databases. (4) multi-threshold method helps to speed the mining performance in large databases without using the projection operations. (5) performs well on dense databases. However, it suffers from the following disadvantages: (1) increases the complexity of the newly proposed upper bounds. (2) does not perform well on sparse databases.

FCHM (Fast Correlated High-utility itemset Miner) algorithm [72] incurs two major drawbacks: (1) Correlation measures must satisfy the DCP. (2) Each measure requires different strategies to calculate and prune to enhance the performance of the corresponding mining task. To address these issues, Hoa et al. [73] proposed the HUIM-class algorithm, namely GMCHM (General Method for Correlated High-utility itemset Mining), based on the EFIM algorithm [51] which considers the high-correlated property among itemsets within customer transaction databases. It stores all the required information to compute the given measure in only one database scan. The GMCHM algorithm utilizes more efficient approaches like High-utility Database Projected (HDP), High-utility-Transaction Merging (HTM), and tighter upper bounds, such as local utility and sub-tree utility, to prune the candidates. The experiment results show that the proposed algorithm performs well as compared to existing FCHM [72] in terms of execution time, memory consumption, and the candidates checked. The speed of GMHCM is up to 7.5 times faster than that of the FCHM algorithm using the *bond* measure and up to 2,600 times faster when $all\_confidence$ measure is 0.1.

EFIM-Closed [64], HMiner-Closed [74], and CHUI-Miner(Max) [75] provide concise representation of HUIs. However, they incur long execution time, high memory consumption, and scalability issues, especially for dense and large datasets. To solve these problems, Hai Duong et al. [76] proposed two efficient algorithms, namely C-HUIM and MaxC-HUIM, first of this kind, to simultaneously mine CHUIs (Closed HUIs) and MaxHUIs (Maximal HUIs) respectively. A novel weak upper bound (WUB) named FWUB and corresponding pruning strategy, named SPWUB is proposed to quickly prune the low utility itemsets. Moreover, two pruning strategies PSNonCHUB (Pruning Strategy of NonCHU Branches in the Process of Constructing CHUI) and LPSNonCHUB (Local Pruning Strategy of NonCHU Branches Without Checking the Subpattern Relationship) are proposed to reduce the search space. PSNonCHUB only need checking the inclusion relationship among a small number of itemsets, while LPSNonCHUB does not need inclusion relationship. The first optimization technique is proposed by designing a new minimum support threshold called *newms* based on the minimum utility threshold *mu* and the maximal value of the TWU. The second optimization technique is designed to reduce the number of inclusion relation checks between itemsets when using the pruning strategy PSNonCHUB. A structure, named MPUN-list, modified version of the PUN-list (PU-tree-Node list) structure [77], is adopted to efficiently store and calculate the information about utility and support of each itemset. The experiment results show that the proposed algorithm is 100 times faster, memory efficient, scalable as compared to the state-of-the-art algorithms, namely CHUI-Miner [78], EFIM-Closed [64], HMiner-Closed [74], and CHUI-Miner(Max) [75] with regards to the execution time, memory consumption and scalablity.

Summary

As discussed above, tree-based HUIM algorithms have shown improvement over level-wise approaches [8, 46, 54] in terms of candidates generation and database scans. The reason is that most of the approaches require only two scans of the database and require less memory as they explore the search space in a depth-first manner. However, tree-based algorithms suffer from the following main disadvantages: (1) tree structure is complex and requires too much information that leads to high memory usage; (2) require more time to recursively process all the prefix-trees to generate candidates; (3) a compact tree is generally expensive to build; (4) a constructed tree may not completely fit in the memory; (5) does not perform well either on dense databases or sparse databases. (6) does not scale well in the case of very large databases. The detailed comparison of the characteristics of tree-based HUIM algorithms are shown in Table 9. Furthermore, the pros and cons of all the tree-based HUIM algorithms are depicted in Table 10.

3.3 Utility-list based high utility itemsets mining

Utility-list-based HUIM algorithms [16–18, 78, 84] have the following advantages over level-wise [8, 46, 54] and tree-based approaches [15, 32, 47, 48, 50, 51]: (1) use vertical [17, 60] and/or horizontal [4] data structure to efficiently perform in one phase only to find the HUIs; (2) the search space spans for the set-enumeration tree [34] obtain the total utility of itemsets to build a utility-list by performing join operations; (3) the upper-bound remaining utility is used to obtain the utility lists; (4) the depth-first search method is used to quickly compute the value of utilities; (5) avoid the costly candidate's generation and utility computation; (6) scalable for the large number of items and transactions; (7) effectively reduce the execution time and memory usage.

Liu et al. [81] proposed HUI-Miner (High Utility Itemset Miner) algorithm to efficiently mine HUIs in one phase. The authors proposed a vertical data structure called a utility-list that stores the utility information of an itemset and the heuristic information to prune the search space. However, the join operations between utility-list of k-itemsets and (k+1)-itemsets consume a high amount of time. Moreover, it suffers from high space and complexity.

The existing utility mining algorithms [14, 32, 42] do not address the challenges when there are longer transactions or the minimum utility threshold is set to be small. To overcome these challenges, Liu et al. [4] proposed d2HUP (Direct Discovery of High Utility Patterns) that uses a pattern-growth approach to directly discover the HUIs by performing only single-phase without the candidates generation. It searches the prefix extension tree using DFS and enumerates an itemset to compute its utility. The original utility information is maintained by the proposed CAUL (Chain of Accurate utility-lists) structure that computes the tight-bound for pruning and directly identifies the HUIs. Moreover, an upper-bound on the utility of prefix extensions of itemsets prunes the search space to directly obtain the HUIs. Look-ahead strategy, based on the closure property[4] [4] and singleton property[5] [4], is incorporated to enhance the efficiency of the algorithm in the dense databases. This work is further extended in [16] that includes several features like efficient computation by pseudo-projection, controlled irrelevant item filtering, and optimizations by partial materialization. However, it suffers from the following disadvantages: (1) the tree structure and CAUL both consume a high amount of memory; (2) the efficiency of the algorithm is low when the look-ahead strategy is of little use; (3) the algorithm obtains the values of upper-bound and utility for a large number of candidates; (4) the performance of the proposed algorithm [16] is not compared with recent algorithms [17, 60].

HUI-Miner [81] performed costly join operations to calculate the utility of each itemset. To address this limitation, Fournier-Viger et al. [17] proposed FHM ((Fast High utility Miner) algorithm and a novel pruning strategy EUCP (Estimated Utility Co-occurrence Pruning). EUCP is based on the EUCS (Estimated Utility Co-occurrence Structure) [17] that reduces the costly join operations by analysis of item co-occurrences. Experimental results show that FHM is up to six times faster and performs up to 95 percent fewer join operations than that of HUI-Miner [81]. However, it has the following disadvantages: (1) consumes slightly high memory than HUI-Miner and has poor performance on dense databases; (2) suffers from high space and complexity; (3) treats all items equally that cannot exactly reflect the characteristics of the real-world databases.

The traditional HUIM algorithms [35, 52] compactly represent HUIs but fail in the case of association rules. To resolve this issue, Sahoo et al. [85] proposed HUCI-Miner (High Utility Closed Itemset Miner) algorithm that extracts HUCIs (High Utility Closed Itemsets) along with their generators. It represents the condensed representation of association rules in support of the confidence framework to mine the HUIs. The proposed algorithm achieves a significant reduction to compress the number of HUIs.

[4] All the prefix extensions of an itemset with relevant items, are high utility itemsets without creating the rest of its subtree.

[5] An itemset is high utility itemset and all its proper subsets are not, without creating the rest of its subtrees.

Table 9 An overview of key characteristics of tree-based high utility itemsets mining algorithms

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
HYP tree [58], Hu et al. (2007)	–	Multiple times	Binary partition tree	High utility, frequent item sets	Recursive procedure	Yield Greedy (YG), YG with one look ahead, Positive correlation, Positive and negative correlation	Positive only	–	–
CTU-Mine [47], Erwin et al. (2007)	One	Twice	CTU-tree	HUIs	Ascending order of TWU values.	TWU	Positive only	Two-phase [12]	Two-phase [12]
IHUP [32], Ahmed et al. (2009)	Two	Multiple times	IHUP$_L$-Tree, IHUP$_{TF}$-Tree, IHUP$_{TWU}$-Tree	HUPs	Item lexicographic order, Decreasing order of transaction frequency of items, Decreasing order of TWU values	TWU	Positive only	Two-phase [12] FUM [8] DCG+ [8]	Two-phase [12]
HUC-Prune [48], Ahmed et al. (2011)	Two	Three times	HUC-tree	High utility patterns (HUPs)	Descending order of TWU values	MTWU	Positive only	Two-phase [12], FUM [8], DCG+ [8]	Two-phase [12]
UMMI [49], Lin et al. (2012)	Two	Three times	HTP MLex-Tree	HUIs	Depth-first search	TWU	Positive only	Two-phase [12], CTU-PRO [79] TWU-mining [80]	Two-phase [12]
UP-Growth [14], UP-Growth+ [15], Tseng et al. (2013)	Two	Twice	UP-Tree MIUT	PHUIs	Descending order of TWU values	DGN, DGU, DLU, DLN	Positive only	IHUP [32], FP-Growth [57]	Two-phase [12]
MU-Growth [59], Yun et al. (2014)	Three	Twice	MIQ-Tree	HUIs	Bottom-up approach	TWU	Positive only	IHUP [32], UP-Growth [14], UP-Growth+ [15]	–
CHUI-Mine [50], Song et al. (2014)	Two	Twice	CHUI-Tree	HUIS	Descending order of TWU values	CPL, HTWUI	Positive only	Two-phase [12], FUM [8], HUC-Prune [48]	Two-phase [12]
EFIM [51, 61], Zida et al. (2015 and 2017)	One	Multiple times	Utility-binary array	HUIs	Depth-first search	LU-Prune, SU-Prune	Positive only	UP-Growth+ [15], HUP-Miner [60], d2HUP [4], HUI-Miner [81], FHM [17]	UP-Growth [14]

Table 9 continued

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
CUARM [62], Shao et al. (2015)	–	Twice	UG-tree	CUAR, AUG	Descending order of TWU values	TWU, Rebuilt transaction utility (RTU)	Positive only	FP-Growth [57], UP-Growth [14]	–
UP-Hist Growth [63], Dawar et al. (2015)	Two	Twice	UP-Hist tree	HUIs	Bottom-up approach	TWU	Positive only	UP-Growth [14], UP-Growth+ [15]	–
EFIM-closed [64], Fournier-Viger et al. (2016)	One	Multiple times	UP-Tree	CHUIs	Depth-first search	Local utility, sub-tree utility and Backward extension pruning	Positive only	CHUD [78]	EFIM [61]
SIQ-tree [65], Ryang et al. (2016)	Two	Once	SIQ-Tree	HUPs	Descending order of TWU values	RPS, REPU	Positive only	IHUP [32]	–
FIA [66], Qu et al. (2018)	Two	Multiple times	Candidate tree	HUIs	Vector Utility		Positive only	UP-Growth+ [15]	Two-phase [12]
EHIL [68], Singh et al. (2019)	One	Multiple times	Utility binary array	HUIs	Depth-first search	Revised Local Utility, Revised Sub-tree Utility	Positive only	FHM+ [69]	EFIM [61]
UT-Miner [70], Dawar et al. (2019)	One	Twice	Utility-tree and HashMap	HUIs	Descending order of TWU values	Exact utility and Remaining Utility	Positive only	FHM [17], mHUIMiner [82], UFH [83], EFIM [61], d2HUP [4], HMiner [18]	–
HUI-PR [71], Wu et al. (2019)	One	Twice	Utility-tree	HUIs	Increasing order of TWU, depth-first search	Strict Sub-tree Utility, strict local utility	Positive only	EFIM [61], d2HUP [4]	EFIM [61]
GMCHM [73], Hoa et al. (2021)	One	Twice	Utility-bin	Correlated HUIs	Depth-first search	TWU, local utility, Sub-tree utility	Positive only	FCHM [72]	EFIM [61]
C-HUIM and MaxC-HUIM [76], Duong et al. (2022)	–	MPUN-list	Quantitative	CHUIs and MaxHUIs	Prefix search tree	SPWUB, PSNonCHUB and LPSNonCHUB	Positive only	CHUI-Miner [78], EFIM-Closed [64], HMiner-Closed [74], and CHUI-Miner(Max) [75]	CHUI-Miner [78]

Table 10 Pros and cons of tree-based high utility itemsets mining algorithms

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
HYP tree [58], Hu et al. (2007)	An approximation method identifies the utility contribution.	IBM, Corporation	It can find segments of data through combinations of few items/rules	The built HYP tree is huge, and it is not memory-efficient.		It contributes towards predefined utility, objective function or performance metric.
CTU-Mine [47], Erwin et al. (2007)	A pattern-growth approach based on a compact data representation named CTU-tree for utility mining.	T10I5D50K, T10I10D10K	The pattern-growth, which avoids candidate generation-and-test, is suitable for dense datasets and long patterns.	The CTU-tree is complex and stores too much information, which may consume more memory. Two-phase algorithm is relatively faster as compared to CTU-Mine when the threshold value is high.	Future work could include the study of sampling based approximations. The thresholds for TWU could be determine from to user-specified utility to reduce the overestimation.	CTU-tree is the extension work of CT-tree.
IHUP [32], Ahmed et al. (2009)	A tree-based approach for incremental and interactive HUP mining.	Chain-store, Kosarak, Mushroom, Retail	The IHUP-tree is more compact than previous trees, and IHUP is significantly faster than IIDS and Two-phase.	It uses the TWU and produces too many HTWUIs in phase 1		The proposed three tree structures with the "Build once mine many" property for HUP mining in the incremental database.
HUC-Prune [48], Ahmed et al. (2011)	It adopts the pattern-growth approach to avoid the level-wise candidate generation process.	Chess, Connect, Kosarak, Mushroom, Retail, T10I4D100K, T10I4D200K, T10I4D600K	It efficiently deals with large dense dataset by avoiding the level-wise candidate generation-and test methodology. Highly scalable and memory efficient algorithm.	The upper-bound is high and the constructed tree is huge.		It find length-one candidates in one database scan, HUC-tree in second scan and actual high utility patterns in final third scan.
UMMI [49], Lin et al. (2012)	The proposed algorithm used two phases, namely, Maximal and Utility.	Chain store, T10I6D100K T10I6D1000K, T20I6D1000K	UMMI is 5, 3, and 7 times faster as compared to CTU-PRO, TWU-mining, and Two-phase respectively. On real dataset, it is 6 times faster than Two-phase.	It requires additional memory to build the HTP tree.		UMMI is linearly scalable with respect to the number of transactions.

Table 10 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
UP-Growth [14], UP-Growth+ [15], Tseng et al. (2013)	Proposed novel algorithm with compact data structure to efficiently discover HUIs.	Accidents, BMSWebView1, Chain-store, Chess, Foodmart, T10I6D100K	UP-tree is more compact than IHUP-tree, and the strategies are powerful to reduce the number of candidates. The enhanced UP-Growth+ significantly decreases the overestimated utilities of PHUIs and effectively reduces the number of candidates.	It consumes more time to recursively process all conditional prefix trees to generate the candidates.		The utility pattern-growth algorithm with more compressed utility pattern tree (UP-tree).
MU-Growth [59], Yun et al. (2014)	It uses the single-pass to construct the tree structure and and highly decrease the number of candidate itemsets.	Accidents, Connect, Foodmart, Retail, T10I4D100K to T10I4D1000K	It performs well on the datasets that have the large number of long transactions or low minimum utility thresholds are set.	High run-time consumption when the total number of transactions is increased in the database.		It decreases the number of candidates by reducing overestimated utilities and pruning the candidate itemsets.
CHUI-Mine [50], Song et al. (2014)	It is based on the pattern-growth approach to avoid the problem of the level-wise candidate generation-and-test strategy.	BMS-POS, Chess, Mushroom, T10I4D100k, T20I4D100k, T5N5D1M	CHUI-Mine is both efficient and scalable as compared to the state-of-the-art algorithms.	The faster tree-based algorithm IHUP is not compared with respect to memory usage.		It mines HUIs by dynamically pruning the CHUI-tree structure.
EFIM [51, 61], Zida et al. (2015, 2017)	EFIM utilizes a horizontal database representation for storing itemset information.	Accidents, BMS, Chess, Connect, Foodmart, Mushroom	It consumes less memory, and its complexity is roughly linear with the number of items in the search space. It performs well on tdense datasets.	Sometimes the recursive projection is time-consuming and uses a lot of memory. The transaction merging technique do not scale well in the case of sparse datasets.	EFIM may be further expanded for HUIM problem such as mining closed+ HUIs, generators of HUIs and on-shelf HUIs.	EFIM uses projection and transaction-merging techniques for reducing the cost of database scans.

Table 10 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
CUARM [62], Shao et al. (2015)	It proposes two factors, namely Contribution and Weight to select the combined patterns of both high utility growth and strong association.	Chain-store, Retail, T20I6D100K, C20D10K	It discovers patterns that consists the combination of both utility increment and high representativeness.	The performance of the proposed algorithm varies dramatically in each dataset.	More interesting pattern selection methods could be future research work.	It is the first method to select the patterns that have high utility without losing the representativeness, called strong association.
UP-Hist Growth [63] Dawar et al. (2015)	It is a pattern-growth recursive algorithm.	Chess, Mushroom, Retail	UP-Hist tree uses the histograms that improve the performance of utility-pattern mining algorithm.	Histograms use extra storage space to store the UP-hist tree.		UP-Hist Growth uses the UP-Hist tree data structure to mine high-utility patterns.
EFIM-closed [64] Fournier-Viger et al. (2016)	The proposed algorithm is a highly efficient approach for closed HUI mining.	Accident, BMS, Chess, Connect, Foodmart, Mushroom	EFIM-closed can be up to 71 times faster and consumes up to 18 times less memory than the state-of-the-art CHUD algorithm.	It scans the database repeatedly in case of dense databases.	Further research may include the EFIM-closed for top-k HUI mining, and high-utility sequence pattern and sequential rule mining.	EFIM-closed is inspired by a checking mechanism used in sequential pattern mining.
SIQ-tree [65] Ryang et al. (2016)	The proposed algorithm constructs SIQ-Tree with single scan and decreases the number of candidate patterns effectively and reduces the overestimation utilities.	Chain-store, Connect, Foodmart, Pumsb, T10I4D100K to T10I4D1000K	The proposed method with SIQ-Tree effectively reduces the number of candidates.	The proposed method requires more computation time as the item size is increased.	The algorithm could by applied to incremental and stream mining.	The restructuring method is based on DGN method.

Table 10 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
FIA [66], Qu et al. (2018)	Candidate tree-based algorithm is proposed for the fast HUI identification.	Accidents, Chain, Chess, Kosarak, Mushroom, Retail, T10I4D100K - T40I10D100K	It performs well when the candidate tree completely fits in the memory.	FIA does not consider the case when the tree does not fit completely in the memory.	The future work might include the fast identification of HUIs from candidate in disk.	BIA is combined with FIA, called BIA-FIA and then compared with the combination of BIA and UP-Growth+.
EHIL [68], Singh et al. (2019)	EHIL presents length constraint-based mining.	Chess, Mushroom	It consumes less run-time and memory usage. More suitable to the real-life applications.	It does not work well on dense datasets. The performance of the proposed algorithm depends on user-defined threshold, minimum length, maximum length and minimum utility.	EHIL may be further extended to closed itemsets mining as well as maximal itemsets mining.	It revises tree-based pruning strategies (Local Utility and Sub-tree Utility) according to the length constraints.
UT-Miner [70], Dawar et al. (2019)	The proposed algorithm mines HUIs in one phase without generating any candidate.	Accidents, Chess, Connect, Chain-Store, Kosarak, Mushroom, Retail	UT-Miner performs well across dense and sparse datasets.	HMiner and UT-Miner perform equally well on the Accidents dataset.		First one-phase tree-based algorithm to mine HUIs.
HUI-PR [71], Wu et al. (2019)	Two new stricter upper bounds are designed to reduce the computation time by refraining from visiting unnecessary nodes of an itemset.	Accidents, Chess, Connect, Foodmart, Mushroom, Retail	The proposed approach generates fewer branches of the search space than EFIM. The proposed method estimates fewer candidate itemsets than the state-of-the-art D2HUP and EFIM algorithms. It performs well on dense datasets.	HUI-PR reduces the run-time of local utility calculation by ignoring some transactions. However, it increases the complexity of the newly proposed upper bounds. It does not perform well on the sparse dataset as HUI-PR could not prune candidate itemsets effectively and wastes a lot of time scanning the dataset.	In the future, it can be extended to design the multi-threads approach in cloud computation model (such as a MapReduce framework) to reveal HUIs in a large-scale dataset.	The proposed approach is extended work of EFIM.

Table 10 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
GMCHM [73], Hoa et al. (2021)	A novel compact list and fast GMCHM algorithm are proposed to mine the compact HUIs to meet the need of decision makers.	Mushroom, Chess, Connect	The proposed algorithm computes the utilities of bigger HUIs without the need of re-scanning the dataset and without generating the candidates. It does not re-scan the dataset when there is change in minimum utility threshold.		The proposed algorithm could be further explored to optimize the memory consumption on the big datasets.	Bigger HUIs are obtained from the HUI of a sub-partition based on the intersection of set of transaction identification.
C-HUIM and MaxC-HUIM [76], Duong et al. (2022)	C-HUIM and MaxC-HUIM mine both CHUIs and MaxHUIs simultaneously.	Connect, Chess, Pumsb, Mushroom, Retail, Chain-store, PowerC, Kddcup99	The proposed algorithms are more than 100 times faster, more memory efficient, and have better scalability than the state-of-the-art algorithms.		The proposed theoretical results could be further extended to discover closed and generator patterns with other interestingness functions in quantitative pattern databases.	MPUN-list efficiently stores and calculates information about each itemset's utility and support.

CHUD [78] overestimates too many low utility of candidates that result in high memory consumption and low run-time. To address the issue, Wu et al. [78] proposed an EU-List (Extended utility-list) structure to maintain the utility information of itemsets in the transaction. EU-List efficiently calculates the utility of itemsets in memory without the original database scan. Further, CHUI-Miner (Closed^{+} High Utility Itemset mining without candidates) approach [78] is proposed that uses the divide-and-conquer method to find all the CHUIs without generating candidates. However, it consumes too much time to recursively process all the conditional prefix-trees.

The author [60] proposed the HUP-Miner algorithm to efficiently discover the HUIs. The partitioned utility-list structure is proposed to maintain the utility information at a granular partition level. Two novel pruning strategies, namely PU-Prune (Partitioned Utility Pruning) and LA-Prune (Look-ahead Utility Pruning) are proposed to limit the mining search space and hence improve the efficiency of the mining process. HUP-Miner performs well on sparse databases. However, it shows poor performance on dense databases. Moreover, the number of partitions is needed to explicitly set by the users.

To enhance the effectiveness of the mining process by pruning the search space using the length constraint, Fournier-Viger et al. [69] proposed FHM+ (Fast High utility itemset Mining+) algorithm to find the HUIs using length constraint. It is an extension work of FHM [17] with LUR (Length upper bounds Reduction). FHM+ reduces the upper bounds on the utility of itemsets using length constraints to prune the search space. The advantage of FHM+ is that the number of patterns is effectively decreased which improves the performance of the mining process. However, the detailed results are not shown to evaluate the efficiency of the LUR concept.

List-based methods [4, 17, 81] require a large number of comparison operations between two given items in the transaction and also need to construct the list for them. To overcome this limitation, Ryang et al. [86] proposed IMHUP (Indexed-list based Mining of High Utility Patterns) algorithm that is based on the IU-List (Indexed Utility-List) structure. IU-List effectively reduces the comparison operations when constructing the local lists to mine the HUIs. Moreover, RUI (Reducing upper-bound utilities in IU-Lists) technique is developed that decreases the search space by reducing the upper-bound utilities in IU-List. Further, CHI (Combining High utility patterns without constructing IU-List) technique is developed that efficiently generates the HUIs from the lists without the need for construction of a local IU-List when the lists consist only of information of the same revised transactions. The experimental results show that the IMHUP algorithm works better in the case of dense and sparse databases as compared to HUI-Miner [81] and FHM [17].

Traditional HUIM algorithms consist of weakly correlated items that lead to incorrect or useless decisions. To avoid this problem, Fournier-Viger et al. [87] proposed an FCHM algorithm to efficiently find the CHIs (Correlated High utility Itemsets) using the *bond* measure [88]. It integrates four strategies, namely DOS (Directly Outputting Single items), PSN (Pruning Supersets of Non-correlated itemsets), PBM (Pruning using the *Bond* Matrix) and AUL (Abandoning Utility-List construction early), to discover CHIs efficiently. The proposed algorithm is more than two orders of magnitude faster as compared to the FHM [17]. It discovers more than five orders of magnitude fewer patterns by pruning a large number of weakly CHIs and mining only CHIs in some cases. However, the detailed results are not shown in the case of PBM and AUL strategies.

EFIM [51] uses expensive sort operations to identify duplicate transactions in the database. To address this issues, Krishnamoorthy [18] proposed HMiner algorithm which utilizes CUL (Compact Utility-List) structure to efficiently store the utility information. It develops a virtual hyperlink structure that discovers duplicate transactions in the database. Further, it applies several pruning strategies (TWU-Prune, U-Prune, LA-Prune, C-Prune and EUCS-Prune) to efficiently mine the HUIs. The proposed algorithm achieves the improvement on the execution time that is ranged from a modest thirty percent to three orders of magnitude across several benchmark databases. Moreover, the requirements of memory consumption also show up to an order of magnitude improvement over the HUI-Miner [81], FHM [17], IMHUP [86] and EFIM [51]. HMiner performs well in the dense regions of both dense and sparse benchmark databases. However, in the case of longer transactions of moderate dense databases, it shows poor performance with regard to memory consumption and run-time.

The previous approaches [16, 17, 61] do not perform well on sparse databases. To address this issue, Peng et al. [82] proposed a one-phase mHUIMiner (modified HUI-Miner) algorithm which shows the best running time on the sparse databases. It incorporates the IHUP tree structure [32] to avoid unnecessary utility-list constructions into the original HUI-Miner algorithm [16]. The mHUIMiner is the fastest algorithm on sparse databases and has comparable performance on dense databases for the benchmark methods. Moreover, it performs efficiently despite the decrease in density. However, the execution time and memory usage are high as the input size increases, but it does not grow exponentially.

The algorithms [16, 17] require high memory and execution time. To overcome the problem, Duong et al. [89] proposed ULB-Miner (Utility-List Buffer Miner) to find the HUIs. They proposed an improved ULB (Utility-List Buffer) structure that efficiently stores and retrieves the utility-list and reuses the memory during the mining process. ULB constructs the utility-list segments in linear time. The proposed algorithm is up to ten times faster and consumes up to six times less memory as compared to HUI-Miner [16] and FHM [17]. Moreover, it achieves better performance on both sparse and dense databases. However, it cannot be directly applied to utility-lists stored in the ULB.

CHUI-Miner [78] suffers from the following drawbacks: (1) needs to conduct the costly join operations to evaluate the utility of each itemset. (2) a large number of candidates are generated to find the set of CHUIs. (3) applicable to an itemset only when its utility-list is fully constructed. (4) considers a large number of non-closed items that significantly reduce the performance of the mining process. To address these issues, Dam et al. [84] proposed the CLS-Miner algorithm to efficiently mine the CHUIs. The proposed algorithm integrates three pruning strategies, namely Chain-EUCP, LBP, and pruning by coverage that prune the search space before constructing the utility lists. The concept of coverage[6] is inspired by the definition of frequent closed itemsets in FIM [67]. Moreover, a pre-check method is also proposed that quickly determines whether an itemset is a subset of another itemset or not. It optimizes the operations of closure computations and subsumption checks that significantly reduce the time to discover the CHUIs. The proposed algorithm outperforms CHUD [20] and CHUI-Miner [78] on both dense and sparse benchmark databases. Moreover, it is linearly scalable in number of transactions and the number of items. However, it needs to store structures in memory for its pruning strategies, Chain-EUCP, and coverage.

The concept of MHUIs (Maximal High Utility Itemsets) [75, 90] is proposed to mine all the HUIs along with their utility values with/without the need for database scans again and requires less memory to store all the results. However, they either require two phases to mine MHUIs or are incomplete. Moreover, they generate a large number of candidates or mine compact forms of HUIs indirectly through CHUIs and remove the non-maximal patterns. To address these issues, Nguyen et al. [91] proposed CHUI-Mine (Maximal) algorithm that efficiently mines MHUIs from the transaction databases. It utilizes pruning techniques, namely EUCP and CUIP (Continuous Unpromising Item Pruning) that significantly prunes the search space to enhance the mining performance. The proposed algorithm significantly reduces the memory requirement and execution time to store the discovered patterns. However, it may generate unnecessary candidate patterns.

Closed and concise rule mining approaches take additional time to construct the lattice[7] to mine closed HUIs. To address this issue, Merugula et al. [92] proposed CG-Algorithm (Closed and Generator Algorithm) that extracts closed and generator HUIs from the same single closure check in the given database. It uses a hash-based data structure to maintain HUIs. The advantage of the CG-Algorithm is that it consumes less memory and low execution time for both closed and generator HUIs over the standard databases.

Many efficient algorithms [20, 64, 78, 84] provide concise profitable commodity combinations to managers. However, operators may need commodities, not only to generate generous profits but also to be purchased by the customers often. To resolve this issue, Wei et al. [93] proposed FCHUIM (Frequent Closed High Utility Itemset Mining) algorithm to find all FCHUIs (Frequent and Closed High Utility Itemsets). They proposed an efficient list structure, named TSL (Total Summary List) to reuse memory and fast access of information of items. A pre-check method is proposed that prunes the search space to reduce the non-closed HUIs efficiently. Moreover, a nested list structure is also used to quickly find FCHUIs in a large number of candidates. The proposed algorithm uses an upper-bound on the utility of items that works the same as the Z-element in [94]. FCHUIM significantly reduces the number of candidates. It shows high performance in the case of dense and sparse databases.

One-phase HUIM algorithms [4, 16, 17] are time-consuming and memory-consuming, especially on dense databases with long transactions. To address these issues, Shan et al. [95] proposed an efficient one-phase HUIM algorithm, named EHUIM-DS, based on a novel data structure. The data structure is used to reorganize the transaction database to obtain all HUIs effectively. It calculates the utility values with one or two scans of ITems Data (ITDs) instead of scanning the utility-list structures or the entire database. Moreover, it significantly reduces memory usage by using a depth-first search. Two upper bounds, namely extension utility and local transaction weighted utility are proposed to prune the search space from width and depth. The experimental results show that the proposed algorithm performs well as compared to the state-of-the-art methods, HUI-Miner [16], d2HUP [4], FHM [17], and UFH [83] in terms of the number of candidates, run-time, and memory usage on the sparse and dense databases.

Wu et al. [96] propose an efficent algorithm, named UBP-Miner (Utility Bit Partition Miner) to improve the utility-list construction process. A novel set of bit-wise operations is proposed called BEO (Bit mErge cOnstruction) to speed up the construction process. Besides, a novel data structure called UBP (Utility Bit Partition) is designed to support BEO. This structure is integrated into a novel UBP-Miner algorithm, which also applies several search space reduction strategies. Experimental results show that UBP-Miner is faster than several state-of-the-art algorithms such as HUI-Miner [16], HUP-Miner [60] and ULB-Miner [89] in terms of run-time, memory usage and scalability on benchmark

[6] The coverage is used to prune low utility itemsets and to quickly calculates the closure of itemsets.

[7] Lattice is an effective approach for data analysis and knowledge discovery to mine the association rules.

datasets. However, the proposed algorithm requires some additional memory.

The existing utility-list-based algorithms [16, 60, 89] are time-consuming and memory-consuming when they store the itemset information for the utility list. To solve this problem, Cheng et al. [97] proposed an efficient one-phase utility-list-based HUIM algorithm, named HUIM-SU to mine HUIs from the transactional dataset. A simplified utility list is designed where each record represents all the utilities of the transactions of an individual item. A construction tree is proposed to reduce the search space based on the simplified utility-list. Moreover, compressed storage is also proposed to reduce the memory usage of the construction tree. To further reduce the search space of the promising candidates, extension utility and local TWU utility are utilized to minimize the number of items. The experimental results show that the proposed algorithm performed better compared to the state-of-the-art algorithms HUI-Miner [16], HUP-Miner [60], FHM [17], and ULB-Miner [89] in terms of the number of candidates, memory usage, and execution time from dense and sparse datasets.

Summary

As discussed above, the utility-list-based HUIM algorithms outperformed level-wise and tree-based approaches on the benchmark databases in terms of efficiency, memory usage, scalability, etc. However, the utility-list-based approaches suffer from the following drawbacks: (1) need to perform the costly join operations between utility-lists of (k+1)-itemsets and k-itemsets that need a high amount of time; (2) suffer from high space and complexity; (3) either perform well on sparse databases or dense databases; (4) avoid promising itemsets which do not appeared in the database; (5) the run-time and memory usage is high as the input size increases; (6) do not scale well in the case of large databases. The detailed comparative summary of utility-list-based HUIM algorithms is shown in Table 11. The pros and cons of all the utility-list based HUIM algorithms are depicted in Table 12. The simplicity of the utility-list structure and the high performance of utility-list-based algorithms have led to the development of numerous utility-list-based algorithms for HUIM and variations of the HUIM problem such as closed high utility itemset mining [84], top-k high utility itemset mining [94, 99], high utility itemset mining in uncertain databases [100], high utility sequential pattern mining [101], and on-shelf high utility itemset mining [102], among others [17, 69, 103]. Although the introduction of the utility-list structure has been a breakthrough in the field of HUIM, however, the utility-list structures still have to be improved. Due to the wide applications of the utility-list structure in high-utility pattern mining, there is an important need to propose a more effective and efficient utility-list structure that can be constructed in linear time and can reduce memory usage.

3.4 Projection-based high utility itemsets mining

To overcome the drawbacks of the utility-list-based approaches, projection-based HUIM algorithms [13, 19, 42, 104] are introduced to recursively project the target items into the projected sub-databases. They have following advantages: (1) reduce the excessive candidate's generation; (2) process efficiently by using the prefix projection to decrease the size of projected sub-databases; (3) improve the efficiency of the mining process by using bi-level and pseudo-projection; (4) perform well on both sparse and dense databases at most support levels; (5) memory-efficient and scalable algorithms.

Two-phase algorithm [12] shows poor performance with dense databases and long patterns. To resolve this problem, Erwin et al. [79] proposed a CTU-PRO algorithm that discovers HUIs using the pattern-growth approach [57]. The CUP-tree (Compressed Utility Pattern tree) is a variant of the CTU-tree [47]. The proposed algorithm uses TWU [12] to prune the search space and discovers the HUIs by avoiding the re-scanning of the database. To extend the work [79], the authors developed an approach, CTU-PROL which mines the HUIs from large databases using the pattern-growth approach [57]. CTU-PROL performs well as compared to the benchmark algorithms, Two-phase [12] and CTU-Mine [47] on the dense and sparse databases at most support levels. However, the global CUP-tree does not fit completely in memory.

CTU-PRO algorithm [79] suffers from memory space. To address this issue, Lan et al. [105] proposed a novel PB (Projection-based) method that uses the indexing mechanism and a pruning strategy to efficiently find the HUIs. It uses a TC (Temporal Candidate) itemset table that quickly stores and obtains significant information on the values of the itemsets in the mining process. Later, this work is extended by [13]. The pruning strategy is based on the TWU [12] that reduces the number of unpromising candidates. The proposed algorithm [13] performs better as compared to the Two-phase [12] and CTU-PRO [79] about the number of candidates, run-time, and memory consumption. However, it generates too many redundant candidates.

One-phase HUIM approaches [16, 60, 61, 69] suffer from high costs concerning memory usage and run-time. To address these issues, Bai et al. [104] proposed SPHUI-Miner (Selective database Projection-based HUI mining algorithm) to enumerate all the HUIs. It uses a compact data format, named HUI-RTPL (High Utility Reduced Transaction Pattern List) that stores unique transactions and cumulative utilities of items in these transactions in the database. Two new upper bounds, namely, tup (Transaction Utility in Projection) and pu (Projection Utility) are proposed that

Table 11 An overview of key characteristics of utility-list-based high utility itemsets mining algorithms

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
HUI-Miner [81], Liu et al. (2012)	One	Once	Utility-list	HUIs	Depth-first search	TWU, Remaining Utility	Positive only	IHUP$_{TWU}$ [32], UP-Growth [14], UPGrowth+	–
d2HUP [4, 16], Liu et al. (2012, 2016)	One	Once	Utility-list, CAUL	HUIs	Depth-first search	Remaining utility upper-bound	Positive only	Two-phase [12], IHUP$_{TWU}$ [32], UP-Growth [14], HUI-Miner [81]	HUI-Miner [81]
FHM [17], Fournier-Viger et al. (2014)	One	Once	Utility-list, EUCS, Hash-map	HUIs	Depth-first search	Remaining utility upper-bound, EUCP	Positive only	HUI-Miner [81]	–
HUCI-Miner [85], Sahoo et al. (2014)	One	Once	Utility-list	HUIs, CHUIs, High utility generators (HG)	–	Remaining utility upper-bound, EUCP	Positive only	FHM [17]	FHM [17]
CHUI-Miner [78], Wu et al. (2015)	One	Twice	Extended Utility-List (EU-List)	CHUIs	Divide-and-conquer	TWU, Remaining utility based and EUCP	Positive only	CHUD [78], HUI-Miner [81]	HUI-Miner [81]
HUP-Miner [60], Krishnamoorthy (2015)	One	Once	Partitioned Utility-List (PUL)	HUIs	Partitioned Utility-List (PUL)	TWU Prune, U-Prune, PU-Prune, LA-Prune	Positive only	HUI-Miner [81]	HUI-Miner [81]
FHM+ [69], Fournier-Viger et al. (2016)	One	Once	Utility-list with EUCS	HUIs	Depth-first search	TWU and revised remaining utility	Positive only	FHM [17]	HUI-Miner [81]
IMHUP [86], Ryang et al. (2016)	One	Twice	IU-List	SHUP	TWU ascending order	TWU, RUI	Positive only	HUI-Miner [81], FHM [17]	HUI-Miner [81]
FCHM [87], Fournier-Viger et al. (2016)	One	Once	Utility-List	Co-related HUIs	Depth-first search	TWU, DOS, PSN, PBM, AUL	Positive only	FHM [17]	FHM [17]
HMiner [18], Krishnamoorthy (2017)	One	Once	Compact utility-list, Virtual hyper-link	HUIs	TWU ascending order	TWU-Prune, U-Prune, LA-Prune, C-Prune, EUCS-Prune	Positive only	EFIM [61], FHM [17], HUP-Miner [60], d2HUP [4]	HUI-Miner [81]
mHUIMiner [82], Peng et al. (2017)	–	–	Tree structure	HUIs	Descending order of TWU values	TWU, Remaining utility	Positive only	IHUP [32], FHM [17], HUI-Miner [81], EFIM [61]	IHUP [32]
ULB-Miner [89], Duong et al. (2018)	One	Twice	Utility-list buffer, EUCS	HUIs	Recursive depth-first search	TWU, EUCP, EA	Positive only	HUI-Miner [81], FHM [17]	HUI-Miner [81]

Table 11 continued

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
CLS-Miner [84], Dam et al. (2019)	One	Twice	Utility-list	CHUIs	Ascending order of TWU	Chain-EUCP, LBP, Coverage	Positive only	CHUD [78], CHUI-Miner [78]	CHUD [78]
CHUI-Mine (Maximal) [91], Nguyen et al. (2019)	One	Twice	EU-list	MHUIs & CHUIs	TWU	EUCS, EUCP, CUIP	Positive only	CHUI-Mine [78]	–
CG-Algorithm [92], Merugula et al. (2020)	–	–	Hash table	CHUI, HUG	HUIs in the order of their length	Index ID, Utility value, Frequency, Closed Flag, Generator Flag	Positive only	LHUI [98]	
FCHUIM [93], Wei et al. (2020)	One	–	Total summary list, Utility-list buffer	FCHUIs	Recursive depth-first search	Extension remaining utility	Positive only	CHUI-Miner [78], CLS-Miner [84]	–
EHUIM-DS [95], Shen et al. (2021)	One	Twice	TID	HUIs	Depth-first search	TWU, Extension utility, Local TWU utility	Positive only	EFIM [61], d2HUP [4], EFIM-Closed [64], CHUD [78], CHUI-Miner [78]	EFIM [61], d2HUP [4]
UBP-Miner [96], Wu et al. (2022)	–	Twice	UBP	HUIs	–	Suru-Prune, LA-Prune, PU-Prune and EUCS-Prune	Positive only	HUI-Miner [16], HUP-Miner [60] and ULB-miner [89]	HUI-Miner [16]
HUIM-SU [97], Cheng et al. (2022)	One	Once	Utility-list	HUIs	Depth-first search	Local TWU	Positive only	HUI-Miner [16], HUP-Miner [60], FHM [17] and ULB-Miner [89]	HUI-Miner [16]

Table 12 Pros and cons of utility-list-based high utility itemsets mining algorithms

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
HUI-Miner [81], Liu et al. (2012)	The one-phase model uses vertical database representation to mine high utility itemsets.	Accidents, Chess, Kosarak, Mushroom, Retail, T10I4D100K, T40I10D100K	It avoids the costly candidate generation and utility computation.	The join operations between utility lists of (k+1)-itemsets and k-itemsets are time-consuming. It suffers from high space and complexity.		It maintains an inverted-list data structure to mine HUIs without any candidate generation in one-phase only.
d2HUP [4, 16], Liu et al. (2012, 2016)	d2HUP transforms a horizontal database into a tree-based structure called CAUL and adopts a pattern-growth strategy to directly discover HUIs in the databases.	Chain Store, Chess, Foodmart, WebView1, T10I6D1M, T20I6D1M	It efficiently obtains the utility of each enumerated itemset and the upper-bound on utilities using the CAUL.	The tree structure and CAUL consume more memory. The algorithm is inefficient where the look-ahead strategy is of little use.	d2HUP could be further applied in high utility sequential pattern mining, parallel and distributed algorithms and their applications in big data analytic.	It directly discovers HUIs without maintaining candidates.
FHM [17], Fournier-Viger et al. (2014)	An improved version of HUI-Miner with a pruning strategy named EUCP. It uses vertical database representation to mine HUIs.	BMS, Chain-store, Kosarak, Retail	FHM reduces the number of join operations by up to 95% and is up to six times faster than the HUI-Miner.	It consumes slightly more memory than HUI-Miner and has poor performance on sparse datasets. It suffers from high space and complexity. It gives equal importance to all items in a database which cannot fully reflect the characteristics of real-world databases.	Other optimizations for itemset mining such as sequential pattern mining and sequential rule mining may be explored in the future research.	It maintains an inverted-list data structure to mine HUIs without any candidate generation in one-phase only.
HUCI-Miner [85], Sahoo et al. (2014)	HUCI-Miner is proposed to mine HUCIs with their generators.	Chess, Chain-store, Foodmart, Mushroom, Retail, T10I4D100K	The proposed algorithm achieves a great reduction in compressing the number of HUIs.			It integrates the concept of minimal generator into HUIs mining.
CHUI-Miner [78], Wu et al. (2015)	CHUI-Mine algorithm dynamically prunes the CHUI-tree.	Chain-Store, Chess, Connect, Foodmart, Mushroom, Retail	Two mechanisms, namely, shared variables and message passing, make the CHUI-tree more compact.	To process recursively all the conditional prefix trees is time-consuming.		To best of our knowledge, this is the first work that address the issue of mining CHUIs without candidate generation.
HUP-Miner [60], Krishnamoorthy (2015)	It uses novel partition utility-list data structure to efficiently mine HUIs.	Chain, Chess, Kosarak, Mushroom, Retail, T10I4D100K, T20I6D100K, T40I10D100K	It performs well on the sparse datasets. The proposed pruning strategies effectively improve the efficiency of high utility itemsets.	It is not efficient on the dense datasets. The number of partitions need to be explicitly set by the users.	The HUP-Miner can be extended to on-shelf and sequential utility pattern. The algorithm can be integrated with other pruning strategies to achieve better performance.	HUP-Miner is an improvement of HUI-Miner algorithm.

Table 12 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
FHM+ [69], Fournier-Viger et al. (2016)	The proposed algorithm mines HUIs while considering length constraints.	Chain-store, Mushroom, Retail	FHM+ efficiently discovers HUIs while considering length constraints.	Detailed results have not shown.	The concept of LUR could be incorporated in high-utility sequential rule mining.	It extends FHM algorithm for HUIM with a novel concept named LUR to reduce the upper bounds on the utility of itemsets using length constraints, and thus prune the search space.
IMHUP [86], Ryang et al. (2016)	It is based on the indexed list-based data structure that reduces the comparison operations when it builds local lists in the HUPM.	Accident, Chain-store, Chess, Connect, Retail, T10I4D100K-T10I4D1000K, T10N10KL1K - T10N100KL10K	It drastically reduces the join and comparison operations with the proposed IU-list data structure. It performs well on the sparse and dense datasets.			It mines HUPs without candidate generation. It uses two techniques, RUI and CHI, to generate HUPs without any construction of conditional lists
FCHM [87], Fournier-Viger et al. (2016)	It integrates the concept of correlation in HUIM to find profitable itemsets that are highly correlated using the bond measure.	Foodmart, Kosarak, Mushroom, Retail	The proposed algorithm is highly efficient and can prune a large number of weakly correlated HUIs.	Detailed results using PBM and ALU strategies are not shown.	FCHM could be considered in incremental HUIM. The novel algorithms could be developed for correlated HUIM based on the EFIM algorithm.	Mining CHIs using FCHM can be much faster than mining HUIs.
Hminer [18], Krishnamoorthy (2017)	It uses a compact utility-list data structure to efficiently store utility information.	Accident, BMS, Chain, Chess, Connect, Foodmart, Kosarak, Mushroom, Pumsb, Retail	It works well in the dense area of both dense and sparse datasets.	It has poor results on both run-time and memory consumption performance for the accidents dataset.	It could be easily applied to other HUI extensions such as HUIs with negative unit profits, top-k HUIs and on-shelf utility mining.	The closed and non-closed utility of an itemset is used to compactly store utility values.
mHUIMiner [82], Peng et al. (2017)	It modifies HUI-Miner that integrates the IHUP-tree structure into the original HUI-Miner algorithm.	Accident, BMS, Chain-Store, Chess, Foodmart, Kosarak, Mushroom, Retail	It performs well on the sparse datasets.	It works only for static dataset. The memory consumption and running time are high as the input size increases.	The future research could include the increment dataset and negative utility values.	It avoids unnecessary utility-list constructions in HUI-Miner by incorporating the tree structure.
ULB-Miner [89], Duong et al. (2018)	It uses the utility-list buffer to reduce the memory consumption and speed up the join operations.	Chain-store, Chess, Connect, Foodmart, Kosarak, Retail	ULB-Miner is 10 times faster and 6 times less memory consumption as compared to FHM and HUI-Miner. It efficiently works on both dense and sparse datasets.	ULB-Miner cannot be directly applied to utility-lists stored in Utility-list buffer.	The proposed structure could be adopted in various optimization techniques such as data stream to mine the itemsets.	It is based on the principle of buffering utility-lists to decrease memory consumption.

Table 12 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
CLS-Miner [84], Dam et al. (2019)	The proposed algorithm utilizes the utility-list structure to directly compute the utilities of itemsets without producing candidates.	Chain-Store, Chess, Connect, Foodmart, Mushroom, Retail	The proposed algorithm has linear scalability with respect to the number of items and the number of transactions.	CLS-Miner needs to store other structures in memory for its pruning strategies such as the EUCS and the coverage of items.	The distributed version of CLS-Miner can be run on cloud-computing platforms for to process the large datasets.	The CLS-Miner algorithm utilizes the utility-list structure and integrates the three proposed pruning strategies and the fast pre-check method.
CHUI-Mine (Maximal) [91], Nguyen et al. (2019)	Compact and lossless representation of HUIs.	Chain-Store, Mushroom, Retail	It significantly reduces the execution time and memory that is required to store the discovered patterns.	It may produce unnecessary candidates patterns.	The execution time and memory consumption of the algorithm could be further reduced by directly mine the MHUIs.	First one-phase algorithm to mine CHUIs.
CG-Algorithm [92], Merugula et al. (2020)	The proposed algorithm extracts Closed and Generator HUIs from the given dataset.	Accident, Chain-Store, Chess, Foodmart, Mushroom, Retail	It occupies less space for both closed and generators.		It is a post pruning algorithm of HUIs, therefore, it could be further improved by deriving them while determining HUIs.	Two operations, closed and generator are performed using a single condition that is either Super-set or Subset check.
FCHUIM [93], Wei et al. (2020)	The proposed algorithm is based on a total summary list structure for storing and retrieving utility lists without repeatedly scanning the database.	Accidents, Chain-store, Chess, Kosarak, Retail	It achieves great performance on the large databases. FCHUIM can efficiently reduce memory consumption.		FCHUIM algorithm could be further extended to frequent and maximal HUIs. It can be introduced to Hadoop or Spark platforms.	It consists both high-frequency and high-utility which allows managers to make decisions to generate more profits from goods that is based on customer's choice.
EHUIM-DS [95], Shen et al. (2021)	The novel data structure is designed by reorganizing the transaction database in find all HUIs effectively in the depth-first search process.	BMS, Chess, Connect, Foodmart, Pumsb, Retail	EHUIM-DS has better performance in memory usage and execution time than the benchmark algorithms.	EHUIM-DS has slightly poor performance than HUI-Miner algorithm on chess dataset.		The upper bounds, extensions utility and local TWU utility reduce the search space from width and depth and minimize the execution time.
UBP-Miner [96], Wu et al. (2022)	The proposed set of bit-wise operations, BEO, is used to improve the join operations of traditional utility-list based algorithms.	Chess, Mushroom, Pumsb, Connect, Retail, Accidents	Fastest list construction. Novel techniques could improve most utility-list based algorithms.	Some additional memory is required.	The proposed algorithm could be further extended in a distributed system and to process data streams.	The data structure, UBP, is integrated with proposed algorithm to reduce the search space.

Table 12 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
HUIM-SU [97], Cheng et al. (2022)	The proposed algorithm obtains all HUIs effectively and reduces memory usage in the depth-first search process.	BMS, Chess, Connect, Foodmart, Pumsb, Retail, Accident, Kosarak	The proposed algorithm shows better performance in terms of the number of candidates, memory usage, and execution time than the state-of-the-art algorithms.	The proposed algoriths have poor performance on some benchmark datasets.	More efficient search techniques could be further used to test the performance on real-world big data scenarios.	The construction tree and compressed storage are used to reduce the search space and memory usage.

effectively prune the search space. Furthermore, two novel data structures, namely SUP-List (Selective database projection utility-list) and Tail-Count list are developed that reduce the number of database scans, and database projection size and only store the relevant information to mine HUIs. The proposed algorithm performs better than benchmark algorithms [4, 15, 16, 60, 61, 69, 81] in computing time, memory consumption, and candidate generation from very condensed databases. Moreover, the proposed algorithm is scalable because it is independent of the order of processed projections which makes it suitable for distributed environment.

HUI-Miner [15], d2HUP [4], and EFIM [61] face the problem of high costs concerning run and memory usage. These algorithms either perform well on dense databases or sparse databases. To address these issues, Jaysawal et al. [19] proposed DMHUPS (Discovering Multiple High Utility Patterns Simultaneously) algorithm to effectively discover the HUIs. It utilizes IUData List (Item Utility Data list) structure to store the information of promising length-1 itemsets along with their positions in the transactions that are used to efficiently obtain the initially projected database. It simultaneously utilizes the utility and tighter extension upper-bound that are used to reduce the search space for multiple potential candidates. The proposed algorithm finds the multiple HUIs simultaneously to efficiently limit the search space. It uses transaction merging and look-ahead strategies to efficiently discover longer patterns. DMHUPS shows high performance as opposed to the UP-Growth+ [15], HUI-Miner [15], d2HUP [4], and EFIM [61] on both dense and sparse databases. Moreover, it is memory efficient and scalable. However, it uses the transaction merging strategy only for dense databases.

HUIPM [106] and FDHUP [107] only take the correlation factor by considering the co-occurring frequency in each transaction. To resolve this challenge, Gan et al. [5] proposed CoHUIM (non-redundant Correlated High Utility Itemset Mining) algorithm to find the CoHUIs (non-redundant Correlated High Utility Itemsets) by spanning the sub-projected databases of candidates. CoHUIs consider both the utility and positive correlation measures. Moreover, an SDC (global Sorted Downward Closure) property is developed to guarantee the global anti-monotonic to identify the complete set of CoHUIs. The proposed algorithm prunes a large number of unpromising candidates and accelerates the performance of the mining process. CoHUIM outperforms benchmark algorithms concerning the run-time and generated patterns. Moreover, it avoids excessive amounts of meaningless and redundant information. Furthermore, the number of obtained CoHUIs is more interestingness and valuable than that of HUIs.

Summary

High utility itemset mining algorithms represent the transaction database through a summarized data structure and mine high-utility itemsets by recursively constructing projected databases from the global data structure. The bottleneck of HUIM algorithms is the exponential search space for exploration and the time spent to construct the projected database during recursive calls. Projection-based algorithms are proposed to represent the transaction database as transactions only and utilize several techniques like closure, transaction merging, etc. to mine patterns efficiently in one phase only. However, projection-based approaches incur several problems: (1) generate too many candidate sets in some cases due to the use of TWU property; (2) do not perform well for high threshold values on few databases. The detailed summary of projection-based HUIM algorithms is shown in Table 13. Furthermore, the pros and cons of projection-based HUIM approaches are depicted in Table 14.

3.5 Miscellaneous approaches of high utility itemsets mining

In this section, several miscellaneous approaches for HUIM are discussed. These approaches use vertical bitmap [109, 110] and/or horizontal bitmap [109] representation to reduce memory usage and grow linearly with the size of the database.

Song et al. [109] proposed BAHUI (Bitmap-based Algorithm for High Utility Itemsets) algorithm which utilizes a divide-and-conquer approach and uses bitmap representation to mine the HUIs. It vertically uses a bitmap to visit the item-

Table 13 An overview of key characteristics of projection-based high utility itemsets mining Algorithms

Algorithm	Phase	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
CTU-PROL [42], Erwin et al. (2008)	Two	Multiple times	CUP-Tree	HUIs	Decreasing order of TWU value	TWU	Positive only	Two-phase [12], CTU-Mine [47]	Two-phase [12]
CTU-PRO [79], Erwin et al. (2007)	Two	Multiple times	CUP-Tree	HUIs	Bottom up approach	TWU,	Positive only	Two-phase [12], CTU-Mine [47]	Two-phase [12]
PB [13, 105], Lan et al. (2010, 2014)	Two	Multiple times	TC, Index table	HUIs	Indexing	TWU	Positive only	Two-phase [12], CTU-Mine [47]	Two-phase [12]
SPHUI-Miner [104], BAI et al. (2018)	One	Multiple times	HUI-RTPL, SPU-List, Tail-Count list	HUIs	Depth-first search	Upper-bound TWU, tup, pu	Positive only	UP-Growth+ [15], HUP-Miner [60], d2HUP [4], FHM [17], HUI-Miner [81], EFIM [61]	EFIM [61]
DMHUPS [19], Jaysawal et al. (2018)	–	–	IUData List	HUPs	Depth-first search	Utility and tighter extension upper-bound	Positive only	EFIM [61], d2HUP [4],	HUI-Miner [81], UP-Growth+ [15]
CoHUIM [5], Gan et al. (2018)	One	–	Utility-list	CoHUIs	Support, Ascending order of items	TWU, Global SDC of kulc	Positive only	FHM [17], HUI-Miner [81]	FHM [17]
SATCHUIM [108], Hidouri et al. (2021)	One	Twice	TID	Closed HUIs	Greedy approach	TWU	Positive only	HUI-Miner [81], d2HUP [4], FHM [17], UFH [83]	–

set lattice, while it horizontally uses a bitmap to calculate the real utilities of candidates. It uses efficient bit-wise operations and significantly reduces memory usage. Furthermore, it only stores the promising HUIs using maximal length and inherits the search from the maximal itemsets mining process. The proposed algorithm is efficient and scalable as compared to the benchmark methods [12, 48]. But it consumes a slightly high memory compared to HUC-Prune [48].

Song et al. [111] proposed IHUI-Mine (Index High Utility Itemsets Mine) algorithm to efficiently mine the HUIs. It uses subsume index [112] data structure for efficient frequent itemset mining that enumerates and prunes the search

Table 14 Pros and cons of projection-based high utility itemsets mining Algorithms

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
CTU-PRO [79], Erwin et al. (2007)	It mines HUIs using pattern-growth approach.	BMS-POS, Retail, T10N5D100K, T5N5DXM, UNI-FORM_10_50K	CTU-PRO performs better than the previous algorithms on both sparse and dense datasets at most support levels.	GlobalCUP-Tree is not completely fit in the memory.	Further research is needed to determine how the thresholds for transaction weighted utility may be varied from the user-specified utility to reduce this overestimate.	CTU-tree is an extension of CFP-Tree and a variant of CTU-Tree.
CTU-PROL [42], Erwin et al. (2008)	It mines HUIs from the large dataset using pattern-growth approach.	BMS-POS, Retail, T10N5D100K, T5N5DXM	It outperforms the state-of-the-art algorithms on the dense and sparse datasets.	In case of Retail dataset, Two-phase is slightly faster than CTU-PROL for high threshold values.	Future work could include the user-specific utility to reduce the overestimate of TWU value. The sampling-based approximation could further study to reduce the overestimation.	It uses TWU with pattern-growth that is based on the compact utility pattern tree data structure.
PB [13, 105], Lan et al. (2010, 2014)	The proposed algorithm uses the projection techniques and novel indexing structure to find the HUIs.	BMS-POS, T10I4N4KD200K	The proposed algorithm using projection techniques, speed-up the run-time and reduces the size of databases to derive the large itemsets.	It generates too many redundant candidates because it adopts the TWU as the upper-bound.	The proposed algorithm can be applied to data streams and supermarket promotion applications.	The concept of projecting tuples from an original database is applied to efficiently mine the HUIs
SPHUI-Miner [104], BAI et al. (2018)	A database projection method that has smaller size and unique data instances for enumeration the set of all HUIs.	Accident,p BMSWebView1, Chain-Store, Chess, Connect, Foodmart, Mushroom, Retail, Webdocs, Movie-Lens	It is suitable for distributed implementation. It finds the set of all HUIs from the very condense databases. The cost of database scan is reduced significantly.		Future work could include the high average utility itemsets for the incremental datasets using HUI-RTPL structure.	This is the first algorithm that performs selective database projection, not only on a data instance but also on dimensions for mining HUI.
DMHUPS [19], Jaysawal et al. (2018)	It simultaneously calculates utility and tighter extension upper-bound values for multiple promising candidates.	Accidents, Chess, Connect, Chain-Store, Kosarak, Mushroom, Retail	It finds multiple high utility patterns simultaneously and efficiently prunes the search space.	DMHUPS uses the transaction merging strategy only for dense datasets.		DMHUPS adopts transaction merging technique similar to EFIM algorithm to reduce the cost of database scans.
CoHUIM [5], Gan et al. (2018)	It extracts non-redundant correlated purchase behaviors while considering the utility and correlation factors.	Chess, Foodmart, Mushroom, Retail, T10I4D100K, T5I2N2KD100K	The proposed algorithm is suitable for real-time applications. It is memory efficient and scalable algorithm.		The novel data structures and pruning techniques could be further developed.	It is first work that considers correlation of items in HUI mining.

Table 14 continued

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
SATCHUIM [108], Hidouri et al. (2021)	The proposed algorithm is based on reduction to enumeration problems of propositional satisfiability.	Chess, Foodmart, Mushroom, Connect, Accidents, Kosarak, Chainstore	The proposed algorithm is efficient and scalable on the benchmark datasets in a reasonable time.	It is challenging task to set the appropriate minimum utility threshold.	A parallel version could be further proposed to improve the performance of SAT-based approach for enumerating (closed) HUIs. The proposed work can be further extended to enumerate the top-*k* (closed) HUIs mining.	The proposed algorithm makes use of symbolic Artificial Intelligence technique, propositional satisfiability, for efficiently enumerating all closed HUIs embedded in a transaction database.

space. Moreover, a discovery algorithm is used to effectively compute the TWU for the HUIs. Furthermore, the real utilities of candidates can be verified from the recorded transactions in the database using a bitmap representation. The proposed algorithm is 5.50, 5.76, 1.37, and 2.73 times faster as compared to the state-of-the-art algorithms, Two-phase [12], FUM [8], HUC-Prune [48], and UMMI [49], respectively about memory consumption, efficiency, and scalability.

Song et al. [110] proposed BPHUI-Mine (Binary Partition-based High Utility Itemsets Mine) algorithm to efficiently find the HUIs. It uses vertical bitmap representation to effectively represent the item expansion based on the binary partition in the transaction database which decreases the processing time and memory consumption. Several pruning strategies such as maximum itemset utility [59] and EUCP [17] are used to prune the search space to efficiently mine HUIs. Furthermore, support count is used to prune HTWUIs [50]. The proposed algorithm outperforms Two-phase [12], FUM [8], HUC-Prune [48], and UMMI [49], respectively concerning memory usage, efficiency, and scalability. It grows linearly with the size of the database. Moreover, it is robust to noise in dense databases.

Hidouri et al. [108] proposed an algorithm, named SATCHUIM (SAT (Satisfiability)-based Closed High Utility Itemset Mining) that makes original use of symbolic Artificial Intelligence technique, i.e. propositional satisfiability, for efficiently enumerating all closed high utility itemsets embedded in a transaction database. The authors use the SAT-based formulation to specify in terms of constraints the task of finding (closed) high utility itemsets over transaction databases. The proposed algorithm performs the tree-based backtrack search procedure, named DPLL (Davis-Putnam-Logemann-Loveland) solver, which is similar to the TWU measure, to prune the search space. The experimental results show that the proposed method performs well as compared to two baselines HUIM algorithm, namely EFIM [51], and d2HUP [4]. The proposed method also shows effectiveness in comparison to three baselines closed HUIM algorithms, namely EFIM-Closed [64], CHUD [35], and CHUI-Miner [78]. However, the number of found itemsets highly depends on the selected threshold values. It decreases when the utility threshold increases and vice versa. Furthermore, the number of patterns can be limited when the minimum utility threshold is large.

Dahiya et al. [113] proposed an optimization technique, named EAHUIM (Enhanced Absolute High Utility Itemset Miner), an advanced version of AHUIM (Absolute High Utility Itemset Miner) from big data. A neutral division approach has been proposed for dividing the search space among the computing nodes. It considers both the parameters, TWU, and length of the item in the subspace to assign it to a node. The storage complexity of the process is reduced by maintaining the subsets of transactions wherever possible instead of the complete transactions. The divide-and-conquer approach is used for extracting the data from the large dataset. Two pruning techniques, namely Absolute Local Utility (ALU) and Absolute Subtree Utility (ASU) are used to prune the search space to significantly improve the mining process. The experimental results prove the improvement of the proposed algorithm over the state-of-the-art algorithms EFIM-Par and PHUI-Miner [114] in terms of the total number of transactions, distinct items, the average number of items per transaction from the benchmark datasets.

Summary

Several miscellaneous approaches for HUIM have been discussed that address the issues of high memory consumption of the level-wise and tree-based algorithms. BAHUI uses horizontal and vertical bitmap representations to mine HUIs by using the divide-and-conquer method. IHUI-Mine uses the subsume index to mine HUIs. BPHUI-Mine is based on the binary partition to represent the transaction dataset. However, these algorithms could be further improved in terms of memory usage. The details of miscellaneous HUIM approaches are shown in Table 15. The pros and cons of all the miscellaneous approaches are depicted in Table 16.

Table 15 An overview of key characteristics of miscellaneous approaches of high utility itemsets mining algorithms

Algorithm	Database Scanning	Data Structure	Mining	Search Type	Pruning Strategy	Utility Value	The State-of-the-art algorithms	Extends
BAHUI [109], Song et al. (2014)	–	Bitmap	HUIs	Divide-and-conquer, Ascending order of TWU	TWU	Positive only	Two-phase [12], FUM [8], HUC-Prune [48]	Two-phase [12]
IHUI-Mine [111], Song et al. (2016)	–	Bitmap, subsume index	HUIs	Depth-first search	TWU	Positive only	Two-phase [12], FUM [8], HUC-Prune [48], UMMI [49]	Two-phase [12]
BPHUI-Mine [110], Song et al. (2016)	Once	Bitmap	HUIs	Ascending order of TWU	Transaction utility-list, Key support count	Positive only	Two-phase [12], FUM [8], HUC-Prune [48], UMMI [49]	Two-phase [12]
EAHUIM [113], Dahiya et al. (2022)	–	HashMap	HUIs	Divide-and-Conquer	ASU and ALU	Positive only	EFIM-Par and PHUI-Miner [114]	EFIM [64]

Table 16 Pros and cons of miscellaneous approaches of high utility itemsets mining

Algorithm	Description	Test Datasets	Pros	Cons	Future Direction	Remarks
BAHUI [109], Song et al. (2014)	The proposed algorithm mines HUIs with bitmap database representation.	Mushroom, Retails, T10I4D100k, T20I10D100k	Using bitmap compression scheme, BAHUI reduces the memory usage and makes use of the efficient bit-wise operations.	BAHUI consumes a slightly more memory than HUC-Prune on some datasets.	Efficient data structures could be further designed to meet the memory requirements.	BAHUI exploits a divide-and-conquer approach to visit itemset lattice by using bitmap vertically, while, it horizontally uses bitmap to calculate the real utilities of candidates.
IHUI-Mine [111], Song et al. (2016)	The proposed algorithm uses the subsume index data structure to efficiently mine the frequent itemsets mining and to prune the HUI search space.	BMS-POS, Chain-store, Mushroom, T15N150D100K, T10N100D1M	IHUI-Mine is 5.50, 5.76, 1.37, and 2.73 times faster than Two-phase, FUM, HUC-Prune, and UMMI, respectively.	The execution times of both IHUI-Mine and HUC-Prune are nearly constant on some benchmark datasets.		The subsume index is extended to discover the HUIs.
BPHUI-Mine [110], Song et al. (2016)	The proposed algorithm uses a vertical bitmap to represent the transaction database.	Mushroom, Retail, T15N50D100K, T30N100D200K	BPHUI-Mine grows linearly with the size of the dataset. It is more robust to noise on the dense datasets.	HUC-Prune has similar memory requirements to BPHUI-Mine when the utility threshold is high.		The proposed algorithm is a new itemset expansion method based on binary partitioning
EAHUIM [113], Dahiya et al. (2022)	EHUIM optimized the AHUIM algorithm from the large dataset by using the divide-and-conquer approach.	Chain-store, Kosarak, BMS2, Connect	EAHUIM outperforms AHUIM and other algorithms for time and space complexity.	It is a challenging task to find the optimal minimal utility threshold. The algorithm needs to be re-executed several time to obtain the desired threshold.	Top-k HUIM algorithms could be incorporated to get the desired number of k itemsets without the need of setting the threshold. Other features, such as mining with negative utilities and privacy-preserving techniques, could be further added to the proposed algorithm.	The proposed work enhances the performance of the AHUIM algorithm for extracting HUIs by embedding various optimizations.

3.6 Summary and discussions

We discussed the various approaches of HUIM for transaction databases such as level-wise, tree-based, utility-list-based, projection-based and miscellaneous. It has been observed that they have the following main advantages: (1) consider the usefulness and semantic information of an itemset; (2) extract the maximum profit for the businesses; (3) use efficient pruning strategies to significantly prune the search space; (4) significantly reduce the excessive database scans; (5) significantly reduce the number of candidates; (6) improve the performance of the mining process; (7) effectively reduce the memory usage and execution time; (8) highly scalable; (9) perform well on various benchmark databases. However, all these approaches are not applicable in the following domains: (1) suitable only for the static databases, not for incremental or dynamic databases; (2) use only positive utility value, however, the real-world businesses consist of both positive and negative utility value; (3) apply only for transaction databases, not for other databases eg. on-shelf, sequential databases, etc; (4) use only for simple mining processes, however, there exists many complex approaches such as sequential pattern mining, data stream, uncertain databases, etc.

Furthermore, we show the horizontal view of all the algorithms. We categorize the existing algorithms into two groups: high utility itemsets based algorithms and closed high utility itemsets based algorithms. Figures 3 and 4 show horizontal classification of HUIs and Closed HUIs mining algorithms, respectively.

Fig. 3 Horizontal view of HUIs mining algorithms for transactional databases

HUIs Mining Algorithms

Algorithm	**Year**
MEU [53]	2004
Two-phase [54, 12]	2005
UMining [46] and UMining_H [46]	2006
HURM [55]	2013
HYP tree [58]	2007
CTU-Mine [47]	2007
IHUP [32]	2009
UMMI [49]	2012
UP-Growth [14], & UP-Growth+ [15]	2010, & 2013
MU-Growth [59]	2014
CHUI-Mine [50]	2014
EFIM [61, 51]	2015, & 2017
CUARM [62]	2015
UP-Hist Growth [63]	2015
FIA [66]	2018
EHIL [68]	2019
UT-Miner [70]	2019
HUI-PR [71]	2019
GMCHM [73]	2021
C-HUIM and MaxC-HUIM [76]	2021
HUI-Miner [81]	2012
d2HUP [4, 16]	(2012, & 2016
FHM [17]	2014
HUP-Miner [60]	2015
FHM+ [69]	2016
IMHUP [86]	2016
FCHM [87]	2016
HMiner [18]	2017
mHUIMiner [82]	2017
ULB-Miner [89]	2018
EHUIM-DS [95]	2021
UBP-Miner [96]	2022
HUIM-SU [97]	2022
CTU-PRO [79], & CTU-PROL [42]	2007, & 2008
PB [105, 13]	2010, & 2014
SPHUI-Miner [104]	2018
DMHUPS [19]	2018
CoHUIM [5]	2018
BAHUI [109]	2014
IHUI-Mine [111]	2016
BPHUI-Mine [110]	2016
EAHUIM [113]	2022

Fig. 4 Horizontal view of Closed-HUIs mining algorithms for transactional databases

CHUIs Mining Algorithms

Algorithm	Year
OOApriori and top-K closed [52]	2003
IIDS, FUM and DCG+ [8]	2008
AprioriHC, AprioriHC-D and CHUD (extended) [20]	2015
HUC-Prune [48]	2011
EFIM-closed [64]	2016
C-HUIM and MaxC-HUIM [76]	2022
HUCI-Miner [85]	2014
CHUI-Miner [78]	2015
CLS-Miner [84]	2019
CHUI-Mine (Maximal) [91]	2019
CG-Algorithm [92]	2020
FCHUIM [93]	2020
SATCHUIM [108]	2021

4 Other high utility itemsets mining algorithms

In this section, we briefly discuss the HUIM approaches for other type of databases, including on-shelf HUIM, HUIM from sequential, uncertain, temporal, incremental, and other databases, databases with negative utility values, data stream mining, periodic mining, and privacy-preserving mining.

4.1 On-shelf high utility itemsets mining

Most of the HUIs remain undiscovered by using the existing traditional HUIM algorithms because the exhibition periods of all items are different in real-time applications. Hence, it may be biased when the items are not always on-shelf. On-shelf utility mining has recently received interest in the data mining field due to its practical considerations. On-shelf utility mining considers not only profits and quantities of items in transactions but also their on-shelf time periods in stores. Lan et al. [33] proposed TP-OHUI (Two-phase algorithm for mining On-shelf High Utility Itemsets in temporal databases) that efficiently and effectively mines the high on-shelf utility items. However, it works only in the case of positive utility. To resolve this challenge, Lan et al. [115] proposed an efficient three-scan mining approach, named TS-HOUN that efficiently discovers the high on-shelf utility itemsets with negative profit from the temporal databases. An effective itemset generation method is developed to avoid generating a large number of redundant candidates and to effectively reduce the number of data scans in mining. Another efficient algorithm, named FOSHU (Faster On-Shelf High Utility itemset miner) [116] is proposed to mine HUIs while considering on-shelf time periods of items, and items having positive and/or negative unit profit. The experiments show that FOSHU can be more than 1000 times faster and use up to 10 times less memory than the state-of-the-art algorithm TS-HOUN [115]. Dam et al. [102] proposed KOSHU (fast top-K On-Shelf High Utility itemset miner) algorithm that mines the top-k high on-shelf utility items having positive and/or negative unit profits while considering the on-shelf time periods of the items. KOSHU introduces three novel strategies, named efficient estimated co-occurrence maximum period rate pruning, period utility pruning and concurrence existing of a pair 2-itemset pruning to reduce the search space. KOSHU also incorporates several novel optimizations and a faster method for constructing utility-lists. Zhang et al. [117] propose two methods, OSUM (On-shelf utility mining) of sequence data (OSUMS) and OSUMS+, to extract on-shelf high-utility sequential patterns. For further efficiency, several strategies are designed to reduce the search space and avoid redundant calculation with two upper bounds time prefix extension utility (TPEU) and time reduced sequence utility (TRSU). In addition, two novel data structures were developed for facilitating the calculation of upper bounds and utilities. OSUMS may consume a large amount of memory and is unsuitable for cases with limited memory, while OSUMS+ has wider real-life applications owing to its high efficiency.

4.2 High utility itemsets mining from sequential databases

The problem of mining high utility sequences aims at discovering subsequences having a high utility (importance) in a quantitative sequential database. High utility sequence mining has been applied in numerous applications. It is quite challenging task to solve this problem due to the combinatorial explosion of the search space when considering sequences, and because the utility measure of sequences does not satisfy the downward-closure property used in pattern mining to reduce the search space [118]. Various extensions of the HUSP problem have been studied such as to hide high utility sequential patterns in databases to protect sensitive information [119] and discovering high-utility sequential rules [119]. Yin et al. [120] propose an efficient algorithm, named USpan to mine for high utility sequential patterns from the sequential database. A lexicographic quantitative sequence tree is designed to extract the complete set of high utility sequences and design concatenation mechanisms for calculating the utility of a node and its children with two effective pruning strategies. The experimental results show that USpan efficiently identifies high utility sequences from large scale data with very low minimum utility.

4.3 High utility itemsets mining from uncertain databases

The traditional mining algorithms are designed to mine the high frequent or high utility patterns by considering support and utility individually from the uncertain datasets. They are not designed to mine the required information from the uncertain datasets which consider both measures (utility and uncertainly) together as a multi-objective optimization problem. The utility measure is considered as an semantic method to access the value of an pattern, on the other hand, the uncertainly measure is considered as an objective method to access the reliability and existence of a pattern. It is a non-trivial tasks to consider both measures to mine HUIs from uncertain datasets. The reason is that both measures are conflicting with each other, thereby, resulting in the useless extracted patterns. Lin et al. [100] propose a a novel framework, named PHUIM (Potential High-Utility Itemset Mining) in uncertain databases, to efficiently discover not only the itemsets with high utilities but also the itemsets with high existence probabilities in an uncertain database based on the tuple uncertainty model. The PHUI-UP algorithm (Potential High-Utility Itemsets Upper-Bound-based mining algorithm) is first presented to mine PHUIs (Potential High-Utility Itemsets) using a level-wise search. Since PHUI-UP adopts a generate-andtest approach to mine PHUIs, it suffers from the problem of repeatedly scanning the database. To address this issue, a second algorithm named PHUI-List (Potential High-Utility Itemsets PU-list-based mining algorithm) is also proposed. This latter directly mines PHUIs without generating candidates, thus greatly improving the scalability of PHUI mining. Lin et al. [121] propose an efficient algorithm, named MUHUI (Mining Uncertain High-Utility Itemsets), is proposed to efficiently discover PHUIs in uncertain data. Based on the PU-list (Probability-Utility-list) structure, the MUHUI algorithm directly mines PHUIs without generating candidates, and can avoid constructing PU-lists for numerous unpromising itemsets by applying several efficient pruning strategies, which greatly improve its performance. MUHUI algorithm scales well when mining PHUIs in large-scale uncertain datasets. Ahmed et al. [122] propose the multi-objective evolutionary approach to discover the high expected utility patterns mining (MOEAHEUPM) framework in a limited time-period from the uncertain database. The proposed algorithm considers the utility and uncertainty simultaneously to extract the set of non-dominated high expected utility patterns (HEUPs) based on the evolutionary computation from the uncertain environment. The proposed algorithm does not need the prior knowledge (minimum utility threshold and minimum uncertain threshold) to discover the information. But, instead of prior knowledge, it mines more meaningful and unique non-dominated patterns to meet the decision.

4.4 High utility itemsets mining from temporal databases

The existing traditional HUIM algorithms generate too many candidates that make it difficult for the user to identify useful items from the large static databases. To resolve this issue, several algorithms are proposed in data stream mining. Chu et al. [123] proposed THUI-Mine (Temporal High Utility Itemsets Mine) algorithm that mines the temporal[8] HUIs from the data stream. The proposed algorithm identifies the temporal HUIs with fewer candidate-generated itemsets and has high performance. Shie et al [90] proposed a GUIDE (Generation of maximal high Utility Itemsets from Data strEams) algorithm that generates compact and insightful items which are high utility as well as maximal from the data stream. Ryang et al. [124] proposed SHU-Grow (Sliding window-based High Utility Grow) algorithm and SHU-Tree (Sliding window-based High Utility Tree) that efficiently mine the HUIs from the continuous data stream. They also present two techniques, namely RGE (Reducing Global Estimated utilities) and RLE (Reducing Local Estimated utilities) that significantly prune the search space and candidates by decreasing the overestimated utilities.

4.5 High utility itemsets mining from incremental databases

The traditional HUIM algorithms are proposed to handle static databases. However, in real-time applications, the transactions are gradually inserted, deleted, or modified in the database. When new transactions occur in the database, new items may emerge and old items become irrelevant. The conventional HUIM approaches run in batch mode only. When they apply to extract data from the updated database, they need to execute from scratch which ignores the previous results and is also time-consuming. In the recent past, several efficient IHUPM (Incremental High Utility Itemsets Mining) algorithms are proposed to handle the inserted transactions in the updated database.

Gan et al. [27] provide a comprehensive survey of incremental HUIM. The authors have comprehensively reviewed ten incremental HUIM algorithms which mainly fall into the following three groups: (1) Apriori-based; (2) tree-based; (3) utility-list-based. For details, the readers may refer to the survey paper on incremental HUIM by [27].

[8] The temporal HUIs are the itemsets whose support is larger than a pre-specified threshold in current time window of the data stream.

Now, we briefly discuss several incremental HUIM algorithms which are not covered by [27]. Lee et al. [125] proposed PIHUP (Pre-large Incremental High Utility Patterns) algorithm that is based on the pre-large concept to effectively discover the HUIs in the incremental database. It uses PIHUP$_L$-tree (PIHUP Lexicographic tree) structure to find the patterns fastly. The proposed algorithm needs only one scan to process the dynamic data. It effectively reduces the redundant operations and memory space as compared to the PRE-HUI algorithm [126]. However, the generated candidate patterns need to maintain the anti-monotone property [1].

Dam et al. [127] proposed a single-phase IncCHUI (Incremental Closed High Utility Itemset miner) that mines closed HUIs from the incremental databases using the incremental utility-list structure. The incremental utility-list structure stores the information of all single items both in the original database and the added transactions. The proposed algorithm constructs the lists of single items by scanning the original database or updated section only once. It uses the CHT (Closed Hash Table) to store the discovered closed HUIs. The proposed algorithm is highly scalable concerning the sizes of the input databases as opposed to the benchmark algorithms. Moreover, it is efficient in memory usage and execution time.

Nguyen et al. [128] proposed a modified version of the EFIM algorithm [51], named MEFIM (Modified EFficient high utility Itemset Mining) by adding the capability to handle the dynamic databases. The authors proposed an optimized version of the MEFIM algorithm, named iMEFIM that uses a novel structure P-set. The P-set structure reduces the number of transaction scans and improves the mining process. The proposed algorithm performs well as compared to the benchmark algorithms on dynamic databases regarding memory usage and run-time. The iMEFIM scans 26 percent fewer transactions than MEFIM. However, the MEFIM algorithm scans the database repeatedly to compute the utility, local utility, and sub-tree utility of each item. The iMEFIM algorithm consumes high memory because of the P-set structure.

Liu et al. [129] proposed an Id2HUP+ (Incremental Direct Discovery of High Utility Patterns) algorithm which adopts a one-phase paradigm of d2HUP [4, 16] by improving relevance-based pruning and upper-bound based pruning and introducing quick merge of identical transactions. The proposed algorithm proposes niCAUL (newly improved Chain of Accurate Utility-Lists) structure to quickly update the dynamic databases. Two pruning strategies, namely absence-based pruning and legacy-based pruning are proposed for incremental mining. The proposed algorithm is up to one-to-three orders of magnitude more efficiently as compared to the benchmark algorithms.

Yun et al. [130] proposed IIHUM (Indexed-list based Incremental High Utility Pattern Mining) algorithm to discover the HUIs from the incremental databases. An IIU-List (Incremental Indexed Utility-List) structure is proposed in a list form to discover the HUIs without any candidate generation. Furthermore, restructuring and pruning techniques are suggested to efficiently process the incremental data. The proposed algorithm efficiently mines the HUIs from incremental databases as compared to the benchmark algorithms.

4.6 High utility itemsets mining from other databases

Traditional HUIM methods deal with positive utility only, however, real-world applications consist of negative utility as well. For example, an outlet may sell items at the loss to cover the cost of the other items in the store or promotion of certain items. The traditional HUIM approaches do not address this issue. In the past decade, several HUIM with negative utility approaches are proposed to mine HUIs with negative utility. Singh et al. [30] provide a comprehensive survey of HUIM with negative utility. This survey includes twelve papers that mainly fall into the following three groups: (1) level-wise; (2) tree-based; (3) utility-list-based. For details, the readers may refer to the survey paper of HUIM with negative utility value by [30]. Now, we briefly discuss several HUIM algorithms for the negative utility which are not covered by [30]. Singh et al. [131] proposed EHNL (Efficient High utility itemsets mining with Negative utility and Length constraints) algorithm that uses length constraints to find HUIs. This is the first work to mine HUIs from the database having negative utility values and considering length constraints. It introduces a minimum length constraint to remove the excessive very small itemsets. Furthermore, it also uses maximum length constraint to restrict the too longer itemsets. EHNL utilizes database projection and transaction merging techniques to reduce the cost of database scans. Moreover, it utilizes a sub-tree-based pruning strategy, based on EHIN [132], to reduce the search space and accelerate the mining process. The proposed algorithm efficiently mines the HUIs with low memory consumption for real databases.

In the conventional HUIM methods, the actual utility values of itemsets are increased along with the increase of their length. This makes it difficult to analyze whether an itemset is better than its subsets or not. To resolve this issue, HAUIM (High Average-Utility Itemsets Mining) algorithms [133] are proposed that mine the average HUIs from the databases. The average utility of an itemset is the summation of all utility values of all items in the appeared transaction divided by the total number of items in the transaction. However, they spend a high amount of time and generate excessive candidates. To address these limitations of traditional HAUIM algorithms, in recent years, more advanced HAUIM algorithms [134, 135] are proposed that perform well in terms of the number of join operations, execution time, memory consumption, and scalability. However, they work only on static databases. To address this issue, incremental HAUIM algorithms [136, 137] are proposed that efficiently mine the average HUIs on the dynamic database in a significant way. Singh et al. [25] provide a comprehensive survey of high average-utility itemsets mining algorithms that can serve as the recent advancement and research opportunities in the area of data mining. For further details, the readers may refer to the paper [25] for the new technological advancements in the domain of HAUIM.

The traditional HUIM algorithms spend more computation time searching for an item from large databases. The issue can be resolved by using evolutionary computation. Two genetic algorithms, namely HUPE$_{umu}$-GRAM (High Utility Pattern Extracting using Genetic algorithm with Ranked Mutation using Minimum Utility threshold) [138] and HUPE$_{wumu}$-GRAM (High Utility Pattern Extracting using Genetic algorithm without Ranked Mutation using Minimum Utility threshold) [138], are proposed to find HUIs with and without minimum utility threshold respectively. But, both these algorithms are inadequate to obtain promising HUIs because the computations are more if the distinct items are very large in the database. Another evolutionary algorithm based on particle swarm optimization, named Binary-Particle Swarm Optimization (BPSO) [139], is proposed to obtain the optimization solutions from the large search space. However, it requires a large number of computations to obtain high accuracy. Another solution to the computational evolutionary is to use the fuzzy theory system to handle the transaction databases based on the crisp sets. Recently, Kumar et al. [140] provide a comprehensive survey on soft computing based on HUIM that includes evolutionary computation and fuzzy-based approaches. For more information, the readers may refer to the paper [140] to get the new advancement and research opportunities in the area of soft computing using HUIM.

The traditional HUIM algorithms utilize the single minimum utility threshold to discover the HUIs. But, in real-time applications, these approaches are unrealistic because it is hard to develop efficient strategies for businesses by using these methods. To overcome this problem, many HUIM algorithms with multiple minimum thresholds [31, 141] are extensively proposed that specify multiple minimum utility thresholds for each item to identify more specific and useful HUIs. These approaches generate more benefits as compared to the HUIM algorithms. The HUIM with multiple minimum utility thresholds can be applied to expert intelligent systems to make more efficient decisions or strategies. However, these approaches consume a high amount of memory.

The traditional HUIM methods do not consider the period constraints and ignore the timestamps of the transactions. Hence, these methods discover profitable HUIs but they seldom occur in the transactions. To address this issue, several algorithms are proposed that efficiently and effectively mine the complete set of periodic HUIs while considering the period constraint and pruning a large number of non-periodic items from the large databases [142]. These methods may provide significant, reliable, and effective solutions in real-time applications.

HUIM algorithms are vulnerable to privacy issues. To address this issue, several privacy-preservation HUIM algorithms are proposed to hide sensitive HUIs [143]. These algorithms, not only generate the high profitable HUIs, but also capable of privacy-preserving the secure or private HUIs. These methods can be applied to intelligent privacy-preserving approaches in the industry environment.

4.7 Summary

As discussed above, the on-shelf HUIM algorithms efficiently and effectively mine the high on-shelf utility itemsets. The temporal HUIM algorithms efficiently mine the HUIs over a time period from the data stream. The incremental HUIM approaches are proposed to efficiently discover the HUIs in the updated databases. HUIMs with negative utility efficiently work to discover the HUIs with negative profit. Average HUIM algorithms effectively discover the HUIs along with the increase of their length. The HUIM algorithms with multiple minimum utility thresholds are used to identify more specific and interesting items that make it easy for businesses to reach more efficient decisions. The periodic HUIM mines the HUIs while considering the periodic constraint. The privacy-preserving HUIM algorithms are capable to hide sensitive HUIs. In short, these approaches perform well in the real world environment.

5 Databases and open-source resources

5.1 Databases

In this survey, the HUIM algorithms for transactional databases consider various real-world [144–146]. A brief description of these real-world is given that are publicly available in the literature. These algorithms consist of dense, sparse, mixed, short transactions, moderate transactions, and long transactions databases. Dense databases have fewer items and longer transactions as compared to sparse databases. Dense databases are generated from games like Chess and species of mushroom that have very few items and longer transaction's length. Sparse databases are generated by retail giants like Walmart, Amazon, etc. that sell millions or billions of products, but a customer usually purchases a few products only. Accident database anonymized traffic accident data. BMS-POS database contains several year' worth of point-of-sale data from a large electronics retailer. The real database BMSWebview-1 is taken from [147]. This database was used in KDD CUP 2000. It contains click-stream data from e-commerce. The Chess database is prepared based on UCI chess. Connect database is prepared based on the UCI connect-4. The Foodmart database consists of customer transactions from a retail store. The database includes 1,112,949 transactions with quantities, 46,086 items, an average transaction length of 7.2, and a utility table. The Mushroom database is prepared based on the UCI mushrooms. OnlineRetail is transformed from the Online Retail database. Pumsb database consists of census data for population and housing. Retail database is the customer transactions from an anonymous Belgian retail store. WebDocs is a real-life transactional database that is available to the data mining community through the FIMI repository. The whole collection contains about 1.7 million documents, mainly written in English, and its size is about 5 GB. It is stored, obtained, and transformed from SQL-Server 2000. Kosarak is a very large database containing 9,90,000 sequences of click-stream data from a Hungarian news portal. The database was converted into SPMF format using the original data. Grocery chain store were received from U-MineBench version 2.0 [146]. The characteristics of all the real-world databases are shown in Table 17.

5.2 SPMF open source framework

SPMF [148] is an open-source data mining framework. It offers more than 254 data mining algorithms. These algorithms cover 14 different categories, including sequential pattern mining, sequential rule mining, sequence prediction, itemset mining, episode mining, periodic pattern mining, graph pattern mining, high-utility pattern mining, association rule mining, stream pattern mining, clustering, time series mining, classification, and text mining. It is written in Java, specialized in pattern mining. It is distributed under the GPL v3 license. It offers several resources such as documentation with examples of how to run each algorithm, a developer's guide, performance comparisons of algorithms, datasets, an active forum, a FAQ, and a mailing list. SPMF presents various kinds of databases for data mining. It also presents database generators for a synthetic transaction database, sequence database, and sequence database with timestamps. It offers a toolbox for calculating statistics about a transaction database with utility information, and about a sequence database. The toolbox can be used to convert a sequence database to a transaction database and vice-versa.

6 Research opportunities and future directions

In recent decades, there are tremendous growth in the field of HUIM. However, there are still lots of challenges with the traditional approaches regarding memory consumption, run-time, scalability and, many more. Hence, there is wide scope for improvement. In this section, we highlight some key research opportunities and future research areas in high utility itemsets mining.

6.1 Compact and concise high utility itemsets mining

Although several algorithms [20, 35, 84] integrate the utility mining and closed itemsets. However, there is still wide scope for the integration of HUIM algorithms with other compact representations [36]. This is an interesting research problem that may be quite useful to further reduce the redundant itemsets from the databases. Moreover, the closed HUIM approaches consume high memory and execution time in the case of dense databases depending on the minimum utility threshold. Hence, efficient algorithms could be developed by introducing novel data structures, pruning strategies, and constraints.

6.2 Dynamic complex data

Another promising research opportunity could be the extension of HUIM algorithms in the area of dynamic complex data, for example, uncertain, dynamic sequential, incremental, and stream data. There is a wide range of scope for improvement so that the algorithms can efficiently handle the dynamic complex data to mine HUIs.

6.3 Constraint based high utility itemsets mining

Although there are lots of efficient HUIM algorithms, the user is mainly interested in the longer and better actionable candidates. The action-ability of HUIM algorithms is useful to decide for the users to increase utility. Several algorithms are proposed that reduce the total number of generated itemsets based on the constraints on the resulting rules. Therefore, the users are more interested in constraint HUIM algorithms. Pei et al. [149] present many constraints for FIM. It can be pushed into HUIM as well. Moreover, length-based HUIM algorithms play a significant role in constraint-based mining that can remove a lot of small itemsets and generate more interesting and actionable high-utility items. Although, in [69], the HUIM algorithm with length thresholds is proposed. But, there are still lots of research opportunities to discover more efficient algorithms to push the constraints as deep as we can.

Table 17 Characteristics of Real-World Databases [144–146]

Database	Transactions	Number of Items	Average Transaction Length	Maximum Length	Density (%)	Type
Accidents	3,40,183	468	33.8	51	7.2239	Moderately dense, moderately long transactions
BMS	59,601	497	4.8	267	0.5052	Sparse, short transactions
BMS-POS	517,255	1657	6.5	164	…	Mixed
BMSWebView1	59,601	497	2.51	267	–	Sparse
Chainstore	11,12,949	46,086	7.3	170	0.0158	Very sparse, short transactions
Chess	3,196	75	37	37	49.3333	Dense, long transactions
Connect	67,557	129	43	43	33.3333	Dense, long transactions
E-commerce	17535	3803	15.4	–	–	–
Foodmart	4, 141	1, 559	4.4	14	0.2838	Very sparse, moderately short transactions
Kosarak	9,90,002	41,270	8.1	2498	0.0196	Very sparse, moderately short transactions
major grocery chain store	1,112,949	46,086	7.2	–	–	Real-world market data
Mushroom	8,124	119	23	23	19.3277	Dense, moderately long transactions
OnlineRetail	541,909	2 603	4.37	–	–	–
Pumsb	49,046	2113	74	74	3.5021	Dense, very long transactions
Retail	88,162	16,470	10.3	76	0.0626	Very sparse, moderately short transactions
Webdocs	1,692,082	5,267,656	71.47	–	–	Sparse, very large

6.4 High utility itemsets mining with big data

Big data deals with Grid computing, Multi-core computing, MapReduce, Graphics Processing Units (GPUs), Spark framework, and Apache Hadoop. The HUIM approaches could be explored to mine the HUIs from the large databases in the area of distributed and parallel algorithms, and their applications in big data analytics, similar to those proposed in [150]. Previous works [151, 152] explore big data and various operations on big data. The multi-threads approach in cloud computation such as the Map-Reduce framework may be further investigated to find the HUIs in large databases. Furthermore, the distributed version of the concise mining algorithm [84] could be further developed to run on cloud computing platforms like Hadoop or Spark platforms and the TensorFlow system, to process very large databases. Hence, there are lots of research opportunities to work on the scalability issue of HUIM algorithms from constraint-based to more collaborative and hybrid models in the future.

6.5 Other problems

The latest preprocessing techniques improve the completeness, consistency, and precision of the algorithms. The previous works [153, 154] utilize the latest preprocessing techniques with HUIM algorithms. This can be further investigated to develop more efficient algorithms. Few researchers combine the association rule mining and classification models in the past [155]. The integration between classification and utility mining could be further investigated. The sampling-based approximation [156] could be investigated to mine the HUIs that significantly make the computation less expensive. There are many research possibilities to extend the variations of the HUIM problems and reuse them in various areas such as high utility association rules [157], incremental HUIs [32, 158], top-k HUIs [94], on-shelf HUIs [116] and periodic HUIs [103]. The HUIM approaches may be further investigated to optimization methods involving the HUIs in data-stream.

7 Conclusions

High utility itemsets mining (HUIM) is a powerful technique for discovering interesting patterns in transactional datasets. The HUIM algorithms have various applications including market basket analysis, web-click stream analysis, web mining, cross-marketing, gene regulation, mobile commerce environment, and e-commerce. In this survey, we presented highlights of the HUIM approaches for transactional databases that might be useful for the readers to select proper methods for their applications. This work is quite appropriate to find suitable algorithms and improve business satisfaction. It is not only beneficial to the retailer to extract the useful information of high utility items and gain more profits but also to the customers to find a better choice of the selection of items.

In this survey, we provided an up-to-date and comprehensive review of HUIM algorithms for transactional databases. In this survey paper, we have discussed the key concepts, algorithms, and applications of HUIM. The paper provided the taxonomy and presented the state-of-the-art algorithms of HUIM for transactional databases that are categorized as level-wise, tree-based, utility-list-based, projection-based, and miscellaneous. It also discussed the pros and cons of each category of HUIM approaches are discussed in-depth. We also presented a summary of other existing HUIM algorithms including on-shelf, sequential databases, uncertain databases, temporal databases, incremental databases, negative utility values, average-utility, soft computing using HUIM, multiple minimum utility thresholds, data stream, periodic mining, and privacy-preserving utility mining. We also presented 16 real-world databases that are utilized by various HUIM approaches. Finally, the paper identified several key research opportunities and future directions for HUIM algorithms.

Author Contributions Author Contributions K.S. conceived analysis idea and presented taxonomy. R.K. developed the theory and analysis of the state-of-the-art approaches. R.K. provided the data for Tables 7 to 17. K.S. implemented the example and tables related to the running example. R.K. performed the analysis part at Section 3. K.S. collected all the required data. K.S. Reviewed and Edited the manuscript. Both the authors analyzed and contributed to the final manuscript.

Data availability and access Our work does not have any data to explore. Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Competing Interests The authors declare no conflicts of interest. The article presents a survey of high utility itemsets mining approaches for transactional databases. No any Funding is received for this work.

Ethical and informed consent for data used This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc, p 487–499
2. Fournier-Viger P, Lin JC-W, Kiran RU, Koh YS (2017) A survey of sequential pattern mining. Data Science and Pattern Recognition 1(1):54–77

3. Yao H, Hamilton HJ, Geng L (2006) A unified framework for utility-based measures for mining itemsets. In Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining, p 28–37. Citeseer
4. Liu J, Wang K, Fung BCM (2012) Direct discovery of high utility itemsets without candidate generation. 2012 IEEE 12th International Conference on Data Mining. Belgium, Brussels, pp 984–989
5. Gan W, Lin JC-W, Fournier-Viger P, Chao H-C, Fujita H (2018) Extracting non-redundant correlated purchase behaviors by utility measure. Knowl-Based Syst 143:30–41
6. Shie B-E, Tseng VS, Yu PS (2010) Online mining of temporal maximal utility itemsets from data streams. SAC '10, New York, NY, USA. Association for Computing Machinery, p 1622–1626
7. Tamilselvi T, Arasu GT (2019) Handling high web access utility mining using intelligent hybrid hill climbing algorithm based tree construction. Clust Comput 22(Suppl 1):145–155
8. Li Y-C, Yeh J-S, Chang C-C (2008) Isolated items discarding strategy for discovering high utility itemsets. Data Knowl Eng 64(1):198–217
9. Zihayat M, Davoudi H, An A (2017) Mining significant high utility gene regulation sequential patterns. BMC Syst Biol 11(6):109
10. Shie B-E, Hsiao H-F, Tseng VS (2013) Efficient algorithms for discovering high utility user behavior patterns in mobile commerce environments. Knowl Inf Syst 37(2):363–387
11. Li H-F, Huang H-Y, Chen Y-C, Liu Y-J, Lee S-Y (2008) Fast and memory efficient mining of high utility itemsets in data streams. In 2008 eighth IEEE international conference on data mining. p 881–886. IEEE
12. Liu Y, Liao W-k, Choudhary A (2005) A two-phase algorithm for fast discovery of high utility itemsets. In Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD'05, Berlin, Heidelberg. Springer-Verlag, p 689–695
13. Lan G-C, Hong T-P, Tseng VS (2014) An efficient projection-based indexing approach for mining high utility itemsets. Knowl Inf Syst 38(1):85–107
14. Tseng VS, Wu C-W, Shie B-E, Yu PS (2010) UP-Growth: An Efficient Algorithm for High Utility Itemset Mining. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, New York, NY, USA. Association for Computing Machinery, p 253–262
15. Tseng VS, Shie B-E, Wu C-W, Yu PS (2013) Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans Knowl Data Eng 25(8):1772–1786
16. Liu J, Wang K, Fung BCM (2016) Mining high utility patterns in one phase without generating candidates. IEEE Trans Knowl Data Eng 28(5):1245–1257
17. Fournier-Viger P, Wu C-W, Zida S, Tseng VS (2014) FHM: Faster High-Utility Itemset Mining Using Estimated Utility Co-occurrence Pruning. In Andreasen T, Christiansen H, Cubero J-C, Raś ZW (eds) Foundations of Intelligent Systems. Springer International Publishing, p 83–92. Cham
18. Krishnamoorthy S (2017) Hminer: Efficiently mining high utility itemsets. Expert Syst Appl 90:168–183
19. Jaysawal BP, Huang J-W (2019) DMHUPS: discovering multiple high utility patterns simultaneously. Knowl Inf Syst 59(2):337–359
20. Tseng VS, Wu C, Fournier-Viger P, Yu PS (2015) Efficient algorithms for mining the concise and lossless representation of high utility itemsets. IEEE Trans Knowl Data Eng 27(03):726–739
21. Fournier-Viger P, Lin JC-W, Nkambou R, Vo B, Tseng VS (2019) High-Utility Pattern Mining: Theory, Algorithms and Applications, vol 51 of Studies in Big Data. Springer
22. Zhang C, Han M, Sun R, Du S, Shen M (2020) A survey of key technologies for high utility patterns mining. IEEE Access 8:55798–55814
23. Rahmati B, Sohrabi MK (2019) A systematic survey on high utility itemset mining. Int J Inf Technol Decis Mak 18(04):1113–1185
24. Suvarna U, Srinivas Y (2019) Efficient High-Utility Itemset Mining Over Variety of Databases: A Survey. p 803–816
25. Singh K, Kumar R, Biswas B (2021) High average-utility itemsets mining: a survey. Appl Intell. p 1–38
26. Gan W, Chun-Wei J, Chao H-C, Wang S-L, Yu PS (2018) Privacy preserving utility mining: A survey. 2018 IEEE International Conference on Big Data (Big Data)
27. Gan W, Lin JC-W, Fournier-Viger P, Chao H-C, Hong T-P, Fujita T-P (2018) A survey of incremental high-utility itemset mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8(2):e1242
28. Gan W, Lin JC-W, Fournier-Viger P, Chao H-C, Tseng VS, Yu PS (2021) A survey of utility-oriented pattern mining. IEEE Trans Knowl Data Eng 33(4):1306–1327
29. Han M, Gao Z, Li A, Liu S, Mu D (2022) An overview of high utility itemsets mining methods based on intelligent optimization algorithms. Knowl Inf Syst 64(11):2945–2984
30. Singh K, Singh SS, Kumar A, Biswas B (2018) High utility itemsets mining with negative utility value: A survey. Journal of Intelligent and Fuzzy Systems 35(6):6551–6562
31. Lin JC-W, Gan W, Fournier-Viger P, Hong T-P, Zhan J (2016) Efficient mining of high-utility itemsets using multiple minimum utility thresholds. Know-Based Syst 113(C):100–115
32. Ahmed CF, Tanbeer SK, Jeong B-S, Lee Y-K (2009) Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans Knowl Data Eng 21(12):1708–1721
33. Lan G-C, Hong T-P, Tseng VS (2011) Discovery of high utility itemsets from on-shelf time periods of products. Expert Syst Appl 38(5):5851–5857
34. Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, SIGMOD '93, New York, NY, USA. Association for Computing Machinery, p 207–216
35. Wu C-W, Fournier-Viger P, Yu PS, Tseng VS (2011) Efficient mining of a concise and lossless representation of high utility itemsets. In Cook DJ, Pei J, Wang W, Zaïane OR, Wu X (eds) 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011. IEEE Computer Society, p 824–833
36. Singh K, Singh SS, Luhach AK, Kumar A, Biswas B (2021) Mining of closed high utility itemsets: A survey 14(1):6–12
37. Lin JC-W, Gan W, Fournier-Viger P, Hong T-P, Tseng VS (2016) Fast algorithms for mining high-utility itemsets with various discount strategies. Adv Eng Inform 30(2):109–126
38. Zihayat M, An A (2014) Mining top-k high utility patterns over data streams. Inf Sci 285(C):138–161
39. Ahmed CF, Tanbeer SK, Jeong B-S (2011) A framework for mining high utility web access sequences. IETE Tech Rev 28(1):3–16
40. Ahmed CF, Tanbeer SK, Jeong B-S, Lee Y-K (2009) Efficient mining of utility-based web path traversal patterns. In Proceedings of the 11th International Conference on Advanced Communication Technology - Volume 3, ICACT'09. pp 2215–2218. IEEE Press
41. Li Y-C, Yeh J-S, Chang C-C (2005) Direct candidates generation: A novel algorithm for discovering complete share-frequent itemsets. In Proceedings of the Second International Conference on Fuzzy Systems and Knowledge Discovery - Volume Part II, FSKD'05. Berlin, Heidelberg, pp 551–560. Springer-Verlag
42. Erwin A, Gopalan R, Achuthan N (2008) Efficient mining of high utility itemsets from large datasets. p 554–561

43. Liu Y-C, Cheng C-P, Tseng VS (2013) Mining differential top-k co-expression patterns from time course comparative gene expression datasets. BMC Bioinforma 14(1):230. Article number: 230
44. Krishnamoorthy S, Roy D (2020) An utility-based storage assignment strategy for e-commerce warehouse management. In 2019 International Conference on Data Mining Workshops (ICDMW), p 997–1004. IEEE
45. Yun C-H, Chen M-S (2007) Mining mobile sequential patterns in a mobile commerce environment. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 37(2):278–295
46. Yao H, Hamilton HJ (2006) Mining itemset utilities from transaction databases. Data and Knowledge Engineering 59(3):603–626
47. Erwin A, Gopalan RP, Achuthan NR (2007) CTU-Mine: An efficient high utility itemset mining algorithm using the pattern growth approach. In 7th IEEE International Conference on Computer and Information Technology (CIT 2007). p 71–76. IEEE
48. Ahmed CF, Tanbeer SK, Jeong B-S, Lee Y-K (2011) HUC-Prune: an efficient candidate pruning technique to mine high utility patterns. Appl Intell 34:181–198
49. Lin M-Y, Tu T-F, Hsueh S-C (2012) High utility pattern mining using the maximal itemset property and lexicographic tree structures. Inf Sci 215:1–14
50. Song W, Liu Y, Li J (2014) Mining high utility itemsets by dynamically pruning the tree structure. Appl Intell 40(1):29–43
51. Zida S, Fournier-Viger P, Lin Jerry C-W, Wu C-W, Tseng VS (2017) EFIM: A fast and memory efficient algorithm for high-utility itemset mining. Knowl Inf Syst 51(2):595–625
52. Chan R, Yang Q, Shen Y-D (2003) Mining high utility itemsets. In Third IEEE international conference on data mining. p 19–26. IEEE Computer Society
53. Yao H, Hamilton HJ, Butz CJ (2004) A Foundational Approach to Mining Itemset Utilities from Databases, vol 4. p 482–486
54. Liu Y, Liao W-k, Choudhary A (2005) A fast high utility itemsets mining algorithm. In Proceedings of the 1st International Workshop on Utility-Based Data Mining, UBDM '05. New York, NY, USA, p 90–99. Association for Computing Machinery
55. Lee D, Park S-H, Moon S (2013) Utility-based association rule mining: A marketing solution for cross-selling. Expert Syst Appl 40(7):2715–2725
56. Barber B, Hamilton HJ (2003) Extracting share frequent itemsets with infrequent subsets. Data Min Knowl Disc 7(2):153–185
57. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. SIGMOD Rec 29(2):1–12
58. Hu J, Mojsilovic A (2007) High-utility pattern mining: A method for discovery of high-utility item sets. Pattern Recogn 40(11):3317–3324
59. Yun U, Ryang H, Ryu KH (2014) High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates. Expert Syst Appl 41(8):3861–3878
60. Krishnamoorthy S (2015) Pruning strategies for mining high utility itemsets. Expert Systems with Applications 42(5):2371–2381
61. Zida S, Fournier-Viger P, Lin JC-W, Wu C-W, Tseng VS (2015) EFIM: A Highly Efficient Algorithm for High-Utility Itemset Mining. Springer International Publishing, Cham, pp 530–546
62. Shao J, Yin J, Liu W, Cao L (2015) Mining actionable combined patterns of high utility and frequency. In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA). p 1–10. IEEE
63. Dawar S, Goyal V (2015) UP-Hist Tree: An Efficient Data Structure for Mining High Utility Patterns from Transaction Databases. In Proceedings of the 19th International Database Engineering & Applications Symposium, IDEAS '15. New York, NY, USA, p 56–61. Association for Computing Machinery
64. Fournier-Viger P, Zida S, Lin JC-W, Wu C-W, Tseng VS (2016) EFIM-Closed: Fast and Memory Efficient Discovery of Closed High-Utility Itemsets. Springer International Publishing, Cham, pp 199–213
65. Ryang H, Yun U, Ryu KH (2016) Fast algorithm for high utility pattern mining with the sum of item quantities. Intelligent Data Analysis 20(2):395–415
66. Qu J-F, Liu M, Xin C, Wu Z (2018) Fast identification of high utility itemsets from candidates. Information 9(5)
67. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min Knowl Disc 8(1):53–87
68. Singh K, Biswas B (2019) Efficient algorithm for mining high utility pattern considering length constraints. Int J Data Warehous Min 15(3):1–27
69. Fournier-Viger P, Lin JC-W, Duong Q-H, Dam T-L (2016) FHM+: Faster High-Utility Itemset Mining Using Length Upper-Bound Reduction. In Trends in Applied Knowledge-Based Systems and Data Science: 29th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2016, Morioka, Japan, August 2-4, 2016, Proceedings, p 115–127, Cham. Springer International Publishing
70. Dawar S, Goyal V, Bera D (2019) A one-phase tree-based algorithm for mining high-utility itemsets from a transaction database. CoRR. arXiv:1911.07151
71. Wu JM-T, Lin JC-W, Tamrakar A (2019) High-utility itemset mining with effective pruning strategies. ACM Trans Knowl Discov Data 13(6)
72. Fournier-Viger P, Zhang Y, Lin JC-W, Dinh D-T, Le HB (2020) Mining correlated high-utility itemsets using various measures. Logic Journal of the IGPL 28(1):19–32
73. Hoa NT, Tao NV (2021) A novel fast algorithm for mining compact high utility itemsets. International Conference on Intelligent Systems Design and Applications. Springer International Publishing, Cham, pp 1325–1335
74. Nguyen LTT, Vu VV, Lam MTH, Duong TTM, Manh LT, Nguyen TTT, Vo B, Fujita H (2019) An efficient method for mining high utility closed itemsets. Information Sciences 495:78–99
75. Cheng-Wei Wu, Philippe Fournier-Viger, Jia-Yuan Gu, and Vincent S. Tseng. *Mining Compact High Utility Itemsets Without Candidate Generation*, pages 279–302. Springer International Publishing, Cham, 2019
76. Duong H, Hoang T, Tran T, Truong T, Le B, Fournier-Viger P (2022) Efficient algorithms for mining closed and maximal high utility itemsets. Knowl-Based Syst 257:109921
77. Deng Z-H (2018) An efficient structure for fast mining high utility itemsets. Appl Intell 48(9):3161–3177
78. Wu C-W, Fournier-Viger P, Gu J-Y, Tseng VS (2016) Mining closed+ high utility itemsets without candidate generation. In 2015 conference on technologies and applications of artificial intelligence (TAAI), p 187–194. IEEE
79. Erwin A, Gopalan RP, Achuthan NR (2007) A bottom-up projection based algorithm for mining high utility itemsets. In Proceedings of the 2nd International Workshop on Integrating Artificial Intelligence and Data Mining - Volume 84, AIDM '07, AUS. Australian Computer Society, Inc, p 3–11
80. Le B, Nguyen H, Cao TA, Vo B (2009) A novel algorithm for mining high utility itemsets. In 2009 First Asian Conference on Intelligent Information and Database Systems, p 13–17. IEEE
81. Liu M, Qu J (2012) Mining high utility itemsets without candidate generation. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12, New York, NY, USA, p 55–64. Association for Computing Machinery
82. Peng AY, Koh YS, Riddle P (2017) mHUIMiner: A Fast High Utility Itemset Mining Algorithm for Sparse Datasets. In: Kim J,

Shim K, Cao L, Lee J-G, Lin X, Moon Y-S (eds) Advances in Knowledge Discovery and Data Mining. Springer International Publishing, Cham, pp 196–207

83. Dawar S, Goyal V, Bera D (2017) A hybrid framework for mining high-utility itemsets in a sparse transaction database. Appl Intell 47(3):809–827
84. Dam T-L, Li K, Fournier-Viger P, Duong Q-H (2019) CLS-Miner: efficient and effective closed high-utility itemset mining. Frontiers of Computer Science 13(2):357–381
85. Sahoo J, Das AK, Goswami A (2014) An algorithm for mining high utility closed itemsets and generators. CoRR, arXiv:1410.2988
86. Ryang H, Yun U (2017) Indexed list-based high utility pattern mining with utility upper-bound reduction and pattern combination techniques. Knowl Inf Syst 51(2):627–659
87. Fournier-Viger P, Lin JC-W, Dinh T, Le HB (2016) Mining correlated high-utility itemsets using the bond measure. In International Conference on Hybrid Artificial Intelligence Systems, p 53–65. Springer
88. Bouasker S, Yahia SB (2015) Key correlation mining by simultaneous monotone and anti-monotone constraints checking. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, New York, NY, USA, p 851–856. Association for Computing Machinery
89. Ramampiaro H, Nørvåg K, Duong Q-H, Fournier-Viger P, Dam T-L (2018) Efficient high utility itemset mining using buffered utility-lists. Appl Intell 48:1859–1877
90. Shie B-E, Philip SY, Tseng VS (2012) Efficient algorithms for mining maximal high utility itemsets from data streams with different models. Expert Syst Appl 39(17):12947–12960
91. Nguyen TDD, Vu Q-B, Nguyen LTT (2019) Efficient algorithms for mining maximal high-utility itemsets. In 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), p 428–433. IEEE
92. Merugula S, Rao MVP (2020) An integrated approach for mining closed and generator high utility itemsets. International Journal of Knowledge-based and Intelligent Engineering Systems 24(1):27–35
93. Wei T, Wang B, Zhang Y, Hu K, Yao Y, Liu H (2020) FCHUIM: efficient frequent and closed high-utility itemsets mining. IEEE Access 8:109928–109939
94. Tseng VS, Wu C, Fournier-Viger P, Yu PS (2016) Efficient algorithms for mining top-k high utility itemsets. IEEE Trans Knowl Data Eng 28(01):54–67
95. Shen W, Zhang C, Fang W, Zhang X, Zhan Z-H, Lin JC-W (2021) Efficient high-utility itemset mining based on a novel data structure. In 2021 IEEE International Smart Cities Conference (ISC2), p 1–6. IEEE
96. Wu P, Niu X, Fournier-Viger P, Huang C, Wang B (2022) UBP-Miner: An efficient bit based high utility itemset mining algorithm. Knowl-Based Syst 248:108865
97. Cheng Z, Fang W, Shen W, Lin JC-W, Yuan B (2022) An efficient utility-list based high-utility itemset mining algorithm. Appl Intell
98. Lan G-C, Hong T-P, Tseng VS et al (2012) A projection-based approach for discovering high average-utility itemsets. J Inf Sci Eng 28(1):193–209
99. Duong Q-H, Liao B, Fournier-Viger P, Dam T-L (2016) An efficient algorithm for mining the top-k high utility itemsets, using novel threshold raising and pruning strategies. Knowl-Based Syst 104:106–122
100. Lin JC-W, Gan W, Fournier-Viger P, Hong T-P, Tseng VS (2016) Efficient algorithms for mining high-utility itemsets in uncertain databases. Know-Based Syst 96(C):171–187
101. Wang J-Z, Huang J-L, Chen Y-C (2016) On efficiently mining high utility sequential patterns. Knowl Inf Syst 49(2):597-627
102. Dam T-L, Li K, Fournier-Viger P, Duong Q-H (2017) An efficient algorithm for mining top-k on-shelf high utility itemsets. Knowl Inf Syst 52(3):621–655
103. Fournier-Viger P, Lin JC-W, Duong Q-H, Dam T-L (2016) PHM: Mining Periodic High-Utility Itemsets. Springer International Publishing, Cham, pp 64–79
104. Bai A, Deshpande PS, Dhabu M (2018) Selective database projections based approach for mining high-utility itemsets. IEEE Access 6:14389–14409
105. Lan G-C, Hong T-P, Tseng VS (2010) Projection-based utility mining with an efficient indexing mechanism. In 2010 International Conference on Technologies and Applications of Artificial Intelligence, pages 137–141. IEEE
106. Ahmed CF, Tanbeer SK, Jeong B-S, Choi H-J (2011) A framework for mining interesting high utility patterns with a strong frequency affinity. Inf Sci 181(21):4878–4894
107. Lin JC-W, Gan W, Fournier-Viger P, Hong T-P, Chao H-C (2017) FDHUP: Fast algorithm for mining discriminative high utility patterns. Knowl Inf Syst 51(3):873–909
108. Hidouri A, Jabbour S, Raddaoui B, Yaghlane BB (2021) Mining closed high utility itemsets based on propositional satisfiability. Data and Knowledge Engineering 136(C)
109. Song W, Liu Y, Li J (2014) BAHUI: Fast and memory efficient mining of high utility itemsets based on bitmap. Int J Data Warehous Min 10(1):1–15
110. Song W, Wang C, Li J (2016) Binary partition for itemsets expansion in mining high utility itemsets. Intelligent Data Analysis 20(4):915–931
111. Song W, Zhang Z, Li J (2016) A high utility itemset mining algorithm based on subsume index. Knowledge and Information Systems 49(1):315–340
112. Song W, Yang B, Xu Z (2008) Index-BitTableFI: An improved algorithm for mining frequent itemsets. Knowledge-Based Systems 21(6):507–513
113. Dahiya V, Dalal S (2022) EAHUIM: Enhanced absolute high utility itemset miner for big data. International Journal of Information Management Data Insights 2(1):100055
114. Chen Y, An A (2016) Approximate parallel high utility itemset mining. Big Data Research 6:26–42
115. Lan G-C, Hong T-P, Huang J-P, Tseng VS (2014) On-shelf utility mining with negative item values. Expert Syst Appl 41(7):3450–3459
116. Fournier-Viger P, Zida S (2015) FOSHU: Faster on-Shelf High Utility Itemset Mining – with or without Negative Unit Profit. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, New York, NY, USA, pages 857–864. Association for Computing Machinery
117. Zhang C, Du Z, Yang Y, Gan W, Yu PS (2021) On-shelf utility mining of sequence data. ACM Trans Knowl Discov Data 16(2)
118. Truong-Chi T, Fournier-Viger P (2019) A Survey of High Utility Sequential Pattern Mining. Springer International Publishing, Cham, pp 97–129
119. Quang MN, Dinh T, Huynh U, Le B (2016) MHHUSP: An integrated algorithm for mining and Hiding High Utility Sequential Patterns. In 2016 Eighth International Conference on Knowledge and Systems Engineering (KSE), p 13–18
120. Yin J, Zheng Z, Cao L (2012) USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. KDD '12, New York, NY, USA, p 660–668. Association for Computing Machinery
121. Lin JC-W, Gan W, Fournier-Viger P, Hong T-P, Tseng VS (2017) Efficiently mining uncertain high-utility itemsets. Soft Comput 21(11):2801–2820
122. Ahmed U, Lin JC-W, Srivastava G, Yasin R, Djenouri Y (2020) An evolutionary model to mine high expected utility patterns from uncertain databases. IEEE Transactions on Emerging Topics in Computational Intelligence 5(1):19–28

123. Chu C-J, Tseng VS, Liang T (2008) An efficient algorithm for mining temporal high utility itemsets from data streams. Journal of Systems and Software 81(7):1105–1117
124. Ryang H, Yun U (2016) High utility pattern mining over data streams with sliding window technique. Expert Syst Appl 57(C):214–231
125. Lee J, Yun U, Lee G, Yoon E (2018) Efficient incremental high utility pattern mining based on pre-large concept. Eng Appl Artif Intell 72(C):111–123
126. Lin C-W, Hong T-P, Lan G-C, Wong J-W, Lin W-Y (2014) Incrementally mining high utility patterns based on pre-large concept. Appl Intell 40(2):343–357
127. Dam T-L, Ramampiaro H, Nørvåg K, Duong Q-H (2019) Towards efficiently mining closed high utility itemsets from incremental databases. Knowl-Based Syst 165:13–29
128. Nguyen LTT, Nguyen P, Nguyen TDD, Vo B, Fournier-Viger P, Tseng VS (2019) Mining high-utility itemsets in dynamic profit databases. Knowl-Based Syst 175:130–144
129. Liu J, Ju X, Zhang X, Fung BCM, Yang X, Yu C (2019) Incremental mining of high utility patterns in one phase by absence and legacy-based pruning. IEEE Access 7:74168–74180
130. Yun U, Nam H, Lee G, Yoon E (2019) Efficient approach for incremental high utility pattern mining with indexed list structure. Futur Gener Comput Syst 95:221–239
131. Singh K, Kumar A, Singh SS, Shakya HK, Biswas B (2019) EHNL: An efficient algorithm for mining high utility itemsets with negative utility value and length constraints. Inf Sci 484:44–70
132. Singh K, Shakya HK, Singh A, Biswas B (2018) Mining of high-utility itemsets with negative utility. Expert Syst 35(6):e12296
133. Hong T-P, Lee C-H, Wang S-L (2011) Effective utility mining with the measure of average utility. Expert Syst Appl 38(7):8259-8265
134. Truong T, Duong H, Le B, Fournier-Viger P, Yun U (2019) Efficient high average-utility itemset mining using novel vertical weak upper-bounds. Knowl-Based Syst 183:104847
135. Truong T, Duong H, Le B, Fournier-Viger P (2020) EHAUSM: An efficient algorithm for high average utility sequence mining. Inf Sci 515:302–323
136. Wu JM-T, Teng Q, Lin JC-W, Cheng C-F (2020) Incrementally updating the discovered high average-utility patterns with the pre-large concept. IEEE Access 8:66788–66798
137. Kim J, Yun U, Yoon E, Lin JC-W, Fournier-Viger P (2020) One scan based high average-utility pattern mining in static and dynamic databases. Future Generation Computer Systems 111:143–158
138. Kannimuthu S, Premalatha K (2014) Discovery of high utility itemsets using genetic algorithm with ranked mutation. Appl Artif Intell 28(4):337–359
139. Lin JC-W, Yang L, Fournier-Viger P, Hong T-P, Voznak M . A binary PSO approach to mine high-utility itemsets. *Soft Comput.*, 21(17):5103–5121, 2017
140. Kumar R, Singh K (2022) A survey on soft computing-based high-utility itemsets mining. Soft Comput 26:6347–6392. Springer
141. Krishnamoorthy S (2018) Efficient mining of high utility itemsets with multiple minimum utility thresholds. Eng Appl Artif Intell 69:112–126
142. Lin JC-W, Zhang J, Fournier-Viger P, Hong T-P, Zhang J (2017) A two-phase approach to mine short-period high-utility itemsets in transactional databases. Adv Eng Inform 33(C):29–43
143. Yun U, Kim J (2015) A fast perturbation algorithm using tree structure for privacy preserving utility mining. Expert Syst Appl 42(3):1149–1165
144. Jr Bayardo RJ, Goethals B, Zaki MJ (eds) (2005) FIMI ’04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004, volume 126 of CEUR Workshop Proceedings. CEUR-WS.org
145. Dua D, Graff C (2017) UCI machine learning repository. Center for Machine Learning and Intelligent Systems
146. Ramanathan YL, Liao N-W-k, Memik G, Ozisikyilmaz B, Pisharath J, Choudhary A (2008) NU-MineBench version 2.0 source code and datasets. Center for Ultra-scale Computing and Information Security (CUCIS)
147. Zheng Z, Kohavi R, Mason L (2001) Real world performance of association rule algorithms. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’01, New York, NY, USA, p 401–406. Association for Computing Machinery
148. Fournier-Viger P, Lin JC-W, Gomariz A, Gueniche T, Soltani A, Deng Z, Lam HT (2016) The SPMF open-source data mining library version 2. In: Berendt B, Bringmann B, Fromont É, Garriga G, Miettinen P, Tatti N, Tresp V (eds) Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, Cham, pp 36–40
149. Pei J, Han J (2002) Constrained frequent pattern mining: A pattern-growth view. SIGKDD Explor Newsl 4(1):31–39
150. Lin YC, Wu C-W, Tseng VS (2015) Mining high utility itemsets in big data. In: Cao T, Lim E-P, Zhou Z-H, Ho T-B, Cheung D, Motoda H (eds) Advances in Knowledge Discovery and Data Mining. Springer International Publishing, Cham, pp 649–661
151. Martn D, Martnez-Ballesteros M, Garca-Gil D, Alcal-Fdez J, Herrera F, Riquelme-Santos JC (2018) Mrqar. Know-Based Syst 153(C):176–192
152. Zhai J, Zhang S, Zhang M, Liu X (2018) Fuzzy integral-based elm ensemble for imbalanced big data classification. Soft Comput 22(11):3519–3531
153. Zhai J, Wang X, Pang X (2016) Voting-based instance selection from large data sets with mapreduce and random weight networks. Inf Sci 367(C):1066–1077
154. Ramírez-Gallego S, Krawczyk B, García S, Woźniak M, Herrera F (2017) A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing 239:39–57
155. Dong G, Zhang X, Wong L, Li J (1999) CAEP: Classification by aggregating emerging patterns. In International Conference on Discovery Science, vol 1721 of Lecture Notes in Computer Science. pp 30–42. Springer
156. Lee SD, Cheung DW, Kao B (1998) Is sampling useful in data mining? a case in the maintenance of discovered association rules. Data Min Knowl Disc 2(3):233–262
157. Sahoo J, Das AK, Goswami A (2015) An efficient approach for mining association rules from high utility itemsets. Expert Syst Appl 42(13):5754–5778
158. Lin C-W, Hong T-P, Lan G-C, Wong J-W, Lin W-Y (2015) Efficient updating of discovered high-utility itemsets for transaction deletion in dynamic databases. Adv Eng Inform 29(1):16–27

Dr. Rajiv Kumar has obtain Ph. D. in Computer Science from Bharthiar University, Coimbatore, India in 2015. He has received M. Tech. in Computer Science and Engineering from Guru Jambheshwar University, Hisar, India in 2005. Currently, he is working as a Assistant Professor, School of Computer Science Engineering and Technology, Bennett University, Greater Noida, Uttar Pradesh, India. He has more than 18 years of experience. He has published more than 30 papers in various repute journals and conferences. He is member of various professional bodies like IEEE, ACM, IEI, IETE, CSI etc. His research interest areas are image processing, data mining, machine learning and deep learning.

Dr. Kuldeep Singh received Ph.D. in in Computer science & Engineering from Indian Institute of Technology (BHU) Varanasi. He received his M.Tech degree in Computer science and Engineering from Guru Jambheshwar University of Science and Technology, Hisar (Haryana). He is working as Assistant Professor at Department of Computer Science in University of Delhi, Delhi. His research interest includes High utility itemsets mining, social network analysis, and data mining. He has more than 13 years of teaching and research experience.