# A Survey of High Utility Sequential Pattern Mining


**Tin Truong-Chi and Philippe Fournier-Viger**

**Abstract** The problem of mining high utility sequences aims at discovering subsequences having a high utility (importance) in a quantitative sequential database. This problem is a natural generalization of several other pattern mining problems such as discovering frequent itemsets in transaction databases, frequent sequences in sequential databases, and high utility itemsets in quantitative transaction databases. To extract high utility sequences from a quantitative sequential database, the sequential ordering between items and their utility (in terms of criteria such as purchase quantities and unit profits) are considered. High utility sequence mining has been applied in numerous applications. It is much more challenging than the aforementioned problems 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. This chapter introduces the problem of high utility sequence mining, the state-of-art algorithms, applications, present related problems and research opportunities. A key contribution of the chapter is to also provide a theoretical framework for comparing upper-bounds used by high utility sequence mining algorithms. In particular, an interesting result is that an upper-bound used by the popular USpan algorithm is not an upper-bound. The consequence is that USpan is an incomplete algorithm, and potentially other algorithms extending USpan.

## 1 Introduction

High Utility Itemset Mining (HUIM) is a popular data mining task, consisting of discovering sets of values having a high utility (importance) in a quantitative transaction database. HUIM extends the problem of Frequent Itemset Mining (FIM), which has been widely studied. HUIM addresses limitations of frequent itemset mining by

T. Truong-Chi (✉)
University of Dalat, Dalat, Vietnam
e-mail: tintc@dlu.edu.vn

P. Fournier-Viger
Harbin Institute of Technology (Shenzhen), Shenzhen, China
e-mail: philfv8@yahoo.com



P. Fournier-Viger et al. (eds.), *High-Utility Pattern Mining*,
Studies in Big Data 51, https://doi.org/10.1007/978-3-030-04921-8_4

considering that items may appear more than once in each transaction and that items may have weights indicating their relative importance to the user. Although HUIM is useful, an important drawback of HUIM is that it ignores information about the sequential ordering of items. Hence, HUIM is inappropriate for many interesting real-life applications where the sequential ordering of quantitative items or itemsets must be considered, e.g. databases consisting of sequences of web accesses and sequences of purchases made by customers over a long period of time.

Motivated by these needs of practical applications, the problem of High Utility Sequence Mining (HUSM) in quantitative sequence databases was proposed [1–3]. It is an interesting and emerging topic that has been studied for about a decade. It has attracted the attention of many researchers and has many practical applications such as discovering interesting patterns in dynamic web log data [1, 2], mobile commerce environments [3–5], gene regulation data [6], and activity-cost event logs from healthcare [7].

The goal of HUSM is to identify all sequences having a high utility (importance) in a database. To represent the importance of patterns in HUSM, each item $a$ in a quantitative sequence database is associated with a unit profit $p(a)$ indicating its relative importance, and each occurrence of $a$ is associated with a quantity $q$ (e.g. indicating the number of units of the item $a$ purchased by a customer in a transaction). Formally, a pair $(a, q)$ is called a $q$-item, a $q$-itemset is a set of $q$-items (e.g. a customer transaction), a $q$-sequence is a list of $q$-itemsets (e.g. the list of transactions made by a customer, ordered by time), and a quantitative sequence database $D'$ consists of a finite set of input $q$-sequences (e.g. multiple sequences of customer transactions). To discover high utility patterns, utility calculations are performed. The utility of a $q$-item $(a, q)$ is the product of the quantity $q$ of $a$ by its unit profit, that is $q * p(a)$. The utility of a $q$-itemset (or $q$-sequence) is the sum of the utility of its $q$ -items (or $q$-itemsets respectively). The utility of a sequence (or sequential pattern) is computed by applying a utility function on all $q$-sequences in $D'$ where it appears. The problem of HUSM is to enumerate all sequences having a utility no less than a predefined minimum utility threshold. For instance, HUSM can be used to discover all subsequences of purchases that yield a profit that is no less than a threshold in sequences of customer transactions.

The relationship between the problem of HUSM, HUIM and Frequent Sequence Mining (FSM) is the following. In the case where each $q$-sequence in a database $D'$ consists of only one $q$-itemset, the problem of HUSM is equivalent to the traditional problem of HUIM. And in the case where all quantities are equal to either 0 or 1 and all unit profit values are set to 1, the problem of HUSM becomes equivalent to that of FSM.

Because the problem of HUSM in quantitative sequence databases is more general than the above problems, it is also more challenging. The key challenges of HUSM are as follows:

- *First*, considering the sequential ordering of itemsets leads to a combinatorial explosion of the search space. In other words, the number of patterns to be considered is much greater in HUSM than in HUIM. Thus, designing efficient HUSM algorithms requires to design effective strategies for search space pruning.

- *Second*, differently from HUIM, a pattern (subsequence) may appear multiple times in a $q$-sequence in HUSM. As a result, the utility of a pattern may be calculated in different ways.
- *Third,* utility calculations in HUSM are more time-consuming than support (occurrence frequency) calculation in FSM and utility calculations in HUIM. The reason is that in HUSM, quantities and unit profits must be considered, and that a pattern may appear multiple times in a $q$-sequence due to the sequential ordering.
- *Fourth*, in FSM a powerful property called the downward closure (*DC*) property is used to efficiently prune the search space. This property states that if a sequence is infrequent (its support is less than a user-specified minimum support threshold *minsup*), then all its super-sequences are also infrequent. However, this nice property does not hold for the utility in HUSM. Thus, other strategies must be found to reduce the search space.

This chapter provides a survey of HUSM that can serve both as an introduction to this problem and a concise overview of recent work for researchers in the field. The chapter introduces the problem of high utility sequence mining, the state-of-art algorithms, applications, present related problems and research opportunities. A key contribution of the chapter is also to provide a formal theoretical framework for comparing upper-bounds used by high utility sequence mining algorithms. In particular, an interesting result is that an upper-bound used by the popular USpan algorithm is not an upper-bound. The consequence is that USpan is an incomplete algorithm, and potentially other algorithms extending USpan.

The rest of this chapter is organized as follows. The definition of the HUSM problem, main properties of upper bounds on the utility of sequences and an algorithm for HUSPM are presented in Sect. 2. Section 3 introduces some extensions and problems related to HUSM. Section 4 discusses research opportunities. Finally, Sect. 5 draws a conclusion.

## 2 Problem Definition and Algorithm

This section first defines the problem of high utility sequence mining and how it generalizes those of high utility itemset mining, frequent sequence mining, and frequent itemset mining. Then, key properties of the utility measure and different search space pruning strategies are presented based on various upper bounds (UBs) on utility. Finally, an algorithm for HUSM is presented.

### *2.1 Definition of the High Utility Sequential Pattern Mining Problem*

The problem of HUSM consists of discovering interesting sequences (sequential patterns) in a Quantitative Sequence DataBases (QSDBs), where pattern interestingness is evaluated using a utility function.

A quantitative sequence database is formally defined as follows. Let $A = \{a_1, a_2, \ldots, a_M\}$ be a set of distinct items occurring in a QSDB. A subset $E \subseteq A$ is called an itemset. Without loss of generality, we assume that items in itemsets are sorted according to a total order $\prec$, such as the lexicographical order. In a QSDB, each item $a$ is associated with a positive number $p(a) \in R_+$ representing its importance (e.g. unit profit), called the external utility. Moreover, a positive quantity $q$ is associated to each occurrence of an item in a QSDB. An item $a$ associated with a quantity $q$ is represented as a pair $(a, q)$ and is called a quantitative item (or briefly $q$-item). A $q$-itemset $E'$ (or $q$-element according to itemset $E$) is defined and denoted as $E^{'} = \{(a, q) \mid a \in E, q \in R_+\}$. For example, a $q$-itemset can represent a set of items with quantities purchased by a customer (a transaction). A $q$-sequence $\alpha'$ is a list of $q$-itemsets $E^{'}_k$, $k = 1 \ldots p$, which is denoted as $\alpha^{'} = E^{'}_1 \rightarrow E^{'}_2 \rightarrow \cdots \rightarrow E^{'}_p$ (or $\alpha^{'} =< E^{'}_1, E^{'}_2, \ldots, E^{'}_p >$). For instance, a sequence of customer transactions can be viewed as a $q$-sequence. The size and length of a $q$-sequence is defined as $size(\alpha^{'}) = p$ and $length(\alpha^{'}) = \sum_{k=1\ldots p} |E^{'}_k|$, where $|E^{'}_k|$ is the number of $q$-items in $E^{'}_k$. A QSDB $D'$ is a finite set of (input) $q$-sequences, $D' = \{\Psi^{'}_i,\ i = 1, \ldots,\ N\}$, where each $q$-sequences $\Psi^{'}_i$ is associated with a unique identifier SID (Sequence IDentifier). For the convenience of the reader, Table 1 summarizes the symbols used in this chapter to denote ($q$-) items, ($q$-) elements, and ($q$-) sequences and input $q$-sequences.

*Example 1* (***QSDB***). A QSDB $D'$ is shown in Table 2, with external utility values provided in Table 3. This database will be used as running example. It contains four $q$-sequences $\Psi^{'}_1$, $\Psi^{'}_2$, $\Psi^{'}_3$ and $\Psi^{'}_4$. Consider the third $q$-sequence $\Psi^{'}_3 = (c,\ 4) \rightarrow (a, 4)(c, 2)(e, 2) \rightarrow (a, 1)(f, 2)$. The $3^{rd}$ $q$-element of that sequence is $\{(a, 1)(f, 2)\}$. If $\Psi^{'}_3$ is a customer transaction sequence, it indicates that a customer purchased 4 units

**Table 1** Notations

| | Convention letters | Example |
|---|---|---|
| Items; $q$-items | Roman letters; (Roman letter, number) | $a, b, c, d, e, f, g, h$; $(a, 2), (b, 5), (c, 1)$ |
| Elements; $q$-elements | Capitalized roman letters; Capitalized roman letters followed by' | $A, B, C, D, E, F$; $A$', $B$', $C$', $D$', $E$', $F$' |
| Sequences; $q$-sequences | Greek letters; Greek letters followed by prime' | $\alpha, \beta, \gamma, \delta, \varepsilon$; $\alpha$', $\beta$', $\gamma$', $\delta$', $\varepsilon$' |
| Input sequence; input $q$-sequences | Capitalized Greek letters Capitalized Greek letters followed by' | $\Psi$, $\Psi_{index}$; $\Psi'$, $\Psi^{'}_{index}$ |

**Table 2** A QSDB $\mathscr{D}'$

| | |
|---|---|
| $\Psi_1$ | $(a, 2)$ $(c, 1)$ $(e, 3) \rightarrow$ $(a, 3)$ $(b, 2) \rightarrow$ $(a, 5)$ $(d, 5) \rightarrow$ $(a, 5)$ $(b, 3)$ $(c, 8) \rightarrow$ $(a, 4)$ $(c, 2)$ $(d, 1)$ $(f, 4)$ |
| $\Psi_2$ | $(b, 4) \rightarrow$ $(a, 2)$ $(c, 4)$ $(e, 3) \rightarrow$ $(a, 3)$ $(d, 2) \rightarrow$ $(a, 1)$ $(c, 4)$ $(d, 1)$ $(f, 1) \rightarrow$ $(a, 4)$ $(b, 3)$ $(c, 3)$ |
| $\Psi_3$ | $(c, 4) \rightarrow$ $(a, 4)$ $(c, 2)$ $(e, 2) \rightarrow$ $(a, 1)$ $(f, 2)$ |
| $\Psi_4$ | $(d, 8) \rightarrow$ $(a, 7)$ $(c, 10)$ $(e, 3) \rightarrow$ $(a, 2)$ $(g, 1) \rightarrow$ $(a, 9)$ $(f, 8)$ |

**Table 3** External utility values

| Item $i$ | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ |
|---|---|---|---|---|---|---|---|
| $p(i)$ | 1 | 3 | 5 | 10 | 2 | 9 | 2 |

of item $a$, followed by purchasing 4, 2 and 2 units of items $a$, $c$ and $e$, respectively, followed by purchasing 1 and 2 units of items $a$ and $f$, respectively. The external utility values of Table 3 can be interpreted as the amount of profit yield by the sale of one unit of each item.

In the special case where each $q$-element of a QSDB contains a single $q$-item, the QSDB is said to be a 1-QSDB. Similarly, if each $q$-element of a QSDB contains at least one $q$-item, a QSDB is said to be a $n$-QSDB. For example, the database of Table 2 is a $n$-QSDB.

The utility of $q$-items, $q$-iemsets, $q$-sequences, and QSDB are defined as follows.

**Definition 1** (*Utility of q-item, q-element, q-sequence and QSDB*) For any $q$-sequence $\alpha^{'} = E_1^{'} \rightarrow E_2^{'} \rightarrow \cdots \rightarrow E_p^{'}$, the utility of a $q$-item $(a, q)$ is defined as $u((a, q)) = p(a) * q$. Similarly, the utility of a $q$-element $E^{'} = \{(a_{i_1}, q_{i_1}), (a_{i_2}, q_{i_2}), \ldots, (a_{i_m}, q_{i_m})\}$ is defined as $u(E^{'}) = \sum_{j=1\ldots m} u((a_{i_j}, q_{i_j}))$. The utility of a $q$-sequence $\alpha^{'}$ is defined and denoted as $u(\alpha^{'}) = \sum_{i=1\ldots p} u(E'_i)$. The utility of the QSDB $D'$ is defined and denoted as $u(D') = \sum_{\Psi' \in D'} u(\Psi^{'})$.

*Example 2* (***Utility of a q-item, q-element and q-sequence***). Consider the QSDB of Tables 2 and 3. The utility of the $q$-element $E^{'} = \{(a, 2)\,(c, 1)\;\,(e, 3)\}$ is $u\left(E'\right) = u((a, 2)) + u((c, 1)) + u((e, 3)) = p(a) * 2 + p(c) * 2 + p(e) * 3 = 1 * 2 + 5 * 2 + 2 * 3 = 18$. The utility of the $q$-sequence $\Psi_3^{'}$ is $u(\Psi_3^{'}) = u((c, 20)) + u((a, 4)) + u((c, 10)) + u((e, 4)) + u((a, 1)) + u((f, 18)) = 20 + 4 + 10 + 4 + 1 + 18 = 57$.

It can be observed that all utility calculations require multiplying quantities of $q$-items in input $q$-sequences with their external utilities. If many utility calculations are performed using the same $q$-items, the corresponding multiplications will be performed several times, which is inefficient. To *avoid repeatedly calculating* the utility of each $q$-item $(a, q)$ in elements of each input $q$-sequence $\Psi' D'$, these utility values can be calculated once. In this case, the value $q$ of each $q$-item $(a, q)$ is replaced by the product $u((a, q)) = p(a) * q$. The result is an equivalent representation of the QSDB $D'$, which is called the *integrated QSDB* of $D'$. For the sake of brevity, it is also denoted as $D'$. In the rest of this chapter, we consider that all QSDBs are represented as integrated QSDBs.

*Example 3* (***Integrated QSDB***). Consider the database QSDB $D'$ of Table 2 and the unit profit values of Table 3. The equivalent integrated-database QSDB $D'$ is shown in Table 4. For example, the $3^{rd}$ $q$-element $\{(a, 1)\,(f, 2)\}$ of $\Psi_3^{'}$ is transformed to $\{(a, 1)(f, 18)\}$ in the corresponding integrated QSDB$D^{'}$, because $u((a, 1)) = p(a) * 1 = 1$ and $u((f, 2)) = p(f) * 2 = 18$. Thereafter, the integrated database of Table 4 will be considered for the running example.

**Table 4** Integrated QSDB $\mathscr{D}'$

| | |
|---|---|
| $\Psi_1'$ | $(a, 2)(c, 5)(e, 6) \rightarrow (a, 3)(b, 6) \rightarrow (a, 5)(d, 50) \rightarrow (a,5)(b,9)(c,40) \rightarrow (a,4)(c,10)(d,10)(f,36)$ |
| $\Psi_2'$ | $(b, 12) \rightarrow (a, 2)(c, 20)(e, 6) \rightarrow (a, 3)(d, 20) \rightarrow (a, 1)(c, 20)(d, 10)(f, 9) \rightarrow (a, 4)(b, 9)(c, 15)$ |
| $\Psi_3'$ | $(c, 20) \rightarrow (a, 4)(c, 10)(e, 4) \rightarrow (a, 1)(f, 18)$ |
| $\Psi_4'$ | $(d, 80) \rightarrow (a, 7)(c, 50)(e, 6) \rightarrow (a, 2)(g, 2) \rightarrow (a, 9)(f, 72)$ |

**Table 5** SDB $\mathscr{D}$ according to $\mathscr{D}'$

| | |
|---|---|
| $\Psi_1$ | $b \rightarrow ace \rightarrow ad \rightarrow acdf \rightarrow abc$ |
| $\Psi_2$ | $b \rightarrow ace \rightarrow ad \rightarrow acdf \rightarrow abc$ |
| $\Psi_3$ | $c \rightarrow ace \rightarrow af$ |
| $\Psi_4$ | $d \rightarrow ace \rightarrow ag \rightarrow af$ |

To obtain a (non-quantitative) sequence database $D$ according to integrated QSDB $D'$, we perform a projection as follows.

The *projection* of a $q$-itemset $E' = \{(a, q) \mid a \in E,\ q \in R_+\}$ to obtain an itemset $E$ is defined and denoted as $E = proj(E')$. The projection of a $q$-sequence $\alpha' = E_1' \rightarrow E_2' \rightarrow \cdots \rightarrow E_p'$ to obtain a sequence $\alpha$ is defined and denoted as: $\alpha = proj(\alpha') = proj(E_1') \rightarrow proj(E_2') \rightarrow \cdots \rightarrow proj(E_p')$. The projection of a QSDB $D'$ to obtain a (non-quantitative) sequence database (SDB) $D$ is defined as $D = proj(D') = \{proj(\Psi') \mid \Psi' \in D'\}$.

*Example 4* (***Sequence database according to*** $D'$) Consider the integrated QSDB $D'$ of Table 4. The (non-quantitative) SDB $D$ according to $D'$ is shown in Table 5. For example, the projection of $\Psi_3' = (c, 20) \rightarrow (a, 4)(c, 10)(e, 4) \rightarrow (a, 1)(f, 18)$ is $\Psi_3 = c \rightarrow ace \rightarrow af$.

A particularity of mining patterns (subsequences) in sequences is that a pattern may appear multiple times in the same sequence (have multiple occurrences). To formally describe occurrences of a sequence $\alpha$ in a $q$-sequence $\beta'$, an order relation is introduced. In the following definitions, $\alpha' = E_1' \rightarrow E_2' \rightarrow \cdots \rightarrow E_p'$ and $\beta' = F_1' \rightarrow F_2' \rightarrow \cdots \rightarrow F_q'$ are two $q$-sequences, and $\alpha = E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow E_p$ and $\beta = F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$ are two sequences.

**Definition 2** (*Order relations over q-sequences and sequences*) A *partial order relation* $\sqsubseteq$ is defined over the set of all $q$-sequences as follows:

1. For any two $q$-elements, $E' = \{(a_{i_1}, q_{i_1}), (a_{i_2}, q_{i_2}), \ldots, (a_{i_m}, q_{i_m})\}$ and $F' = \{(a_{j_1}, q_{j_1}), (a_{j_2}, q_{j_2}), \ldots, (a_{j_n}, q_{j_n})\}$, $E'$ is contained in $F'$ and denoted as $E' \sqsubseteq F'$, iff[1] there exist natural numbers $1 \leqslant k_1 < k_2 < \cdots < k_m \leqslant n$ such that $(a_{i_l}, q_{i_l}) = (a_{j_{k_l}}, q_{j_{k_l}})$, *i.e.* $a_{i_l} = a_{j_{k_l}}$ and $q_{i_l} = q_{j_{k_l}}, \forall l = 1, \ldots, m$.

[1] iff means "if and only if"

2. $\alpha^{'}$ is contained in $\beta^{'}$ (or $\alpha^{'}$ is a sub $q$-sequence of $\beta^{'}$, $\beta^{'}$ is a super $q$-sequence of $\alpha^{'}$) and denoted as $\alpha^{'} \sqsubseteq \beta^{'}$ (or $\beta^{'} \sqsupseteq \alpha^{'}$) iff $p \leqslant q$ and there exist $p$ integers, $1 \leqslant j_1 < j_2 < \ldots < j_p \leqslant q$ such that $E_k^{'} \sqsubseteq F_{j_k}^{'}, \forall k = 1, \ldots, p$; and $\alpha^{'} \sqsubset \beta^{'} \Leftrightarrow (\alpha^{'} \sqsubseteq \beta^{'} \wedge \alpha^{'} \neq \beta^{'})$.

Similarly, for simplicity, we also use $\sqsubseteq$ to define the partial order relation over the set of all sequences, defined as follows: $\alpha \sqsubseteq \beta \Leftrightarrow \exists p$ positive integers, $1 \leqslant j_1 < j_2 < \cdots < j_p \leqslant q$: $E_k \subseteq F_{j_k}, \forall k = 1, \ldots, p$, and $\alpha \sqsubset \beta \Leftrightarrow (\alpha \sqsubseteq \beta \wedge \alpha \neq \beta)$.

*Example 5* ($\sqsubseteq$ ***relation***) In the running example, $q$-sequence $\alpha^{'} = (a, 2)(e, 6) \rightarrow (a, 3) \rightarrow (c, 20) \sqsubseteq \Psi_2^{'}$, and sequence $\alpha = proj(\alpha^{'}) = ae \rightarrow a \rightarrow c \sqsubseteq \Psi_2$.

To describe whether a sequence appears or not in a $q$-sequence, the concept of match between a sequence and a $q$-sequence is presented.

**Definition 3** (*Match of a sequence in a q-sequence*) A $q$-sequence $\alpha^{'}$ matches a sequence $\alpha$ iff $proj(\alpha^{'}) = \alpha$, which is denoted as $\alpha^{'} \sim \alpha$. A sequence $\alpha$ is contained in a $q$-sequence $\beta^{'}$, which is denoted as $\alpha \sqsubseteq \beta^{'}$, iff $proj(\beta^{'}) \sqsupseteq \alpha$, i.e. there exists a sub-$q$-sequence $\alpha^{'} \sqsubseteq \beta^{'}$ such that $\alpha^{'} \sim \alpha$. Let $U(\alpha, \beta^{'}) = \{\alpha^{'} \mid \alpha^{'} \sqsubseteq \beta^{'} \wedge \alpha^{'} \sim \alpha\}$ be the set of all sub-$q$-sequences $\alpha^{'}$ of $\beta^{'}$ that matches $\alpha$ (or all occurences $\alpha^{'}$ of $\alpha$ in $\beta^{'}$).

*Example 6* (***Match***) The $q$-sequence $\alpha^{'} = (a, 2)(e, 6) \rightarrow (a, 3) \rightarrow (c, 20) \sqsubseteq \Psi_2^{'}$ matches $\alpha = ae \rightarrow a \rightarrow c$. Hence, $\alpha \sqsubseteq \Psi_2^{'}$. The three occurrences of $\alpha$ in $\Psi_2^{'}$ are $U(\alpha, \Psi_2^{'}) = \{(a, 2)(e, 6) \rightarrow (a, 3) \rightarrow (c, 20), (a, 2)(e, 6) \rightarrow (a, 3) \rightarrow (c, 15), (a, 2)(e, 6) \rightarrow (a, 1) \rightarrow (c, 15)\}$.

To discover sequences having a high utility in a QSDB, it is necessary to define how to calculate the utility of a sequence in a $q$-sequence. In HUIM, calculating the utility of a pattern is relatively simple because a pattern (an itemset) appears at most once in each record of a quantitative transaction database. Thus, a pattern has a single utility value for each record. In HUSM, a pattern (sequence) $\alpha$ may appear multiple times in a $q$- sequence $\Psi^{'}$, and thus may have multiple utility values in the same $q$-sequence. For example, the sequence $\alpha = ae \rightarrow a \rightarrow c$ appears three times in the $q$- sequence $\Psi_2^{'}$ as shown in Example 6. Thus the utility of a sequence $\alpha$ in a $q$ -sequence $\Psi^{'}$ can be calculated in multiple ways.

In some studies [1, 2], the utility of $\alpha$ in $\Psi^{'}$ is calculated as the sum or maximum of the utility values of all items of $\alpha$ appearing in $\Psi^{'}$ depending on whether $\alpha$ has multiple distinct occurrences in $\Psi^{'}$ or not. A drawback of such definitions is that computing the utility of sequences is time-consuming, the utility values may be very large, and that these values may represent the personal behaviors of customers rather than the behavior of most customers [8]. Moreover, in some studies [1], only 1-QSDBs are considered rather than the more general $n$-QSDBs. To simplify utility calculations and provide a more suitable definition for the needs of many real-world applications, recent studies have mostly defined the utility of a sequence using the maximum form, that is $u_{max}(\alpha, \Psi^{'}) = max\{u(\alpha^{'}) \mid \alpha^{'} \in U(\alpha, \Psi^{'})\}$ [8–12].

**Definition 4** (*Maximum utility measure of sequences*) Consider a sequence $\alpha$ that is contained in a $q$-sequence $\beta^{'}$, i.e. $\alpha \sqsubseteq \beta^{'}$. The *maximum utility* (actual utility, or briefly utility) of $\alpha$ in $\beta^{'}$ is denoted and defined as $u_{max}(\alpha,\ \beta^{'}) = max\{u(\alpha^{'}) \mid \alpha^{'} \in U(\alpha,\ \beta^{'})$. Similarly, the maximum utility of $\alpha$ in $D'$ is denoted and defined as $u_{max}(\alpha, D') = \sum_{\Psi^{'} \in \rho(\alpha)} u_{max}(\alpha, \Psi^{'})$, or more concisely denoted as $u_{max}(\alpha)$, where $\rho(\alpha) = \{\Psi^{'} \in D' \mid \Psi^{'} \sqsupseteq \alpha\}$ is the set of all input $q$-sequences $\Psi^{'}$ in $D'$ containing $\alpha$.

Based on the above definition, the problem of high utility sequence mining is defined as follows.

**Definition 5** (*Problem definition*) A sequence $\alpha$ is said to be a *high utility* (HU) sequence (or HU sequential pattern) if its utility in $D'$ is not less than a user-defined minimum utility threshold $mu$, that is $u_{max}(\alpha) \geqslant mu$, and $\alpha$ is called a *low utility* (LU) sequence if $u_{max}(\alpha) < mu$. Let $HUS(mu)$ or briefly $HUS = \{\alpha \mid u_{max}(\alpha) \geqslant mu\}$ denote the set of all high utility sequences. The problem of high utility sequence mining (HUSM) is to discover $HUS$.

*Example 7* (***High utility sequence mining***) Consider the integrated QSDB $D'$ of Table 4 and $mu = 350$. The sequence $\alpha = d \rightarrow ac \rightarrow af$ appears in the two following input $q$-sequences: $\rho(\alpha) = \{\Psi_1^{'}, \Psi_4^{'}\}$. The occurrences of $\alpha$ in the first $q$-sequence are $U(\alpha,\ \Psi_1^{'}) = \{(d, 50) \rightarrow (a, 5)(c, 40) \rightarrow (a, 4)(f,\ 36)\}$, so $u_{max}\left(\alpha, \Psi_1^{'}\right) = 135$. Similarly, $u_{max}\left(\alpha, \Psi_4^{'}\right) = 218$. Thus, $u_{max}(\alpha) = 135 + 218 = 353 \geqslant mu$ and $d \rightarrow ac \rightarrow af$ is a HU sequence.

Algorithms for discovering patterns in databases generally search for patterns by starting from patterns containing a single item, and then consider larger patterns by appending items to these patterns one at a time. In HUIM, the process of adding an item to a pattern is called an $i$-extension. It consists of adding an item $y$ to an itemset $A$ to obtain an itemset $A \cup \{y\}$. In HUSM, the process of extending a pattern (sequence) $\alpha$ with an item is more complicated because there two ways of appending an item to a pattern are considered. These two types of *extensions* are called $i$-extension and $s$-extension, respectively. For example, consider the sequence $\alpha = ce \rightarrow a$ and the item $d$. By appending the item $d$ to $\alpha$ using these two types of extensions, two different sequences are obtained: $ce \rightarrow ad$ and $ce \rightarrow a \rightarrow d$.

**Definition 6** (*Extensions*) A sequence $\alpha$ can be extended by two types of extensions. The *itemset extension* (or briefly, *i -extension*) of $\alpha$ with $\beta$ such that $\forall a \in E_p$, $\forall b \in F_1$, $a \prec b$, is a sequence denoted and defined as $\alpha \diamond_i \beta = E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow (E_p \cup F_1) \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$. The *sequence extension* (or *s -extension*) of $\alpha$ with $\beta$, denoted as $\alpha \diamond_s \beta$, is the sequence $\alpha \diamond_s \beta = E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow E_p \rightarrow F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$. A *forward extension* (or briefly *extension*) of $\alpha$ with $\beta$, $\gamma = \alpha \diamond \beta$, can be either $\alpha \diamond_i \beta$ or $\alpha \diamond_s \beta$.

*Example 8* (***Extensions***) Consider the sequence $\alpha = ce \rightarrow a$ and the item $d$. The *i-extension* of the sequence $\alpha$ with $d$ is $ce \rightarrow ad$. The *s-extension* of the sequence $\alpha$ with $d$ is $ce \rightarrow a \rightarrow d$. Both extensions are forward extensions of $\alpha$.

Note that in the case where each $q$-sequence $\Psi^{'}$ of a QSDB $D^{'}$ contains a single $q$-itemset (*i.e.* $size(\Psi^{'}) = 1$), then only $i$-extensions need to be considered to find all high utility sequences, and HUSM becomes equivalent to the problem of HUIM. In that case, the QSDB is a Quantitative Transaction DataBase (QTDB), and $u_{max}\left(\alpha,\ \Psi^{'}\right)$ is the utility of a $q$-itemset, as in HUIM.

Based on the concepts of extensions, the concepts of prefix, suffix and projected databases are defined next, which are used by most algorithms for HUSM and will be useful to discuss properties of the HUSM problem in this chapter.

**Definition 7** (*Prefix, suffix*) Consider a sequence $\gamma$ obtained by performing a forward extension of a sequence $\alpha$ with a sequence $\beta$, *i.e.* $\gamma = \alpha \diamond \beta$. In that case, $\alpha$ is called a *prefix* of $\gamma$ and $\beta$ is called the *suffix* of $\gamma$ *w.r.t.* $\alpha$. In addition, if $\delta$ is the *smallest prefix* of $\gamma$ (according to $\sqsubseteq$) containing $\alpha$, then it is denoted as $\delta = pref(\gamma, \alpha)$. The corresponding suffix $\beta$ of $\gamma$ *w.r.t.* $\delta$, i.e. $\gamma = \delta \diamond \beta$, is denoted as $suf(\gamma, \alpha)$. Similarly, we also have corresponding concepts of $pref(\Psi', \alpha)$ and $suf(\Psi', \alpha)$.

*Example 9* (***Prefix, suffix***) Let $\alpha = ae \rightarrow a \rightarrow c$. In the SDB $D$ in Table 5 and integrated QSDB $D'$ in Table 4, $pref(\Psi_1, \alpha) = ace \rightarrow ab \rightarrow ad \rightarrow abc$, so $suf(\Psi_1, \alpha) = acdf$, and $pref(\Psi_1^{'}, \alpha) = (a, 2)(c, 5)(e, 6) \rightarrow (a, 3)(b, 6) \rightarrow (a, 5)(d, 50) \rightarrow (a, 5)(b, 9)(c, 40)$, so $suf(\Psi_2^{'}, \alpha) = (a, 4)(c, 10)\ (d, 10)(f, 36)$.

**Definition 8** (*Projected database*) Consider the *non-quantitative* sequence database (*SDB*) $D$ corresponding to $D'$, $D = proj(D')$. A *projected database* (*PDB*) of $D$ *w.r.t.* $\alpha$ is defined and denoted as $D_\alpha = \{suf(\Psi, \alpha) | \Psi \in D \wedge \Psi \sqsupseteq \alpha\}$. Similarly, $D_\alpha^{'} = \left\{suf\left(\Psi',\ \alpha\right) \middle| \Psi' \in D' \wedge \Psi' \sqsupseteq \alpha\right\}$.

*Example 10* (***Projected database***) Consider the QSDB $D'$, SDB $D$ shown in Tables 4 and 5, the sequence $\alpha = ae \rightarrow a \rightarrow c$. This sequence appears in $\Psi_1$ and $\Psi_2$. It is found that $suf(\Psi_1, \alpha) = acdf$ and $suf(\Psi_2, \alpha) = df \rightarrow abc$. Thus, $D_\alpha = \{acdf, df \rightarrow abc\}$. Similarly, $suf(\Psi_1^{'}, \alpha) = (a, 4)(c, 10)(d, 10)(f, 36)$ and $suf(\Psi_2^{'}, \alpha) = (d, 10)(f, 9) \rightarrow (a, 4)\ (b, 9)(c, 15)$, so $D_\alpha^{'} = \{(a, 4)\ (c, 10)\ (d, 10)\ (f, 36)$, $(d, 10)\ (f, 9) \rightarrow (a, 4)\ (b,\ 9)\ (c, 15)\}$.

In HUSM, the search space of sequences is usually represented a *prefix tree* which is a tree structure, where the root represents the null sequence, and each other tree node represents a candidate sequence. In that tree, each child of a node is an $i$-extension or $s$-extension of that node.

The problem of HUSM is more general than the problem of FSM. In FSM, the goal is to discover all sequences having a support (occurrence frequency) that is not less than a user-defined minimum support threshold *minsupp*. The support measure is defined as follows.

**Definition 9** (*Support of a sequence*) The support of a sequence $\alpha$ is defined as the number of super-$q$-sequences (in $D'$ w.r.t. $\sqsubseteq$) of $\alpha$, that is $supp(\alpha) = |\rho(\alpha)|$, where $\rho(\alpha) = \{\Psi^{'} \in D' \mid \Psi^{'} \sqsupseteq \alpha\}$.

*Example 11* (***Support***) In the running example, the sequence $\alpha = ae \rightarrow a \rightarrow c$ appears in two $q$-sequences $\rho(\alpha) = \{\Psi_1^{'}, \Psi_2^{'}\}$. Thus, $supp(\alpha) = 2$.

To transform the problem of HUSM into FSM, the utility of a $q$-item $(a, q)$, $q$-itemset $E'$ and $q$-sequence $\Psi' = E_1^{'} \rightarrow E_2^{'} \rightarrow \cdots \rightarrow E_p^{'}$ are redefined as follows, $u((a, q)) \equiv 1$ (i.e. all items have the same importance), $u(E^{'}) = \prod_{(a_{ij}, q_{ij}) \in E'} u((a_{i_j}, q_{i_j}))$ and $u(\Psi^{'}) = \prod_{i=1 \ldots p} u(E_i^{'})$. Then, we have $u_{max}(\alpha, \Psi^{'}) = 1$ iff $\Psi^{'} \in \rho(\alpha)$, thus $u_{max}(\alpha) = \sum_{\Psi^{'} \in \rho(\alpha)} u_{max}(\alpha, \Psi^{'}) = |\rho(\alpha)| = supp(\alpha)$. Hence, we obtain the traditional problem of FSM in non-quantitative Sequence DataBases (SDBs) when replacing *minutil* with the user-specified minimum support threshold *minsupp*. In other words, the utility measure is more general than the traditional support measure and the FSM problem on SDBs is a special case of HUSM on QSDBs.

To find all patterns for a pattern mining problem in a reasonable time, it is necessary to design a search procedure to explore the search space of all patterns, and strategies to prune parts of the search space that do not contain the desired patterns. In problems such as FSM or frequent itemset mining, a powerful property is used to reduce the search space, called the downward closure (DC or equivalently anti-monotonic – $AM$) property. It allows to efficiently prune many infrequent patterns. The DC property states that the support of a pattern $\alpha$ is always greater or equal to the support of all its super-patterns. Thus, if a pattern $\alpha$ is infrequent, i.e. $supp(\alpha)$ is less than a predefined minimum support threshold *minsupp*, then all its supper-patterns are also infrequent and can be immediately pruned from the search space.

The problem of HUSM is more challenging than FSM because utility measures such as $u_{max}$ are neither anti-monotonic nor monotonic. In other words, for a QSDB, there may exist patterns $\alpha$, $\beta$, $\gamma$ and $\delta$ such that $\alpha \sqsubset \beta$, $\gamma \sqsubset \delta$ and $u(\alpha) < u(\beta)$ and $u(\delta) < u(\gamma)$. For instance, consider $D'$ in Table 4 and the utility measure $u_{max}$. For $\alpha = ace \rightarrow a \rightarrow f$, $\beta = ace \rightarrow a \rightarrow af$ and $\delta = ace \rightarrow a \rightarrow a \rightarrow f$, we have that $\delta \sqsupset \alpha \sqsubset \beta$ and $u_{max}(\beta) = 245 > u_{max}(\alpha) = 231 > u_{max}(\delta) = 59$. To overcome the lack of a DC property, upper bounds $UB$ on the utility $u_{max}$ have been proposed which satisfy an $AM$ or weaker $AM$ properties. For example, in [2, 4, 9], an upper bound (UB) on $u_{max}$ named $SWU$ (Sequence-Weighted Utility) was proposed, which satisfies the $AM$ property. If $SWU(\alpha) < mu$ for a sequence $\alpha$, an algorithm can prune all its super-sequences $\beta \sqsupseteq \alpha$, because $u_{max}(\beta) \leqslant SWU(\beta) \leqslant SWU(\alpha) < mu$. Note that the upper bound is a natural generalization of the $SWU$ (Transaction-Weighted Utility) upper bound used in HUIM [13].

In summary, there are three main challenges in HUSM. *First*, compared with HUIM, sequencing between itemsets in HUSM leads to a combinational explosion of the search space. Assume that $L$ is the maximum length of input $q$-sequences in a database $D^{'}$ containing $M$ distinct items ($L$ can be greater than $M$). Then, in the worst case, the maximum number $MaxNP$ of patterns is $O(M^L)$ in HUSM, while in HUIM or FIM, $MaxNP$ is only $O(2^M)$. *Second*, since a sequential pattern (or sequence) may have multiple occurences in each input $q$-sequence, computing the utility of patterns is more complicated and consumes much more time compared to calculating their supports in FIM and FSM, or the utility of itemsets in HUIM. As a result, the computational complexity of HUSM is usually much higher than that of HUIM. *Third*, the $AM$ property does not hold for the utility measure $u_{max}$. Thus, well-known efficient algorithms as well as search space pruning strategies for mining

frequent sequences or high utility itemsets cannot be directly applied to the problem of HUSM. Hence, designing UBs which satisfy *AM* or weaker *AM* properties is key to effectively reduce the search space for mining $HUS$, and obtain efficient algorithms.

## 2.2 Upper Bounds on $u_{max}$ and their Key Properties

UBs are critical to mine patterns efficiently. Several UBs on $u_{max}$ have been proposed to prune LU sequences. Thus, an important question that arises is how to select appropriate UB(s) when designing an HUSM algorithm.

In the field of pattern mining, researchers often compare UBs in terms of their values. It can be shown that some UBs are tighter. Intuitively, one may think that tighter UBs are better as they provide a smaller overestimation of the utility of patterns. But this is not always true. Different UBs have different pruning effects (some UBs can be used for depth or width pruning), and their ability at reducing the size of a projected database or other upper bounds may vary.

This subsection first defines what an UB is. Then it proposes a general framework for evaluating upper-bounds in terms of tightness, pruning effects and other properties. This subsection discusses in details the SWU [9], MEU and LEU UBs presented in [12]. Moreover, towards the end of the subsection, additional UBs are discussed namely the *PEU* [11] and *CRoM* [10] UBs.

**Definition 10** (*Upper bound*) A utility measure $UB$ is called an *upper bound* (UB) on $u_{max}$ iff $u_{max}(\alpha) \leqslant ub(\alpha)$, for any sequence $\alpha$. For any two UBs $ub_1$ and $ub_2$ on $u_{max}$, $ub_1$ is said to be *tighter* than $ub_2$ and denoted as $ub_1 \ll ub_2$ iff $ub_1(\alpha) \leqslant ub_2(\alpha)$, $\forall\ \alpha$. Moreover, $ub_1$ is said to be *strictly tighter* than $ub_2$ iff $ub_1 \ll ub_2$ and $\exists \alpha$ : $ub_1(\alpha) < ub_2(\alpha)$.

The SWU [9], MEU and LEU UBs are defined as follows.

**Definition 11** ($ub_{rem}$ *upper bound on* $u_{max}$ *in a q-sequence*) Assume that $\alpha \sqsubseteq \beta' = F_1' \rightarrow F_2' \rightarrow \cdots \rightarrow F_q'$, i.e. $\exists \alpha' = E_1' \rightarrow E_2' \rightarrow \cdots \rightarrow E_p' \sqsubseteq \beta' : \alpha' \sim \alpha$. Thus, there exist $p$ integers $1 \leqslant i_1 < i_2 < \cdots < i_p \leqslant q$ such that $E_k' \sqsubseteq F_{i_k}'$, $\forall k = 1, \ldots, p$. In that case, the index $i_p$ is said to be the *ending* of $\alpha$ in $\beta'$, denoted as $end(\alpha, \beta')$ or $end(\alpha', \beta')$ and the last item of $\alpha$ in $F_{i_p}'$ is called the *ending item* and is denoted as $e_{i_p}$. Then, the *remaining* $q$-sequence of $\alpha$ in $\beta'$ w.r.t. $\alpha'$ (or the ending $i_p$) is the rest of $\beta'$ after $\alpha'$ (or after the ending item $e_{i_p}$), which is denoted as $rem(\alpha, \beta', \alpha')$ (or $rem(\alpha, \beta', i_p)$). The measure $ub_{rem}(\alpha, \beta') = max\{u(\alpha') + u(rem(\alpha, \beta', \alpha')) \mid \alpha' \in U(\alpha, \beta')\}$ is an upper bound on $u_{max}$ in $\beta'$ based on the remaining utilities $u(rem(\alpha, \beta', \alpha'))$. For an ending $i_p$ of $\alpha$ in $\beta'$, let $u(\alpha, \beta', i_p) = max\left\{u(\alpha') \mid \alpha' \in U\left(\alpha, \beta'\right) \wedge end(\alpha', \beta') = i_p\right\}$. Then, $ub_{rem}(\alpha, \beta') = max\{u(\alpha, \beta', i_p) + u(rem(\alpha, \beta', i_p)) \mid$*for all ending* $i_p$ *of* $\alpha$ *in* $\beta'\}$.

**Definition 12** (*SWU, MEU and LEU upper bounds on* $u_{max}$)
a. The *Sequence-Weighted Utility* (*SWU*) [9] of $\alpha$ is denoted and defined as $SWU(\alpha) = \sum_{\Psi' \in \rho(\alpha)} u(\Psi')$.
b. The *Maximum Extension Utility* (*MEU*) [12] of $\alpha$ is defined and denoted as $MEU(\alpha) = \sum_{\Psi' \in \rho(\alpha)} ub_{rem}(\alpha, \beta')$.
c. The *Looser extension Utility* (*LEU*) of a sequence $\beta = \alpha \diamond y$ for a prefix $\alpha$ is defined and denoted as $LEU(\beta) = \sum_{\Psi' \in \rho(\beta)} ub_{rem}(\alpha, \Psi')$ if $\alpha \neq <>$, and $LEU(y) = SWU(y)$ if $\alpha = <>$. Here, the notation $<>$ denotes the empty sequence. The *LEU* UB is used by the *LAS* (Look Ahead Strategy) [12].

The relationship between the *SWU*, *LEU* and *MEU* UBs and $u_{max}$ is presented in the next theorem. Basically, the three measures are upper-bounds on $u_{max}$ and the *SWU* is the loosest of the three upper-bounds. Thus, if one wants to propose a new upper-bound *ub* on a utility measure such as $u_{max}$, it should respect at least two conditions. First, *ub* should truly be an UB, and second, it should be tighter than the largest well-known *SWU* UB.

**Theorem 1** (Relation between *UBs* on $u_{max}$)

$$u_{max} \ll MEU \ll LEU \ll SWU.$$

*That is, SWU, LEU and MEU are UBs on* $u_{max}$*, and MEU (or SWU) is the tightest (or largest respectively) UB among the three above UBs.*

To prove Theorem 1, the following lemma is needed.

**Lemma 1** (Anti-monotonicity w.r.t. forward extension of $ub_{rem}(\alpha, \Psi')$) *For any extension* $\beta = \alpha \diamond \delta$ *of* $\alpha$*, we have* $\rho(\beta) \subseteq \rho(\alpha)$ *and* $ub_{max}(\beta, \Psi') \leqslant ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$, $\forall \Psi' \in \rho(\beta)$.

*Proof* If $\beta = \alpha \diamond \delta \sqsupset \alpha$, then $\forall \Psi' \in \rho(\beta)$, $\exists \beta' \sqsubseteq \Psi' : proj(\beta') = \beta \sqsupset \alpha$, so $\exists \alpha' \sqsubset \beta' \sqsubseteq \Psi' : proj(\alpha') = \alpha$. Thus, $\Psi' \in \rho(\alpha)$, i.e. $\rho(\beta) \subseteq \rho(\alpha)$. Without loss generality, we can assume that the sequence $\delta$ only consists of an item $x$, $\delta = x$, i.e. $\beta = \alpha \diamond x$. Let $p = size(\alpha)$, $k = |\alpha[p]|$ and for any $\Psi' \in \rho(\beta)$, assume that $\beta^{*'} \in U(\beta, \Psi')$ such that $ub_{rem}(\beta, \Psi') = u(\beta^{*'}) + u(rem(\Psi', \beta^{*'}))$. Call $\alpha^{*'} = \beta^{*'}[1 \ldots p] = \beta_1^{*'} \rightarrow \beta_2^{*'} \rightarrow \cdots \rightarrow \beta_p^{*'}$ or $\alpha^{*'} = \beta^{*'}[1 \ldots p-1] \diamond_s \beta^{*'}[p][1 \ldots k]$ if $\beta = \alpha \diamond_s x$ or $\beta = \alpha \diamond_i x$, respectively. Then, we always have $\alpha^{*'} \sqsubseteq \Psi'$ and $proj(\alpha^{*'}) = \alpha$, so $\alpha^{*'} \in U(\alpha, \Psi')$ and $ub_{max}(\beta, \Psi') \leqslant ub_{rem}(\beta, \Psi') \leqslant u(\alpha^{*'}) + u(rem(\Psi', \alpha^{*'})) \leqslant max\{u(\alpha') + u(rem(\Psi', \alpha')) \mid \alpha' \in U(\alpha, \Psi')\} = ub_{rem}(\alpha, \Psi')$.

*Proof of Theorem 1*. For any sequence $\beta$ and $\Psi' \in \rho(\beta)$, it is clear that $u_{max}(\beta, \Psi') \leqslant max\{u(\alpha') + u\left(rem(\alpha, \beta', \alpha')\right) \mid \alpha \in' U(\alpha, \beta')\} = ub_{rem}(\beta, \Psi') \leqslant u(\Psi')$. Thus, by summing for all $\Psi' \in \rho(\beta)$, $u_{max}(\beta) \leqslant MEU(\beta) \leqslant SWU(\beta)$, i.e. $u_{max} \ll MEU \ll SWU$. To prove that $MEU \ll LEU \ll SWU$, consider any extension $\beta$ of $\alpha$ with an item $x$, $\beta = \alpha \diamond x \sqsupset \alpha$, and $\Psi' \in \rho(\beta)$. If $\alpha = <>$, due to $LEU(\beta) = SWU(\beta)$, then $MEU(\beta) \leqslant LEU(\beta) = SWU(\beta)$. If $\alpha \neq <>$, by Lemma 1, we

also obtain $ub_{rem}(\beta,\ \Psi^{'}) \leqslant ub_{rem}(\alpha,\ \Psi^{'}) \leqslant u(\Psi^{'})$. Thus, $MEU(\beta) \leqslant LEU(\beta) \leqslant SWU(\beta)$. Hence, in all cases, we always have $MEU(\beta) \leqslant LEU(\beta) \leqslant SWU(\beta)$.

*Example 12* (*UBs MEU, LEU and SWU*) For the sequence $\alpha = d \rightarrow a \rightarrow a, \rho(\alpha) = \{\Psi_1^{'}, \Psi_2^{'}, \Psi_4^{'}\}$. For the $q$-sequence $\Psi_4^{'}$, we have $u(\Psi_4^{'}) = 228$. There are two endings of $\alpha$ in $\Psi_4^{'}$, which are $a_3$ and $a_4$. According to $a_3$, the first occurrence $\alpha^{'}$ of $\alpha$ in $\Psi_4^{'}$ is $(d,80) \rightarrow (a,7) \rightarrow (a,2)$ and the corresponding remaining $q$-sequence is $rem\left(\alpha, \Psi_4^{'}, \alpha^{'}\right)$ or $rem\left(\alpha, \Psi_4^{'}, a_3\right) = (g,\ 2) \rightarrow (a, 9)\ (f, 72)$. For the second ending $a_4$, there are two occurences $\alpha^{'}$ of $\alpha$ in $\Psi_4^{'}$, that is $(d, 80) \rightarrow (a, 7) \rightarrow (a, 9)$ and $(d, 80) \rightarrow (a, 2) \rightarrow (a, 9)$. The corresponding remaining $q$-sequence is $rem\left(\alpha, \Psi_4^{'}, a_4\right) = (f,\ 72)$. Thus, $u_{max}\left(\alpha,\ \Psi_4^{'}\right) = max\{89,\ 96,\ 91\} = 96, ub_{rem}\left(\alpha,\ \Psi_4^{'}\right) = max\{89 + 83,\ 96 + 72,\ 91 + 72\} = 172$. After performing similar calculations for $\Psi_1^{'}$ and $\Psi_2^{'}$, we obtain $u_{max}(\alpha) = 59 + 25 + 96 = 180$, $SWU(\alpha) = 191 + 131 + 228 = 550$, $MEU(\alpha) = 115 + 49 + 172 = 338$ and $LEU(\alpha) = \sum_{i=1,2,4} ub_{rem}\left(\delta,\ \Psi_i^{'}\right) = 164 + 88 + 228 = 480$, where $\delta = d \rightarrow a$. Thus, $u_{max}(\alpha) < MEU(\alpha) < LEU(\alpha) < SWU(\alpha)$, i.e. $MEU$ and $LEU$ are strictly tighter than $LEU$ and $SWU$, respectively.

Intuitively, one may think that tighter UBs are always better than looser UBs. But this intuition is not completely correct. In general, to evaluate different UBs of sequences, besides considering their values, we need to additionally examine their anti-monotonic-like properties and their pruning effects for reducing the search space. These concepts are introduced in Definitions 13 and 14 of the proposed generic framework for comparing UBs. Based on this, we can design different strategies for width or depth pruning, or reducing the size of projected databases (PDBs) or reducing these UBs on PDBs that have been just reduced Theorem 2, or gradually tightening candidate item sets for $i$- and $s$-extensions of a sequence during the mining process Proposition 1. For example, although $MEU$ is tighter than $LEU$ and $SWU$ (i.e. $MEU$ is only better in term of values), but it should not only use the tightest UB $MEU$ to prune LU candidate sequences, because the three UBs satisfy different anti-monotonic or weaker anti-monotonic properties. Thus, they will have different effects for pruning or tightening candidate item sets for extensions as it will be shown in Example 13. The following anti-monotonic-like properties are defined for discussing UBs.

**Definition 13** (*Anti-monotonic property and its weaker extensions*) A *measure* (*e.g. support*, *utility* $u_{max}$ or *upper bounds* $SWU$, $LEU$, $MEU$) $ub$ of sequences is called:
a. *anti-monotonic* (or *downward closure*) and denoted as $AM(ub)$ iff $ub(\beta) \leqslant ub(\alpha)$, $\forall \beta \sqsupseteq \alpha$.
b. *anti-monotonic w.r.t. forward extension*, denoted as $AMF(ub)$, iff $ub(\beta) \leqslant ub(\alpha)$, for any (*forward*) extension $\beta = \alpha \diamond \delta \sqsupseteq \alpha$.
c. *anti-monotonic w.r.t. backward extension*, denoted as $AMB(ub)$, iff for any *backward extension* $\alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$, i.e. $\alpha \diamond \varepsilon \diamond y \sqsupset \alpha \diamond y$ with $\varepsilon \neq <>$ and $\forall \diamond \in \{\diamond_i, \diamond_s\}$, $ub(\alpha \diamond \varepsilon \diamond y) \leqslant ub_{max}(\alpha, y)$, where $ub_{max}(\alpha, y) = max\{ub(\alpha \diamond_i y), ub(\alpha \diamond_s y)\}$.

d. *anti-monotonic w.r.t. bi-direction extension*, denoted as $AMBiDi(ub)$, iff $AMF(ub)$ and $AMB(ub)$.

The $AMF$, $AMB$ and $AMBiDi$ properties will be called *anti-monotone-like* and they are strictly weaker than $AM$. In other words, if $AM(ub)$, then $AMBiDi(ub)$, $AMF(ub)$ and $AMB(ub)$. In the rest of this chapter, we will generally consider $ub$ to be one of the three presented UBs on $u_{max}$, that are the $SWU$, $LEU$ or $MEU$. Besides the above properties of UBs, an UB can have two types of pruning effects, defined as follows.

**Definition 14** (*Pruning effects of UBs*) An upper bound $ub$ on $u_{max}$ (*e.g.* $SWU$, $LEU$, $MEU$) of sequences is said to have a:
a. *depth* pruning effect, which is denoted as $DP(ub)$, iff $u_{max}(\beta) \leqslant ub(\alpha)$ for any (*forward*) extension of $\alpha$, $\beta = \alpha \diamond \delta \sqsupseteq \alpha$. An UB $ub$ having this pruning effect is called a *depth* UB (*DepthUB*);
b. *width* pruning effect, which is denoted as $WP(ub)$, iff $DP(ub)$ and for any *backward extension* $\alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$ with the same prefix $\alpha$, i.e. $\alpha \diamond \varepsilon \diamond y \sqsupset \alpha \diamond y$ with $\varepsilon \neq <>$ and $\forall \diamond \in \{\diamond_i, \diamond_s\}$, $u_{max}(\alpha \diamond \varepsilon \diamond y) \leqslant ub_{max}(\alpha, y)$. An UB $ub$ having this pruning effect is called a *width* UB (*WidthUB*).

A QSDB where each database entry is a $q$-sequence, is said to be represented in Horizontal DB Form (HDF). This is the case for the databases considered in the examples until now (e.g. Tables 2 and 4). For such database, a HUSM algorithm will typically start from patterns (sequence) containing single items and gradually extend these patterns by appending items to find larger sequences. When exploring the search space by extending a sequence with items, an algorithm may produce patterns which do not appear in the QSDB. Processing patterns that do not appear in a QSDB can waste a considerable amount of time. To overcome this drawback, several HUSM algorithms create projected databases (PDBs or pseudo-PDBs) for each sequence that is considered. By scanning the PDB of a sequence, it is possible to find the set of all items that can extend a sequence to generate patterns that exist in a QSDB. Although this can avoid the problem of considering non-existing sequences, a drawback of this approach is that creating and scanning PDBs can be time-consuming and require a considerable amount of memory. To address this problem, based on only the $AMB$ property, the following proposition allows to tighten or reduce the set of candidate items used for extending sequences without considering PDBs. Consider the task of extending a sequence $\alpha$. Let $I_{ub}(\alpha) = \{y \in A \mid y \succ lastItem(\alpha) \wedge ub(\alpha \diamond_i y) \geqslant mu\}$ and $S_{ub}(\alpha) = \{y \in A \mid ub(\alpha \diamond_s y) \geqslant mu\}$ be two sets of candidate items that can be used to extend a sequence $\alpha$ by $i$- and $s$-extensions, respectively. Moreover, let the set of all items that can extend a sequence $\alpha$ be defined as $IS_{ub}(\alpha) = I_{ub}(\alpha) \cup S_{ub}(\alpha)$. Similarly, the set of all items that can extend a sequence resulting from extending $\alpha$ with an item $x$ is defined as $IS_{ub}(\alpha, x) = IS_{ub}(\alpha \diamond_i x) \cup IS_{ub}(\alpha \diamond_s x)$. Recall that $<>$ denotes the empty sequence. It is clear that $I_{ub}(<>) = S_{ub}(<>) = IS_{ub}(<>)$ and $S_{SWU}(<>) = S_{LEU}(<>)$, because $LEU(y) = SWU(y)$. Based on these definitions, a third pruning effect for UBs is formalized, which consists of reducing the number of items to be considered for extending a sequence.

**Proposition 1** (Tightening $IS_{ub}$ effect of ub such that AMB (ub)) *Let ub be an upper bound on the utility $u_{max}$. If AMB (ub) then ub has the tightening $IS_{ub}$ effect, that is, for any item x in A, $IS_{ub}(\alpha, x) \subseteq IS_{ub}(\alpha)$. Thus, $u_{max}(\beta) \leqslant ub(\beta) \leqslant ub_{max}(\alpha, y)$, for any backward extension $\beta = \alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$. Hence, we say that ub has the effect of tightening the $IS_{ub}$ set, denoted as TE(ub).*

*Proof* Assume that *AMB*(*ub*). Furthermore, we consider any *backward* extension $\beta = \alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$ with $\varepsilon \neq <>$. Since *AMB* (*ub*), $u_{max}(\beta) \leqslant ub(\beta) \leqslant ub_{max}(\alpha, y)$, $\forall \diamond \in \{\diamond_i, \diamond_s\}$. Especially with $\varepsilon = x$, if $y \in IS_{ub}(\alpha \diamond_i x) \cup IS_{ub}(\alpha \diamond_s x)$, i.e. $mu \leqslant ub(\beta)$, then $mu \leqslant ub_{max}(\alpha, y)$, so $y \in IS_{ub}(\alpha)$ and $IS_{ub}(\alpha, x) \subseteq IS_{ub}(\alpha)$.

This proposition is very useful for the following reason. Consider a sequence $\alpha$ and the set $IS_{ub}(\alpha)$ of items that has been considered for extending $\alpha$. Now consider a sequence $\beta = \alpha \diamond x$. Based on Proposition 1, to find the set $IS_{ub}(\alpha, x)$ of items that can extend $\beta$, it is not necessary to consider all possible items (the set $A$). Instead, we can only consider the items in $IS_{ub}$. By applying this idea, the set of candidate items used to extend $\beta$ can be greatly reduced. This is an optimization that can improve the performance of HUSM and does not require creating or scanning PDBs.

This pruning effect is illustrated with an example. Consider the *LEU* upper bound, which satisfies the *AMB* property, as it will be shown in Theorem 3. Furthermore, consider the database of the running example with $mu = 350$, and that the items to be considered initially for extending the empty sequence is $S_{LEU}(<>) = \{acdef\}$. To determine the set $IS_{LEU}(a)$ of candidate items that can extend item $a$, the naïve approach is to consider all items appearing in the database ($A = \{a, b, c, d, e, f, g\}$). If we instead apply Proposition 1, we know that $IS_{ub}(a) \subseteq S_{LEU}(<>)$. Thus, we only need to examine five candidate items for extending the sequence $a$. This strategy allows to reduce the number of items to be considered without creating or scanning the projected database $D'_a$. Then, it is found that $I_{LEU}(a) = \{cef\}$, $S_{LEU}(a) = \{af\}$, and thus $IS_{LEU}(a) = \{acef\}$. Similarly, to determine the set $IS_{LEU}(ac)$ of items which can extend a sequence $ac$, Proposition 1 states that $IS_{LEU}(ac)$ is a subset of $IS_{LEU}(a)$. Thus, we only need to consider four items in $IS_{LEU}(a)$ to extend $ac$. Then, we obtain $IS_{LEU}(ac) = \{aef\}$ ($\subseteq IS_{LEU}(a) \subseteq IS_{LEU}(<>)$). This pruning effect can be similarly applied for other sequences.

Several algorithms rely on PDBs to discover high utility sequences for the aforementioned reasons. Because a projected database can be large and still contain many items, it is desirable to use UBs to reduce the number of items to be considered from a PDB. To reduce PDBs and UBs when searching for high utility sequences, the set of *irrelevant* items w.r.t. an UB *ub* in PDB $D_\alpha$ is denoted as $IRS_{ub}(\alpha) = S_{ub}(<>) \setminus IS_{ub}(\alpha) = \{y \in S_{ub}(<>) \mid ub_{max}(\alpha, y) < mu\}$ for $\alpha \neq <>$, and $IRS_{ub}(<>) = A \setminus S_{ub}(<>) = \{y \in A \mid ub(y) < mu\}$ for $\alpha = <>$. The following theorem allows to design strategies for depth or width pruning as well as for reducing PDBs and UBs based on the *AMF* or *AMBiDi* properties of any upper bound *ub*.

**Theorem 2** (Pruning and Reducing PDB and UB strategies) *Let ub be any UB on* $u_{max}$.
*a. If AM* $(ub)$*, then AMBiDi* $(ub)$.
*b. (Depth Pruning Strategy DPS*$(ub)$ *based on AMF*$(ub)$*). If AMF*$(ub)$ *(e.g. AMF*$(MEU)$*) and ub* $(\alpha) < mu$*, then DP*$(ub)$*, so we can deeply prune the whole branch*$(\alpha)$ *of the prefix tree (the tree consisting of all forward extensions of* $\alpha$*).*
*c. (Width Pruning Strategy* $-WPS(ub)$*, Reducing PDB and UBs strategy*$-Red(ub)$ *based on AMBiDi*$(ub)$*). If AMBiDi*$(ub)$ *and* $ub_{max}(\alpha \diamond y) < mu$*, then* $WP(ub)$*, i.e. all both forward and backward extensions (bi-directional extensions) of* $\alpha \diamond y$*,* $\alpha \diamond y \diamond \delta$ *and* $\alpha \diamond \varepsilon \diamond y$*, can be pruned, which is denoted as* $WPS(ub)$*. Moreover, we can reduce PDBs by removing from* $D_\alpha$ *all items in* $IRS_{ub}(\alpha)$ *and UBs of all sequences with prefix* $\alpha$ *by calculating again UBs on such reduced PDB. This is denoted as Red*$(ub)$.

*Proof* a. Obviously, for any extension $\beta = \alpha \diamond \delta \sqsupseteq \alpha$, by Definition 13, if $AM(ub)$, then $AME(ub)$ and $AMB(ub)$, or $AMBiDi(ub)$.
b. If $AMF(ub)$ and $ub(\alpha) < mu$, then $ub(\alpha \diamond \delta) \leqslant ub(\alpha) < mu$, i.e. we can prune all (*forward*) extensions $\alpha \diamond \delta$ of $\alpha$.
c. If $AMBiDi(ub)$ (i.e. $AMF(ub)$ and $AMB(ub)$) and $ub_{max}(\alpha, y) < mu$, by assertion b., then we only need to prove that we can prune all *backward* extensions $\beta = \alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$. This is correct, because $ub(\beta) \leqslant ub_{max}(\alpha, y) < mu$ by Proposition 1. Furthermore, for any $\gamma = \alpha \diamond \varepsilon \diamond y \diamond \delta$ (i.e. the sequence $\gamma$ represents an arbitrary sequence in the PDB $D'_\alpha$ containing $y$ and having the same prefix $\alpha$), we always have $u_{max}(\gamma) \leqslant ub(\gamma) \leqslant ub(\alpha \diamond \varepsilon \diamond y) \leqslant ub_{max}(\alpha, y) < mu$, i.e. any sequence $\gamma$ in the PDB $D'_\alpha$ containing $y$ must be a low utility sequence. Thus, we can discard such item $y$ from $D'_\alpha$.

Note that, due to the generality of Definition 13, strategies for depth or width pruning as well as for reducing PDBs presented in Theorem 2 can be also applied for other measures such as the support. Another such measure is the probability of sequences which satisfy the $AM$ property and is used in the extended problem of HU-probability sequence mining in uncertain QSDBs (see Sect. 3.1).

The three UBs presented until now ($MEU$, $LEU$ and $SWU$) satisfy various anti-monotonic-like properties. Their properties are presented in the following theorem.

**Theorem 3** (Anti-monotonic-like properties of UBs) $AM(SWU)$, $AMF(MEU)$, $AMBiDi(LEU)$.

*Proof* For any sequences $\alpha \sqsubseteq \beta$, then $\rho(\alpha) \supseteq \rho(\beta)$, so $SWU(\alpha) \geqslant SWU(\beta)$, *i.e.* $AM(SWU)$.

If $\beta = \alpha \diamond \delta \sqsupset \alpha$, by Lemma 1, $ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi'), \forall \Psi' \in \rho(\beta) \subseteq \rho(\alpha)$, so $MEU(\beta) \leqslant MEU(\alpha)$ and $AMF(MEU)$.

*Proof of* $AMF(LEU)$: To prove that $LEU(\beta) \leqslant LEU(\alpha), \forall \beta = \alpha \diamond \delta \sqsupset \alpha$, without loss of generality, we can assume that $\delta$ only consists of an item $y$. If $\alpha = \varepsilon \diamond x \sqsubset \beta = \alpha \diamond y = \varepsilon \diamond x \diamond y$ with $\varepsilon \neq <>$ and $x, y \in A$, then, by Lemma 1,

$\forall \Psi^{'} \in \rho(\beta) \subseteq \rho(\alpha)$, $\varepsilon \diamond x \sqsupset \varepsilon$, $ub_{rem}\left(\varepsilon \diamond x,\ \Psi^{'}\right) \leqslant ub_{rem}(\varepsilon, \Psi^{'})$, so $LEU(\beta) = \sum_{\Psi^{'} \in \rho(\beta)} ub_{rem}(\varepsilon \diamond x,\ \Psi^{'}) \leqslant \sum_{\Psi^{'} \in \rho(\alpha)} ub_{rem}(\varepsilon,\ \Psi^{'}) = LEU(\alpha)$. Otherwise, if $\varepsilon = <>$, $\delta = y$, i.e. $\alpha = x \sqsubset \beta = x \diamond y$, then $\forall \Psi^{'} \in \rho(\beta) \subseteq \rho(\alpha), LEU(\beta) = \sum_{\Psi^{'} \in \rho(\beta)} ub_{rem}(x,\ \Psi^{'}) \leqslant \sum_{\Psi^{'} \in \rho(x)} ub_{rem}(x,\ \Psi^{'}) = MEU(x) \leqslant SWU(x) = LEU(\alpha)$.

*Proof of AMB(LEU)*: For $\alpha = <>$, $LEU(\varepsilon \diamond y) = \sum_{\Psi^{'} \in \rho(\varepsilon \diamond y)} ub_{rem}\left(\varepsilon,\ \Psi^{'}\right) \leqslant \sum_{\Psi^{'} \in \rho(\varepsilon \diamond y)} u\left(\Psi^{'}\right) \leqslant \sum_{\Psi^{'} \in \rho(y)} u\left(\Psi^{'}\right) = SWU(y) = LEU(y)$ because $\rho(\varepsilon \diamond y) \subseteq \rho(y), \forall \diamond \in \{\diamond_i, \diamond_s\}$. Thus, $LEU_{max}(\varepsilon \diamond y) \leqslant LEU(y)$.

For $\alpha \neq <>$ and $\varepsilon \neq <>$, consider any backward extension $\beta = \alpha \diamond \varepsilon \diamond y$ of $\alpha \diamond y$. For $\beta = \alpha \diamond_i \varepsilon \diamond_i y$ and $size(\varepsilon) = 1$, then $\beta = \alpha \diamond_i (\varepsilon \diamond_i y) \sqsupseteq \alpha \diamond_i y$, by Lemma 1, $\rho(\beta) \subseteq \rho(\alpha \diamond_i y)$, $LEU(\beta) = \sum_{\Psi^{'} \in \rho(\beta)} ub_{rem}\left(\alpha \diamond_i \varepsilon,\ \Psi^{'} \in\right) \leqslant \sum_{\Psi^{'} \in \rho(\alpha \diamond_i y)} ub_{rem}(\alpha, \Psi^{'}) = LEU(\alpha \diamond_i y)$; otherwise, then $\beta \sqsupseteq \alpha \diamond_s y$, so $\rho(\beta) \subseteq \rho(\alpha \diamond_s y)$, $LEU(\beta) = \sum_{\Psi^{'} \in \rho(\beta)} ub_{rem}(\alpha \diamond \varepsilon,\ \Psi^{'}) \leqslant \sum_{\Psi^{'} \in \rho(\alpha \diamond_s y)} ub_{rem}\left(\alpha,\ \Psi^{'}\right) = LEU(\alpha \diamond_s y)$. Thus, in all cases, we always have $LEU(\beta) \leqslant \max\{LEU(\alpha \diamond_i y), LEU(\alpha \diamond_i y)\} = LEU_{max}$
$(\alpha \diamond y)$, i.e. $AMB(LEU)$. Hence, we have $AMBiDi(LEU)$.

These properties of the *MEU, LEU*, and *SWU* UBs are illustrated with an example.

*Example 13* (***Different pruning and reducing effects of UBs***) Consider the database of the running example and $mu = 350$. It is first found that $u(\Psi_1^{'}) = 191$, $u(\Psi_2^{'}) = 131$, $MEU(d) = 471$ and $MEU(f) = 163 < LEU(g) = 228 < LEU(b) = 322 < mu$. Then, by $MEU$, the whole $branch(f)$ consisting of $f$ and all its (forward) extensions can be *deeply* pruned from the search tree early ($f$ is thus called a leaf node). Furthermore, using $LEU$ and $S_{LEU}(<>) = acdef$, we can remove from $D'$ (or *widely* prune) two irrelevant items in $IRS_{LEU}(<>) = \{b, g\}$ and reduce all UBs of remaining items in $S_{LEU}(<>)$. For example, we have the reduced values of $u\left(\Psi_1^{'}\right) = 191 - u((b, 6)) - u((b, 9)) = 176$, $u\left(\Psi_2^{'}\right) = 131 - u((b, 12)) - u((b, 9)) = 110$ and $MEU(d) = 471 - u((b, 9)) - u((b, 9)) - u((g, 2)) = 451$. After removing two irrelevant items $b$ and $g$ of $IRS_{LEU}(<>)$ from $D'$, a part of the prefix tree is shown in Fig. 1, where each node $\alpha$ together with its utilities are represented in the brief form $\alpha_{\rho(\alpha)}^{u_{max}, MEU, LEU}$.

Since $MEU(af) = 169 < mu$, $LEU(ad) = 286 < mu$ and $MEU(a \diamond_i x) = 469 \geqslant mu$, $\forall x \in \{c, e, f\}$, then $I_{LEU}(a) = cef$. Similarly, $MEU(a \rightarrow f) = 173 < mu$, $S_{LEU}(a) = af$, so $IS_{LEU}(a) = acef$ and $IRS_{LEU}(a) = S_{LEU}(<>) \setminus IS_{LEU}(a) = d$. Thus, we can deeply prune branches starting from nodes $ad$, $af$, $a \rightarrow c$, $a \rightarrow d$, $a \rightarrow e$, $a \rightarrow f$, and remove the irrelevant item $d$ from the PDB $D_a^{'}$ and reduce all UBs of $a$ and its remaining child nodes $ac$, $ae$ and $a \rightarrow a$. After the reduction, since two reduced values $MEU(ae) = 294$ and $MEU(a \rightarrow a) = 272$ are less than $mu$, we can additionally deeply prune two branches $branch(ae)$ and $branch(a \rightarrow a)$. Similarly, for the remaining $branch(ac)$ not yet pruned, we have $I_{LEU}(ac) = \{e\}$, $S_{LEU}(ac) = \{af\}$, so $IRS_{LEU}(ac) = IS_{SWU}(<>) \setminus IS_{LEU}(ac) = bcdg$. After discarding additionally the irrelevant item $c$ from $D_{ac}^{'}$ and reducing all UBs of $ac$, we obtain the reduced value $MEU(ac) = 294 < mu$. Thus, $branch(ac)$ is

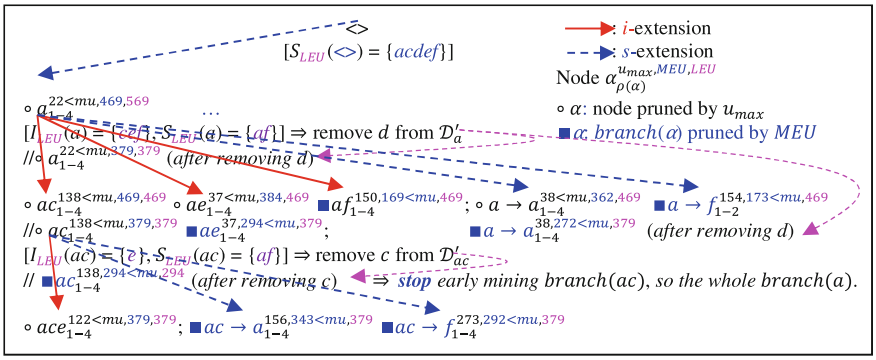


**Fig. 1** Illustration of pruning and reducing strategies

also pruned and we can stop mining $branch(a)$ early compared to if the *Red reducing strategy* was not used.

**Remarks**. Based on the proposed general framework for comparing UBs, some important remarks are presented.

a. The $SWU$-based width pruning strategy is applied in almost all algorithms for mining $HUS$. The $SWU$-based strategy for reducing PDBs and the SWU was proposed in [5, 8] and the strategy for reducing PDBs based on UBs satisfying the *AMBiDi* property has been utilized in [12]. The depth-pruning strategy based on UBs satisfying the *AMF* property has been proposed in [9–12].

Note that although the *Width Pruning Strategy* can be applied with both the $LEU$ and $SWU$ UBs, $LEU \ll SWU$, so $IRS_{SWU}(\alpha) \subseteq IRS_{LEU}(\alpha)$. It is thus sufficient to only use $LEU$ as $ub$ in Theorem 2c. In other words, $LEU$ is really better than the $SWU$ in terms of value as well as effect for pruning the search space and reducing other UBs.

b. Although $MEU$ is tighter than $LEU$, the latter has a stronger bi-direction pruning effect compared to the former because $LEU$ allows to additionally reduce PDBs and UBs as shown in Example 13. However, note that we cannot apply the reducing strategy $Red$ for $MEU$, i.e. $Red(MEU)$ is incorrect. *Indeed*, assume conversely that $Red(MEU)$ is true. Consider $mu = 350$ and $\alpha = d$. Since $MEU(df) = 84$ and $MEU(d \rightarrow f) = 286$ are less than $mu$ and we eliminate the irrelevant item $f \in IRS_{MEU}(d)$ from $D'_d$. Then, the sequence $\beta = d \rightarrow ac \rightarrow af$ containing $f$ cannot be found in all extensions of $d$, but this sequence is a high utility sequence since $u_{max}(\beta) = 353 > mu$, i.e. the HU sequence $\beta$ is missing in the final set $HUS$. Thus, simultaneously integrating both $MEU$ and $LEU$ into algorithms for mining $HUS$ is really necessary.

c. Due to the fact that a sequence may have multiple occurences in an input $q$-sequence in HUSM, designing an upper-bound $UB$ on $u_{max}$ is not trivial and it may thus result in making some mistakes. To ensure the correctness and usefulness of an UB $ub$, this latter should satisfy at least two properties: (1) it must really be an UB on $u_{max}$, and (2) it should be tighter than the $SWU$, which is the largest UB commonly

used in HUSM. If one tightens or reduces an upper bound $UB$ too much, it may not be an UB anymore, and algorithms based on $UB$ may miss some HU sequences. In other words, these algorithms are incomplete. On the other hand loosening an $UB$ too much may make it greater than the $SWU$. In that case, the upper bound may not be useful. *Indeed*, consider an integrated QSDB $D' = \{\Psi' = (a, 2)(c, 5)(e, 3) \rightarrow (b, 1) \rightarrow (b, 2) \rightarrow (d, 1) \rightarrow (b, 80) \rightarrow (c, 4)(e, 1) \rightarrow (d, 3) \rightarrow (c, 3)(e, 2)\}$.

(*i*). For example, for $\alpha = a \rightarrow b \rightarrow d \rightarrow c$, then $u_{max}(\alpha) = 88 < MEU(\alpha) = 90$. Consider a measure called *tub*,which is tighter than the $MEU$ and is defined as $tub(\alpha) = \sum_{\Psi' \in \rho(\alpha)}(u(\alpha, \Psi', i_p) + u(rem((\alpha, \Psi', i_p)))$, where $i_p$ is the *first ending* or *pivot* of $\alpha$ in $\Psi'$, $i_p = 6$. Then, $tub \ll MEU$ and $AMF(tub)$. Since $tub(\alpha) = 9 + 9 = 18$, $tub(\alpha) < u_{max}(\alpha) < MEU(\alpha)$, i.e. *tub* is not an UB on $u_{max}$. For $mu = 20$, then $tub(\alpha) < mu$. Hence, if the *tub*-based depth pruning strategy is applied to prune $\alpha$ and its extensions, then some HU sequences such as $\beta = \alpha \diamond_i e$ may be missing in the set $HUS$, because $u_{max}(\beta) = 90 > mu$.

The *tub* UB is commonly known as *SPU* (*Sequence-Projected Utilization*) and was introduced in the USpan algorithm [9, 14]. Because USpan relies on this upper-bound to reduce the search space, it can miss patterns and is thus an incomplete algorithm. This is important implications since several algorithms are derived from USpan and thus may also be incomplete. This includes algorithms such as TUS for top-k HU sequence mining [14], HUSP-NIV for mining HUSs with negative item values [15], PHUSM for mining periodic HUSs [16], HHUSP and MSPCF for hiding HUSs [17].

(*ii*). The $MEU$ and $LEU$ UBs are similar to the $PEU$ (*Prefix Extension Utility*) and $RSU$ (*Reduced Sequence Utility*) used in [11], but the two formers are more simple. Note that, $MEU$ only satisfies $AMF$, and does not satisfy $AMB$ and $AM$. *Indeed*, for the *backward* extension $b \rightarrow d$ of $d$, we have $MEU(b \rightarrow d) = 96 > MEU(d) = 94$, i.e. $not(AMB(MEU))$. This remark is important, since if $AM(MEU)$, the application of the reduced strategy *Red* for $MEU$ as discussed in Remark.b may lead to missing some HU sequences.

Similarly, consider another measure $tub'$, tighter than $MEU$, which is defined as $tub'(\alpha) = \sum_{\Psi' \in \rho(\alpha)} ub'_{rem}(\alpha, \Psi')$, where $ub'_{rem}(\alpha, \Psi') = max\{ub'_{rem}(\alpha, \Psi', i_p) \mid \forall\ ending\ i_p\ of\ \alpha\ in\ \Psi'\}$, $ub'_{rem}(\alpha, \Psi', i_p) = u(rem(\alpha, \Psi', i_p)) + u(\alpha, \Psi', i_p)$ if $u(rem(\alpha, \Psi', i_p)) > 0$, and otherwise $ub'_{rem}(\alpha, \Psi', i_p) = 0$. Then, $tub' \ll MEU$. For $\alpha = a \rightarrow b \rightarrow d \rightarrow ce$, we have $u_{max}(\alpha) = 90$, $MEU(\alpha) = 90$, $tub'(\alpha) = ub'_{rem}(\alpha, \Psi') = max\{max\{9; 10\} + 8;\ 0\} = 18$, because for the last occurrence $\alpha' = (a, 2) \rightarrow (b, 80) \rightarrow (d, 3) \rightarrow (c, 3)(e, 2)$ of $\alpha$ in $\Psi'$ according to the ending $i_p$=8, $u(rem(\alpha, \Psi', i_p))$=0, so $ub'_{rem}(\alpha, \Psi', i_p) = 0$. Thus, $tub'(\alpha) < u_{max}(\alpha) \leqslant MEU(\alpha)$. In other words, $tub'$ is not an UB on $u_{max}$, and it is also called the $PEU$ (*Prefix Extension Utility*) upper bound, and is used in Theorem 4 of [11] of the HUS-Span algorithm. Fortunately, when $u(rem(\alpha, \Psi', i_p)) = 0$, an extension of $\alpha$ in such $\Psi'$ will be terminated, thus $tub'$ only can result in missing patterns when the pruning condition "if ($tub'(\alpha) < mu$) then *stop mining branch*($\alpha$)" is executed before displaying the result sequence "if ($u_{max}(\alpha) \geqslant mu$) then *output* $\alpha$".

(*iii*). For $\alpha = P \rightarrow i$, where $P \equiv b$, $i \equiv d$, $S \equiv \Psi'$, an UB named *CRoM* [10] was defined as $CRoM(\alpha) \equiv CRoM(P, i) = RMUB(P, S, i) = u(P, S) + S^{ru}(i, m+1)$, where $m = CSeq_P^{last\_IS}(S, 1) = 2$ indicates that $b$ first appears in the $2^{nd}$ itemset of $S$, $u(P, S) = u_{max}(P, S) = max\{u((b, 1)), u((b, 2)), u((b, 80))\} = 80$ and the remaining utility of $i$ (including itself) in $S$ after the $3^{rd}$ itemset: $S^{ru}(i, m+1) = u((d, 1) \rightarrow (b, 80) \rightarrow (c, 4)(e, 1) \rightarrow (d, 3) \rightarrow (c, 3)(e, 2)) = 94$. Thus, $CRoM(\alpha) = 80 + 94 = 174$. Meanwhile, $SWU(\alpha) = u(\Psi') = 107$ and $MEU(\alpha) = 96$. Hence, $CRoM(\alpha) > SWU(\alpha) > MEU(\alpha)$. In other words, the *CRoM* UB is larger than the $SWU$. Thus, in some cases, this UB may not be useful for pruning the search space.

## 2.3 Algorithms

Early HUSM algorithms discover HU sequences in two phases. UL and US [2] are such algorithms, which perform a breadth-first and depth-first search, respectively. In the first phase, they find the set $HUS_{SWU}$ of all HU sequences w.r.t. $SWU$. In the second phase, they calculate the utility of sequences by scanning the QSDB to output $HUS_{SWU}$ only those having a utility ($u_{max}$) that is no less than the threshold *minutil*. Two-phase algorithms have two important limitations, especially for low *minutil* values. The first one is that a considerable amount of memory may be spent to store the set $HUS_{SWU}$. The second limitation is that scanning the QSDB to compute the utility of candidate sequences found in the first phase can be very time-consuming.

To overcome these two limitations, HU candidate sequences are maintained in a *prefix tree* which consists of the null sequence as its root, and where each tree node represents a candidate sequence and each child node is its *i*- or *s-extension*. Each node (or sequence) in the prefix tree is stored in a utility-based data structure such as a utility-matrix [9], temporal sequence (TS) table [8], utility lists [10] or utility chains [11], and utility-linked (UL)-lists *CSeq* [12]. These data structures represent a pattern (sequence) by storing not only the sequence identifiers (SID) of input-sequences containing it, but also information about its utility and remaining utility. This information allows quickly computing the actual utility $u_{max}$ and all UBs of a considered sequence without scanning QSDB or PDBs.

In more details, a QSDB $D'$ can be represented in vertical database format (VDF), where each item $x$ is associated with a *utility-chain* structure named $UC(x)$ [11]. This structure is an extension of the *IDList* structure used in FSM [18]. Especially, the bitset implementation of *IDList* [19] has been used and proved its efficiency in terms of *execution time and memory* consumption in many well-known algorithms for frequent sequence mining such as ClaSP [20], CM-ClaSP [21], FCloSM and FGenSM [22], FGenCloSM and MaxGenCloSM [23]. For a given sequence $\alpha$, the structure $UC$ of $\alpha$ is defined as $UC(\alpha) = \{(SID, UL) \mid \Psi'_{SID} \in D' \text{ and } \Psi'_{SID} \sqsupseteq \alpha\}$, where the utility list $UL = \{tup(end) = (end, u, u_{rem})\}$ is a list of tuples according to each ending *end* of $\alpha$ in $\Psi'_{SID}$ with $u = u(\alpha, \Psi'_{SID}, end)$ and $u_{rem} = u(rem(\alpha, \Psi'_{SID}, end))$.

| Sid | ***a***: $UL$={($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (1,2,189), (2,3,175), (3,5,164), (4,5,109), (5,4,56) |
| 2 | (2,2,117), (3,3,88), (4,1,67), (5,4,24) |
| 3 | (2,4,33), (3,1,18) |
| 4 | (2,7,141), (3,2,83), (4,9,72) |

| Sid | ***b***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (2,6,169), (4,9,100) |
| 2 | (1,12,119), (5,9,15) |

| Sid | ***c***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (1,5,184), (4,40,60), (5,10,46) |
| 2 | (2,20,97), (4,20,47), (5,15,0) |
| 3 | (1,20,37), (2,10,23) |
| 4 | (2,50,91) |

| Sid | ***d***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (3,50,114), (5,10,36) |
| 2 | (3,20,68), (4,10,37) |
| 4 | (1,80,148) |

| Sid | ***e***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (1,6,178) |
| 2 | (2,6,91) |
| 3 | (2,4,19) |
| 4 | (2,6,85) |

| Sid | ***f***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 1 | (5,36,0) |
| 2 | (4,9,28) |
| 3 | (3,18,0) |
| 4 | (4,72,0) |

| Sid | ***g***:{($end$, $u$, $u_{rem}$)} |
|---|---|
| 4 | (3,2,81) |

**Fig. 2** The vertical representation of the integrated QSDB$D'$

For example, as shown in Fig. 2, we have $\rho(a) = \{\Psi_i^{'}, i = 1, \ldots, 4\}$. The sequence $a$ appears in the 2nd and 3rd itemsets of $\Psi_3^{'}$. According to endings $end_1 = 2$ and $end_2 = 3$, the first ending is $end_1$, $u = u(a,\ \Psi_3^{'},\ 2) = 4$, $u_{rem} = u((c, 10)(e, 4) \rightarrow (a,\ 1)(f,\ 18)) = 33$ and we obtain the first tuple $tup(end_1) = (2,\ 4,\ 33)$ and similarly the second tuple $tup(end_2) = (3,\ 1,\ 18)$. Thus, the $UL$ according to $\Psi_3^{'}$, ($SID = 3, UL$), in $UC(a)$ is $\{(2,\ 4,\ 33), (3,\ 1,\ 18)\}$. Other $UL$s are computed in the same way.

However, how can the $UC$ structure of a sequence $\alpha$ be used to calculate its utility $u_{max}$ and its UB values? During the first QSDB scan, $\rho(\alpha) = \{\Psi_i^{'} \mid (i, UL) \in UC(\alpha)\}$, $SWU(\alpha) = \sum_{(i,UL)\in UC(\alpha)} u(\Psi_i^{'})$ and all values $\{u(\Psi_i^{'}), \Psi_i^{'} \in D^{'}\}$ are computed once, $u_{max}(\alpha) = \sum_{(i,UL)\in UC(\alpha)} u_{max}(\alpha,\ \Psi_i^{'})$, $MEU(\alpha) = \sum_{\Psi_i^{'}\in\rho(\alpha)} ub_{rem}(\alpha, \Psi_i^{'})$, where for each $(i, UL) \in UC(\alpha)$ or $\Psi_i^{'} \sqsupseteq \alpha$, $u_{max}(\alpha,\ \Psi_i^{'}) = max\{tup.u \mid tup \in UL\}$, $ub_{rem}(\alpha, \Psi_i^{'}) = max\{tup.u + tup.u_{rem} \mid tup \in UL\}$, and the value $LEU(\alpha)$ of $\alpha = \delta \diamond y$ is computed based on the already calculated value $MEU(\delta)$ or $SWU(\alpha)$ depending on whether $\delta$ is the non-null sequence or not, respectively. For example, for $\alpha = b$, we have $u_{max}(\alpha) = max\{6,\ 9\} + max\{12,\ 9\} = 21, MEU(\alpha) = max\{6 + 169,\ 9 + 100\} + max\{12 + 119,\ 9 + 15\} = 175 + 131 = 306, LEU(\alpha) = SWU(\alpha) = 191 + 131 = 322$, because the prefix of $b$ is null.

Another important question is how to calculate the $UC(\beta)$ of the $i$-extension $\beta = \alpha\diamond_i y$ with $y \succ lastItem(\alpha)$ (or $s$-extension $\beta = \alpha\diamond_s y$) of a sequence $\alpha$ with an item $y$ based on their $UC$s, $UC(\alpha)$ and $UC(y)$. For any sequence $\alpha$ and each element ($SID, UL$) of $UC(\alpha)$, we denote briefly any tuple ($end, u,\ u_{rem}$) in the tuple list $UL$ as $t(\alpha, SID)$ (or briefly $t(\alpha)$ in the unambiguous context for each fixed

$SID$ and $t(\alpha).end = end$), $UC(\alpha).SIDs = \{SID \mid (SID,\ UL) \in UC(\alpha)\}$. For each $SID \in UC(\alpha).SIDs \cap UC(y).SIDs$ and each fixed $t(y)$, consider all $t(\alpha) \in UL(\alpha)$ in $(SID,\ UL(\alpha)) \in UC(\alpha)$ such that $t(\alpha).end = t(y).end$ (or $t(\alpha).end < t(y).end$, respectively) and create the corresponding tuple $t(\beta)$ as follows: $t(\beta).end = t(y).end$, $t(\beta).u_{rem} = t(y).u_{rem}$ and $t(\beta).u = t(\alpha).u + t(y).u$ (or $t(\beta).u = max\{t(\alpha).u + t(y).u \mid t(\alpha).end < t(y).end\}$, respectively). Then, we add the new tuple $t(\beta) = (t(\beta).end, t(\beta).u, t(\beta).u_{rem})$ to the utility list $UL$ of the element $(SID, UL)$ of $UC(\beta)$.

For example, *first*, consider $\alpha = a$, $y = b$ and the $i$-extension $\beta = \alpha \diamond_i y = ab$, shown in Fig. 3a. There exist two $SID \in UC(a).SIDs \cap UC(b).SIDs = \{1, 2\}$. For instance, according to $SID = 1$, there exist two pair of tuples $(t(a), t(b))$ such that $t(a).end = t(b).end \in (\{2,\ 4\})$. For example, the first pair is $t(a) = (2,\ 3,\ 175)$ and $t(b) = (2, 6, 169)$, then $t(a).end = t(b).end = 2$, so $t(\beta).end = 2$, $t(\beta).u = 3 + 6 = 9$, $t(\beta).u_{rem} = t(b).u_{rem} = 169$. For the second pair, $t(a) = (4, 5, 109)$ and $t(b) = (4, 9, 100)$, then $t(a).end = t(b).end = 4$, so $t(\beta).end = 4$, $t(\beta).u = 5 + 9 = 14$, $t(\beta).u_{rem} = t(b).u_{rem} = 100$, *i.e.* the sequence $\beta = ab$ appears two times in $\Psi_1'$ according two sub $q$-sequences $(a,\ 3)(b, 6)$ and $(a, 5)(b, 9)$ with the corresponding utilities of 9 and 14, respectively. Thus, we obtain the first element $ele_1 = (SID = 1, \{(2, 9, 169), (4, 14, 100)\})$ of $UC(ab)$. Similarly, we also have the second element $ele_2 = (SID = 2, \{(5, 13, 15)\})$ of $UC(ab)$ and finally, $UC(ab) = \{ele_1, ele_2\}$. Thus, $u_{max}(\beta) = max\{9, 14\} + max\{13\} = 27$, $MEU(\beta) = max\{9 + 169, 14 + 100\} + max\{13 + 15\} = 178 + 28 = 206$, $SWU(\beta) = 191+131=322$, $LEU(\beta) = ub_{rem}(a,\ \Psi_1') + ub_{rem}(a,\ \Psi_2') = max\{2 + 189, 3 + 175, 5 + 164,\ 5 + 109, 4 + 56\} + max\{2 + 117, 3 + 88, 1 + 67, 4 + 24\} = 191 + 119 = 310$ and $u_{max}(\beta) < MEU(\beta) < LEU(\beta) < SWU(\beta)$.

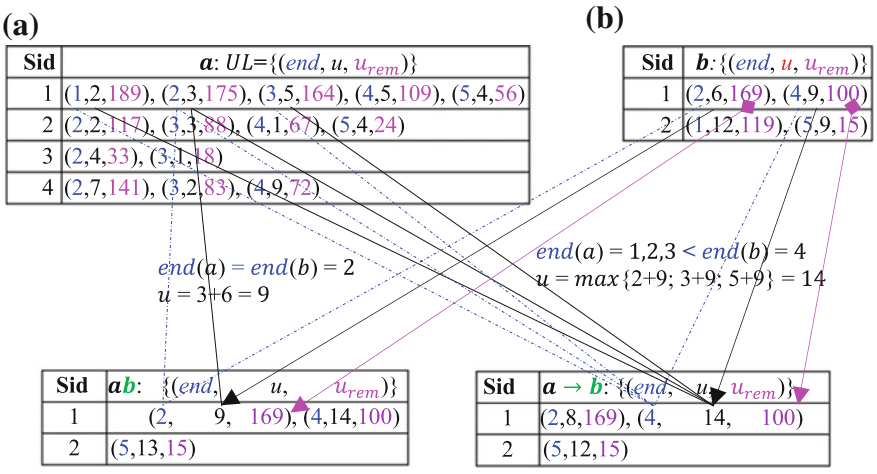


**Fig. 3** **a** The $UC(ab)$. **b** The $UC(a \rightarrow b)$

*Second*, the $UC$ of the $s$-extension of $\alpha = a$ with $y = b$, $\beta = a \rightarrow b$, is shown in Fig. 3b. There are also two $SID \in UC(a).SIDs \cap UC(b).SIDs = \{1, 2\}$. For example, with $SID = 1$, for the first tuple $t(b) = (2, 6, 169)$, then there is the unique tuple $t(a) = (1, 2, 189)$ such that $t(a).end = 1 < t(b).end = 2$, so $t(\beta).end = t(b).end = 2$, $t(\beta).u_{rem} = t(b).u_{rem} = 169$, $t(\beta).u = 2 + 6 = 8$ and we receive the first tuple $(2, 8, 169)$ of the utility list $UL(\beta)$ according to $SID = 1$. Moreover, for the second tuple $t(b) = (4, 9, 100)$, then there are three tuples $t(a) \in \{(1, 2, 189), (2, 3, 175), (3, 5, 164)\}$ such that $t(a).end < t(b).end = 4$, so $t(\beta).end = t(b).end = 4, t(\beta).u = max\{2 + 9, 3 + 9, 5 + 9\} = 14$, $t(\beta).u_{rem} = t(b).u_{rem} = 100$. We obtain thus the second tuple $(4, 14, 100)$ of $UL(\beta)$ and the first element of $UC(a \rightarrow b)$, $ele_1^{'} = (SID = 1, \{(2, 8, 169), (4, 14, 100)\})$. In the same way, we also have the second element $ele_2^{'} = (SID = 2, \{(5, 12, 15)\})$ of $UC(a \rightarrow b)$. Hence, $UC(a \rightarrow b) = \left\{ele_1^{'}, ele_2^{'}\right\}$. Hence, $u_{max}(\beta) = max\{8, 14\} + max\{12\} = 26$, $MEU(\beta) = max\{8 + 169, 14 + 100\} + max\{12 + 15\} = 177 + 27 = 204$, $SWU(\beta) = 191 + 131 = 322$, $LEU(\beta) = ub_{rem}(a, \Psi_1^{'}) + ub_{rem}(a, \Psi_2^{'}) = 191 + 119 = 310$ and $u_{max}(\beta) < MEU(\beta) < LEU(\beta) < SWU(\beta)$.

Based on the $UC$ structure and an UB that provides the *depth* pruning ability $DepthUB$ (e.g. the $MEU$), $WidthUB$ such that $WidthUB(y) = SWU(y), \forall y \in A$ (e.g. $LEU$), which has the *width* pruning ability, and modifying the USpan algorithm [9], we present a complete algorithm named HUSPM for mining high utility sequential pattern. The pseudocode of the main procedure of this algorithm is given in Fig. 4.

First, it scans the QSDB $D'$ to compute the vector $ISU = (u(\Psi'), \Psi' \in D')$ used to compute the $SWU$ of items, the set $S_{WidthUB}(<>)$ of all HU candidate items w.r.t. $WidthUB$ (line 1) and discard from $D'$ all irrelevant *low-WidthUB* items in $IS_{WidthUB}(<>) = A \setminus S_{WidthUB}(<>)$ (line 2). Then, the recursive procedure SearchHUS is called for each candidate items in $S_{WidthUB}(<>)$.

In the first line in SearchHUS($\alpha$, $IS$, $mu$) (Fig. 5), if $u_{max}(\alpha) \geqslant mu$, the HUS $\alpha$ is output. Next, if $DepthUB(\alpha) < mu$, then the whole $branch(\alpha)$ is deeply pruned by $DepthUB$, i.e. we can stop SearchHUS and backtrack the search procedure. Based on $WidthUB$, the width pruning strategy is applied by the procedure WidthPruning_ReducingPDB shown in Fig. 6. Afterward, if the reduced value $DepthUB(\alpha) < mu$, then we can stop mining the $branch(\alpha)$ (line 4). Then, Search-

$\mathcal{HUS}$ **HUSPM**($\mathcal{D}'$, $mu$)
. *Input*: QSDB $\mathcal{D}'$, $mu$.
. *Output*: all HU sequences of $\mathcal{HUS}$.
1. Scan once $\mathcal{D}'$ to calculate the vector $ISU$ and $S_{WidthUB}(<>)$ := {*high-WidthUB* items};
2. **Remove** from $\mathcal{D}'$ all *irrelevant* items in $\mathcal{IS}_{WidthUB}(<>) = \mathcal{A} \setminus S_{WidthUB}(<>)$;
3. **for** each item $i \in S_{WidthUB}(<>)$ **do**
4. **SearchHUS**($i$, $S_{WidthUB}(<>)$, $mu$);

**Fig. 4** The HUSPM algorithm for mining the $\mathcal{HUS}$ set

**SearchHUS**($\alpha$, $IS$, $mu$)
. *Input*: pattern $\alpha$, extension candidate set $IS$, threshold $mu$.
. *Output*: HU sequence $\alpha$ in $\mathcal{HUS}$.
1. **if** ($u_{max}(\alpha) \geqslant mu$) **then** output HU sequence $\alpha$;
2. **if** ($DepthUB(\alpha) < mu$) **then return**; // *deeply pruning branch(α) by DepthUB*
3. **WidthPruning_ReducingPDB**($\alpha$, $IS$, $WidthUB$, $mu$, $newI$, $newS$);
4. **if** ($DepthUB(\alpha) < mu$) **then return**; // *deeply pruning branch(α) by reduced DepthUB*
5. $newIS := newI \cup newS$;
6. **for** each item $i \in newS$ **do**
7. **SearchHUS**($\alpha \diamond_s i$, $newIS$, $mu$);
8. **for** each item $i \in newI$ **do**
9. **SearchHUS**($\alpha \diamond_i i$, $newIS$, $mu$);

**Fig. 5** The SearchHUS procedure

**WidthPruning_ReducingPDB**($\alpha$, $IS$, $WidthUB$, $mu$, $newI$, $newS$)
. *Input*: pattern $\alpha$, candidate item set $IS$ for extensions of $\alpha$'s prefix, measure $WidthUB$, threshold $mu$.
. *Output*: two candidate item sets $NewI$ and $newS$ for $i$- and $s$- extensions of $\alpha$, PDB $\mathcal{D}'_\alpha$ and UBs of α are *reduced.*
1. **for** each item $i \in IS$ **do** { // *candidate items for extensions of α based on WidthUB*
2. **if** ($i \succ lastItem(\alpha)$ and $WidthUB(\alpha \diamond_i i) \geqslant mu$) **then** $newI := newI \cup \{i\}$;
3. **if** ($WidthUB(\alpha \diamond_s i) \geqslant mu$) **then** $newS := newS \cup \{i\}$;
4. }
5. **for** each item $i \notin (newI \cup newS)$ **do** // *reduce PDB* $\mathcal{D}'_\alpha$
6. **remove** $i$ from $\mathcal{D}'_\alpha$; // *remove irrelevant item* $i$ from $\mathcal{D}'_\alpha$
7. **Reduce** UBs of $\alpha$ based on reduced $\mathcal{D}'_\alpha$;

**Fig. 6** The WidthPruning_ReducingPDB procedure

HUS is recursively called for extensions of $\alpha$ with items in $newS$ and $newI$ (lines 6–9). In WidthPruning_ReducingPDB, lines 1–4 search two candidate item sets for $i$- and $s$- extensions of $\alpha$, $NewI$ and $NewS$, based on the set $IS$ of its prefix. Next, the strategy for reducing PDBs and UBs is utilized in lines 5–7.

Consider Example 1 and $mu = 350$. By applying HUSPM, we obtain $HUS = \{d \rightarrow ac \rightarrow af\}$ with $u_{max}(d \rightarrow ac \rightarrow af) = 353$.

Note that, although the HUSPM algorithm is designed for HUSM, however since it is only based on two general measures $DepthUB$ and $WidthUB$ which have respectively *depth* and *width* pruning abilities, and $Red(WidthUB)$, it is easy to modify HUSPM or extend it to mine other *interesting* types of HU sequences such as HU-probability sequences from uncertain QSDBs or HU sequences with multiple minimum utility thresholds as shown in Sect. 3.

**Other algorithms for HUSM**. Algorithms for HUSM in [1, 2] named UL and US are applied on static and dynamic web log 1-QSDBs, using an incremental IUWAS-tree structure, where the utility of a sequence is computed in sum or maximum form depending if a sequence has multiple occurrences or not in an input $q$-sequence. Calculation is complex because the algorithm must first determine which situation

holds for a given sequence and -sequence. Moreover, if the sum of utilities of all distinct occurrences in a $q$-sequence is used, the patterns may be influenced by the personal buying behaviors of some customers and be less representative of the behavior of most customers. But obtaining an overview of the behavior of all customers is often the goal of HUSM, when applied to market basket data. The UL and US algorithms can perform multiple QSDB or PDB scans. Because they are two-phase algorithms, they can consume a large amount of memory to maintain the set of high-SWU sequences as well as much time to calculate the actual utility of all sequences by scanning again the QSDB.

Using the maximum utility and a database-projection projection approach, Shie has proposed a one-phase algorithm named UM-Span [4] for HUSM in mobile commerce environments. It only considers 1-QDSBs associated with paths of location IDs where each path has only one utility value. Similarly, in [8], a one-phase algorithm named PHUS is proposed, which uses a database projection approach, a temporal sequence (TS) table structure and an improved strategy by removing unpromising (or low-SWU) items from PDBs. All above algorithms only use the SWU as UB to prune irrelevant candidate sequences. However, since the SWU UB is still quite loose, these algorithms may generate too many candidates.

In [9], the authors have proposed the USpan algorithm for the general problem of HUSM using three UBs, namely the SWU, SPU (*Sequence-Projected Utilization*) and SRU *(Sequence-Reduced Utility*) [24] having *width* and *depth* pruning effects. Using the LQS-tree to represent the search space and a utility matrix structure, the USpan algorithm is designed to efficiently mine HU sequences. Unfortunately, as demonstrated in this chapter, the SPU is not really an upper bound on $u_{max}$. Thus, using them to prune candidate sequences may result in missing some HU sequences. In other words, USpan is an incomplete algorithm.

The HupsExt algorithm and an efficient strategy for pruning candidates before candidate pattern generation based on an upper bound named CRoM was introduced in [10]. However, the CRoM upper bound has the drawback that it can be larger than the SWU.

In [11], two UBs that are tighter than the SWU, named PEU (*Prefix Extension Utility*) and RSU (*Reduced Sequence Utility*) have been proposed. Based on them and utility-chain ($UC$) structure was defined and the algorithm HUS-Span was proposed for HUSM. However, as shown in the discussion, the PEU is not an UB on $u_{max}$. Thus, HUS-Span can also miss patterns.

To more efficiently mine top-$k$ HU sequences (see Sect. 3.3) as well as HU sequences, Lin et. al. have introduced three pruning strategies named MEUs (*Maximal Extension Utility Strategy*), LAS (*Look Ahead Strategy*) and IPS (*Irrelevant Item Pruning Strategy*) based on the $MEU$ UB [12], which is tighter than the $SWU$. The three strategies are the $DPS$, $WPS$ and $Red$ strategies shown above.

Table 6 presents some recent algorithms for mining HU sequences and their characteristics.

**Table 6** HUSM algorithms

| Algorithm | Search type | Number of phases | Data representation | UBs |
|---|---|---|---|---|
| UL, US [2] | breadth-first, depth-first | Two | Horizontal database | SWU |
| UM-Span [4] | depth-first | One | Prefix tree | SWU |
| PHUS [8] | depth-first | One | Temporal Sequence Table | SWU |
| USpan [9] | depth-first | One | LQS-tree, Utility Matrix | SWU, SPU, SRU [24] |
| HupsExt [10] | depth-first | One | Prefix tree | SWU, CRoM |
| hline HUS-Span [11] | depth-first | One | Utility-chain | SWU, PEU, RSU |
| HUSPM | depth-first | One | Utility-chain | MEU, LEU [12] |

## 3 Extensions of the Problem

Several extensions of the HUSM problem have been proposed. This section provides an overview of the main extensions.

### *3.1 Mining High Utility-Probability Sequential Patterns in Uncertain Databases*

A limitation of HUSM is that it is focused on mining HU sequences in precise data. But in real-world, data is often uncertain. This can be the case for data collected from sensors in wireless networks that are inaccurate due to the quality of sensors or because they are operating in a noisy environment. To address this limitation, the problem of high utility-probability sequence mining (HUPSM) in uncertain sequence databases (USDBs) was introduced [25]. Different from HUSM, in HUPSM, each input $q$-sequence $\Psi^{'}$ in an USDB is associated with a positive existence probability $p(\Psi^{'})$. The probability of a sequence $\alpha$ is defined and denoted as $p(\alpha) = \sum_{\Psi' \in \rho(\alpha)} p(\Psi^{'})/|PS|$, where $PS = \sum_{\Psi \in' D'} P(\Psi')$ is the probability sum of all input $q$-sequences in $D'$ and $p\,(\alpha) \in [0; 1]$. Then, $\alpha$ is called a *high utility-probability* sequence (HUPS) if $u_{max}(\alpha) \geqslant mu$ and $p(\alpha) \geqslant mp$, where $mp \in ((0; 1])$ is a user-specified minimum expected support threshold (or minimum probability threshold). The problem of HUPSM is to discover all HUPSs. In the case where $mp = min\left\{p(\Psi^{'}) \mid \Psi^{'} \in USDB\right\}/|PS|$, or all probabilities $p(\Psi^{'})$ are equal to a constant $p$ (i.e. all $\Psi^{'} \in USDB$ have the same importance) and $mp = p/|PS|$, we obtain the normal problem of HUSM. That is, HUPSM is an extension of HUSM that generalizes HUSM.

Since the operator $\rho$ is anti-monotonic (i.e. $\rho(\alpha) \supseteq \rho(\beta)$, $\forall \beta \sqsupseteq \alpha$), we have $AM(p)$. Based on $AM(p)$ and $AM(SWU)$, an algorithm named P-HUSPM (Projection HUSPM) has been proposed in [25]. Note that, since $AM(p)$, the width pruning strategy $WPS(p)$ and the reducing PDB strategy $Red(p)$ in Corollary 1 can be additionally applied for the probability measure $p$ by inserting a procedure like WidthPruningReducingPDB($\alpha$, $IS$, $p$, $mp$, $newIp$, $newSp$) for the probability $p$ after the line 3 of the SearchHUS procedure. Integrating this procedure into SearchHUS for HUPSM, replacing $S_{SWU}(<>)$ with $S_{SWU-prob}(<>) = \{a \in A \mid SWU(a) \geqslant mu \wedge p(a) \geqslant mp\}$, adding $(p(\alpha) \geqslant mp)$ into the condition in line 3 and calling ($newI$, $newS$) as one of two pairs ($newI$, $newS$) and ($newIp$, $newSp$) such that it has the smallest total size, we could then expect that it would prune much more unpromising sequences compared to P-HUSPM, because $WidthUB \ll SWU$ and using additionally both strategies $DPS(DepthUB)$ and $WPS(WidthUB)$, $Red(WidthUB)$ is better than utilizing only $WPS(SWU)$ and $Red(SWU)$.

## 3.2 High-Utility Sequential Pattern Mining with Multiple Minimum Utility Thresholds

Another important limitation of HUSM is that it finds all HU sequences under a single minimum utility threshold, so that all items in sequences are treated as having the same importance. But this issue is not suitable for many real-word applications. To deal with this issue, the problem of HUSM with multiple minimum utility thresholds was proposed [12]. To avoid missing items that are rare but important, each item $a$ in a QSDB is associated with a minimum utility threshold $mu(a)$. The minimum utility threshold of a sequence $\alpha$, denoted as $MIU(\alpha) = min\{mu(x) \mid x \in \alpha\}$ is the least $mu$ value among all its items. Different from traditional HUSM (with a single minimum utility threshold), in the problem (MultiMU_HUSM) of mining HU sequences with multiple minimum utility thresholds, a sequence $\alpha$ is called a HU sequence if $u_{max}(\alpha) \geqslant MIU(\alpha)$ and we must find the complete set $HUSP = \{\alpha \mid u_{max}(\alpha) \geqslant MIU(\alpha)\}$ of all HU sequential patterns. To efficiently prune the search space by preserving a downward closure-like property for any upper bound $ub$ on $u_{max}$ such that the $AMF(ub)$ property which is weaker than $AM(ub)$ holds, we only need to replace the fixed threshold $mu$ of HUSM with a dynamic potential minimum utility threshold for each sequence $\alpha$, defined as $PMIU(\alpha) = min\{mu(x) \mid x \in \alpha \vee (x \in rem(\alpha, \Psi', first\ ending(\alpha, \Psi')) \wedge \Psi' \in \rho(\alpha))\}$. In more details, if $ub(\alpha) < PMIU(\alpha)$, then for all forward extensions (but not super sequences) $\beta$ of $\alpha$, $u_{max}(\beta) < MIU(\beta)$, i.e. all extensions of $\alpha$ (including itself) cannot be HU sequences. *Indeed*, because $AMF(ub)$, $PMIU(\alpha) \leqslant MIU(\alpha)$ and $PMIU$ is monotonic, so $ub(\alpha) < PMIU(\alpha) \leqslant MIU(\alpha)$ and $u_{max}(\beta) \leqslant ub(\beta) \leqslant ub(\alpha) < PMIU(\alpha) \leqslant PMIU(\beta) \leqslant MIU(\beta)$. Note that, in the particular case where all items have the same importance $mu$, we obtain the traditional HUSM problem, i.e. MultiMU_HUSM is an extension of HUSM and is more general than

HUSM. To obtain the set $HUSP$ based on the procedure SearchHUS, we should replace the conditions "if $(u_{max}(\alpha) \geqslant mu)$ then" or "if $(ub(\alpha) \geqslant mu)$ then" for an UB $ub$ with "if $(u_{max}(\alpha) \geqslant MIU(\alpha))$ then" or "if $(ub(\alpha) \geqslant PMIU(\alpha))$ then", respectively.

### 3.3 Top-k High Utility Sequential Pattern Mining

Although algorithms for HUSM can discover all HU sequences under a predefined minimum utility threshold $mu$, it is very difficult for users to determine a suitable threshold $mu$ for obtaining the most valuable patterns. Due to the complexity of QSDBs and the sensitivity of the threshold, for a same threshold, some QSDBs may produce millions of sequences while other QSDBs may generate nothing. A challenge is thus to tune the threshold for obtaining a specified number of interesting patterns. But this is not easy since choosing an appropriate threshold requires being familiar with database characteristics which are usually invisible to users. Thus fine-tuning the threshold to obtain enough but not too many patterns can be very time-consuming.

Top-$k$ high utility sequential pattern mining addresses this problem by letting users specify the desired number of top-$k$ HU sequences instead of setting a threshold. A sequence $\alpha$ is called a *top-k high utility* sequence if there are less than $k$ sequences whose utilities are no less than $u_{max}(\alpha)$. The problem of top-$k$ HU sequence mining (top-$k$ HUSM) is to discover the complete set $T$ of top-$k$ HU sequences. The task of top-$k$ HUSM has been applied to gene regulation data [6]. Call $mu^* = min\{u_{max}(\alpha) | \alpha \in T\}$ the optimal minimum utility threshold to find the top-$k$ HU sequences. Then the problem of (top-$k$ HUSM) is to find all sequences $\alpha$ such that $u_{max}(\alpha) \geqslant mu^*$. The main solution for this problem is to design effective strategies which allow to raise as fast as possible the threshold $mu$ to $mu^*$ during the mining process while not missing any top-$k$ HU sequence by only pruning parts of the search spaces that do not contain top-$k$ HU sequences. Based on the USpan algorithm [9], the TUS algorithm was proposed for top-$k$ HUSM [14]. It uses a fixed-size sorted list named TUSList to dynamically maintain the top-$k$ high utility sequential patterns, and a temporal threshold $mu$ to prune unpromising candidates. Moreover, TUS utilizes a pre-insertion strategy to effectively raise $mu$ to a reasonable level before starting the mining process and a SPU-based sorting concatenation order strategy. In the pre-insertion strategy, all input $q$-sequences and all items (1-sequences) together with their utilities are inserted into TUSList. This strategy can reduce the number of unpromising candidates that are generated. In the SPU-based sorting concatenation order strategy, concatenation items having larger SPU upper bound values are extended first expecting that corresponding sequences will have a high utility and may thus help to raise the $mu$ threshold faster.

In [11], based on the HUS-Span algorithm for HUSM and different search strategies, authors have proposed the breadth first search-based algorithm TKHUS-Span$_{BFS}$ and the hybrid search-based algorithm TKHUS-Span$_{Hybrid}$ for efficiently mining top-$k$ HU sequences. In experiments, the former has proved that it is faster

because it generates less candidates. However it runs out of memory in some cases. In situations where memory is limited, the later can be applied to achieve a better performance. It is clear that for any $mu$, if $k = |HUS(mu)|$, the problem of top-$k$ HUSM is equivalent to the problem of HUSM.

### 3.4 Mining Periodic High Utility Sequential Patterns

Well-known algorithms for HUSM often discover a huge number of HU sequences but many of those are irrelevant for some applications. Mining the top-$k$ HU sequences is a solution to this problem. Other solutions consist of using other measures to assess how interesting a sequence is. For instance, in a recent study, it was proposed to find HUSs that periodically appear in a QSDB. An application of this definition is market basket analysis, where a retail store manager may want to identify sets of products that are regularly purchased. Periodic patterns can provide interesting insights about customers' behavior and be useful to develop or adapt marketing strategies. For each sequence $\alpha$, assume that $\rho(\alpha) = \left\{\Psi' \in D' \mid \Psi' \sqsupseteq \alpha\right\} = \{\Psi'_{i_1}, \Psi'_{i_2}, \ldots, \Psi'_{i_k}\}$, where $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant N = |D'|$. Two $q$-sequences $\alpha'_p$, $\alpha'_q$ in $\rho(\alpha)$ with $p < q$ are said to be consecutive w.r.t. $\alpha$ if $\nexists \alpha'_r \in \rho(\alpha)$ such that $p < r < q$. Their period is denoted and defined as the number of $q$-sequences between $\alpha'_p$ and $\alpha'_q$, that is $pe(\alpha'_p, \alpha'_q) = q - p$. The periods of $\alpha$ is a list of periods denoted as $ps(\alpha) = \{i_1 - i_0, i_2 - i_1, \ldots, i_{k+1} - i_k\} = \bigcup_{0 \leqslant j \leqslant k} \{i_{j+1} - i_j\}$, where $i_0 = 0$ and $i_{k+1} = N$. The maximum, minimum and average periodicity of $\alpha$ are respectively defined and denoted as $maxper(\alpha) = max(ps(\alpha))$, $minper(\alpha) = min(ps(\alpha))$, $avgper(\alpha) = (\sum_{p \in ps(\alpha)} p)/|ps(\alpha)|$. It is clear that $avgper(\alpha) = |D'|/(\rho(\alpha) + 1)$. A sequence is called a periodic HU sequence (PHUS) if $(maxper(\alpha) \leqslant maxPer$, $minper(\alpha) \geqslant minPer$ and $minAvg \leqslant avgper(\alpha) \leqslant maxAvg)^{(*)}$, where $maxPer$, $minPer$, $maxAvg$ and $minAvg$ are user-predefined thresholds. The problem of periodic HU sequence mining (PHUSM) is to find all PHUSs. Based on the USpan algorithm [9], a post-processing algorithm named PHUSPM [16] was designed for PHUSM by replacing simply the condition "if $(u_{max}(\alpha) \geqslant mu)$" in USpan with "if $(u_{max}(\alpha) \geqslant mu$ and $^{(*)})$". If setting $maxPer = maxAvg = N$, $minPer = minAvg = 1$, the problem of PHUSM is equivalent to HUSM.

Moreover, to reduce the number of HU sequences and consider specific requirements of users, another extension of HUSM is the problem of HUSM with *constraints* proposed in [5]. In that study, an algorithm named IM-Span was proposed for mining interesting mobile sequential patterns by pushing constraints in terms of utility, support and patterns deeply into the mining process. When constraints are set to specific values, the traditional problem of HUSM is obtained.

### 3.5 Related Problems

**High Utility Episode Mining**. A problem related to HUSM is that of mining all high utility episodes (HUE) in a complex event sequences (*CS*), where each complex event is a $q$-element associated with a time point [26] and the utility of an episode $\alpha$ is calculated as the sum of utilities of $q$-elements according to minimal occurences of $\alpha$ in *CS*. By incorporating the concept of utility into episode mining and based on the EWU model (*Episode-Weighted Utilization* model), an algorithm named UP-Span (*Utility ePisodes mining by Spanning prefixes*) has been designed for efficiently mining all HUEs.

**Hiding High Utility Sequential Patterns**. Although several algorithms have been proposed for mining high utility sequential patterns, an issue is that personal or sensitive information may be revealed by these algorithms. *Privacy Preserving* Data Mining (PPDM) has emerged as an interesting research topic in recent years. Hiding HUS is useful for applications such as those related to business, healthcare and security. Privacy preserving data mining aims at hiding private information so that it cannot be found by data mining algorithms. In HUSM, all high utility sequential patterns can be hidden so that adversaries cannot mine them from a sanitized database for a given threshold value. Based on USpan, the authors in [17] have designed two algorithms, HHUSP (*Hiding High Utility Sequential Patterns*) and MSPCF (*Maximum Sequential Patterns Conflict First*), for hiding HUSs.

**High Utility Sequential Pattern Mining from Incremental QSDB and Evolving Data Streams**. Most HUSM algorithms are designed for static QSDB. In dynamic QSDB, when a new $q$-sequence is added to a QSDB, discovering patterns from scratch to update results is very time-consuming. Hence, a projection-based incremental algorithm has been designed for HUSM from incremental database in [27] based on an index strategy and by extending the PHUS algorithm in [8]. Similarly, the HUSP-Stream algorithm [28] is used to discover all HUSs from a data streams based on a Sliding Window model.

**Distributed and Parallel High Utility Sequential Pattern Mining**. Most HUSM algorithms are based on the assumption that data can fit into the main memory of a computer. However, this assumption does not hold for large sequence datasets. An effective solution for mining big data is using *parallel* algorithms in a *distributed environment.* Thus, a new framework for mining HUSPs in big data has been proposed in [29]. The authors have designed a distributed and parallel algorithm called BigHUSP to mine HU sequences efficiently using multiple MapReduce-like steps to process data in parallel in a distributed environment, while applying some pruning strategies to reduce the search space. The proposed algorithm decreases computational and communication costs drastically, especially on large QSDBs.

**Mining High Utility Sequences with Negative Item Values**. Algorithms presented above are designed to mine HU sequences in QSDBs, where all $q$-items in database are only associated with positive values. However, in some applications, $q$-sequences in QSDBs may consists of items having *negative* unit profit values. For example, for cross-selling, a product such as a cartridge may be sold at a negative

profit when it is packed with another one such as a printer that provides a high positive return. Authors in [15] have proposed the HUSP-NIV algorithm for mining HU sequences with negative item values, based on two width, depth pruning strategies, by extending the USpan algorithm [9], and using a negative sequence pruning strategy. Note that, the utility of input $q$-sequences in QSDB used for computing the upper bound $SWU$ is calculated as the sum of utilities of only items having positive external utility values.

As other algorithms based on USpan, HUP-NIV extends USpan for solving an extended problem related to HUSM. It uses the $SPU$ as an upper bound on $u_{max}$ to deeply prune branches of the prefix tree early. However, as discussed in Remark c., the SPU is not an upper bound on $u_{max}$. Thus, using the $SPU$ to prune branches of the search tree early can result in missing some HUSs in the final set as shown in Remark c.($i$). In other words, HUP-NIV maybe also an incomplete algorithm.

**High Utility Sequential Rule Mining**. Another limitation of HUSM is that it does not provide a measure of the confidence or probability that a pattern will be followed. Having such measure is useful in several applications such as product recommendation. So far, few algorithms have been designed for mining high utility sequential rules (HUSR) [7, 30]. In [30], the problem of high-utility sequential rule mining in QSDBs is proposed and formalized with the assumption that all input $q$-sequences in QSDBs cannot contain the same item more than once. Based on a compact UtilityTable structure and several optimizations, a one-phase algorithm HUSRM has been designed for mining all high-utility sequential rules. HUSRs have also been applied for activity-cost event log analysis in the healthcare domain [7].

## 4 Research Opportunities

Because HUSM is more general than high utility itemset mining (HUIS), HUSM also has many research opportunities. Some of those are related to improving the performance of algorithms, for example, in terms of designing better upper-bounds, data structures and algorithms, and also to design parallel or distributed implementations. There are also several opportunities and challenges for applying HUSM to the real-word. For instance, one could propose novel interestingness/utility measures which are more suitable or useful in some real-life applications, and integrate various types of constraints. Developing techniques for visualizing results and interactively exploring patterns is also important.

## 5 Conclusion

This chapter has introduced the problem of high-utility sequential pattern mining. It is an interesting and important research topic that has many real-word applications. Some main techniques for pruning the search space based on upper bounds on the

actual utility as well as reducing projected databases and these upper bounds and the HUSPM algorithm for high utility sequence (HUS) mining have been presented. Then, some extensions of HUSM have been introduced to overcome some of the limitations of HUSM, for example, to discover top-*k* HUSs, periodic HUSs, high utility-probability sequences in uncertain quantitative sequence databases, mining HUSs with constraints or with multiple minimum utility thresholds. Finally, some related problems have been discussed such as discovering HUSs with negative utility values, HUSs from incremental datasets or streams, high-utility sequential rules. Lastly, research opportunities have been briefly discussed.

## References

1. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S.: Mining high utility web access sequences in dynamic web log data. In: 2010 11th ACIS International Conference Software Engineering AI Networking and Parallel/Distributed Computing (SNPD) (2010a)
2. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S.: A novel approach for mining high-utility sequential patterns in sequence databases. ETRI J. **32**, 676–686 (2010b)
3. Shie, B.E., Hsiao, H., Tseng, V.S., Yu, P.S.: Mining high utility mobile sequential patterns in mobile commerce environments. In: DASFAA (2011)
4. Shie, B.E., Cheng, J.H., Chuang, K.T., Tseng, V.S.: A one-phase method for mining high utility mobile sequential patterns in mobile commerce environments. In: Advanced Research in Applied Artificial Intelligence, pp. 616–626 (2012)
5. Shie, B.E., Yu, P.S., Tseng, V.S.: Mining interesting user behavior patterns in mobile commerce environments. Appl. Intell. **38**, 418–435 (2013)
6. Zihayat, M., Davoudi, H., An, A.: Top-k utility-based gene regulation sequential pattern discovery. In: Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference (2016a)
7. Dalmas, B., Fournier-Viger, P., Norre, S.: TWINCLE: a constrained sequential rule mining algorithm for event logs. In: Proceedings 9th International KES Conference (IDT-KES 2017). Springer (2017)
8. Lan, G.C., Hong, T.P., Tseng, V.S., Wang, S.L.: Applying the maximum utility measure in high utility sequential pattern mining. Expert Syst. Appl. **41**(11), 5071–5081 (2014)
9. Yin, J., Zheng, Z., Cao, L.: USpan: an efficient algorithm for mining high utility sequential patterns. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2012)
10. Alkan, O.K., Karagoz, P.: CRoM and HuspExt: improving efficiency of high utility sequential pattern extraction. IEEE Trans. Knowl. Data Eng. **27**(10), 2645–2657 (2015)
11. Wang, J.Z., Huang, J.L., Chen, Y.C.: On efficiently mining high utility sequential patterns. Knowl. Inf. Syst. **49**(2), 597–627 (2016)
12. Lin, J.C.W., Zhang, J., Fournier-Viger, P.: High-utility sequential pattern mining with multiple minimum utility thresholds. In: Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data (2017)
13. Liu, Y., Liao, W.K., Choudhary, A.N.: A two-phase algorithm for fast discovery of high utility itemsets. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hanoi, Vietnam (2005)
14. Yin, J., Zheng, Z., Cao, L., Song, Y., Wei, W.: Efficiently mining top-k high utility sequential patterns. In: 2013 IEEE 13th International Conference on Data Mining (ICDM) (2013)
15. Xu, T., Dong, X., Xu, J., Dong, X.: Mining high utility sequential patterns with negative item values. Int. J. Pattern Recogn. Artif. Intell. **31**(10), 1–17 (2017) (1750035)

16. Dinh, T., Huynh, V.N., Le, B.: Mining periodic high utility sequential patterns. In: In Asian Conference on Intelligent Information and Database Systems (2017)
17. Dinh, T., Quang, M.N., Le, B.: A Novel approach for hiding high utility sequential patterns. In: Proceedings International Symposium Information and Communication Technology (2015)
18. Zaki, M.J.: SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1), 31–60 (2001)
19. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, New York, NY (2002)
20. Gomariz, A., Campos, M., Marin, R., Goethals, B.: ClaSP: an efficient algorithm for mining frequent closed sequences. In: Proceedings of 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia (2013)
21. Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast vertical mining of sequential patterns using co-occurrence information. In: Proceedings of 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2014 (2014)
22. Bac, L., Hai, D., Tin, T., Fournier-Viger, P.: FCloSM, FGenSM: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. In: Knowledge and Information Systems (2017)
23. Hai, D., Tin, T., Bac, L.: Efficient algorithms for simultaneously mining concise representations of sequential patterns based on extended pruning conditions. Eng. Appl. Artif. Intell. **67**, 197–210 (2018)
24. Yin, J. Z. Z. C. L. S. Y. a. W. W.: Efficiently mining top-k high utility sequential patterns. In: 2013 IEEE 13th International Conference on Data Mining (ICDM) (2013)
25. Zhang, B., Lin, J.C.W., Fournier-Viger, P., Li, T.: Mining of high utility-probability sequential patterns from uncertain databases. PLoS One **12**(7), 1–21 (2017)
26. Wu, C.W., Lin, Y.F., Yu, P.S., Tseng, V.S.: Mining high utility episodes in complex event sequences. In: KDD 2013 Conference (2013)
27. Dave, U., Shah, J.: Efficient mining of high utility sequential pattern from incremental sequential dataset. Int. J. Comput. Appl. **122**(12), 22–28 (2015)
28. Zihayat, M., Wu, C.W., An, A., Tseng, V.S.: Mining high utility sequential patterns from evolving data streams. In: Proceedings of the ASE Big Data and Social Informatics 2015 (2015)
29. Zihayat, M., Hut, Z.Z., An, A., Hut, Y.: Distributed and parallel high utility sequential pattern mining. In: Big Data (Big Data), 2016 IEEE International Conference (2016b)
30. Zida, S., Fournier-Viger, P., Wu, C.W., Lin, J.C.W., Tseng, V.S.: Efficient mining of high utility sequential rules. In: Proceedings 11th International on Conference on Machine Learning and Data Mining (MLDM 2015). Springer, LNAI 9166 (2015)