EHUSM: Mining High Utility Sequences with a Pessimistic Utility Model

Tin Truong
University of Dalat,
Dalat, Vietnam
tintc@dlu.edu.vn

Anh Tran
University of Dalat,
Dalat, Vietnam
anhtn@dlu.edu.vn

Hai Duong
University of Dalat,
Dalat, Vietnam
haidv@dlu.edu.vn

Bac Le
VNU - HCMC, University of Science,
Ho Chi Minh City, Vietnam
lhbac@fit.hcmus.edu.vn

Philippe Fournier-Viger
Harbin Institute of Technology (Shenzhen),
Shenzhen, China
philfv@hit.edu.cn

ABSTRACT

The problem of mining all high utility sequences ($\mathcal{HUS}$) in a quantitative sequence database (QSDB) has attracted the attention of many researchers. It has many applications such as identifying highly profitable purchase patterns of customers in retail stores. However, a sequence (pattern) may appear multiple times in the same input sequence. To handle this situation, most studies adopt an optimistic perspective as they measure the utility of a sequence as the maximum utility among its occurrences. As a result, the utility of a pattern may seem higher than it actually is. Hence, taking business decisions on the basis of a pattern found by traditional high utility sequence mining algorithms may be risky. To address this issue, this paper proposes to adopt a pessimistic model for measuring the utility of patterns, using a novel measure called the minimum utility measure (u_{min}). Although discovering patterns using u_{min} is desirable to reduce risks for decision-making, designing an efficient algorithm to find patterns using this measure is challenging. The reason is that u_{min} does not satisfy the downward closure property (DCP) generally used to reduce the search space in sequence mining. To overcome this challenge, two novel upper bounds on u_{min} are proposed, which are tighter than the well-known sequence-weighted utility upper-bound, and satisfy anti-monotone-like properties that can be weaker than the DCP. Based on these properties, two novel depth and width pruning strategies are designed to eliminate low utility candidate sequences from the search space early. These theoretical results are the basis of a novel algorithm named EHUSM (Efficient High Utility Sequence Mining) for efficiently mining all high utility sequences using u_{min}. Experimental studies on both real-life and synthetic QSDBs show that the proposed algorithm is efficient in terms of runtime and memory.

KEYWORDS

Quantitative sequence database, minimum utility measure, high utility sequence, upper bound, pruning strategy, pessimistic model.

1 INTRODUCTION

The problem of frequent sequence mining (FSM) in sequential databases consists of discovering sequences that appear frequently in a set of sequences. It extends the problem of frequent itemset mining in traditional transaction databases by considering the sequential ordering of itemsets. Although FSM is popular, it is unsuitable for many real-life applications, since patterns (sequences) are discovered on the basis of their occurrence frequency rather than their utility (importance). Thus, FSM can discover numerous frequent sequences that are unimportant to the user and miss many important but infrequent sequences. To overcome this limitation of FSM, the problem of high utility sequence mining ($\mathcal{HUSM}$) in quantitative sequence databases was introduced, where each item is associated with a *quantity* (e.g. a purchase quantity) and a *quality* that indicates its relative importance (e.g. its unit profit). For example, in market basket analysis, $\mathcal{HUSM}$ consists of discovering sequences representing highly profitable sequences of purchases (called high utility sequences), which can be used for marketing purposes. $\mathcal{HUSM}$ also has many other applications such as analyzing web logs [1], data from mobile commerce environments [2], gene regulation [3], and healthcare activity-cost event logs [4]. $\mathcal{HUSM}$ is a generalization of FSM, and it is more challenging than FSM due to the additional consideration of utility information in sequences and the fact that the utility measure is not monotonic nor anti-monotonic, which makes it unsuitable to reduce the search space [5]. For these reasons, in the last decade, many researchers have studied this problem [6], [7], [8], [9], [10], [11] as well as its extensions [12], [13], [14], [15], [16], [17], [18], [19], [20], [21].

Related work. In previous work related to $\mathcal{HUSM}$, authors have mostly calculated the utility of a sequence using the maximum utility measure u_{max}. The u_{max} value of a sequence α in each input quantitative sequence Ψ' is calculated as the maximum utility among those of all occurrences of α in Ψ'. Thus, previous studies adopt an optimistic perspective. A key challenge of $\mathcal{HUSM}$ is that the u_{max} measure does not satisfy the downward closure property (DCP), a powerful property for efficiently reducing the search space in FSM. To address this problem, many researchers proposed upper bounds (UB) on u_{max} that restore the DCP. The first such UB is the SWU (Sequence-Weighted Utility), which has been used in several algorithms [5], [7], [8]. It was

demonstrated that all super sequences of a sequence having a low utility w.r.t. the SWU also have a low utility w.r.t. SWU and u_{max}. This property is used to reduce the search space. However, since the SWU measure is not a tight UB on u_{max}, the number of candidate sequences considered by *SWU* based algorithms can still be very large. To more efficiently reduce the search space, different UBs that are tighter than the SWU have been proposed, and were shown to improve the performance of $\mathcal{HUSM}$ [5], [9], [13], [11]. More information about these UBs is provided in Section 3.

However, in real-world applications, using an optimistic perspective for measuring the utility can be risky. For example, consider that investors want to expand a business by opening new branches, and that the historical business information about the corporation is stored in a QSDB. Using the optimistic approach of previous work on $\mathcal{HUSM}$ to analyze this database may result in taking unnecessary risks. The reason is that the utility of a sequence is calculated as the maximum utility among its occurrences. Thus, the utility of a pattern (sequence) may seem higher than it actually is in most input sequences. In fact, a sequential pattern may appear to have a high utility in some input quantitative sequences due to occurrences with large purchase quantities, while other occurrences of the pattern may have much smaller purchase quantities. Thus, measuring the utility of a pattern using the maximum utility of its occurrences is misleading, as it only provides information based on the largest purchase quantities in each input sequence. Hence, using such high utility patterns in investment can lead to taking important risks as it overestimates the quantities typically purchased by customers. To reduce risk, it is often desirable to use a more conservative approach by expecting that the minimum amount of utility is obtained from each input sequence. This paper thus proposes to calculate the utility of a pattern (sequence) using its minimum utility in each quantitative sequence (using a novel u_{min} measure) instead of its maximum utility (using u_{max}).

Contributions. The main contributions of this paper are the following. The problem of $\mathcal{HUSM}$ with the novel u_{min} measure is formalized. Two new UBs named RBU and LRU on u_{min} are proposed, which are tighter than the traditional SWU UB. Moreover, two efficient strategies are designed for pruning the search space. Based on these strategies, a novel efficient algorithm named EHUSM is introduced for mining high utility sequences. Finally, an experimental evaluation is conducted on both real-life and synthetic QSDBs to assess the runtime and memory consumption of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 introduces preliminary concepts. Section 3 presents the two novel UBs, pruning strategies and algorithm. Section 4 describes the experimental evaluation. Finally, Section 6 draws the conclusion.

2 PRELIMINARY CONCEPTS

This section introduces fundamental concepts and notations related to high utility sequence mining in quantitative sequence databases using the novel *minimum* utility measure u_{min}, and the order relation on quantitative sequences and extensions of sequences.

Definition 1 (*Quantitative sequence database*). Let $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$ be a set of distinct items. Each item $a_k \in \mathcal{A}$, $1 \leqslant k \leqslant M$, is associated with a positive number called *quality* $p(a_k)$ which represents its relative importance (e.g. unit profit). Item quality values are stored in a quality vector $P(\mathcal{A}) = (p(a_1), p(a_2), \ldots, p(a_M))$. A quantitative item (or briefly q-item) is a pair (a, q), where $a \in \mathcal{A}$ and $q \in R_+$ is a positive number representing the purchase quantity of item a. A subset E of $\mathcal{A}$, $E \subseteq \mathcal{A}$, is called an itemset. Without loss of generality, we assume that items in itemsets are sorted according to a total order such as the lexicographical order $\prec$. A quantitative itemset (or q-itemset) according to E is denoted and defined as $E' = \{(a_i, q_i) \mid a_i \in E, q_i \in R_+\}$ and E is called the projected itemset of E', which is denoted as $E = proj(E')$. A list of q-itemsets, E'_k, $k = 1, \ldots, p$, where p is a positive integer, denoted as $\alpha' = E'_1 \rightarrow E'_2 \rightarrow \ldots \rightarrow E'_p$, is called a q-sequence. Furthermore, we define $\alpha'[k] \stackrel{\text{def}}{=} E'_k$. The projected sequence α of a q-sequence α' is defined as $\alpha = proj(\alpha') \stackrel{\text{def}}{=} proj(E'_1) \rightarrow proj(E'_2) \rightarrow \cdots \rightarrow proj(E'_p)$. Moreover, we define $\alpha[k] \stackrel{\text{def}}{=} proj(E'_k)$. A q-sequence is a null q-sequence, denoted as $<>$, if all its itemsets are empty. A quantitative sequence database (QSDB) $\mathcal{D}'$ consists of a finite number of input q-sequences, $\mathcal{D}' = \{\Psi'_i,\ i = 1, \ldots, N\}$, and a quality vector $P(\mathcal{A})$. The size of the q-sequence α' is $size(\alpha') \stackrel{\text{def}}{=} p$, and its length is $length(\alpha') \stackrel{\text{def}}{=} \sum_{k=1}^{p} |proj(E'_k)|$, where $|E|$ is the number of items in itemset E.

Table 1: Notation

Type	Representation	Example
Item	Roman letter	*a*, *b*, *c*
q-item	(Roman letter, number)	(*a*, 2), (*b*, 5)
Event	Capitalized roman letter	*A*, *B*, *C*
q-Event	Capitalized roman letter followed by '	*A'*, *B'*, *C'*
Sequence	Greek letter	α, β, γ
q-sequence	Greek letter followed by prime '	α', β', γ'
Input q-sequence	Capitalized Greek letter followed by '	Ψ', Ψ'_{index}

For the convenience of the reader, Table 1 summarizes the notation used in the rest of this paper to denote (q-) items, (q-) events, (q-) sequences and input q-sequences.

Let $\alpha' = E'_1 \rightarrow E'_2 \rightarrow \cdots \rightarrow E'_p$, $\beta' = F'_1 \rightarrow F'_2 \rightarrow \cdots \rightarrow F'_q$ be two q-sequences, and $\alpha = E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow E_p$, $\beta = F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$ be two arbitrary sequences.

Definition 2 (*Partial order relation* $\sqsubseteq$). Consider any two q-events $E' = \{(a_{i_1}, q_{i_1}), \ldots, (a_{i_m}, q_{i_m})\}$, $F' = \{(a_{j_1}, q_{j_1}), \ldots, (a_{j_n}, q_{j_n})\}$ and $m \leqslant n$. The q-event E' is said to be contained in F' and denoted as $E' \sqsubseteq F'$, iff [1] there exist natural numbers $1 \leqslant k_1 <$

[1] "iff" means "if and only if".

$k_2 < \cdots < k_m \leqslant n$ such that $a_{i_l} = a_{j_{k_l}}$ and $q_{i_l} = q_{j_{k_l}}, \forall l = 1, \ldots, m$. Then, α' is contained in β' and denoted as $\alpha' \sqsubseteq \beta'$ (or $\beta' \sqsupseteq \alpha'$) iff $p \leqslant q$ and there exist p positive integers, $1 \leqslant j_1 < j_2 < \cdots < j_p \leqslant q$: $E'_k \sqsubseteq F'_{j_k}$, $\forall k = 1, \ldots, p$; and $\alpha' \sqsubset \beta' \Leftrightarrow (\alpha' \sqsubseteq \beta' \wedge \alpha' \neq \beta')$. Similarly, for simplicity, we also use $\sqsubseteq$ to define the containment relation over all sequences as follows: $\alpha \sqsubseteq \beta$ (or β is a super-sequence of α) $\Leftrightarrow \exists p$ 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 $\alpha \sqsubset \beta \Leftrightarrow (\alpha \sqsubseteq \beta \wedge \alpha \neq \beta)$. The sequence α is contained in the q-sequence β', which is denoted as $\alpha \sqsubseteq \beta'$ (or $\beta' \sqsupseteq \alpha$), iff $proj(\beta') \sqsupseteq \alpha$. Let $\rho(\alpha) \stackrel{\text{def}}{=} \{\Psi' \in \mathcal{D}' \mid \Psi' \sqsupseteq \alpha\}$ denote the set of all input q-sequences containing α.

Definition 3 (*Utilities of q-elements*). The utilities of q-item (a, q), q-event $E' = \{(a_{i_1}, q_{i_1}), \ldots, (a_{i_m}, q_{i_m})\}$, q-sequence α' and QSDB $\mathcal{D}'$ are respectively defined and denoted as $u((a, q)) \stackrel{\text{def}}{=} p(a) * q$, $u(E') \stackrel{\text{def}}{=} \sum_{j=1..m} u((a_{i_j}, q_{i_j}))$, $u(\alpha') \stackrel{\text{def}}{=} \sum_{i=1..p} u(E'_i)$ and $u(\mathcal{D}') \stackrel{\text{def}}{=} \sum_{\Psi' \in \mathcal{D}'} u(\Psi')$.

To avoid repeatedly calculating the utility u of each q-item (a, q) in itemsets of all input q-sequences $\Psi' \in \mathcal{D}'$, we calculate all utility values, and replace each q-item q by $u\big((a, q)\big) = p(a) * q$ in input q-sequences. This results in an equivalent representation of the QSDB $\mathcal{D}'$ that is called the integrated QSDB of $\mathcal{D}'$. For the sake of brevity, it is denoted as $\mathcal{D}'$. Hereafter, we only consider the integrated QSDB $\mathcal{D}'$. For example, Table 2 shows an integrated QSDB $\mathcal{D}'$, which will be used as running example in this paper.

Table 2: Integrated QSDB $\mathcal{D}'$

ID	Input q-sequence
Ψ'_1	(*a*, 2)(*c*, 5)(*e*, 6)→(*a*, 3)(*b*, 6)→(*a*, 5)(*d*, 50)→ (*a*, 5)(*b*, 9)(*c*,40)→ (*a*, 4)(*c*, 10)(*d*, 10)(*f*, 36)
Ψ'_2	(*b*, 12)→(*a*, 2)(*c*, 20)(*e*, 6)→(*a*, 3)(*d*, 20) →(*a*, 1)(*c*,20)(*d*, 10)(*f*, 9) →(*a*, 4)(*b*, 9)(*c*, 15)
Ψ'_3	(*c*, 20)→(*a*, 4)(*c*, 10)(*e*, 4)→(*a*, 1)(*f*, 18)
Ψ'_4	(*d*, 80)→(*a*, 7)(*c*, 50)(*e*, 6)→(*a*, 2)(*g*, 2)→(*a*, 9)(*f*, 72)

Definition 4 (*Novel minimum utility measure of a sequence*). Assume that $\alpha \sqsubseteq \beta'$. Let $U(\alpha, \beta') \stackrel{\text{def}}{=} \{\alpha' | \alpha' \sqsubseteq \beta' \wedge proj(\alpha') = \alpha\}$ be the set of all occurrences α' of α in β'. The *minimum* utility (or briefly utility) of α in β' and in $\mathcal{D}'$ are denoted and defined as $u_{min}(\alpha, \beta') \stackrel{\text{def}}{=} min\{u(\alpha') | \alpha' \in U(\alpha, \beta')\}$, and $u_{min}(\alpha, \mathcal{D}')$ or briefly $u_{min}(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} u_{min}(\alpha, \Psi')$, respectively.

The sequence α is said to be a high utility (HU) sequence if its utility in $\mathcal{D}'$ is not less than a user-defined mu threshold, that is $u_{min}(\alpha) \geqslant mu$. Otherwise, α is said to be a low utility (LU) sequence. The ***problem of high utility sequence mining*** ($\mathcal{HUSM}$) with u_{min} considered in this paper is to discover the set $\mathcal{HUS} \stackrel{\text{def}}{=} \{\alpha \mid u_{min}(\alpha) \geqslant mu\}$.

For example, consider the sequence $\alpha = d \rightarrow ac \rightarrow a$, $(\alpha) = \{\Psi'_1, \Psi'_2, \Psi'_4\}$. The sequence α first appears in Ψ'_4 as the sub-q-sequence $\alpha' = (d, 80) \rightarrow (a, 7)(c, 50) \rightarrow (a, 2)$ of Ψ'_4, $\alpha = proj(\alpha')$ and $u(\alpha') = 80 + 7 + 50 + 2 = 139$. Hence, $\alpha \sqsubseteq \Psi'_4$, $\alpha' \in U(\alpha, \Psi'_4)$. Similarly, α also appears in Ψ'_4 as the sub-q-sequence $\alpha'' = (d, 80) \rightarrow (a, 7)(c, 50) \rightarrow (a, 9)$ of Ψ'_4 with the utility $u(\alpha'') = 146$. Hence, $U(\alpha, \Psi'_4) = \{\alpha', \alpha''\}$ and $u_{min}(\alpha, \Psi'_4) = min\{u(\alpha'), u(\alpha'')\} = 139$. In the same way, we also have $u_{min}(\alpha, \Psi'_1) = 99$, $u_{min}(\alpha, \Psi'_2) = 45$. Thus, $u_{min}(\alpha) = u_{min}(\alpha, \Psi'_1) + u_{min}(\alpha, \Psi'_2) + u_{min}(\alpha, \Psi'_4) = 99 + 45 + 139 = 283$. For the running example and $mu = 350$, since $\beta = d \rightarrow ac \rightarrow af$ with $u_{min}(\beta) = 353 > mu$ is the unique HU sequence, thus $\mathcal{HUS} = \{\beta\}$.

Definition 5 (*i-extension and s-extension of a sequence*). The i-extension (or s-extension) of α and β is defined and denoted as $\alpha \diamond_i \beta \stackrel{\text{def}}{=} E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow (E_p \cup F_1) \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$, where $a \prec b$, $\forall a \in E_p$, $\forall b \in F_1$ (or $\alpha \diamond_s \beta \stackrel{\text{def}}{=} E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow E_p \rightarrow F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$, respectively). The i-extension (or s-extension) of α and β is defined and denoted as $\alpha \diamond_i \beta \stackrel{\text{def}}{=} E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow (E_p \cup F_1) \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$, where $a \prec b, \forall a \in E_p, \forall b \in F_1$ (or $\alpha \diamond_s \beta \stackrel{\text{def}}{=} E_1 \rightarrow E_2 \rightarrow \cdots \rightarrow E_p \rightarrow F_1 \rightarrow F_2 \rightarrow \cdots \rightarrow F_q$, respectively). A *forward* extension (or briefly extension) of α with β, denoted as $\gamma = \alpha \diamond \beta$, can be either $\alpha \diamond_i \beta$ or $\alpha \diamond_s \beta$. In that case, α is said to be a prefix of γ. To mine sequences, the search space of candidate sequences can be viewed as a prefix tree, which consists of the null sequence as its root, and where each node represents a candidate sequence, and each child node of a node is its extension. In the following, $branch(\alpha)$ denotes the set consisting of α and all its (forward) extensions.

Beside, any sequence $\beta = \alpha \diamond y$ where α is a non-null prefix, can be extended in a backward manner using a sequence ε. The sequence $\gamma = \alpha \diamond \varepsilon \diamond y$ such that $\gamma \sqsupseteq \beta$ is called a backward extension of β (by ε w.r.t. the last item $y = lastItem(\beta)$).

For instance, the sequence $\alpha = be$, $branch(\alpha)$ consists of sequence α and its forward extensions $be \rightarrow \varepsilon$, $bef \rightarrow \varepsilon$, $beg \rightarrow \varepsilon$, and $befg \rightarrow \varepsilon$, $\forall \varepsilon$ (including <>). For $\beta = be$, the backward extensions of β are bce, bde, $bcde$, and backward extensions of $b \rightarrow e$ are sequences $bC \rightarrow \varepsilon \rightarrow e$, $\forall C \subseteq cdefg$ (e.g. $bc \rightarrow \varepsilon \rightarrow e$, $bcdefg \rightarrow \varepsilon \rightarrow e$) and $b \rightarrow \varepsilon \rightarrow De$, $\forall D \subseteq abcd$ (e.g. $b \rightarrow \varepsilon \rightarrow ace$, $b \rightarrow \varepsilon \rightarrow abde$).

A main challenge of $\mathcal{HUSM}$ with $u_{min}$is that the u_{min} measure is neither monotonic nor anti-monotonic, i.e. $\exists \beta \sqsupset \alpha$ and $\gamma \sqsubset \delta$: $u_{min}(\beta) > u_{min}(\alpha)$ and $u_{min}(\gamma) > u_{min}(\delta)$. Thus, the DCP does not hold for u_{min}. The DCP is a strong property used in FSM for efficiently pruning unpromising branches of the prefix tree (the search space). Indeed, for $\alpha = ace \rightarrow a \rightarrow f$, $\beta = ace \rightarrow a \rightarrow af$ and $\delta = ace \rightarrow a \rightarrow a \rightarrow f$, then $\delta \sqsupset \alpha \sqsubset \beta$, $u_{min}(\beta) = 243 > u_{min}(\alpha) = 229 > u_{min}(\delta) = 57$. To address this problem, upper bounds on u_{min} satisfying anti-monotone-like properties that may be weaker than the DCP are proposed in the next section.

3 THE PROPOSED UPPER BOUNDS, PRUNING STRATEGIES AND ALGORITHM

3.1 Two Upper Bounds on u_{min}

Before presenting the two proposed upper bounds on u_{min}, we introduce the concept of ending and remaining q-sequence of a sequence in a q-sequence. Assume that $\alpha' \sqsubseteq \Psi'$ and $\alpha = proj(\alpha') \sqsubseteq \Psi'$, where $\Psi' = F'_1 \rightarrow F'_2 \rightarrow \cdots \rightarrow F'_q \in \mathcal{D}'$, i.e. there exist p integers, $1 \leqslant i_1 < i_2 < \cdots < i_p \leqslant q$: $E'_k \sqsubseteq F'_{i_k}$ and $E_k = proj(E'_k) \subseteq proj(F'_{i_k})$, $\forall k = 1, \ldots, p$. The index i_p is called an ending of α' (or α) in Ψ', denoted as $end(\alpha', \Psi')$ (or $end(\alpha, \Psi')$). The last item of α in F'_{i_p} is called an ending item according to i_p and is denoted as e_{i_p}. Then, the remaining q-sequence of α in Ψ' w.r.t. the ending index i_p is the rest of Ψ' after the ending item e_{i_p} and is denoted as $rem(\alpha, \Psi', i_p)$. Let $i_p^* \stackrel{\text{def}}{=} FEnd(\alpha, \Psi')$ be the first ending of α in Ψ'. If $\alpha = <>$, then as convention $i_p^* \stackrel{\text{def}}{=} FEnd(<>, \Psi') \stackrel{\text{def}}{=} 0$ and $rem(<>, \Psi', i_p^*) = \Psi'$. For each ending $i_p = end(\alpha, \Psi')$, we define $u(\alpha, \Psi', i_p) \stackrel{\text{def}}{=} min\{u(\alpha') \mid \alpha' \in U(\alpha, \Psi') \wedge end(\alpha', \Psi') = i_p\}$. Furthermore, we define $ub_{rem}(\alpha, \Psi') \stackrel{\text{def}}{=} u\big(\alpha, \Psi', i_p^*\big) + u(rem(\alpha, \Psi', i_p^*))$ the remaining utility-based upper bound on u_{min} of α in Ψ' for $\alpha \neq <>$, and $ub_{rem}(<>, \Psi') \stackrel{\text{def}}{=} u(\Psi')$ if $\alpha = <>$.

A measure ub is said to be an upper bound (UB) on u_{min} iff $u_{min}(\alpha) \leqslant ub(\alpha)$, $\forall \alpha$. For two UBs ub_1 and ub_2, ub_1 is said to be tighter than ub_2, denoted as $ub_1 \ll ub_2$, iff $ub_1(\alpha) \leqslant ub_2(\alpha)$, $\forall \alpha$, and ub_1 is said to be strictly tighter than ub_2, iff $ub_1 \ll ub_2$ and $\exists\, \beta$: $ub_1(\beta) < ub_2(\beta)$. For briefly, we set $ub_{max}(\alpha \diamond \varepsilon) \stackrel{\text{def}}{=} max\{ub(\alpha \diamond_i \varepsilon), ub(\alpha \diamond_s \varepsilon\}$, $ub_{max}(\alpha \diamond \varepsilon \diamond \delta) \stackrel{\text{def}}{=} max\{ub(\alpha \diamond \varepsilon \diamond \delta)$, $\forall \diamond \in \{\diamond_i, \diamond_s\}\}$ and define $IS_{ub}(\alpha) \stackrel{\text{def}}{=} \{y \in \mathcal{A} \mid ub_{max}(\alpha \diamond y) \geqslant mu\}$ as the set of all candidate items for extensions of α w.r.t. the ub UB on u_{min}, because if $y \notin IS_{ub}(\alpha)$, then for $\diamond \in \{\diamond_i, \diamond_s\}$, $u_{min}(\alpha \diamond y) \leqslant ub_{max}(\alpha \diamond y) < mu$.

Based on these concepts, we propose two novel UBs, named RBU and LBU, which are tighter than the well-known SWU UB, i.e. $RBU \ll LRU \ll SWU$. Although RBU is better than LRU in terms of values (i.e. providing a tighter UB), the width pruning effect of LRU is stronger than the depth pruning effect of RBU.

Definition 6 (*Upper bounds on* u_{min}). Consider any item $y \in \mathcal{A}$ and non-null sequence α.

a. The traditional SWU UB (*Sequence Weighted Utility*) is defined as $SWU(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} u(\Psi')$ [5].

b. Two novel UBs named RBU (*Remaining-Based Utility*) and LRU (*Looser Remaining Utility*) are defined as follows: $RBU(\alpha) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha)} ub_{rem}(\alpha, \Psi')$, $LRU(y) \stackrel{\text{def}}{=} SWU(y)$ and $LRU(\alpha \diamond y) \stackrel{\text{def}}{=} \sum_{\Psi' \in \rho(\alpha \diamond y)} ub_{rem}(\alpha, \Psi')$.

For example, consider the sequence $\beta = ae \rightarrow a \rightarrow c$, $\rho(\beta) = \{\Psi'_1, \Psi'_2\}$. For Ψ'_1, $u_{min}(\beta, \Psi'_1) = 21$, since β first appears in Ψ'_1 as the sub-q-sequence $\beta' = (a,2)(e,6) \rightarrow (a,3) \rightarrow (c,40)$, the first ending of β in Ψ'_1 is $i_p^* \stackrel{\text{def}}{=} FEnd\ (\beta, \Psi'_1) = 4$, and the corresponding remaining q-sequence is $rem(\beta, \Psi'_1, i_p^*) = (a,4)(c,10)(d,10)(f,36)$, so $u(rem(\beta, \Psi'_1, i_p^*)) = 60$. Similarly, β also appears in Ψ'_1 as the sub-q-sequence $\beta'' = (a,2)(e,6) \rightarrow (a,5) \rightarrow (c,40)$, then $u\big(\beta, \Psi'_1, i_p^*\big) = min\{u(\beta'), u(\beta'')\} = min\{51,53\} = 51$ and $ub_{rem}(\beta, \Psi'_1) = u(\beta, \Psi'_1, i_p^*) + u(rem(\beta, \Psi'_1, i_p^*)) = 111$. In the same way, $u_{min}(\beta, \Psi'_2) = 24$, $ub_{rem}(\beta, \Psi'_2) = 31 + 47 = 78$, so $u_{min}(\beta) = u_{min}(\beta, \Psi'_1) + u_{min}(\beta, \Psi'_2) = 45$, $RBU(\beta) = ub_{rem}(\beta, \Psi'_1) + ub_{rem}(\beta, \Psi'_2) = 189$. Similarly, for $\alpha = ae \rightarrow a$, we also have $ub_{rem}(\alpha, \Psi'_1) = 186$, $ub_{rem}(\alpha, \Psi'_2) = 99$. Hence, $\beta = \alpha \rightarrow c$, $LRU(\beta) = ub_{rem}(\alpha, \Psi'_1) + ub_{rem}(\alpha, \Psi'_2) = 285$, $SWU(\beta) = u(\Psi'_1) + u(\Psi'_2) = 191 + 131 = 322$. Hence, $u_{min}(\beta) \leqslant RBU(\beta) \leqslant LRU(\beta) \leqslant SWU(\beta)$.

Before further discussing the properties of the UBs, we show that the ρ operator and the ub_{rem} UB on u_{min} are anti-monotonic in each input q-sequence.

Lemma 1 (*Properties of* ρ *and* ub_{rem}).

a. (*Anti-monotonicity of* ρ) $\rho(\alpha) \supseteq \rho(\beta)$, $\forall \beta \sqsupseteq \alpha$.

b. (*Anti-monotonicity w.r.t. forward extension of* ub_{rem} *in each input* q*-sequence*) $u_{min}(\alpha, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$, $\forall \Psi' \in \rho(\alpha)$, and $ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$, $\forall \beta = \alpha \diamond \varepsilon \sqsupseteq \alpha$ and $\forall \Psi' \in \rho(\beta)$.

Proof.

a. For any $\Psi' \in \rho(\beta)$, then $\Psi' \sqsupseteq \beta \sqsupseteq \alpha$, and hence $\Psi' \in \rho(\alpha)$.

b. For any $\Psi' \in \rho(\alpha)$, since $\{\alpha' \in U(\alpha, \Psi') \wedge end(\alpha', \Psi') = i_p^*\} \subseteq U(\alpha, \Psi')$, $u_{min}(\alpha, \Psi') \leqslant u\big(\alpha, \Psi', i_p^*\big) \leqslant u\big(\alpha, \Psi', i_p^*\big) + u(rem(\alpha, \Psi', i_p^*)) = ub_{rem}(\alpha, \Psi')$, where $i_p^* \stackrel{\text{def}}{=} FEnd(\alpha, \Psi')$. For any $\Psi' \in \rho(\alpha)$, define $\alpha'_{min}(\alpha, \Psi')$ (or briefly α'_{min}) as the q-sequence in $U(\alpha, \Psi')$ that yields the minimum value $u\big(\alpha, \Psi', i_p^*\big)$, that is $u(\alpha'_{min}) = u\big(\alpha, \Psi', i_p^*\big)$, and $rem(\alpha'_{min}, \Psi')$ the rest of Ψ' after α'_{min}. To prove that $ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$, for any forward extension of α, $\beta = \alpha \diamond \delta \sqsupset \alpha$, without loss the generality, we can assume that the sequence δ only consists of an item x, i.e. $\beta = \alpha \diamond x$ and define $p \stackrel{\text{def}}{=} size(\alpha)$. In the following, we denote the k^{th}itemset of β' as $\beta'[k]$ and the k^{th} item of $\beta'[i]$ as $\beta'[i][k]$. For any $\Psi' \in \rho(\beta)$, consider the two following cases.

(*i*) $\beta = \alpha \diamond_s x$: $\forall k: 1 \leqslant k \leqslant p$, $u(\beta'_{min}[k]) \leqslant u(\alpha'_{min}[k])$ and $u(\beta'_{min}[p+1]) + u(rem(\beta'_{min}, \Psi')) \leqslant u(rem(\alpha'_{min}, \Psi'))$. Thus, $ub_{rem}(\beta, \Psi') = \sum_{1 \leqslant k \leqslant p+1} u(\beta'_{min}[k]) + u(rem(\beta'_{min}, \Psi')) \leqslant \sum_{1 \leqslant k \leqslant p} u(\alpha'_{min}[k]) + u(rem(\alpha'_{min}, \Psi')) = ub_{rem}(\alpha, \Psi')$.

(*ii*) $\beta = \alpha \diamond_i x$: $\forall k: 1 \leqslant k \leqslant p-1$, $u(\beta'_{min}[k]) \leqslant u(\alpha'_{min}[k])$ and $u(\beta'_{min}[p]) + u(rem(\beta'_{min}, \Psi') \leqslant u(rem(\alpha'_{min}, \Psi')) \leqslant u(\alpha'_{min}[p]) + u(rem(\alpha'_{min}, \Psi'))$, so $ub_{rem}(\beta, \Psi') = \sum_{1 \leqslant k \leqslant p} u(\beta'_{min}[k]) + u(rem(\beta'_{min}, \Psi')) \leqslant \sum_{1 \leqslant k \leqslant p} u(\alpha'_{min}[k]) + u(rem(\alpha'_{min}, \Psi')) = ub_{rem}(\alpha, \Psi')$. Thus, $ub_{rem}(\alpha \diamond \delta, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$. □

The two following theorems show the relationships between the three aforementioned UBs in terms of their values and *anti-monotone-like* properties.

Theorem 1 (*Relationships between UBs on* u_{min})

$$u_{min} \ll RBU \ll LRU \ll SWU.$$

Thus, SWU, LRU and RBU are gradually tighter UBs on u_{min}.

Proof.

For any forward extension. $\beta = \alpha \diamond y \sqsupset \alpha$, $\alpha \neq <>$, and $\Psi' \in \rho(\beta)$, by Lemma 1, $ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi') \leqslant u(\Psi')$. Thus, $RBU(\beta) \leqslant \sum_{\Psi' \in \rho(\beta)} ub_{rem}(\alpha, \Psi') = LRU(\beta) \leqslant SWU(\beta)$

and $RBU(y) \leqslant SWU(y) = LRU(y)$. Hence, $RBU \ll LRU \ll SWU$. Beside, for $i_p^* \stackrel{\text{def}}{=} FEnd(\alpha, \Psi')$, we have $\{\alpha' \in U(\alpha, \Psi') \wedge end(\alpha', \Psi') = i_p^*\} \subseteq U(\alpha, \Psi')$, $u_{min}(\alpha, \Psi') \leqslant u(\alpha, \Psi', i_p^*) \leqslant u(\alpha, \Psi', i_p^*) + u(rem(\alpha, \Psi', i_p^*)) = ub_{rem}(\alpha, \Psi')$, Hence, $u_{min}(\alpha) \leqslant RBU(\alpha)$ and $u_{min} \ll RBU$. □

Theorem 2 (*Anti-monotone-like property of* RBU, LRU *and* SWU).

a. SWU is *anti-monotonic*, denoted as $\mathcal{AM}(SWU)$, *i.e.* $SWU(\alpha) \geqslant SWU(\beta)$, for any super-sequence β of α.

b. RBU is *anti-monotonic w.r.t forward extensions*, denoted as $\mathcal{AMF}(RBU)$, *i.e.* $RBU(\alpha) \geqslant RBU(\beta)$, $\forall$ *forward extension* $\beta = \alpha \diamond \varepsilon$ of α.

c. LRU is *anti-monotonic w.r.t bi-directional (forward and backward) extensions*, denoted as $\mathcal{AMBi}(LRU)$, i.e. $\mathcal{AMF}(LRU)$ and $LRU_{max}(\beta) \geqslant LRU_{max}(\gamma)$ $\forall$ *backward extension* $\gamma = \alpha \diamond \varepsilon \diamond y$ of $\beta = \alpha \diamond y$. Moreover, $IS_{LRU}(\alpha \diamond y) \subseteq IS_{LRU}(\alpha)$.

Proof.

a. $\forall \beta \sqsupseteq \alpha$, due to $\rho(\beta) \subseteq \rho(\alpha)$. Hence, $SWU(\beta) \leqslant SWU(\alpha)$, i.e. $\mathcal{AM}(SWU)$.

b. Moreover, for any $\beta = \alpha \diamond \varepsilon$ such that $\beta \sqsupseteq \alpha$ and $\Psi' \in \rho(\beta) \subseteq \rho(\alpha)$, by Lemma 1, then $ub_{rem}(\beta, \Psi') \leqslant ub_{rem}(\alpha, \Psi')$. Hence, $RBU(\beta) \leqslant RBU(\alpha)$, i.e. $\mathcal{AMF}(RBU)$.

c. To prove that $LRU(\beta) \leqslant LRU(\alpha), \forall \beta = \alpha \diamond \varepsilon \sqsupset \alpha$, without loss generality, we can assume that ε only consists of an item y. Assume that $\alpha = \delta \diamond x \sqsubset \beta = \delta \diamond x \diamond y$ with $x, y \in \mathcal{A}$, then $\delta \diamond x$ is a forward extension of δ. By Lemma 1, $LRU(\beta) = \sum_{\Psi \in \rho(\beta)} ub_{rem}(\delta \diamond x, \Psi') \leqslant \sum_{\Psi \in \rho(\beta)} ub_{rem}(\delta, \Psi') \leqslant \sum_{\Psi \in \rho(\alpha)} ub_{rem}(\delta, \Psi') = LRU(\alpha)$. Hence, $\mathcal{AMF}(LRU)$.

To prove that $LRU_{max}(\gamma) \leqslant LRU_{max}(\beta)$, $\forall \gamma = \alpha \diamond \varepsilon \diamond y$, $\beta = \alpha \diamond y$ such that $\gamma \sqsupset \beta$, first consider that $\gamma = \alpha \diamond_i \varepsilon \diamond_i y$ and $size(\varepsilon) = 1$. We have $\gamma \sqsupseteq \beta = \alpha \diamond_i y$ and by Lemma 1, $\rho(\gamma) \subseteq \rho(\alpha \diamond_i y)$. Hence, $LRU(\gamma) = \sum_{\Psi \in \rho(\gamma)} ub_{rem}(\alpha \diamond_i \varepsilon, \Psi') \leqslant \sum_{\Psi \in \rho(\gamma)} ub_{rem}(\alpha, \Psi') \leqslant \sum_{\Psi \in \rho(\alpha \diamond_i y)} ub_{rem}(\alpha, \Psi') = LRU(\alpha \diamond_i y)$. For all other cases, for each $\gamma \in \{\alpha \diamond_i \varepsilon \diamond_s y, \alpha \diamond_s \varepsilon \diamond_i y, \alpha \diamond_s \varepsilon \diamond_s y\}$, we have $\alpha \diamond_s y \sqsubseteq \gamma$. Thus, $\rho(\gamma) \subseteq \rho(\alpha \diamond_s y)$ and $LRU(\gamma) = \sum_{\Psi \in \rho(\gamma)} ub_{rem}(\alpha \diamond \varepsilon, \Psi') \leqslant \sum_{\Psi \in \rho(\gamma)} ub_{rem}(\alpha, \Psi') \leqslant \sum_{\Psi \in \rho(\alpha \diamond_s y)} ub_{rem}(\alpha, \Psi') = LRU(\alpha \diamond_s y)$. Hence, $\forall \diamond \in \{\diamond_i, \diamond_s\}$, $LRU(\gamma) \leqslant max\{LRU(\alpha \diamond_i y), LRU(\alpha \diamond_s y)\} = LRU_{max}(\beta)$ and $LRU_{max}(\gamma) = max\{LRU(\alpha \diamond \varepsilon \diamond y), \forall \diamond \in \{\diamond_i, \diamond_s\}\} \leqslant LRU_{max}(\beta)$.

For any $z \in IS_{LRU}(\alpha \diamond y)$, by the above proof, $mu \leqslant LRU_{max}(\alpha \diamond y \diamond z) \leqslant LRU_{max}(\alpha \diamond z)$, i.e. $z \in IS_{LRU}(\alpha)$. □

Obviously, the *anti-monotone-like* properties $\mathcal{AM}$, $\mathcal{AMBi}$ and $\mathcal{AMF}$ are *gradually weaker*, i.e. $\mathcal{AM}(ub) \Rightarrow \mathcal{AMBi}(ub) \Rightarrow \mathcal{AMF}(ub)$, for any UB ub on u_{min}. For example, $\mathcal{AMBi}(LRU) \Rightarrow \mathcal{AMF}(LRU)$, $\mathcal{AM}(SWU) \Rightarrow \mathcal{AMBi}(SWU) \Rightarrow \mathcal{AMF}(SWU)$.

3.2 Two Pruning Strategies

Based on Theorem 2, we design two strategies for pruning low utility candidate branches of the prefix search tree. These strategies only prune LU sequences, which ensures that all HU sequences can be found. Consider any item $y \in \mathcal{A}$ and non-null sequence α.

Depth pruning strategy based on RBU UB *(*$\mathcal{DPS}(RBU)$*).* If $RBU(\alpha) < mu$, then we can prune the $branch(\alpha)$, because $u_{min}(\beta) \leqslant RBU(\beta) \leqslant RBU(\alpha) < mu$, for any forward extensions β of α.

Width pruning strategy based on LRU or SWU UBs *(*$\mathcal{WPS}(LRU)$ or $\mathcal{WPS}(SWU)$*).* If $LRU(\alpha \diamond y) < mu$ or $SWU(\alpha \diamond y) < mu$, then we can prune not only all forward extensions of $\alpha \diamond y$ (or $branch(\alpha \diamond y)$) but also all its backward extension branches by applying $IS_{LRU}(\alpha \diamond y) \subseteq IS_{LRU}(\alpha)$.

Thus, if $LRU(y) < mu$ (or equivalently $SWU(y) < mu$), then we can remove such irrelevant item y from QSDB $\mathcal{D}'$, because $u_{min}(\gamma) \leqslant SWU(\gamma) \leqslant SWU(y) = LRU(y) < mu$, for any super-sequence γ of y. Then, all UBs values according to the updated $\mathcal{D}'$ should be also updated and may be decreased.

Similarly, if $LRU_{max}(\alpha \diamond y) < mu$, such irrelevant item y can be eliminated from the projected database $\mathcal{D}'_\alpha$ of α. The reason is that for any sequence $\gamma = \alpha \diamond \varepsilon \diamond y \diamond \delta$ having the same prefix α, which represents an arbitrary sequence in the PDB $\mathcal{D}'_\alpha$ containing y, we always have $u_{min}(\gamma) \leqslant LRU(\gamma) \leqslant LRU(\alpha \diamond \varepsilon \diamond y) \leqslant LRU_{max}(\alpha \diamond \varepsilon \diamond y) \leqslant LRU_{max}(\alpha \diamond y) < mu$. In other words, any sequence γ in $\mathcal{D}'_\alpha$ containing y is also LU. In this context, the projected database $\mathcal{D}'_\alpha$ of α consists of all remaining sequences $rem(\alpha, \Psi', i_p^*)$, where $\Psi' \in \rho(\alpha)$ and $i_p^* = FEnd(\alpha, \Psi')$ is the first ending of α in Ψ'. Then, all UBs values according to the reduced $\mathcal{D}'_\alpha$ should be calculated and they can be reduced.

When extending a sequence $\beta = \alpha \diamond y$ with an item z from $\mathcal{A}$, we may generate candidate sequences that do not appear in $\mathcal{D}'$. This results in unnecessary calculations. To address this problem, one can create a projected database (PDB) according to β. However, creating and scanning multiple PDBs is very costly. Based on the inclusion relationship $IS_{LRU}(\beta) \subseteq IS_{LRU}(\alpha)$ of Theorem 2.c, another way to overcome this problem is to only extend β with an item z (i.e. $z \in IS_{LRU}(\beta)$) if z is in the candidate item set $IS_{LRU}(\alpha)$ of the previous step. Note that the IS_{LRU} sets become gradually smaller during the mining process. Thus, this decreases the time for mining $\mathcal{HUS}$.

It is observed that $LRU(y) = SWU(y)$, $\forall y \in \mathcal{A}$. The pruning effect of the RBU UB - the tightest UB among the three discussed UBs - is weaker than that of the LRU and SWU UBs (the pruning effect of the two latter UBs is stronger). In general, an UB ub that is tighter than some others can be applied more often (when $ub(\alpha) < mu$) but its pruning effect is weaker. Beside, since both LRU and SWU UBs have the same width pruning effect, but LRU is tighter than SWU, LRU is always better than the SWU UB. Thus, to efficiently prune HU candidate sequences, it is sufficient to use the two proposed tighter LRU and RBU UBs.

For example, for $mu = 350$, since $SWU(b) = LRU(b) = 322$, $SWU(d) = LRU(d) = 228$ are less than mu, and the SWU and LRU values of remaining items in the candidate

item set $IS = \{a, c, d, e, f\}$ are greater than mu (e.g. $SWU(a) = LRU(a) = 609$). Then, by $\mathcal{WPS}(SWU)$ or $\mathcal{WPS}(LRU)$, not only the items b and g can be removed from $\mathcal{D}'$ but the upper bound values of remaining items can be decreased. For example, the new updated $LRU(a)$ is 569 (< 609). Similarly, because $LRU(ad) = LRU(a \rightarrow d) = 286 < mu$, $IS_{LRU}(a) = acef$ and by $\mathcal{WPS}(LRU)$, we can completely prune $branch(ad)$ and $branch(a \rightarrow d)$ by $\mathcal{DPS}(LRU)$, but also eliminate many of their backward extension branches such as $branch(acd)$, $branch(ac \rightarrow d)$, $branch(ae \rightarrow d)$, $branch(a \rightarrow c \rightarrow d)$, and $branch(ace \rightarrow d)$, because $IS_{LRU}(ac)$, $IS_{LRU}(ae)$, $IS_{LRU}(a \rightarrow c)$ are subsets of $IS_{LRU}(a)$ and $IS_{LRU}(ace) \subseteq IS_{LRU}(ac) \subseteq IS_{LRU}(a)$. Thus, $\mathcal{WPS}$ is strictly stronger than $\mathcal{DPS}$. To illustrate the depth pruning effect of RBU, consider values $RBU(a \rightarrow f) = 164$ and $RBU(f) = 154$ which are less than mu. Using $\mathcal{DPS}(RBU)$, the branches $branch(a \rightarrow f)$ and $branch(f)$ are pruned. Although RBU is tighter than LRU, RBU does not have the width pruning effect. *Indeed*, assume conversely that $\mathcal{WPS}(RBU)$. If we remove f from $\mathcal{D}'$, then the sequence $\alpha = d \rightarrow ac \rightarrow af$ containing f with $u_{min}(\alpha) = 353 > mu$ which is a HU sequence, will not be found in the final $\mathcal{HUS}$ result set.

Discussion. Instead of using the minimum utility u_{min} in $\mathcal{HUSM}$, previous studies considered the maximum utility measure u_{max} by considering an optimistic perspective. This latter measure is defined as $u_{max}(\alpha) \stackrel{def}{=} \sum_{\Psi' \in \rho(\alpha)} u_{max}(\alpha, \Psi')$, where $u_{max}(\alpha, \Psi') \stackrel{def}{=} max\{u(\alpha') | \alpha' \in U(\alpha, \Psi')\}$ ([5], [8], [9], [13], [11]). It is neither anti-monotonic nor monotonic. Thus, upper bounds on u_{max} that satisfy anti-monotonic-like properties have been proposed, e.g. SWU [8], SPU [5], $CRoM$ [9], MEU and LEU [11]. For example, $MEU(\alpha) \stackrel{def}{=} \sum_{\Psi' \in \rho(\alpha)} MEU(\alpha, \Psi')$ and for $\beta = \alpha \diamond y$, $LEU(\beta) \stackrel{def}{=} \sum_{\Psi' \in \rho(\beta)} MEU(\alpha, \Psi')$ if $\alpha \neq <>$, and $LEU(y) \stackrel{def}{=} SWU(y)$ if $\alpha = <>$, where $MEU(\alpha, \Psi') \stackrel{def}{=} max\{u(\alpha') + u(rem(\alpha, \Psi', i_p)) | \forall \alpha' \in U(\alpha, \Psi'), i_p = end(\alpha', \Psi')\}$ [11]. Then, $u_{min} \ll u_{max} \ll MEU \ll LEU \ll SWU$, RBU and LRU UBs on u_{min} are strictly tighter than MEU and LEU UBs on u_{max}, respectively. Obviously, the number of high utility sequences using u_{max} is not less than that when using u_{min}.

For example, consider another integrated QSDB $\mathcal{D}'' = \{\Psi'_1 = (a, 2)(c, 5)(e, 3) \rightarrow (b, 1) \rightarrow (b, 2) \rightarrow (d, 1) \rightarrow (b, 80) \rightarrow (c, 2) \rightarrow (c, 4)\ (e, 1) \rightarrow (d, 3) \rightarrow (c, 6) \rightarrow (c, 3)(e, 2), \Psi'_2 = (a, 5)(e, 2) \rightarrow (d, 4) \rightarrow (c, 3)(e, 4)\}$ and $mu = 50$. Then, the sequence $\alpha = b \rightarrow d$ is a HU sequence using u_{max} but not u_{min}, because $u_{max}(\alpha) = 83 \geqslant mu > u_{min}(\alpha) = 2$, and the number of HU sequences using u_{max} or u_{min} are respectively 4128 or 2752. For $\beta = a \rightarrow b \rightarrow d \rightarrow ce$, we have $u_{min}(\beta) = 9$, $RBU(\beta) = 23$, $LRU(\beta) = 25$, $u_{max}(\beta) = 90$, $MEU(\beta) = 90$, $LEU(\beta) = 96$ and $SWU(\beta) = 115$. It is observed that $u_{min}(\beta) < RBU(\beta) < LRU(\beta) < u_{max}(\beta) = MEU(\beta) < LEU(\beta) < SWU(\beta)$. Hence, RBU and LRU are only UBs on u_{min} but not on u_{max}.

Moreover, note that the measure SPU ([5], [12]) is not really an UB on u_{max}, and the $CRoM$ UB on u_{max} [9] is too large, i.e. it may be greater than the traditional SWU UB. *Indeed*, SPU can be defined as $SPU(\alpha) \stackrel{def}{=} \sum_{\Psi' \in \rho(\alpha)} [u_p(\alpha, \Psi') + u(rem(\alpha, \Psi', i^*_p))]$, where $i^*_p \stackrel{def}{=} FEnd(\alpha, \Psi')$ is the first ending (or *pivot*) of α in Ψ' and $ub_p(\alpha, \Psi') \stackrel{def}{=} max\{u(\alpha') | \forall \alpha' \in U(\alpha, \Psi')$ and $end(\alpha', \Psi') = i^*_p\}$. Then, $SPU(\beta)$ (= 24) $< u_{max}(\beta)$. Thus, the well-known USpan [5] algorithm for mining $\mathcal{HUS}$ using u_{max} and its SPU UB is an incomplete algorithm, i.e. it may miss some HU sequences. Similar, for the $CRoM$ UB (see more details in [9]) and $\alpha = b \rightarrow d$, then $CRoM(\alpha) = 182$, $SWU(\alpha) = 115$. Hence, $CRoM(\alpha) > SWU(\alpha)$ and the performance of the HuspExt [9] algorithm that relies on $CRoM$ may decrease in such cases.

3.3 The EHUSM Algorithm

Based on the above theoretical results, we design an efficient algorithm named **EHUSM** (Efficient High Utility Sequence Mining) for mining the set $\mathcal{HUS}$ of all high utility sequences when considering the u_{min}measure. The pseudo-code is shown in Algorithm 1. It takes as input a QSDB $\mathcal{D}'$ and the mu threshold. At the first level of the prefix-tree, the algorithm applies $\mathcal{WPS}(SWU)$ or $\mathcal{WPS}(LRU)$. The algorithm scans QSDB $\mathcal{D}'$ once to calculate the set $IS \stackrel{def}{=} \{i \in \mathcal{A} | SWU(i) \geqslant mu\}$ of HU candidate items. Then, all irrelevant items (items in $\mathcal{A} \backslash IS$) are removed from $\mathcal{D}'$ (lines 1-2). Next, the procedure **SearchHUS** is called for each item $i \in IS$ (line 3).

Algorithm 1. EHUSM($\mathcal{D}'$, mu)

. *Input*: QSDB $\mathcal{D}'$, the mu threshold.
. *Output*: all HU sequences in $\mathcal{HUS}$.
1. Scan once $\mathcal{D}'$ to find $IS = \{i \in \mathcal{A} | SWU(i) \geqslant mu\}$;
2. Remove from $\mathcal{D}'$ all irrelevant items in $\mathcal{A} \backslash IS$.
3. **for** each item $i \in IS$ **do SearchHUS**(i, IS, mu);

Algorithm 2. SearchHUS(α, IS, mu)

. *Input*: sequence α, a candidate item set IS for extensions, the mu threshold.
. *Output*: α if it is a HU sequence.
1. **if** ($RBU(\alpha) < mu$) **then return**; // *depth pruning strategy*
2. **if** ($u_{min}(\alpha) \geqslant mu$) **then** output α;
3. $newI = newS = \emptyset$; // *candidate item sets for extensions of* α
4. **for** each item $i \in IS$ **do** { // *width pruning strategy*
5. **if** ($i >$*lastItem*(α) and $LRU(\alpha \diamond_i i) \geqslant mu$) **then** $newI = newI \cup \{i\}$;
6. **if** ($LRU(\alpha \diamond_s i) \geqslant mu$) **then** $newS = newS \cup \{i\}$;
7. }
8. $newIS := newI \cup newS$;
9. Remove from PDB $\mathcal{D}'_\alpha$ all irrelevant items i in $IS \backslash newIS$.
10. **if** ($RBU(\alpha) < mu$) **then return**; // $\mathcal{DPS}$ *after reducing* $\mathcal{D}'_\alpha$
11. **for** each item $i \in newI$ **do SearchHUS**($\alpha \diamond_i i$, $newIS$);
12. **for** each item $i \in newS$ **do SearchHUS**($\alpha \diamond_s i$, $newIS$);

The recursive **SearchHUS** procedure (Algorithm 2) takes as input a sequence α, a candidate item set IS, and the mu threshold. The procedure applies the depth pruning strategy $\mathcal{DPS}(RBU)$ in line 1. If $u_{min}(\alpha) \geqslant mu$, the HU sequence α is output (line 2). Next, in lines 4-9, based on the inclusion $NewIS \stackrel{def}{=} IS_{LRU}(\alpha \diamond i) \subseteq IS \stackrel{def}{=} IS_{LRU}(\alpha)$, the width pruning strategy $\mathcal{WPS}(LRU)$ is applied

for extensions of α. Afterward, $\mathcal{DPS}(RBU)$ is used again in line 10. Finally, the **SearchHUS** procedure is recursively called for each item using the two candidate item sets $newI$ and $newS$ to perform i-extensions and s-extensions, respectively (lines 11-12). Theorems 1 and 2 guarantee the correctness of EHUSM using two strategies $\mathcal{WPS}$ and $\mathcal{DPS}$. They are based on LRU and RBU UBs and help to prune unpromising branches in the prefix-tree early without missing HU sequences.

The execution of EHUSM is illustrated as follows. For $mu = 350$, after performing line 1 of Algorithm 1, two irrelevant items b and d are removed from $\mathcal{D}'$ (line 2) because $LRU(b) = SWU(b) = 322$, $LRU(d) = 228$. Afterward, updated values of RBU and LRU of all remaining items in the set $IS = \{a, c, d, e, f\}$ from the reduced database $\mathcal{D}'$ (without b and d) may be also reduced. *Indeed*, for instance, consider the original QSDB $\mathcal{D}'$, where $LRU(e)$ $(= SWU(e)) = 609$. In the reduced database $\mathcal{D}'$ after discarding two items b and d, the new updated $LRU(e)$ is 569 (< 609).

Now, we briefly show the execution of the SearchHUS(α, IS) procedure (Algorithm 2). For the sequence $\alpha = f$, since $RBU(f) = 154 < mu$, $branch(f)$ is deeply pruned (line 1). For the sequence $\alpha = e$, $LRU(e)$ = 569 > $RBU(e) = 369$ > mu > $u_{min}(e) = 22$. Hence lines 1-2 are skipped. Moreover, by lines 3-8, because $LRU(ef) = 0$, $LRU(e \rightarrow a) = LRU(e \rightarrow f) = 369$ (> mu), and LRU values of all s-extensions $e \rightarrow y$ of e are less than mu, for $y \in \{c, d, e\}$, then $newI = \emptyset$ and $newS = af$ ($\subseteq IS$), so $newIS = af$ and $IS \backslash newIS = cde$. After deleting all irrelevant items c, d, e from $\mathcal{D}'_e$, then we obtain the reduced $\mathcal{D}'_e$ = {(a, 3)⟶(a, 5)⟶(a, 5)⟶(a, 4)(f, 36), (a, 3)⟶(a, 1)(f, 9)⟶(a, 4), (a, 1)(f, 18), (a, 2)⟶(a, 9)(f, 72)}. Hence, the updated value $LRU(e)$ = 194 (< 569) is less than mu and $branch(e)$ is pruned earlier than if $\mathcal{D}'_e$ was not reduced (line 10). SearchHUS is applied for other sequences in a similar way. Finally, EHUSM returns only one HU sequence $\alpha = d \rightarrow ac \rightarrow af$, with $u_{min}(\alpha) = 353$.

4 EXPERIMENTAL EVALUATION

Experiments were performed on a computer having an Intel(R) Core (TM) i5-2320, 3.0 GHz CPU and 4 GB of memory, running Windows 8.1. EHUSM is the first algorithm for mining high utility sequences based on the minimum utility measure, u_{min}. It is implemented using the Java SE 1.8 programming language.

Table 3: Parameters of the IBM Synthetic Data Generator

Parameter	Meaning
D	Number of sequences (in thousands) in the database
C	Average number of itemsets per sequence
T	Average number of items per itemset
N	Number of different items (in thousands) in the database
S	Average number of items in maximal sequences
I	Average number of itemsets in maximal sequences

In the experiments, we consider two well-known benchmark real-life datasets named Gazelle and Snake [22], and three synthetic datasets generated using the IBM Quest data generator (obtained from [22]) using parameters described in Table 3. Characteristics of the considered datasets are shown in Table 4.

Table 4: Characteristics of databases

Database	#seq.	#items	Avg. length	Data type
Gazelle	59,601	497	2.5	web click stream
Snake	163	20	60.6	protein sequences
D4C7T5N5S6I4	4,000	5,000	28.7	Synthetic
D0.5C10T15N2S6I4	500	2,000	127.7	Synthetic
D5C10T5N5S6I4	5,000	5,000	42.9	Synthetic

4.1 Comparison of UBs

In a first experiment, we compare the pruning effect of the proposed RBU and LRU UBs with the well-known SWU UB on two QSDBs, the real-life Gazelle and the synthetic D4C7T5N5S6I4 datasets. In this experiment, the mu threshold is varied and the runtime and number of candidate sequences are measured. Six versions of the EHUSM algorithms are compared, respectively using the *SWU*, *LEU*, *MEU*, *RBU*, *LBU*, and both the *RBU* and *LBU* UBs, to reduce the search space.

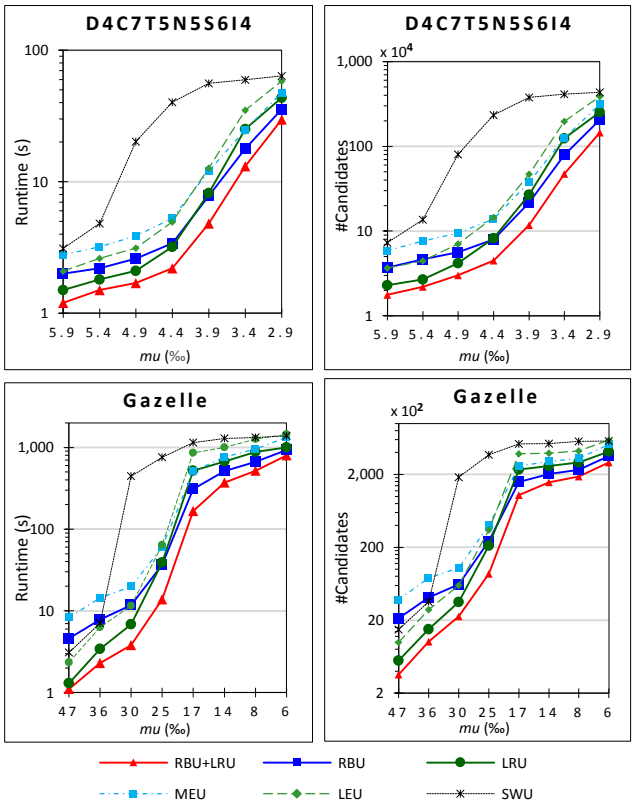

Figure 1: Pruning effect of UBs.

Results are shown in Figure 1. It is observed that using only the RBU or LRU UB is usually better than using the SWU, *MEU* and *LEU* because much less candidate sequences (*#Candidates*) are generated. Moreover, it is observed on D4C7T5N5S6I4 that using LRU can result in generating less candidates than when using RBU for *high* mu values (e.g. higher than 0.44%). The reason is

that the pruning condition ($LRU(\alpha) < mu$) of LRU can be frequently applied and the width pruning effect of LRU is stronger than the depth pruning effect of RBU.

For *low* mu thresholds (e.g. smaller than 0.44%), RBU is better than LRU since the pruning condition ($RBU(\alpha) < mu$) of RBU is applied more often than LRU, although the pruning effect of RBU is weaker. The version of EHUSM using both RBU and LRU is always faster by 8, 2 and 2 (or 24, 4 and 4) times on average compared to the version using only the SWU, MEU and LEU on the D4C7T5N5S6I4 (or Gazelle) dataset, respectively.

On overall, these results show that the two proposed upper bounds and corresponding pruning strategies are very effective for pruning the search space and that they should be used together. Thus, for the following experiments, we only consider the version of EHUSM (using both RBU and LRU), to evaluate the scalability of EHUSM in terms of runtime and memory usage w.r.t. the mu parameter and database parameters.

4.2 Influence of the *mu* parameter

The second experiment assesses the influence of the mu parameter on time, memory consumption and number of discovered HU sequences (#$\mathcal{HUS}$) for the datasets of Table 4.

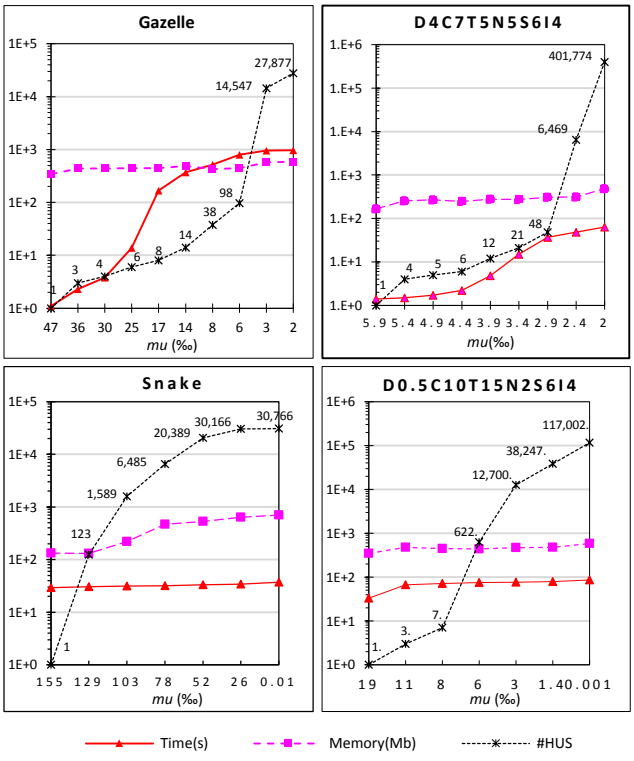

Figure 2: Influence of varying the *mu* threshold.

Results depicted in Figure 2 show that as mu is decreased, the runtime of EHUSM increases and memory usage does not significantly change on Gazelle and D4C7T5N5S6I4 for low database density $\delta \stackrel{\text{def}}{=} \frac{T}{N} = 1\%$. In particular, for very low mu values on Gazelle (from 6% to 2%) and D4C7T5N5S6I4 (from 2.9% to 2%), although the number of HU sequences (#$\mathcal{HUS}$) increases, execution time and memory usage only slightly increase. On the dense Snake and D0.5C10T15N2S6I4 databases (with a high density of $\delta = 7.5\%$), when mu is set to very high values (e.g. $mu = 15.5\%$ for Snake and $mu = 1.9\%$ for D0.5C10T15N2S6I4) few patterns are found (e.g. #$\mathcal{HUS}$ = 1). However, the number of HU sequences increases very quickly when mu is decreased (e.g. for $mu = 0.0001\%$ on D0.5C10T15N2S6I4), while the runtime slightly fluctuates.

4.3 Influence of database parameters

To evaluate the influence of database characteristics on the efficiency of EHUSM, a third experiment was performed where parameters of synthetic datasets (see Table 3) were varied. In the following, a parameter letter followed by the wildcard (*) symbol indicates that the parameter is varied for that dataset. Figure 3 shows how the time, memory consumption, and number of HU candidates (#Candidates) generated by EHUSM vary when dataset parameters are increased. It is observed that the algorithm has nearly linear scalability in terms of runtime.

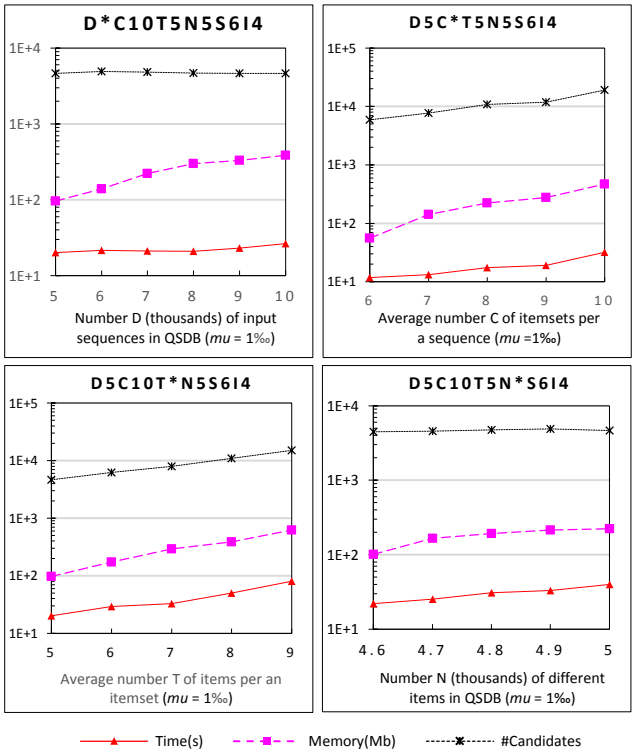

Figure 3: Influence of database parameters.

Overall, the above experiments have shown that the proposed algorithm is efficient in terms of runtime and memory usage for both real-life and synthetic datasets. It is also observed that using simultaneously the two novel RBU and LRU UBs in EHUSM is much more efficient than using the traditional SWU UB.

5. CONCLUSION

This paper proposed a novel utility measure u_{min} to evaluate sequences in QSDBs. Moreover, two new upper bounds on u_{min}, named RBU and LRU are proposed, which are tighter than the traditional SWU UB. Two depth and width pruning strategies are developed based on the two UBs to prune low utility branches of the prefix search tree early. Based on these strategies, an efficient algorithm named EHUSM has been designed for mining the set of high utility sequences $\mathcal{HUS}$. To our best knowledge, this is the first algorithm for mining $\mathcal{HUS}$ based on u_{min}. Experiments conducted on both real-life and synthetic QSDBs have shown that using the two proposed UBs considerably decreases the number of candidate sequences, as well as the runtime and memory consumption of EHUSM.

The two novel upper bounds and corresponding pruning strategies proposed in this paper could be applied to mine concise representations of $\mathcal{HUS}$ such as the sets of all closed and generator high utility sequences. Mining generator and closed sequences is useful as they often have very small cardinalities and summarize the set of $\mathcal{HUS}$. Extending EHUSM for these problems will be considered in future work.

ACKNOWLEDGMENTS

This work is funded by Vietnam's National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2017.300.

REFERENCES

[1] Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., "Mining high utility web access sequences in dynamic web log data," in *Proc. Software Engineering AI Networking and Parallel/Distributed Computing (SNPD), 2010 11th ACIS International Conference*, 2010a.

[2] 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," *Advanced Research in Applied Artificial Intelligence*, pp. 616-626, 2012.

[3] Zihayat, M., Davoudi, H., An, A., "Top-k utility-based gene regulation sequential pattern discovery," in *Proc. Bioinformatics and Biomedicine (BIBM), 2016 IEEE International Conference*, 2016a.

[4] Dalmas, B., Fournier-Viger, P., Norre, S., "TWINCLE: A Constrained Sequential Rule Mining Algorithm for Event Logs," in *Proc. 9th International KES Conference (IDT-KES 2017), Springer*, 2017.

[5] Yin, J., Zheng, Z., Cao, L., "USpan: an efficient algorithm for mining high utility sequential patterns," in *Proc. of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[6] Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., "A novel approach for mining high-utility sequential patterns in sequence databases," *ETRI Journal*, vol. 32, p. 676–686, 2010b.

[7] Shie, B. E., Yu, P. S., Tseng, V. S., "Mining interesting user behavior patterns in mobile commerce environments," *Appl. Intell.*, vol. 38, pp. 418-435, 2013.

[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 Systems with Applications*, vol. 41, no. 11, pp. 5071-5081, 2014.

[9] Alkan, O. K., Karagoz, P., "CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 10, pp. 2645-2657, 2015.

[10] Zihayat, M., Hut, Z. Z., An, A., Hut, Y., "Distributed and parallel high utility sequential pattern mining," in *Proc. Big Data (Big Data), 2016 IEEE International Conference*, 2016b.

[11] Lin, J. C. W., Zhang, J., Fournier-Viger, P., "High-Utility Sequential Pattern Mining with Multiple Minimum Utility Thresholds," in *Proc. Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, 2017.

[12] Yin, J., Zheng, Z., Cao, L., Song, Y., Wei, W., "Efficiently mining top-k high utility sequential patterns," in *Proc. 2013 IEEE 13th International Conference on Data Mining (ICDM)*, 2013.

[13] Wang, J. Z., Huang, J. L., Chen, Y. C., "On efficiently mining high utility sequential patterns," *Knowledge and Information Systems*, vol. 49, no. 2, pp. 597-627, 2016.

[14] Wu, C. W., Lin, Y. F., Yu, P. S., Tseng, V. S., "Mining High Utility Episodes in Complex Event Sequences," in *Proc. KDD'13 Conference*, 2013.

[15] Zihayat, M., Wu, C. W., An, A., Tseng, V. S., "Mining high utility sequential patterns from evolving data streams," in *Proc. of the ASE Big Data & Social Informatics 2015*, 2015.

[16] Zida, S., Fournier-Viger, P., Wu, C. W., Lin, J. C. W., Tseng, V. S., "Efficient Mining of High Utility Sequential Rules," in *Proc. 11th Intern. Conference on Machine Learning and Data Mining (MLDM 2015). Springer, LNAI 9166*, 2015.

[17] Dave, U., Shah, J., "Efficient Mining of High Utility Sequential Pattern from Incremental Sequential Dataset," *International Journal of Computer Applications*, vol. 122, no. 12, pp. 22-28, 2015.

[18] Dinh, T., Quang, M. N., Le, B., "A Novel approach for hiding high utility sequential patterns," in *Proc. Int. Symp. Information and Communication Technology*, 2015.

[19] Dinh, T., Huynh, V.N., Le, B., "Mining Periodic High Utility Sequential Patterns," in *Proc. Asian Conference on Intelligent Information and Database Systems*, 2017.

[20] Xu, T., Dong, X., Xu, J., Dong, X., "Mining High Utility Sequential Patterns with Negative Item Values," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 31, no. 10, pp. 1750035.1-17, 2017.

[21] Zhang, B., Lin, J. C. W., Fournier-Viger, P., Li, T., "Mining of high utility-probability sequential patterns from uncertain databases," *PLoS ONE*, vol. 12, no. 7, pp. 1-21, 2017.

[22] Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., "SPMF: a Java Open-Source Pattern Mining Library," *J. Machine Learning Research*, vol. 15, no. 1, pp. 3389-3393, 2014.