High-Utility Sequential Pattern Mining with Multiple Minimum Utility Thresholds

Jerry Chun-Wei Lin[1], Jiexiong Zhang[1], and Philippe Fournier-Viger[2]

[1] School of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
[2] School of Natural Sciences and Humanities
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
jerrylin@ieee.org, jiexiong.zhang@foxmail.com, philfv@hitsz.edu.cn

Abstract. High-utility sequential pattern mining is an emerging topic in recent decades and most algorithms were designed to identify the complete set of high-utility sequential patterns under the single minimum utility threshold. In this paper, we first propose a novel framework called high-utility sequential pattern mining with multiple minimum utility thresholds to mine high utility sequential patterns. A high-utility sequential pattern with multiple minimum utility thresholds algorithm, a lexicographic sequence (LS)-tree, and the utility-linked (UL)-list structure are respectively designed to efficiently mine the HUSPs. Three pruning strategies are then introduced to lower the upper-bound values of the candidate sequences, and reduce the search space by early pruning the unpromising candidates. Substantial experiments on real-life datasets show that our proposed algorithms can effectively and efficiently mine the complete set of HUSPs with multiple minimum utility thresholds.

Keywords: Data mining · Sequence · High-utility sequential pattern · Multiple thresholds

1 Introduction

Sequential pattern mining (SPM) [1–4] has been emerging as an interesting and critical topic in recent years. The main target of SPM is to discover the set of frequent sequences measured by a user-specified minimum support threshold. This process may, however, not be informative for decision makers since they cannot discover the patterns with high profit or having great impact. To address this issue, high-utility itemset mining (HUIM) has been introduced [5, 6] to consider both the quantity and the unit of profit of itemsets to mine the high-utility itemsets (HUIs). Considering the ordered sequences in real-life situations, high-utility sequential pattern mining (HUSPM) [7–10] was introduced for mining more informative sequential patterns. However, HUSPM meets an important limitation since it necessitates to measure all patterns with a single minimum utility threshold for finding the complete set of high-utility sequential patterns (HUSPs). Utilizing a single threshold for all itemsets or sequences in databases

means that they are treated as the same importance, which is not convincing in real-world situations.

In this paper, we first design a new framework called high-utility sequential pattern mining with multiple minimum utility thresholds for mining the set of HUSPs. It allows to set different thresholds for items instead of a single minimum utility threshold and avoid the "rare item" problem [11]. Besides, a lexicographic-sequence (LS)-tree is introduced as the search space to mine the complete set of HUSPs. A novel compressed utility-linked (UL)-list structure is further designed to store the information of patterns. Three pruning strategies are then developed to reduce the search space and improve mining performance of the designed algorithm, which can be observed in the experiments.

2 Preliminaries and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of distinct items. A quantitative itemset, denoted as $v = [(i_1, q_1)\ (i_2, q_2)\ \ldots\ (i_c, q_c)]$, is a subset of I and each item in the quantitative itemset is associated with a quantity (internal utility). An itemset, denoted as $w = [i_1, i_2, \ldots, i_c]$, is a subset of I without quantities. A quantitative sequence is an ordered list of one or more quantitative itemsets, which is denoted as $s = <v_1, v_2, \ldots, v_d>$. A sequence is an ordered list of one or more itemsets without quantities, which is denoted as $t = <w_1, w_2, \ldots, w_d>$. For convenience, we use "q-" as the abbreviation of "quantitative". Thus, the "q-sequence" indicates the sequences with quantities, and "sequence" indicates the sequences without quantities, which can be also defined for the "q-itemset". For example, $<[(a, 2)\ (b, 1)], [(c, 3)]>$ is a q-sequence while $<[ab], [c]>$ is a sequence. $[(a, 2)\ (b, 1)]$ is a q-itemset and $[ab]$ is an itemset. A quantitative sequential database is a set of transactions $D = \{S_1, S_2, \ldots, S_n\}$, where each transaction $S_q \in D$ is a q-sequence, and has a unique identifier q, named its *SID*. In addition, each item in D is associated with a profit (external utility), and denoted as $pr(i_j)$.

Table 1. A quantitative sequential database

SID	**Q-sequence**
S_1	$<[(a{:}2)(c{:}3)], [(a{:}3)(b{:}1)(c{:}2)], [(a{:}4)(b{:}5)(d{:}4)], [(e{:}3)]>$
S_2	$<[(a{:}1)(e{:}3)], [(a{:}5)(b{:}3)(d{:}2)], [(b{:}2)(c{:}1)(d{:}4)(e{:}3)]>$
S_3	$<[(e{:}2)], [(c{:}2)(d{:}3)], [(a{:}3)(e{:}3)], [(b{:}4)(d{:}5)]>$
S_4	$<[(b{:}2)(c{:}3)], [(a{:}5)(e{:}1)], [(b{:}4)(d{:}3)(e{:}5)]>$
S_5	$<[(a{:}4)(c{:}3)], [(a{:}2)(b{:}5)(c{:}2)(d{:}4)(e{:}3)]>$
S_6	$<[(f{:}4)], [(a{:}5)(b{:}3)], [(a{:}3)(d{:}4)]>$

A running example of the quantitative sequential database is shown in Table 1, which consists of 6 transactions and 6 items. The external utility of each item is defined in *profit-table* as: $\{pr(a){:}5, pr(b){:}3, pr(c){:}4, pr(d){:}2, pr(e){:}1, pr(f){:}6\}$. From the given example, it can be seen that $[(a{:}2)(c{:}3)]$ is the first q-itemset in a transaction S_1. The quantity of an item (a) in this q-itemset is 2, and its utility is calculated as $(2{\times}5)(= 10)$.

Definition 1. *The minimum utility threshold of an item (i_j) in a quantitative sequential database D is denoted as $mu(i_j)$, and the multiple minimum utility threshold table (denoted as MMU-table) consists of the minimum utility threshold of each item in D, which can be defined as:*

$$MMU\text{-}table = \{mu(i_1), mu(i_2), \ldots, mu(i_m)\}. \quad (1)$$

In Table 1, we assume that the *MMU-table* is defined as *MMU-table* = {$mu(a)$, $mu(b)$, $mu(c)$, $mu(d)$, $mu(e)$, $mu(f)$} = {500, 500, 500, 200, 500, 70}.

Definition 2. *The minimum utility threshold of a sequence t is denoted as $MIU(t)$, which is the least mu value among the items in t and defined as:*

$$MIU(t) = min\{mu(i_j)|i_j \in t\}. \quad (2)$$

In Table 1, $MIU(<[a]>) = min\{mu(a)\} = 500$, and $MIU(<[ad]>) = min\{mu(a), mu(d)\} = min\{500, 200\} = 200$.

Definition 3. *The utility of an item (i_j) in a q-itemset v is denoted as $u(i_j, v)$, and defined as:*

$$u(i_j, v) = q(i_j, v) \times pr(i_j), \quad (3)$$

where $q(i_j, v)$ is the quantity of (i_j) in v, and $pr(i_j)$ is the profit of (i_j).

In Table 1, the utility of an item (c) in the first q-itemset of S_1 is calculated as: $u(c, [(a:2)(c:3)]) = q(c, [(a:2)(c:3)]) \times pr(c)$ (= 3×4)(= 12).

Definition 4. *The utility of a q-itemset v is denoted as $u(v)$ and defined as:*

$$u(v) = \sum_{i_j \in v} u(i_j, v). \quad (4)$$

In Table 1, $u([(a{:}2)(c{:}3)]) = u(a, [(a{:}2)(c{:}3)]) + u(c\ , [(a{:}2)(c{:}3)])$ (= 2×5 + 3×4)(= 22).

Definition 5. *The utility of a q-sequence $s = <v_1, v_2, \ldots, v_d>$ is denoted as $u(s)$ and defined as:*

$$u(s) = \sum_{v \in s} u(v). \quad (5)$$

In Table 1, $u(S_1) = u([(a{:}2)(c{:}3)]) + u([(a{:}3)(b{:}1)(c{:}2)]) + u([(a{:}4)(b{:}5)(d{:}4)]) + u([(e{:}3)])$ (= 22 + 26 + 43 + 3)(= 94).

Definition 6. *The utility of a quantitative sequential database D is denoted as $u(D)$ and defined as:*

$$u(D) = \sum_{s \in D} u(s). \quad (6)$$

In Table 1, $u(D) = u(S_1) + u(S_2) + u(S_3) + u(S_4) + u(S_5) + u(S_6)$ (= 94 + 67 + 56 + 67 + 76 + 81)(= 441).

Definition 7. *Given a q-sequence* $s = <v_1, v_2, \ldots, v_d>$ *and a sequence* $t = <w_1, w_2, \ldots, w_{d'}>$*, iff* $d = d'$ *and the items in* v_k *are the same as the items in* w_k *for* $1 \leq k \leq d$*, t matches s, which is denoted as* $t \sim s$*.*

In Table 1, <[ac], [abc], [abd], [e]> matches S_1. However, the target sequence may have multiple matches in a q-sequence. For example, <[a],[b]> has three matches as <[a:2],[b:1]>, <[a:2],[b:5]> and <[a:3],[b:5]> in S_1. This feature brings more challenges to the designed framework.

Definition 8. *Given two itemsets* w *and* w'*,* w *is said to be contained in* w' *as* $w \subseteq w'$ *iff* w *is a subset of* w' *. Given two q-itemsets* v *and* v'*,* v *is said to be contained in* v' *as* $v \subseteq v'$ *iff for any item in* v*, there exists the same item having the same quantity in* v'*.*

In Table 1, an itemset [ac] is contained in the itemset [abc]. The q-itemset [(a:2)(c:3)] is contained in [(a:2)(b:1)(c:3)], but is not contained in [(a:2)(b:3)(c:1)].

Definition 9. *Given two sequences* $t = <w_1, \ldots, w_d>$ *and* $t' = <w'_1, \ldots, w'_{d'}>$*,* t *is said to be contained in* t' *as* $t \subseteq t$*' iff there exists an integer sequence* $1 \leq k_1 \leq k_2 \leq \ldots \leq d'$ *such that* $w_j \subseteq w'_{k_j}$ *for* $1 \leq j \leq d$*. Given two q-sequences* $s = <v_1, \ldots, v_d>$ *and* $s' = <v'_1, \ldots, v'_{d'}>$*,* s *is said to be contained in* s' *as* $s \subseteq s'$ *iff there exists an integer sequence* $1 \leq k_1 \leq k_2 \leq \ldots \leq d'$ *such that* $v_j \subseteq v'_{k_j}$ *for* $1 \leq j \leq d$*. For convenience, we use* $t \subseteq s$ *to indicate that* $t \sim s_k \wedge s_k \subseteq s$*.*

In Table 1, <[(a:2)],[(e:3)]> and <[(a:4)],[(e:3)]> are contained in S_1, but <[(a:1)],[e:3]> and <[(a:4)],[(e:4)]> are not contained in S_1.

Definition 10. *The utility of a sequence* t *in a q-sequence* s *is denoted as* $u(t, s)$ *and defined as:*

$$u(t, s) = max\{u(s_k) | t \sim s_k \wedge s_k \subseteq s\}. \tag{7}$$

In Table 1, $u(<[a],[b]>, S_1) = max\{u(<[a:2],[b:1]>), u(<[a:2],[b:5]>), u(<[a:3],[b:5]>)\}$ $= max\{13, 25, 30\} = 30$. From this example, it shows that a sequence has multiple utility values in a q-sequence, which is much different from traditional SPM and HUIM.

Definition 11. *The utility of a sequence* t *in a quantitative sequential database* D *is denoted as* $u(t)$ *and defined as:*

$$u(t) = \sum_{s \in D} \{u(t, s) | t \subseteq s\}. \tag{8}$$

In Table 1, $u(<[a],[b]>) = u(<[a],[b]>, S_1) + u(<[a],[b]>, S_2) + u(<[a],[b]>,$ $S_3) + u(<[a],[b]>, S_4) + u(<[a],[b]>, S_5)$ $(= 30 + 31 + 27 + 37 + 35)(= 160)$.

Definition 12 (High-Utility Sequential Pattern, HUSP). *A sequence* t *in a quantitative sequential database* D *is defined as a high-utility sequential pattern (HUSP) iff its utility is no less than the minimum threshold of* t *as:*

$$HUSP \leftarrow \{t | u(t) \geq MIU(t)\}. \tag{9}$$

In Table 1, $u(<[a],[b]>)(= 160)$ and $MIU(<[a],[b]>)(= 500)$; $<[a],[b]>$ is not a HUSP since $u(<[a],[b]>)(= 160) < MIU(<[a],[b]>) (= 500)$.

Problem Statement: Given a quantitative sequential database and a *MMU-table*, the problem of high-utility sequential pattern mining with multiple minimum utility thresholds is to discover the complete set of HUSPs whose utility values are no less than their *MIU* values.

3 Proposed Framework

Based on the above concepts, a baseline high-utility sequential pattern algorithm with multiple minimum utility thresholds is first designed. The proposed algorithm first scans the database to find the 1-sequences for building the lexicographic sequence (LS)-tree. For each node in the LS-tree, a corresponding projected database is built and consisting of the utility-linked (UL)-lists transformed from the transactions in the original database, which is used to calculate the actual utilities and upper-bound values of the generated candidates. Each UL-list can be used to represent each transaction (q-sequence). To generate the child nodes (supersets) of the node in the LS-tree, the *I-Concatenation* and *S-Concatenation* operations are used to combine the candidate HUSPs with items, forming new candidate HUSPs. Each new candidate HUSP (child node) will be evaluated to determine whether it is an actual HUSP and whether the algorithm should explore its supersets (child nodes). The above processes are recursive performed until no candidates are required to be determined. After that, the set of HUSPs are then returned.

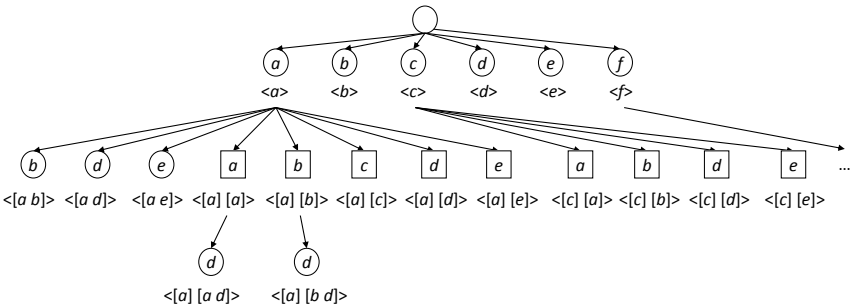

Fig. 1. A lexicographic sequence (LS)-tree

3.1 Lexicographic Sequence (LS)-tree

In the designed algorithm, a lexicographic sequence (LS)-tree is built to ensure the **completeness** and **correctness** for mining the HUSPs. The database is first scanned to find the satisfied 1-sequences against their *PMIU* value (which will be described below). The LS-tree is then built from 1-sequences by adopting a depth-first search strategy. The child nodes the are combination results

of *I-Concatenation* or *S-Concatenation* of the parent node. Fig. 1 shows a LS-tree built from the running example of Table 1. In Fig. 1, a circle represents an *I-Concatenation* sequence while the square represents a *S-Concatenation* sequence. Notice that nodes in the LS-tree represent the candidates (search space) of the HUSPs.

3.2 Utility-Linked (UL)-list Structure

In the designed algorithm, the utility and upper-bound values of candidates are calculated from each transaction. This process has the problem of multiple matches, and requires more computations. To handle this situation, we introduce a novel compact utility-linked (UL)-list structure to store the utility information of each transaction. This structure efficiently helps generate the utility of the *I-Concatenation* and *S-Concatenation* sequences for later mining process. The UL-list structure of S_1 is shown in Table 2.

Table 2. The utility-linked (UL)-list structure of S_1

U&P information	<[(a, 10, 84, 3)(c, 12, 72, 5)], [(a, 15, 57, 6)(b, 3, 54, 7) (c, 8, 46, -)], [(a, 20, 26, -)(b, 15, 11, -)(d, 8, 3, -)], [e, 3, 0, -]>
header_table	(a, 1) (b, 4) (c, 2) (d, 8) (e, 9)

For the header_table in the UL-list structure, it represents the set of distinct items with their first occurrence positions in the transformed transaction. In Table 2, the distinct items of S_1 are (a), (b), (c), (d), and (e) and their first occurrence positions in S_1 are respectively 1, 4, 2, 8 and 9. For the U&P (utility and position) information, each element respectively represents the **1).item name**, the **2).utility of the item**, the **3).remaining utility of the item**, and the **4).next position of the item**. In Table 2, the utility of the item (a) in the first element is calculated as 10 in S_1; the total utility except the item (a) in S_1 (named as remaining utility) is calculated as 84, and the next position of the item (a) in S_1 is found as 3. The utility and the remaining utility of item can be used to calculate the utility values and the upper-bound values of patterns respectively. The next position of item will be used for concatenation and selecting the maximal utility values (based on definition 10) and the maximal upper-bound values of patterns. For each node in the LS-tree, transactions containing this node (sequence) are transformed into a utility-linked (UL)-list and attached to the projected database of this node. The utilities and upper-bound values of the candidates can be easily calculated from the projected database based on the UL-list structure.

3.3 Concatenations

In the designed algorithm, two operations such as *I-Concatenation* and *S-Concatenation* are used to generate the child nodes (supersets) of the processed nodes.

Definition 13. Given a sequence t and an item i_j, the *I-Concatenation* of t with i_j is to append i_j to the last itemset of t, denoted as $<t \oplus i_j>_{I-Concatenation}$; the *S-Concatenation* of t with i_j is to add i_j as a new itemset to the last of t, denoted as $<t \oplus i_j>_{S-Concatenation}$.

For example, given a sequence $t = <[a],[b]>$ and a new item (c), we can obtain that $<t \oplus c>_{I-Concatenation} = <[a],[bc]>$ and $<t \oplus c>_{S-Concatenation} = <[a],[b],[c]>$. Based on those two operations, the candidates can be thus easily formed in the search space for mining the HUSPs.

As mentioned before, the UL-list structure can be used to find the utilities and the upper-bound values of the candidates for deriving HUSPs. A sequence may, however, find the multiple matches in a q-sequence, which makes a sequence have multiple utilities in a q-sequence. Thus, it necessitates to find the positions of the matches to calculate the utilities and the upper-bound values of the processed node (sequence). For convenience, the position of the last item within each match is defined as the **concatenation point**, and the first concatenation point is named as **start point**. For example in Table 1, assume a sequence $t = <[a],[b]>$, it has three matches in S_1 as $<[a:2],[b:1]>$, $<[a:2],[b:5]>$ and $<[a:3],[b:5]>$. The concatenation points of t in S_1 respectively are 4, 7 and 7, and the start point is 4. The candidate items for *I-Concatenation* are the items appearing in the same itemsets where concatenation points appear. In the above example, the candidate items for *I-Concatenation* are $\{(c:2),(d:4)\}$. The items in the itemsets after the start point are the candidate items for *S-Concatenation*. In the above example, the start point $(= 4)$ appears in the second itemset. Hence, the items after the second itemset are candidate items for *S-Concatenation*, which are $\{(a:4),(b:5),(d:4),(e:3)\}$. Besides, to discover the complete set of HUSPs, the designed algorithm enumerates all candidates by two concatenations in *lexicographic-ascending* order.

3.4 Proposed Algorithm

Since the downward closure property does not hold in the problem of high-utility sequential pattern mining with multiple minimum utility thresholds, new downward closure property is required to reduce the search space for mining the HUSPs. Details are described below.

Definition 14. *Given two q-sequences s and s', if $s \subseteq s'$, the extension of s in s' is the rest of s' after s, which can be denoted as $<s' - s>_{rest}$. Given a sequence t and a q-sequence s, if $t \subseteq s$, the extension of t in s is the rest of s after s_k, which can be denoted as $<s - t>_{rest}$, where s_k is the first match of t in s.*

For example, given two q-sequences $s = <[a:2],[b:5]>$ and S_1 in Table 1, the extension of s in S_1 is $<S_1 - s>_{rest} = <[(d:4)],[(e:3)]>$. Given a sequence $t = <[a],[b]>$, there exists three matches of t in S_1, and the first one is $<[a:2],[b:1]>$. Thus, $<S_1 - t>_{rest} = <[(c:2)],[(a:4)(b:5)(d:4)],[(e:3)]>$.

Definition 15. *The extension items of a sequence t in a quantitative sequential database D is denoted as $I(t)_{rest}$ and defined as:*

$$I(t)_{rest} = \{i_j | i_j \in < s - t >_{rest} \wedge t \subseteq s \wedge s \in D\}. \tag{10}$$

In the above example, $I(<[a],[b]>)_{rest} = \{a, b, c, d, e\}$.

To speed up mining process and maintain the downward closure property, a sequence-weighted utilization (SWU) [12] was thus proposed to maintain the sequence-weighted downward closure (SWDC) property for mining the HUSPs. Based on the SWDC property, it can ensure that if the SWU of a sequence t is less than a threshold, the utility of t is less than the threshold; the utilities of the supersets of t are also less than the threshold. Numerous unpromising candidates can be pruned. The SWU of a sequence t is still much larger than the actual utility of t. Thus, the potential utility (PU) model [12] was also introduced to estimate the lower upper-bound values of the candidates. However, in the designed framework, the thresholds for sequences are different; the SWDC and PU properties cannot be directly applied. To address this issue, we propose the following theorems to maintain the downward closure property in the proposed framework.

Definition 16. *The potential minimum utility threshold of a sequence t in a quantitative sequential database D is denoted as $PMIU(t)$ and defined as:*

$$PMIU(t) = min\{mu(i_j) | i_j \in t \vee i_j \in I(t)_{rest}\}. \tag{11}$$

In Table 1, $PMIU(<[a],[b]>) = min\{mu(i_j) | i_j \in <[a],[b]> \vee i_j \in \{a, b, c, d, e\}\} = min\{500, 500, 500, 200, 500\} = 200$.

Theorem 1. *Given a sequence t, let UB be the upper-bound of t, if $UB < PMIU(t)$, then t and the supersets of t cannot be HUSPs.*

Proof. let t' be a superset of t, we can obtain that $\{i_j | i_j \in t'\} \subseteq \{i_j | i_j \in t \vee i_j \in I(t)_{rest}\}$ and $I(t')_{rest} \subseteq I(t)_{rest}$. Based on the definition of upper-bound, $u(t') \leq UB$ and $u(t) \leq UB$. Then, we have $u(t') \leq UB < PMIU(t) = min\{mu(i_j) | i_j \in t \vee i_j \in I(t)_{rest}\} \leq min\{mu(i_j) | i_j \in t' \vee i_j \in I(t')_{rest}\} = PMIU(t') \leq min\{mu(i_j) | i_j \in t'\} = MIU(t'), u(t) \leq UB < PMIU(t) = min\{mu(i_j) | i_j \in t \vee i_j \in I(t)_{rest}\} \leq min\{mu(i_j) | i_j \in t\} = MIU(t)$. Thus, we can obtain that t and t' cannot be the HUSPs.

From theorem 1, it can be seen that if the upper-bound of a sequence t is less than the $PMIU$ of t, then t and the supersets of t are not HUSPs. The upper-bound (UB) is the maximal possible utility, which can be either the SWU and PU of t, and the $PMIU$ is the least minimum utility threshold of the sequence. With the help of the above theorems, we can ensure that the designed algorithm maintains the completeness and correctness of the discovered HUSPs. Based on the proposed framework and the above theorem, the baseline algorithm for HUSPM with multiple minimum utility thresholds and PGrowth mining approach are respectively introduced in Algorithm 1, Algorithm 2 and Algorithm 3.

The **PGrowth** function uses the depth-first search strategy to enumerate the possible sequences in our defined *alphabetic-ascending* order. The enumerated sequences are based on the *I-Concatenation* and *S-Concatenation* operations. After the concatenation operations, new sequences are measured by **Judge** function. Details are given in Algorithm 3.

Algorithm 1: Proposed algorithm

Input: D, a quantitative sequential database; *utable*, a utility table containing the unit profit of each item; *MMU-table*, a table containing the minimum utility threshold of each item.

Output: The set of $HUSPs$.

```
1  scan D to: 1). calculate u(s) for each s ∈ D; 2). build the UL-list for each s ∈ D;
2  HUSPs ← ∅;
3  for each i_j ∈ D do
4      PD(<i_j>)←{the UL-list of s|<i_j>⊆ s ∧ s ∈ D};
5      calculate SWU(<i_j>), u(<i_j>), PMIU(<i_j>), and MIU(<i_j>);
6      if SWU(<i_j>)≥ PMIU(<i_j>) then
7          if u(<i_j>)≥MIU(<i_j>) then
8              HUSPs←HUSPs∪<i_j>;
9          PGrowth(<i_j>, PD(<i_j>), HUSPs);
10 return HUSPs;
```

Algorithm 2: **PGrowth**(*prefix*, $PD(prefix)$, $HUSPs$)

```
1  scan PD(prefix) to get C^I; // measured by SWU
2  for each i_j ∈ C^I do
3      Judge(<prefix⊕i_j>_{I−Concatenation}, PD(prefix), HUSPs);
4  scan PD(prefix) to get C^S; // measured by SWU
5  for each i_j ∈ C^S do
6      Judge(<prefix⊕i_j>_{S−Concatenation}, PD(prefix), HUSPs);
```

Algorithm 3: **Judge**(*prefix'*, $PD(prefix)$, $HUSPs$)

```
1  PD(prefix')←{the UL-list of s|prefix'⊆ s ∧ s ∈ PD(prefix)};
2  calculate u(prefix'), PU(prefix'), PMIU(prefix'), and MIU(prefix');
3  if PU(prefix')≥PMIU(prefix') then
4      if u(prefix')≥MIU(prefix') then
5          HUSPs←HUSPs∪prefix';
6      PGrowth(prefix', PD(prefix'), HUSPs);
```

3.5 Designed Pruning Strategies

The designed baseline algorithm can effectively discover the complete set of HUSPs. However, the search space of the algorithm is still large since the SWU and PU are the over-estimated upper-bound values, which are both larger than

the actual utility of the pattern. To reduce a large number of candidates, three strategies are designed to improve the performance of the baseline algorithm. First, we will introduce a tighter upper-bound for mining HUSPs. Details are respectively given below.

Definition 17. *The maximal extension utility of a sequence t in a q-sequence s is denoted as $MEU(t, s)$ and defined as:*

$$MEU(t, s) = max\{u(s_k) + u(< s - s_k >_{rest})|t \sim s_k \wedge s_k \subseteq s\}. \quad (12)$$

In Table 1, given a sequence t = <[a],[b]>, t has 3 matches in S_2, which are <[a:1],[b:3]>, <[a:1],[b:2]> and <[a:5],[b:2]>. Thus, u(<S_2 - <[a:1],[b:3]>>$_{rest}$) = u(<[(d:2)],[(b:2)(c:1)(d:4)(e:3)]>) = 25; we cal also obtain that u(<S_2 - <[a:1],[b:2]>>$_{rest}$) = u(<[(c:1)(d:4)(e:3)]>) = 15 and u(<S_2 - <[a:5],[b:2]>>$_{rest}$) = u(<[(c:1)(d:4)(e:3)]>) = 15. The utilities of 3 matches respectively are 14, 11 and 31. Thus, MEU(<[a],[b]>,S_2) = max\{14 + 25, 11 + 15, 31 + 15\} = 46.

Definition 18. *The maximal extension utility of a sequence t in D is denoted as $MEU(t)$ and defined as:*

$$MEU(t) = \sum_{s \in D} \{MEU(t, s)|t \subseteq s\}. \quad (13)$$

In Table 1, given a sequence t = <[a],[b]>, $MEU(t)$ is calculated as (67 + 46 + 37 + 48 + 54) (= 252), which is smaller than $PU(t)$ (= 279).

Theorem 2. *Given a quantitative sequential database D, and two sequences t and t'. If t⊆ t', we can obtain that*

$$MEU(t') \leq MEU(t). \quad (14)$$

Proof. Suppose s is a transaction in D and contains t and t'. Let s_q be a q-sequence satisfying $\{u(s_q) + u(<s$ - $s_q>_{rest})\} = MEU(t, s)$, where $t \sim s_q \wedge s_q \subseteq s$. Let $s_{q'}$ be a q-sequence satisfying $\{u(s_{q'}) + u(<s$ - $s_{q'} >_{rest})\} = MEU(t', s)$ where $t' \sim s_{q'} \wedge s_{q'} \subseteq s$. Since $t \subseteq t'$, we can divide t' into two parts as the prefix t and the extension e such that $t + e = t'$. Similarly, the $s_{q'}$ can also be divided into two parts as the prefix $s_{q'_t}$ matching t and the extension $s_{q'_e}$ matching e such that $s_{q'_t} + s_{q'_e} = s_{q'}$. Thus, $MEU(t', s) = \{u(s_{q'}) + u(< s - s_{q'} >_{rest})\} = \{u(s_{q'_t}) + u(s_{q'_e}) + u(< s - s_{q'} >_{rest})\} \leq \{u(s_{q'_t}) + u(< s - s_{q'_t} >_{rest})\} \leq \{u(s_q) + u(< s - s_q >_{rest})\} = MEU(t, s)$. We can thus obtain that $MEU(t') = \sum_{s \in D}\{MEU(t', s)|t' \subseteq s\} \leq \sum_{s \in D}\{MEU(t, s)|t' \subseteq s\} \leq \sum_{s \in D}\{MEU(t, s)|t \subseteq s\} = MEU(t)$.

Theorem 2 indicates that if the MEU value of a sequence t is less than the minimum utility threshold, the MEU values of the supersets of t are less than the minimum utility threshold.

Theorem 3 (MEU Strategy, MEUS). *Given a quantitative sequential database D and a sequence t, we can obtain that*

$$MEU(t) \leq PU(t) \leq SWU(t) \quad (15)$$

Proof. Since $u(s_k) \leq max\{u(s_k)|t \sim s_k \wedge s_k \subseteq s\} = u(t,s)$ and $u(< s - s_k >_{rest}) \leq u(< s - t >_{rest})$, $MEU(t,s) = max\{u(s_k) + u(< s - s_k >_{rest})|t \sim s_k \wedge s_k \subseteq s\} \leq u(t,s) + u(< s - t >_{rest}) \leq u(s)$. Thus $MEU(t) = \sum_{s \in D}\{MEU(t,s)|t \subseteq s\} \leq \sum_{s \in D}\{u(t,s) + u(< s - t >_{rest})|t \subseteq s\} = PU(t) \leq \sum_{s \in D}\{u(s)|t \subseteq s\} = SWU(t)$.

Theorem 3 indicates that the *MEU* model is a tighter upper-bound compared to the *PU* and *SWU* models. With the *MEU* model, the designed algorithm can reduce more candidates than that of *PU* and *SWU* models. The *MEU* model can be used to estimate the utility values of the candidate sequences and the supersets of candidate sequences. Based on the definition of HUSP and the above theorems, we can obtain that if the *MEU* of a sequence t is less than the *PMIU* of t, t and the supersets of t will not be considered as the HUSPs.

For a candidate sequence t, amounts of candidate items are still required to be processed for *I-Concatenation* and *S-Concatenation*. To reduce the number of candidate sequences in advance, we then propose the look ahead strategy (LAS) to remove the unpromising candidate items.

Theorem 4 (Look Ahead Strategy, LAS). *Given a sequence t and a quantitative sequential database D, two situations can be considered to generate the supersets as:*

1) if i_j is a I-Concatenation candidate item of t, the maximal utility of $<t \oplus i_j>_{I-Concatenation}$ is no more than $\sum_{s \in D}\{MEU(t,s)| < t \oplus i_j >_{I-Concatenation} \subseteq s\}$.

2) if i_j is a S-Concatenation candidate item of t, the maximal utility of $<t \oplus i_j>_{S-Concatenation}$ is no more than $\sum_{s \in D}\{MEU(t,s)| < t \oplus i_j >_{S-Concatenation} \subseteq s\}$.

Proof. For 1), let $t' = <t \oplus i_j>_{I-Concatenation}$ for convenience. From definition 17 and theorem 2, we have $u(t') \leq MEU(t')$ and $MEU(t',s) \leq MEU(t,s)$. Thus $u(t') \leq MEU(t') = \sum_{s \in D}\{MEU(t',s)|t' \subseteq s\} \leq \sum_{s \in D}\{MEU(t,s)|t' \subseteq s\}$. In the same way, 2) is true.

Based on the LAS, it can be used to quickly remove the unpromising candidate items for *I-Concatenation* and *S-Concatenation* of sequence t without calculating the *MEU* values of $<t \oplus i_j>_{I-Concatenation}$ and $<t \oplus i_j>_{S-Concatenation}$. Since the upper-bound in LAS can be calculated from the utility linked lists of t, LAS can remove unpromising candidate items in advance. As a result, the algorithm reduces the computation cost by exploring a smaller set of candidate items for concatenation of t.

Theorem 5 (Irrelevant Item Pruning Strategy, IPS). *Given a sequence t and an item $i_j \in I(t)_{rest}$, the maximal utility of $< t \oplus i_j >_{I-Concatenation}$ or $< t \oplus i_j >_{S-Concatenation}$ is no more than $\sum_{s \in D}\{MEU(t,s)| < t \oplus i_j >_{I-Concatenation} \subseteq s \vee < t \oplus i_j >_{S-Concatenation} \subseteq s\}$.*

Proof. For $< t \oplus i_j >_{I-Concatenation}$, based on theorem 4, it can be found that $\sum_{s \in D}\{MEU(t,s)| < t \oplus i_j >_{I-Concatenation} \subseteq s \vee < t \oplus i_j >_{S-Concatenation} \subseteq$

$s\} \geq \sum_{s \in D}\{MEU(t,s)| < t \oplus i_j >_{I-Concatenation} \subseteq s\} \geq u(< t \oplus i_j >_{I-Concatenation}$). For $< t \oplus i_j >_{S-Concatenation}$, it can be proven in the same way.

With the help of IPS, the remaining utility values of candidate sequences in each transaction decrease since many irrelevant items can be greatly pruned. As a result, the MEU values of candidate sequences can be greatly decreased; more candidates will be removed by the IPS.

4 Experimental Results

For the convenience, the baseline and the baseline algorithm with three pruning strategies are respectively named as **Baseline** and **Baseline_S** in the experiments. All the algorithms were implemented in Java and experiments were carried out on a personal computer equipped with an Intel Core2 i7-4790 CPU and 8GB of RAM, running the 64-bit Microsoft Windows 7 operating system. To automatically set different minimum utility thresholds for items, we then defined that: $mu(i_j) = max\{\beta \times u(i_j), LMU\}$, where LMU is a user-specified parameter for the least minimum utility threshold, and $u(i_j)$ is the utility of each item i_j in the database. The β is a constant factor to adjust the mu values of items. Four real-life [13] datasets were used in the experiments to evaluate the performance of our proposed approaches. More details of the used datasets can be found in [13].

4.1 Runtime

Fig. 2 shows the runtime of the compared approaches under a fixed β with various LMU. From Fig. 2, it can be seen that the Baseline_S algorithm outperforms the Baseline algorithm. The reason is that the Baseline_S algorithm has tighter upper-bound to prune the unpromising candidates early than that of the Baseline algorithm. It can be seen that the runtime of the proposed approaches increases along with the decrease of LMU. When LMU is set as a low value, the runtime of the Baseline algorithm sharply increases due to a very large candidates, such as in Fig. 2(b), Fig. 2(c) and Fig. 2(d). Notice that there are no results of the Baseline algorithm when LMU is set as a very low value, such as in Fig. 2(c) and Fig. 2(d). The reason is that the large dataset with some long sequences causes a very large number of candidates. Thus, the designed Baseline approach cannot return the results within a reasonable time.

4.2 Number of Candidates

The number of candidates and the number of actual discovered HUSPs of the compared approaches under a fixed β with various LMU is evaluated and shown in Fig. 3. From Fig. 3, it can be seen that the number of candidates of the Baseline_S algorithm is much less than that of the Baseline algorithm. This shows that the designed pruning strategies can greatly reduce the unpromising

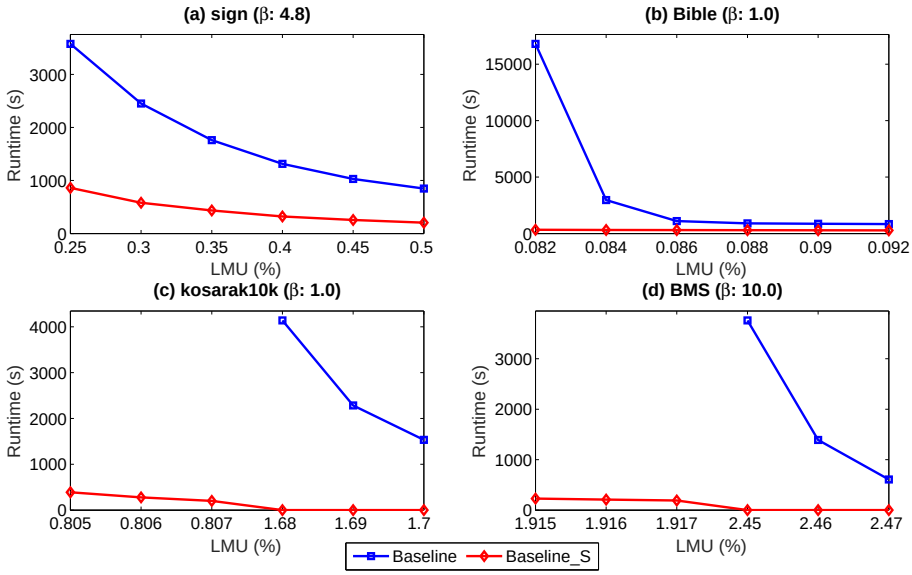

Fig. 2. Runtime under fixed β with various LMU

candidates for mining the HUSPs, and the requirements of runtime can be greatly improved. From the results, it also can be seen that the difference of the number of candidates is more obvious than that of the runtime. This is also reasonable since it takes computations to apply the pruning strategies for reducing the number of candidates.

When the LMU is set lower, it is obvious to see that the Baseline_S algorithm generates much less candidates than that of the Baseline algorithm. Especially, the Baseline algorithm has no results in Fig. 3(c) and Fig. 3(d). Notice that the number of candidates sharply decreases in Fig. 3(c). The reason is that there are a large number of candidates and patterns having similar utility values within a very small interval in this dataset.

5 Conclusion

In this paper, we propose a novel high-utility sequential pattern mining with multiple minimum utility thresholds framework for mining HUSPs. Based on the proposed framework, a baseline approach is first proposed. With the help of the designed LS-tree, UL-list structure, and the properties of HUSPM, the proposed algorithm can discover the complete set of HUSPs with multiple minimum utility thresholds. To improve the performance of the baseline algorithm, three pruning strategies are then introduced to lower the upper-bound value of the sequences and reduce the search space to find the HUSPs. Results are then evaluated to show the effectiveness and efficiency for mining HUSPs in terms of runtime and number of candidates.

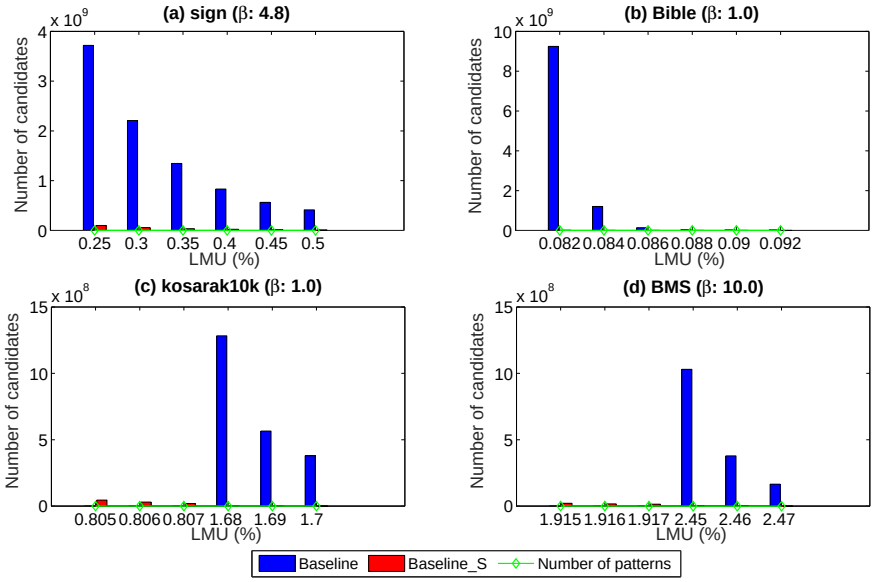

Fig. 3. Number of candidates under fixed β with various LMU

Acknowledgment

This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 6150309, and by the CCF-Tencent Project under grant No. IAGR20160115.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: The International Conference on Data Engineering. pp. 3–14 (1995)
2. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.: Freespan: frequent pattern-projected sequential pattern mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 355–359 (2000)
3. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Science and Pattern Recognition 1(1), 54–77 (2017)
4. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions Knowledge and Data Engineering 16(11), 1424–1440 (2004)
5. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. In: IEEE International Conference on Data Mining. pp. 19–26 (2003)
6. Liu, Y., Liao, W., Choudhary, A.N.: A two-phase algorithm for fast discovery of high utility itemsets. In: Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. pp. 689–695 (2005)
7. Yin, J., Zheng, Z., Cao, L., Song, Y., Wei, W.: Efficiently mining top-k high utility sequential patterns. In: IEEE International Conference on Data Mining. pp. 1259–1264 (2013)

8. Lan, G., Hong, T., Tseng, V.S., Wang, S.: Applying the maximum utility measure in high utility sequential pattern mining. Expert Systems with Applications 41(11), 5071–5081 (2014)
9. Alkan, O.K., Karagoz, P.: Crom and huspext: Improving efficiency of high utility sequential pattern extraction. IEEE Transactions Knowledge and Data Engineering 27(10), 2645–2657 (2015)
10. Wang, J., Huang, J., Chen, Y.: On efficiently mining high utility sequential patterns. Knowledge and Information Systems 49(2), 597–627 (2016)
11. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 337–341 (1999)
12. Yin, J., Zheng, Z., Cao, L.: Uspan: an efficient algorithm for mining high utility sequential patterns. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 660–668 (2012)
13. Fournier-Viger, P., Lin, J.C., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The SPMF open-source data mining library version 2. In: The European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 36–40 (2016)