



High average-utility sequential pattern mining based on uncertain databases

Jerry Chun-Wei Lin¹ · Ting Li² · Matin Pirouz³ · Ji Zhang⁴ · Philippe Fournier-Viger⁵

Received: 26 March 2018 / Accepted: 12 July 2019 / Published online: 22 July 2019
© Springer-Verlag London Ltd., part of Springer Nature 2019

Abstract

The emergence and proliferation of the internet of things (IoT) devices have resulted in the generation of big and uncertain data due to the varied accuracy and decay of sensors and their different sensitivity ranges. Since data uncertainty plays an important role in IoT data, mining the useful information from uncertain dataset has become an important issue in recent decades. Past works focus on mining the high sequential patterns from the uncertain database. However, the utility of a derived sequence increases along with the size of the sequence, which is an unfair measure to evaluate the utility of a sequence since any combination of a high-utility sequence will also be the high-utility sequence, even though the utility of a sequence is merely low. In this paper, we address the limitation of the previous potential high-utility sequential pattern mining and present a potentially high average-utility sequential pattern mining framework for discovering the set of potentially high average-utility sequential patterns (PHAUSPs) from the uncertain dataset by considering the size of a sequence, which can provide a fair measure of the patterns than the previous works. First, a baseline potentially high average-utility sequential pattern algorithm and three pruning strategies are introduced to completely mine the set of the desired PHAUSPs. To reduce the computational cost and accelerate the mining process, a projection algorithm called PHAUP is then designed, which leads to a reduction in the size of candidates of the desired patterns. Several experiments in terms of runtime, number of candidates, memory overhead, number of discovered pattern, and scalability are then evaluated on both real-life and artificial datasets, and the results showed that the proposed algorithm achieves promising performance, especially the PHAUP approach.

Keywords High average-utility sequential pattern mining · Sequential patterns · Uncertain database · Data mining

Abbreviations

ARM	Association rule mining
<i>auub</i>	Average-utility upper-bound value
AU list	Average-utility list
FIM	Frequent itemset mining
HAUIM	High average-utility itemset mining

HAUIs	High average-utility itemsets
HTWUIs	High transaction-weighted utilization itemsets
HUIM	High-utility itemset mining
HUIs	High-utility itemsets
HUSPs	High-utility sequential patterns
HUSPM	High-utility sequential pattern mining
PHAUSPM	Potential high average-utility sequential pattern mining
PHAUSPs	Potential high average-utility sequential patterns
PHAUB	The designed baseline algorithm
PHAUP	The designed projection-based algorithm
PHAUUBDC	Potential high average-utility upper-bound downward closure
PHAUUBSPs	Potential high average-utility upper-bound sequential patterns
PHUSPs	Potential high-utility sequential patterns
SPM	Sequential pattern mining
SWDC	Sequential weighted downward closure
<i>suub</i>	Sequence utility upper-bound value
TWU	Transaction-weighted utility
TWDC	Transaction-weighted down closure
UFIM	Frequent itemset mining on uncertain databases
UFIs	Frequent itemsets in uncertain databases
μ	Minimum expected support threshold
δ	Minimum high average-utility threshold

1 Introduction

With the proliferation of the internet of things (IoT) devices, such as sensors, mobile phone, RFID tags, and actuators, among others, a huge amount of data is collected per second. Depending on different domains and applications, several factors such as frequency, utility, weight, and interestingness are, respectively, considered to mine the required information for decision making. Traditional algorithms such as frequent pattern mining [2–4,14], sequential pattern mining [5,31,44], or high-utility itemset mining [21,25–27,34,38,39] are, respectively, presented to find the specific patterns in varied domains and applications. For example, frequent itemset mining or association rule mining shows the frequency relationship among the item/sets. However, the discovered knowledge only contributes to a small percentage of the total profit produced by the sale of all the items, while infrequent itemsets may actually make larger contributions toward the profit (e.g., breads can be sold much more than diamonds in a month but diamonds bring a much higher profit to the retailer than that of bread). Moreover, sequential pattern mining provides the complete set of frequent subsequences that reveals the relationships of the ordered events and elements. However, most existing works focus on finding the relationships of the sequential patterns in the binary database; thus, only the occurrence is considered in traditional sequential pattern mining. The high-utility itemset mining focuses on discovering the set of high-utility itemsets, which provides more information and insights for decision making. Considering both utility and sequential factors in transaction data, the issue of high-utility sequential pattern mining has been studied, and many existing works [6,7,17,40] focus on developing efficient algorithms to improve the mining performance and reduce the candidate size. However, the above algorithms only concern the binary dataset, where data is either present or absent in the transaction dataset.

Uncertainty plays an important role in IoT data [10,29] since the collected raw data from the IoT sensors or devices may be influenced by the environment, sensitivity, and their location. For example, suppose that several thermometers are set in a room to measure the temperature, and the degrees from those thermometers will not be the same due to some thermometers may be close to the window, and some of them may be close to the heater. Considering the basket market analysis, an itemset with high utility and high existence probability (uncertainty) is useful to make efficient and effective decisions. For example, a customer buys milk in 100 quantities, and the profit for a milk is 2 dollars with a probability value as 0.91; he/she also buys a PC, and the profit for a PC is 100 dollars with the probability value as 0.05. Based on this situation, a retailer may introduce a large amount of milk into shops and reduce the number of PC. The reason is that although the profit of milk is much smaller than that of a PC, but the probability to purchase milk is much higher than that of a PC. Several existing works consider uncertainty as an important factor into pattern mining works, such as uncertain frequent itemset mining [11,19,36], uncertain sequential pattern mining [13], or high-utility itemset mining over uncertain dataset [18,24]. Moreover, the high-utility sequential pattern mining [42] concerns the IoT data and present the potential high-utility sequential pattern mining framework to derive both the high-utility and high-probability patterns from the sequential and uncertain dataset. Yet, it is observed that the utility of an item or set increases along with its size, i.e., the length of the itemset. Such results may result in improper decisions or ill-designed sales strategies. To mitigate this problem, Hong et al. [15] presented a level-wise average-utility algorithm to find the set of high average-utility itemsets. The set of high average-utility itemsets is discovered by dividing the total utility of an itemset in a transaction by its length (that is, the number of items in the transaction). This average concept provides a fair measure to evaluate the importance of the derived patterns. Several extensions [16,20,22, 23,28] of high average-utility itemset mining were presented. The above works fail, however, to consider the intrinsic sequential item ordering in the transactions. Thus, the motivation of this paper aims at the following goals.

- Consider the uncertainty, especially the IoT data, with the high-utility sequential pattern mining to retrieve the information for decision making.
- Consider the length of patterns and provide a fair measure of the discovered patterns, thus the average-utility concept is adopted here to solve the limitations of increasing utility along with the pattern size.
- Develop efficient algorithms and pruning strategies to speed up computation and reduce the number of candidates for discovering the desired patterns.

Based on the above motivations, the potentially high average-utility sequential pattern mining (PHAUSPM) framework is presented here to consider the uncertainty, utility, and sequence ordering of the data, and average-utility concept to mine more meaningful and useful patterns for decision making. As we know, the utility of itemset increases with its size or length. Considering the total utility of the itemsets irrespective of their size may lead to an unfair evaluation regarding whether an itemset is a profitable pattern or not. In addition, data uncertainty should also be considered as the data collected from real-life applications, such as sensor networks or experimental environments, are not (very) precise. For instance, the sensors deployed in a wireless sensor network collect the data with a certain probability from the real-world environment that measures, for example, temperature or humidity. Sequence ordering also plays a critical role for decision making especially when analyzing user's purchase behaviors for recommender systems. Given the impact of the above factors, the scientific contributions of the proposed approaches in this paper are organized as follow.

- An innovative high average-utility sequential pattern mining framework is first proposed to discover the potentially high average-utility sequential patterns (PHAUSPs) from uncertain datasets. The average-utility of a sequence is used as a measure to evaluate its usefulness or importance, which shows a fair measure than that of the traditional utility approach.
- A baseline PHAUB algorithm is proposed to discover the set of PHAUSPs in a level-wise manner. Three pruning strategies are also designed to reduce the number of unpromising candidates in the mining process, which can also reduce the computational cost.
- A projection PHAUP algorithm is further developed to accelerate the mining speed by reducing the size of the processed sequence dataset. Thus, the size of the projected database decreases with the increase of the determined patterns.
- We have conducted comprehensive experiments on both real-life and artificial datasets to demonstrate the promising efficiency and effectiveness performance of the proposed algorithms under varying threshold settings (including different minimum high average-utility thresholds and minimum expected supports) in terms of their execution time, the number of candidates generated, the required space overhead, the number of discovered patterns, and scalability with regard to large datasets.

2 Related work

Frequent itemset mining (FIM) or association rule mining (ARM) considers the frequency of the itemsets, which can only detect interesting patterns in the binary database. High-utility itemset mining (HUIM) was studied [26,38,39] to consider both the quantity and unit profit of the items while mining the high-utility itemsets (HUIs). Because of the major challenges in searching, the transaction-weighted utility (TWU) model [26] was proposed to mine HUIs based on the generate-and-test mechanism. The TWU algorithm holds the transaction-weighted downward closure (TWDC) property, which is used to maintain the designed high transaction-weighted utilization itemsets (HTWUIs) to reduce the search space when detecting HUIs. Several methods [21,27,34] were, respectively, proposed to mine the interesting patterns based on the TWU model.

A major limitation of the existing HUIM lies in the fact that it only quantifies patterns using their utilities, which is often unfair for short itemsets. To mine better patterns after their lengths is considered, high average-utility itemset mining (HAUIM) [15,20,21,28] was presented. It offers a more fair measure called the average utility to evaluate the interestingness of the discovered patterns. An itemset is deemed to be a high average-utility itemset (HAUI) if the division of its utility by its size exceeds a user-specified minimum average-utility count. The two-phase TPAU algorithm [15] is the first algorithm of the HAUIM, which presents an upper bound for the average utility (*auub*) to conduct an estimation of pattern utilities within the search space. The *auub* model is able to effectively reduce the size of search space while ensuring that the discovered patterns in HAUIM are complete and correct. As the TPAU algorithm is a hierarchical method, it has similar limitations as the TWU algorithm does. To improve the performance of mining HAUIs, the projection-based PAI algorithm [16] was proposed which leverages an innovative pruning technique. The HAUP-tree structure and a mining algorithm called HAUP-growth [20] were proposed to avoid multiple database scans. Each node in a HAUP-tree keeps the purchase quantities of its prefix items; therefore, the HAUIs can be directly obtained from it. Despite its efficiency, this method incurs high memory overhead especially when the size of the transaction is very long. The HAUI-tree approach

was proposed [28] to mine HAUIs using an indexing table structure. Thanks to this method, the total number of candidates involved in mining HAUIs can be significantly reduced. In addition, a new structure was proposed to accelerate the calculation of the values of itemsets and minimize space overhead. An innovative, more efficient HAUI-Miner method [22] was proposed with two pruning strategies to mine HAUIs based on a compressed average-utility (AU) list structure. The AU list structure was created to store useful and relevant information for the purpose of HAUI mining; thus, the mining performance can be greatly increased. Lin et al. [23] then developed three pruning strategies to speed up the mining process of the HAUIs. Thus, a tighter upper bound is then provided instead of the traditional *auub* model, and the search space of the candidates can be greatly reduced. However, the above approaches do not consider the realistic problems of the dataset such that the data may consist of the uncertainty factor, especially for the data collected by the mobile devices. Thus, the mined patterns somehow could not make a precise decision since all items/products are then treated as the same importance with binary representation. Also, the sequence ordering of the items/products is not considered, which shows better purchase behaviors of the customers.

High-utility sequential pattern mining (HUSPM) [6,7,17,40] was introduced to consider sequential utilities and orderings of items in the database, which is used to discover the set of high-utility sequential patterns (HUSPs). It combines the ideas of high-utility itemset mining (HUIM) [21,26,27,34,38,39], as well as the sequential pattern mining (SPM) [5,31]. Ahmed et al. [6] first introduced two algorithms named UL and US to mine HUSPs; the former used the level-wise approach to mine HUSPs and the latter was based on a pattern-growth method to discover the HUSPs. Yin et al. [40] presented an efficient method to discover HUSPs. A structure named lexicographic quantitative sequence tree (LQS-tree) was proposed to discover the set of HUSPs. The necessary information of each node in LQS-tree was stored in a matrix, which is used to avoid multiple unnecessary database scans. To achieve a better space and speed performance, two heuristic pruning strategies based on the property known as sequential weighted downward closure (SWDC) were proposed. Lan et al. [17] presented the HUSPM algorithm using the projection mechanism and designed the maximum utility measure and the sequence-utility upper-bound (*suub*) model for efficiently mining the HUSPs. An innovative indexing technique and the sequence utility table, which has the actual utilities and upper bound of candidate utilities, were leveraged to improve the mining performance. Alkan et al. [7] proposed a HuspExt method using the idea of cumulate rest of match (CRoM) to implement a tighter upper-bound for candidate utilities. Mining top-*k* high-utility sequential patterns [35] was proposed, making it possible to extract the required information without the use of the minimum threshold. A projection-based TKHUS-Span [41] was designed to mine top-*k* HUSPs with two tight utility upper bounds to mine HUSPs.

Although the aforementioned methods consider the utility and sequence ordering of the discovered patterns, they can only process the precise data. However, many of the real-life datasets are uncertain. Frequent itemset mining on uncertain databases (UFIM) was presented by considering two different models, namely the expected support [11] and the probabilistic frequentness [8]. Chui et al. [11] first proposed the support model and the UApriori algorithm to discover the frequent itemsets in uncertain databases (UFIs). An itemset is deemed as the UFI if its expected support is no less than the pre-defined minimum support threshold. The UApriori algorithm discovers the UFIs using a level-wise Apriori-like algorithm. Wang et al. [36] then developed a model-based approach, which can be used to reduce the size of the discovered candidates while comparing to the UApriori algorithm. Leung et al. [19] then proposed a pattern-growth UFP-growth algorithm that is based on the UF-tree structure extended from the well-known frequent pattern (FP)-tree [14]. Aggarwal et al. [1] later introduced the UH-mine algorithm [1], which applied the depth-first search and divide-

and-conquer strategies to accelerate the mining process. Muzamml et al. [30] presented three algorithms named BFS, DFS, and PGA under the expected support measure to mine frequent sequential patterns from uncertain databases. The BFS and DFS algorithms use a generate-and-test approach, while the PGA algorithm used a pattern-growth procedure to mine UFSPs. Several extensions consider to mine the high-utility patterns from the uncertain databases [9,37]. Zhang et al. [42] first considers the high-utility sequential pattern mining from the uncertain database based on the expected support model. However, their algorithm is still based on the TWU model, thus the utility of the sequence can be dramatically increased along with the sequence size.

Considering the world semantics, Berbecker et al. [8] proposed the probabilistic frequentness model. An itemset is treated as a frequent itemset in the probabilistic frequent model if its frequentness probability exceeds a specified minimum probability threshold. Sun et al. [32] presented two p-Apriori and TODIS [32] algorithms to discover UFIs based on top-down/bottom-up manners with a designed probabilistic frequent pattern (p-FP) structure. Tong et al. [33] then compared various algorithms based on the expected support/probabilistic frequentness models. Zhao et al. [35] introduced two models for handling the problem of uncertain sequential pattern mining using the U-PrefixSpan algorithm. It is the extensions of the PrefixSpan [31] for revealing the set of high frequent itemsets in the uncertain databases.

In summary, we can observe that some works consider the utility and sequence ordering to find the meaningful and useful high-utility sequential patterns; however, only one work [42] addresses the problem of high-utility sequential pattern mining in the uncertain database. Yet, the utility of the discovered patterns may increase along with the size of the pattern size, which is an unfair measure to evaluate the discovered patterns. High average-utility itemset mining can thus be used to solve such limitations but yet, none of the existing works aim at combining those important factors together to provide a fair measure for detection of realistic and meaningful patterns.

3 Preliminaries and problem statement

Suppose that the m identical items are contained in a set of I such as $I = \{i_1, i_2, \dots, i_m\}$, and an uncertain sequential database is denoted as D containing n sequences such as $D = \{S_1, S_2, \dots, S_n\}$. Each sequence has a relationship such that $S_q \in D$ ($1 \leq q \leq n$), in which q is a unique number in the database and called as sequential identifier (SID). For each sequence S_q in the database, it contains a number of itemsets as $\{I_1, I_2, \dots, I_k\}$ and each I_r is a set of distinct items with their purchase quantities as $q(i_j, I_r, S_q)$ and unique probabilities $p(i_j, I_r, S_q)$. Moreover, an unit profit table is called *utable*, and it has the relationship such that $utable = \{p(i_1), p(i_2), \dots, p(i_m)\}$. Two user-defined thresholds, respectively, named minimum expected support threshold ($= \mu$) and the minimum high average-utility threshold ($= \delta$) are then pre-defined to discover the set of HAUSPs, which is the high average-utility sequential patterns.

Table 1 presents a simple uncertain sequential database. There are four sequences and six distinct items in the database, such as (a), (b), (c), (d), (e), and (f). The corresponding *utable* is shown as $utable = \{a:2, b:1, c:3, d:4, e:1, f:2\}$. In this example, two thresholds μ ($= 35\%$) and δ ($= 9.23\%$) are, respectively, set by users' preference.

Definition 1 A sequence contains k items is defined as k -sequence.

For example, the S_1 is a 5-sequence in the running example because 5 items are contained in S_1 . The sequence $\langle (a), (b) \rangle$ is a 2-sequence because it has 2 items.

Table 1 An uncertain sequential database

SID	Sequence (item, quantity)	Probability
S_1	$\langle (a, 3), (b, 4), [(a, 1), (c, 1), (e, 2)] \rangle$	0.6
S_2	$\langle [(b, 1), (c, 1)], [(a, 1), (d, 1)], [(a, 2), (b, 2)] \rangle$	0.8
S_3	$\langle (f, 1), [(f, 1), (d, 1)], [(b, 4), (c, 1)] \rangle$	0.5
S_4	$\langle (b, 1), (c, 1), [(a, 1), (b, 2)], [(a, 4), (b, 1)] \rangle$	0.9

Definition 2 A sequence $S_x = \langle I_{x1}, I_{x2}, \dots, I_{xn} \rangle$ is a subsequence of $S_y = \langle I_{y1}, I_{y2}, \dots, I_{ym} \rangle$, denoted as $S_x \subseteq S_y$.

For example, $\langle (b), (a) \rangle$ is a subsequence of S_4 in Table 1 since $(a) \subseteq (a, b)$.

Definition 3 The internal utility of an item i_j is denoted as $iu(i_j, I_k, S_q)$, we also can have a relationship such that $i_j \in I_k \in S_q$ and can be defined as:

$$iu(i_j, I_k, S_q) = q(i_j, I_k, S_q) \times p(i_j), \tag{1}$$

where $p(i_j)$ is the unit profit of the item i_j in the database.

For example, S_1 can be defined as three itemsets such that $\langle I_1, I_2, I_3 \rangle$, and $I_1 = [(a, 3)]$, $I_2 = [(b, 4)]$, and $I_3 = [(a, 1), (c, 1), (e, 2)]$. The internal utilities of an item (a) in S_1 can be calculated as $iu(a, I_1, S_1) (= 6)$ and $iu(a, I_3, S_1) (= 2)$, respectively.

Definition 4 The maximum utility of the items within the sequence is concerned as the utility of the item within the sequence, which is denoted as $imu(i_j, S_q)$ and defined as:

$$imu(i_j, S_q) = \max\{iu(i_j, I_k, S_q) | i_j \in I_k \wedge I_k \in S_q\}. \tag{2}$$

For the item (a) in S_1 , $iu(a, I_1, S_1) = 6$ and $iu(a, I_3, S_1) = 2$; thus, the $imu(a, S_1) = \max\{iu(a, I_1, S_1), iu(a, I_3, S_1)\} = \max\{6, 2\} = 6$.

Definition 5 The average-utility of an item i_j in a sequence S_q is denoted as $iau(i_j, S_q)$, and defined as:

$$iau(i_j, S_q) = \frac{imu(i_j, S_q)}{1}. \tag{3}$$

For the item (a) in S_1 , the $iau(a, S_1) = \frac{6}{1} = 6$.

Definition 6 The utility for a subsequence s in S_q is denoted as $su(s, S_q)$, which is to obtain the maximum utility of all combinational results of the items in S_q .

For example of sequence S_4 , a sequence $s = \langle (b), (a) \rangle$ is its one of the subsequences; two combinations of s appeared in S_4 . The utilities of the two combinations are calculated as $(1 \times 1) + (1 \times 2) = 3$ and $(1 \times 1) + (4 \times 2) = 9$. Thus, the utility of S in S_4 can thus be obtained as $\max\{3, 9\} = 9$.

Definition 7 The average-utility of a subsequence s in a sequence S_q is defined as the utility of s in S_q divided by the length of s , which is denoted as $sau(s, S_q)$.

For example, the average-utility of the subsequence $s = \langle (b), (a) \rangle$ in S_4 is calculated as $\frac{9}{2} = 4.5$.

Definition 8 The probability of a sequence s within a sequence S_q is denoted as $sp(S_q, s)$ ($s \subseteq S_q$), such that $sp(S_q, s) = sp(S_q)$.

For example, because $\langle(a)\rangle \subseteq S_1$, $\langle(b), (a)\rangle \subseteq S_4$ and $sp(S_1) = 0.6$, $sp(S_4) (= 0.9)$, $sp(\langle(a)\rangle, S_1) (= 0.6)$, and $sp(\langle(b), (a)\rangle, S_1) (= 0.9)$.

Definition 9 The probability and average-utility of a sequence s in the uncertain sequential database can be, respectively, defined as:

$$sau(s) = \sum_{s \subseteq S_q \wedge S_q \in D} sau(s, S_q), \tag{4}$$

$$sp(s) = \sum_{s \subseteq S_q \wedge S_q \in D} sp(s, S_q) = \sum_{s \subseteq S_q \wedge S_q \in D} sp(S_q). \tag{5}$$

For example, two sequences S_1, S_2 and S_4 contain a sequence $s = \langle(b), (a)\rangle$. We also can have that $sp(S_1, s) = sp(S_1) (= 0.6)$, $sp(S_2, s) = sp(S_2) (= 0.8)$, $sp(S_4, s) = sp(S_4) (= 0.9)$ and $sau(S_1, s) (= 10)$, $sau(S_2, s) (= 4)$, $sau(S_4, s) (= 3)$.

Definition 10 The utility of a sequence S_q in an uncertain sequential database D is denoted as $SU(S_q)$, which can be defined as:

$$SU(S_q) = \sum_{i_j \in I_k \wedge I_k \in S} su(i_j, I_k, S_q) \tag{6}$$

For example, the utility of S_1 is calculated as: $SU(S_1) (= 6 + 4 + 2 + 3 + 2) (= 17)$.

Definition 11 The total utility is denoted as TSU in an uncertain sequential database D , which is to sum up the utilities of all sequences in D and defined as:

$$TSU = \sum_{S_k \in D} SU(S_k). \tag{7}$$

For example, the total utility of Table 1 is calculated as $TSU (= 17 + 16 + 15 + 17) (= 65)$, where $SU(S_1) (= 17)$, $SU(S_2) (= 16)$, $SU(S_3) (= 15)$, and $SU(S_4) (= 65)$.

Problem statement: The problem of potentially high average-utility sequential patterns mining formulated as the discovery of the complete set of potentially high average-utility sequential patterns (PHAUSPs) from uncertain sequential databases. In an uncertain sequential database, a sequence s is considered as a PHAUSP if it satisfies two conditions as follows:

$$\text{PHAUSPs} \leftarrow \{s \mid sp(s) \geq |D| \times \mu \wedge sau(s) \geq TSU \times \delta\}. \tag{8}$$

Therefore, the discovered PHAUSPs demonstrate the average-utility value of a sequential pattern with the probability. For example, suppose that a milk is purchased first then a caviar is purchased afterward. Originally, the profit of a milk could be 2 dollars and the profit of a caviar is about 1000 dollars. Thus, the average-utility of this sequential pattern such as “*buying a milk is followed by a caviar purchase*” is $(2 + 1000)/2 = 501$ dollars. This purchase behavior can have a purchase probability with 0.05. Another average-utility of a sequential pattern such as “*buying a PC is followed by purchasing a notebook*” can be calculated as $(100 + 300)/2 = 200$ dollars. This purchase behavior can have a purchase probability of 0.9. In this situation, the retailer may consider to have more amounts of the latter pattern than the first one. The reason is that the latter one brought lower average-utility value to the retailer, but it has a higher purchase probability than the first sequential pattern.

Table 2 The discovered PHAUSPs from the running example

Sequence	Probability	Average-utility	Sequence	Probability	Average-utility
$\langle(a)\rangle$	2.3	9	$\langle(b), (a)\rangle$	2.3	11
$\langle(b)\rangle$	2.8	6	$\langle(a), (a, b)\rangle$	1.7	6
$\langle(c)\rangle$	2.8	6	$\langle(b), (a, b)\rangle$	1.7	6
$\langle(a), (a)\rangle$	2.3	12	$\langle(b), (a), (a)\rangle$	1.7	6
$\langle(a), (b)\rangle$	2.3	8	$\langle(c), (a, b)\rangle$	1.7	7
$\langle(a, b)\rangle$	1.7	7	$\langle(c), (a), (a)\rangle$	1.7	7
$\langle(c), (a)\rangle$	1.7	9	$\langle(c), (a), (a, b)\rangle$	1.7	6

From the example given in Table 1, the resulting set of PHAUSPs is given in Table 2 under the minimum expected support threshold μ being 35% and the minimum high average-utility threshold δ being 9.23%.

To reduce the search space, the downward closure property should be maintained since the correctness and completeness of the discovered PHAUSPs should be held. Thus, it is necessary to over-estimate an upper-bound value of the PHAUSP. Definitions for maintaining the downward closure property are given as follows.

Definition 12 The sequence-maximum utility of a sequence S_q is denoted as $smu(S_q)$, which is the maximum utility of items in a sequence S_q , and defined as:

$$smu(S_q) = \max\{u(i_j, S_q) | i_j \in S_q\}. \tag{9}$$

For instance, the sequence-maximum utility of S_1 is quantified as $\max\{6, 4, 2, 3, 2\} = 6$.

Definition 13 The sequence average-utility upper-bound of a sequence s in D is denoted as $suub(s)$, which is the sum of the sequence-maximum utility of all sequences containing s , and defined as:

$$suub(s) = \sum_{s \subseteq S_q \wedge S_q \in D} smu(S_q). \tag{10}$$

For example, the $suub$ of the sequence $s = \langle(b), (a)\rangle$ is calculated as $(6 + 4 + 8) = 18$ as s appears in S_1, S_2 and S_4 and we have $smu(S_1) = 6, smu(S_2) = 4, smu(S_4) = 8$.

4 Proposed potentially high average-utility sequential pattern mining framework

In the potentially high average-utility sequential pattern mining (PHAUSPM) framework, we, respectively, design (1) a high probability average-utility upper-bound sequential pattern (PHAUBSP) to hold the downward closure property for level-wise mining process; (2) a novel pattern called potential high average-utility sequential pattern (PHAUSP) is designed to consider utility, uncertainty, sequence ordering and average concept to provide a fair measurement; (3) a baseline algorithm called PHAUB to discover the potentially high average-utility sequential patterns in a level-wise manner; (4) three pruning strategies are developed to reduce the size of the candidate and speed up computation in mining progress; (5) an improved PHAUP algorithm is developed to speed up the PHAUB algorithm using the projection method to reduce the processed database size. To handle the sequence data in the PHAUSPM framework, the concepts of the sequence are first introduced and discussed below.

Definition 14 (Matching sequence). Given two sequences $S_x = \langle I_1, I_2, \dots, I_n \rangle$ and $S_y = \langle I'_1, I'_2, \dots, I'_m \rangle$ and items in the sequences are sorted in lexicographic order, S_x matches S_y if the $(S_x - \text{first.item}.I_1) = (S_y - \text{last.item}.I'_m)$.

For example, $\langle (a), (b), (a, c) \rangle$ matches two sequences $\langle (b), (a, c), (e) \rangle$ and $\langle (b), (a, c, e) \rangle$ but does not match $\langle (a), (b), (a, e) \rangle$.

Thus, based on the matching sequence, we can discover the frequency of the specific patterns in the sequence dataset. Two ways can be performed for generating the sequence patterns, which are the S-Concatenation and I-Concatenation. Details are given below.

Definition 15 (S-Concatenation). Given two k -sequences S_x and S_y and S_x matches S_y . If the final $I'_m \in S_y$ has one item within it only, a sequence S_{xy} is then formed by appending I'_m into S_x , which is defined as S_x joins with S_y using **S-Concatenation**.

For example, $\langle (a), (b), (a, c) \rangle$ matches the sequences $\langle (b), (a, c), (e) \rangle$ and a sequence $\langle (a), (b), (a, c), (e) \rangle$ obtains the **S-Concatenation** relationship with the above sequences.

Definition 16 (I-Concatenation). Given two k -sequences S_x and S_y and S_x matches S_y . If the final $I'_m \in S_y$ has more than one item within it, a sequence S_{xy} is obtained by appending the last item in I'_m into the last position in S_x , which is defined as S_x joins with S_y using **I-Concatenation**.

For example, $\langle (a), (b), (a, c) \rangle$ matches the sequences $\langle (b), (a, c, e) \rangle$ and a sequence $\langle (a), (b), (a, c, e) \rangle$ has **I-Concatenation** relationship of those two sequences.

Thus, given a set of $(k-1)$ -sequences, the k -sequences can be formed using **I-Concatenation** and **S-Concatenation**. For example, if k is equal to 2, which is used to generate the 2-sequence. Thus, each 1-sequence will be concatenated with itself by **S-Concatenation**, and each 1-sequence joins all 1-sequences located after it by both **I-Concatenation** and **S-Concatenation** (this process may be based on the sorting order of the sequence). In addition, each sequence joins all 1-sequences located (in a sorted order) ahead of it using **S-Concatenation**.

For instance, the 1-sequences are, respectively, defined as $\langle (a) \rangle$, $\langle (b) \rangle$ and $\langle (c) \rangle$. Take $\langle (b) \rangle$ as an example to illustrate the steps. For the $\langle (b) \rangle$, it generates four sequences such as $\langle (b), (b) \rangle$, $\langle (b, c) \rangle$, $\langle (b), (c) \rangle$, and $\langle (b), (a) \rangle$. For each k -sequence ($k \geq 2$) S_x in the set, assume that S_y is a sequence located after or before S_x in the set, if S_x matches S_y and the last element in S_y has only one item, S_x joins S_y by **S-Concatenation**. If S_x matches S_y and the last element in S_y has more than one item, S_x joins S_y using **I-Concatenation**. For instance, given a set of 2-sequences such that $\langle (a, b) \rangle$, $\langle (a), (b) \rangle$, $\langle (b, c) \rangle$, $\langle (b), (c) \rangle$. For the 2-sequence $\langle (a), (b) \rangle$, it can generate the sequences such that $\langle (a), (b, c) \rangle$, $\langle (a), (b), (c) \rangle$.

Based on the above definitions, we can obtain the lemma as follows to define and prove the upper-bound value of a sequence pattern in a uncertain dataset.

Lemma 1 *The suub of a sequence s in an uncertain sequential database is defined as $suub(s)$, and any of its supersets has less suub values than itself.*

Proof Let S^{k-1} be one of the subsets of S^k , in which S^k is defined as a k -sequence. We can obtain that $S^{k-1} \subseteq S^k$; the sequence IDs of S^{k-1} is the subset of S^k , thus:

$$\begin{aligned}
 suub(S^k) &= \sum_{S^k \subseteq S_q \wedge S_q \in D} smu(S_q) \leq \sum_{S^{k-1} \subseteq S_q \wedge S_q \in D} smu(S_q) = suub(S^{k-1}). \\
 &\Rightarrow suub(S^k) \leq suub(S^{k-1}).
 \end{aligned}$$

□

To obtain the uncertainty factor of the sequence pattern, the following lemma is given to ensure that the desired pattern holds downward closure property in the mining progress.

Lemma 2 *The probability sp of a sequence s in an uncertain sequential database is defined as $sp(s)$, and any of its subsets have larger or equal sp values than itself.*

Proof Let S^{k-1} be one of the subsets of S^k , in which S^k is defined as a k -sequence. We can obtain that $S^{k-1} \subseteq S^k$, the sequence IDs of S^{k-1} is the subset of S^k , thus:

$$\begin{aligned}
 sp(S^k) &= \sum_{S^k \subseteq S_q \wedge S_q \in D} sp(S_q) \leq \sum_{S^{k-1} \subseteq S_q \wedge S_q \in D} sp(S_q) = sp(S^{k-1}). \\
 \Rightarrow sp(S^k) &\leq sp(S^{k-1}).
 \end{aligned}$$

□

Based on the above lemmas, a theorem can be derived to ensure that the discovered patterns possess the completeness and correctness properties.

Theorem 1 (High-probability DC property, HPDC property) *If a sequential pattern is a high probability sequential pattern in an uncertain sequential database, the HPDC property holds for any of the high probability sequential pattern (HPSP).*

Proof Let S^{k-1} be one of the subsets of S^k , in which S^k is defined as a k -sequence. Since $sp(S_q, S^k) = sp(S_q)$. For any sequence S_q in an uncertain sequential database D , $sp(S_q, S^k) = sp(S_q, S^{k-1})$. Since S^{k-1} is a subset of S^k , the set of *SIDS* of S^{k-1} is as the subset of S^k , according to Lemma 1, we can have that $sp(S^k) \leq sp(S^{k-1})$. Thus, if S^k is a HPSP, its probability is no less than the minimum expected support count, that is $sp(S^k) \geq |D| \times \mu$; $sp(S^{k-1}) \geq |D| \times \mu$. □

Corollary 1 *Let S^k sequence be a HPSP; thus, any of its subsets S^{k-1} belong to HPSP.*

Corollary 2 *Let S^k sequence not be a HPSP; thus, non-superset S^{k+1} will be a HPSP.*

To ensure that the designed algorithms hold the completeness and correctness properties of the discovered patterns, an upper-bound value of the pattern is defined below. Since this upper-bound value holds the downward closure property, the required patterns can be correctly and completely mined. Definition of the upper-bound pattern is presented below.

Definition 17 A sequence s in an uncertain sequential database is defined as a high probability average-utility upper-bound sequential pattern (PHAUUBSP) if 1) $sp(s) \geq |D| \times \mu$, and 2) $suub(s) \geq TSU \times \delta$.

For example, user defined the minimum expected support as μ ($= 35\%$); we can have the minimum expected count as $\delta = (4 \times 35\%)(= 1.4)$. The minimum average-utility threshold is set as 9.23% , and we can have the minimum average-utility count as $\delta = (65 \times 9.23\%)(= 6)$. A sequence $\langle (b), (a) \rangle$ is called as a PHAUUBSP because its $suub(\langle (b), (a) \rangle) = smu(S_1) + smu(S_2) + smu(S_4) = 6 + 4 + 8 = 18 > 6$, and its $sp(\langle (b), (a) \rangle) = (0.6 + 0.8 + 0.9)(= 2.3 > 1.4)$.

Based on the above definitions, we can prove that the designed algorithm holds the potentially high average-utility upper-bound downward closure (PHAUUBDC) property as follows.

Theorem 2 (Potentially high average-utility upper-bound downward closure property, PHAUUBDC property). *Let S^{k-1} be one of the subsets of S^k , in which S^k is defined as a k -sequence. If S^{k-1} is a PHAUUBSP, based on the PHAUUBDC property, we can have that 1) $suub(S^{k-1}) \geq suub(S^k)$ and 2) $sp(S^{k-1}) \geq sp(S^k)$.*

Proof Since $S^{k-1} \subseteq S^k$, the sequence IDs of S^{k-1} is as subset of S^k . Thus:

$$suub(S^k) = \sum_{S^k \subseteq S_q \wedge S_q \in D} smu(S_q) \leq \sum_{S^{k-1} \subseteq S_q \wedge S_q \in D} smu(S_q) = suub(S^{k-1}).$$

$$\Rightarrow suub(S^k) \leq suub(S^{k-1}).$$

□

Thus, we can conclude that the actual PHAUSP is the subset of the PHAUUBSP since the PHAUUBSP holds the upper-bound value of the PHAUSP. We then can derive the following theorem.

Theorem 3 ($PHAUSP \subseteq PHAUUBSP$). *The PHAUUBDC property of PHAUUBSP ensures that $PHAUSP \subseteq PHAUUBSP$ s. It suggests that if a sequential pattern is not a PHAUUBSP, none of its supersets is either PHAUUBSP or PHAUSP.*

Proof $\forall S^{k-1} \in D$, and S^{k-1} is defined as a $(k-1)$ -sequence, we can obtain:

1) According to Theorem 2, if S^{k-1} is not a PHAUUBSP, none of its supersets S^k is PHAUUBSP. Thus, $suub(S^k) \leq suub(S^{k-1}) < TSU \times \delta$ and $sp(S^k) \leq sp(S^{k-1}) < |D| \times \mu$.
 2)

$$sau(S^{k-1}) = \sum_{S^{k-1} \subseteq S_q \wedge S_q \in D} su(S_q, S^{k-1}) \leq \sum_{S^{k-1} \subseteq S_q \wedge S_q \in D} smu(S_q) = suub(S^{k-1}).$$

$$\Rightarrow sau(S^{k-1}) \leq suub(S^{k-1}).$$

□

Thus, if S^{k-1} is not a PHAUUBSP, it suggests that S^{k-1} does not satisfy the following two conditions: $sp(S^{k-1}) \geq |D| \times \mu$ and $suub(S^{k-1}) \geq TSU \times \delta$; S^{k-1} would not be a PHAUSP. Thus, any of its supersets S^k will not be either PHAUUBSP or PHAUSP.

4.1 Three pruning strategies

To accelerate the mining performance, an average-utility correlated structure (AUCS) is designed to obtain the $suub$ value of 2-itemsets. Since the $suub$ value is the upper-bound value of the sequential pattern, the AUCS is therefore used to remove those unpromising candidates of 2-itemsets in advance.

Lemma 3 (Average-utility correlated structure, AUCS) *If the $suub$ of 2-sequence is not larger than the minimum average-utility sequential count, any superset of this 2-sequence will not be either a PHAUUBSP or a PHAUSP.*

Proof Let S^2 be a 2-sequence, and S^k be a k -sequence ($k \geq 3$), which is a superset of S^2 . Since $suub(S^k) \leq suub(S^{k-1})$ and $PHAUSP \subseteq PHAUUBSP$ holds; if a 2-sequence S^2 has $suub(S^2) < TSU \times \mu$, S^2 is not a PHAUUBSP and any supersets of S^2 w.r.t. a k -sequence ($k \geq 3$) is neither a PHAUUBSP nor PHAUSP. □

Table 3 The built AUCS of the running example

Sequence	suub	Sequence	suub
$\langle(a, a)\rangle$	18	$\langle(b, c)\rangle$	8
$\langle(a, b)\rangle$	12	$\langle(b, c)\rangle$	14
$\langle(a, b)\rangle$	18	$\langle(c, c)\rangle$	0
$\langle(a, c)\rangle$	6	$\langle(c, b)\rangle$	12
$\langle(a, c)\rangle$	6	$\langle(c, a)\rangle$	12
$\langle(b, b)\rangle$	12	$\langle(b, a)\rangle$	18

From the given example, the constructed AUCS are given in Table 3.

According to the above lemmas and theorems, three pruning strategies can be designed to speed up the mining performance of the PHAUSPs by reducing of the search space for the candidates.

Strategy 1 *If the probability and the suub of a sequence S^k do not follow the two conditions of: 1) $sp(S^k) \geq |D| \times \mu$ and 2) $suub(S^k) \geq TSU \times \delta$, this sequence is ignored as none of its supersets is a PHAUSP.*

Rationale 1 *According to Theorems 2 and 3, this strategy can be correctly held.*

Strategy 2 *Let S^k be a k -sequence, and if any $(k-1)$ -subsequence of the S^k is not a HPSP, S^k and all its supersets cannot be PHAUSP either.*

Rationale 2 *This pruning strategy holds according to Theorem 1.*

Strategy 3 *Let S^k be a k -sequence ($k \geq 3$). If the suub of a 2-sequence $s \subseteq S^k$ such that $suub(s)$ is minimum high average-utility sequential count based on the built AUCS, S^k is not a PHAUUBSP and will not be a PHAUSP either. Moreover, none of its superset is a PHAUSP.*

Rationale 3 *Based on Lemmas 1 and 3, this pruning strategy can be correctly obtained.*

4.2 Revised database

In the proposed algorithm framework, it is required that the database be revised to keep the set of 1-PHAUUBSPs for future mining. Specifically, the original database is revised by pruning the unpromising candidates if they do not meet the pre-defined thresholds (μ and δ). The pseudo-code to obtain the satisfied 1-PHAUUBSPs is presented in Algorithm 1.

The input of Algorithm 1 includes: (1) an uncertain database is defined as D ; (2) a unit profit table *utable*; (3) the minimum high average-utility threshold δ ; (4) the minimum expected support threshold μ . The revised procedure takes the steps as follows. The input database is firstly scanned to find the sequence-maximum utility for each sequence in D (Lines 1 to 2). Then, the 1-items of high potential average-utility upper-bound sequential patterns are kept in the set of PHAUUBSPs (Lines 3 to 6) and the unqualified items are deleted from the database. When all the unpromising items have been removed from D (Lines 7 to 8), the database itself becomes sanitized, which is called D' ; the discovered set of PHAUUBSPs¹ and D' are finally returned (Line 9).

Algorithm 1: RevisedDB($D, \text{utable}, \mu, \delta$)

Input: D , an uncertain sequential database; utable , the profit table; μ , minimum expected support threshold; δ , minimum high average-utility threshold.

Output: D' , the sanitized database; PHAUUBSPs, the set of potentially high average-utility upper-bound sequential patterns.

```

1 for each sequence  $S_q$  in  $D$  do
2   scan  $D$  to find  $\text{smu}(S_q)$ ;
3 for each  $i_j \in D$  do
4   calculate  $\text{suub}(i_j)$  and  $\text{sp}(i_j)$ ;
5   if  $\text{sp}(i_j) \geq |D| \times \mu$  and  $\text{suub}(i_j) \geq \text{TSU} \times \delta$  then
6     PHAUUBSPs1  $\leftarrow i_j \cup \text{PHAUUBSPs}^1$ ;
7   else
8     remove  $i_j$  from  $D$ ;
9 return PHAUUBSPs1,  $D'$ ;

```

4.3 Proposed level-wise PHAUB algorithm

The baseline algorithm named PHAUB mines the PHAUSPs on a level-by-level basis. First, a breadth-first search is performed in the algorithm to mine PHAUUBSPs. Using the PHAUUBDC property and pruning Strategy 1, the unsatisfied sequences are pruned in advance; thereby the search space is significantly reduced. The revised database is then scanned again to obtain the actual PHAUSPs for each sequence in the set of PHAUUBSPs. The pseudo-code is given in Algorithm 2.

Algorithm 2: Proposed PHAUB Algorithm

Input: D' , a revised uncertain sequential database; utable , the profit table; μ , minimum expected support threshold; δ , minimum high average-utility threshold; PHAUUBSPs¹, the set of 1-PHAUUBSPs.

Output: PHAUSPs, the set of potentially high average-utility sequential patterns.

```

1 set  $k:=2$ ;
2 while PHAUUBSPsk-1  $\neq \text{null}$  do
3    $C_k := \text{Apriori}(\text{PHAUUBSPs}^{k-1})$ ;
4   for each  $S^k \in C_k$  do
5     if  $\exists s \subseteq S^k \wedge (s \subseteq \text{AUCS} \wedge \text{suub}(s) < \text{TSU} \times \delta)$  then
6       remove  $s$  from  $C_k$ ;
7     else
8       calculate  $\text{sp}(S^k)$  and  $\text{suub}(S^k)$ ;
9       if  $\text{sp}(S^k) \geq |D'| \times \mu$  and  $\text{suub}(S^k) \geq \text{TSU} \times \delta$  then
10        PHAUUBSPsk  $\leftarrow \text{PHAUUBSPs}^k \cup S^k$ ;
11   $\text{cands\_set} \leftarrow \text{PHAUUBSPs}^k$ ;
12   $k := k + 1$ ;
13 for each sequence  $s \in \text{cands\_set}$  do
14  calculate  $\text{sau}(s)$ ;
15  if  $\text{sau}(s) \geq \text{TSU} \times \delta$  then
16    PHAUSPs  $\leftarrow \text{PHAUSPs} \cup s$ ;
17 return PHAUSPs ;

```

The inputs of the baseline PHAUB algorithm include: (1) a revised D , called D' ; (2) an unit profit table $utable$; (3) the minimum average-utility threshold, μ ; (4) the minimum expected support threshold, δ . The initial value of k is initially set as 2 (Line 1), and the set of PHAUUBSPs is discovered in a level-wise manner through a loop (Lines 2 to 12). In the $(k-1)$ -th iteration of the loop, the set of PHAUUBSPs ^{k} is retrieved as follows. The generate-and-test procedure is firstly performed to generate the k -sequence by combining two $(k-1)$ -sequence (Line 3). For each candidate sequence S^k , the pruning Strategy 3 is applied to prune the unpromising sequence in advance which cannot further be the PHAUUBSP (Lines 5 to 6). The $suub$ and sp values for S^k are then calculated to find the satisfied sequences and put them into the set of PHAUUBSPs ^{k} (Lines 8 to 10). This procedure is performed in a recursive fashion until the set of PHAUUBSPs ^{$k-1$} becomes *null* (Line 2). Finally, the database is re-scanned to obtain the PHAUSPs from the sequences in the set of PHAUUBSPs (Lines 13 to 16), which are returned as the final results (Line 17).

This algorithm is trivial but effectively discovers the PHAUSPs and the correctness and completeness are maintained. To speed up the mining performance, an improved algorithm named PHAUP is introduced as follows.

4.4 The proposed PHAUP algorithm

Because the PHAUB algorithm uses a level-wise approach to mine the PHAUSPs, high computational and space overhead are required during the mining process. To solve this problem, we present an improved PHAUP algorithm. The PHAUP algorithm uses the projection technique to project the smaller database of the processed sequence, whereby the computational and space overhead can be considerably reduced. The related definitions are presented as follows.

Definition 18 Let S_a and S_b be two sequences defined as $S_a = \langle I_1, I_2, \dots, I_n \rangle$ and $S_b = \langle I'_1, I'_2, \dots, I'_m \rangle$ ($1 \leq m \leq n$). Suppose that all items of each sequence are sorted alphabetically. The sequence S_b is termed a prefix of S_a if 1) $I'_i = I_i$ ($1 \leq i \leq m - 1$); 2) $I'_m \subseteq I_m$ and all items in $(I_m - I'_m)$ are sorted in their alphabetic order.

For example, $\langle (b) \rangle$, $\langle (b), (c) \rangle$, $\langle (b), (c), (d) \rangle$, $\langle (b), (c), (d), (b) \rangle$ are all the prefix of the sequence $\langle (b), (c), (d), (b), (c) \rangle$, but not the prefix of the sequence $\langle (b), (d) \rangle$, $\langle (b), (c), (d), (c) \rangle$.

Definition 19 Let s and S_q be two sequences satisfying that $s \subseteq S_q$. A subsequence of S_q is considered to be the projection of s if the following two conditions are satisfied: (1) the sequence has prefix s and (2) no proper super-sequence of s satisfying that it is not only a subsequence S_q but also has prefix S , represented by $S_q|s$. The projected database of a sequence s in the revised uncertain sequential database D' represents the collection of all projected sequences of each sequence in the database with regard to the sequence s , represented by $D'|s$.

As an example, the projected sequence of $\langle (b) \rangle$ for the sequence $\langle (b), (c), (d), (b), (c) \rangle$ is $\langle -, (c), (d), (b), (c) \rangle$. Also, the projected database of the sequence $\langle (a) \rangle$ in Table 1 is presented in Table 4.

The pseudo-code description of the PHAUP algorithm is given in Algorithm 3.

The inputs of the PHAUP algorithm include: (1) D' , a revised uncertain sequential database; (2) $utable$, a table containing the unit profit of each item; (3) δ , the average-minimum utility threshold; (4) μ , the minimum expected support threshold; (5) the set of $PHAUBSP_s^1$, that is, the PHAUUBSPs of length 1. For each l -sequence s in

Table 4 The projected database of the sequence $\langle(a)\rangle$

SID	Sequence	Probability
1	$\langle(a, 3), (b, 4), [(a, 1), (c, 1)]\rangle$	0.6
2	$\langle[(a, 1)], [(a, 2), (b, 2)]\rangle$	0.8
3	$\langle[(a, 1), (b, 2)], [(a, 4), (b, 1)]\rangle$	0.9

Algorithm 3: Proposed PHAUP Algorithm

```

Input:  $D'$ , a revised uncertain sequential database;  $utable$ , the profit table;  $\mu$ , minimum expected support threshold;  $\delta$ , minimum high average-utility threshold;  $PHAUUBSPs^1$ , the set of 1- $PHAUUBSPs$ .
Output:  $PHAUSPs$ , the set of potentially high average-utility sequential patterns.
1 for each sequence  $s \in PHAUUBSPs^1$  do
2   scan  $D'$  to generate projected database  $D'|_S$  of  $s$ ;
3   set  $k := 1$ ;
4   call Project( $D'|_S, utable, \delta, \mu, k$ );
5 for each sequence  $s \in PHAUUBSPs$  do
6   calculate  $sau(s)$ ;
7   if  $sau(s) \geq TSU \times \delta$  then
8      $PHAUSPs \leftarrow PHAUSPs \cup s$ ;
9 return  $PHAUSPs$ ;
    
```

$PHAUUBSPs^1$, a scan of the input database is executed to generate the projected database $D'|_s$ (Line 2). The initial k value is then set as 1 (Line 3), and the **Project** process is then performed to obtain the projected database of S (Line 4). In the **Project** function (c.f. Algorithm 4), each itemset X' with a prefix X is recursively assessed in order to generate the set of $PHAUUBSPs$ completely (Lines 1 to 10). First, it produces the $(k+1)$ -itemset X' which consists of the prefix itemset X and each $S \in PHAUUBSPs^1$ (Line 1). Then, they are added to the set of C^{k+1} (Line 2). For the itemsets in C^{k+1} , they are processed and the $db|_{X'}$ is performed to obtain the projected sub-database for the $db|_{X'}$ of X' (Line 4), $sp(X')$ and $suub(X')$ (Line 5). Two conditions are then assessed to check whether its supersets have to be explored and checked for the subsequent projection search (Lines 6 to 9). If an itemset is obtained in the set of $PHAUUBSPs$, the Project procedure will be continuously executed (Line 8). After that, the **Project** procedure return $PHAUUBSPs$ with a prefix X (Line 10). The detailed description of the **Project** procedure is given as follows.

5 An illustrative example

In this section, we will present an example to better explain the ideas behind the proposed algorithms. The uncertain sequential database in this example has been presented in Table 1. For the two algorithms we proposed, the database will be revised to remove unpromising candidates. A scan of the database is conducted for the calculation of the expected support and $suub$ of each 1-sequence. In this example, all the resulting 1-sequences are presented in Table 5.

After the above step, the $\langle(a)\rangle$, $\langle(b)\rangle$ and $\langle(c)\rangle$ are regarded as the PHAUUBSPs and added into $PHAUUBSPs^1$ (a set to keep 1-sequences). Then, an individual item in the database will be removed from the database using Algorithm 1 if it is not in the set $PHAUUBSPs^1$.

Algorithm 4: Proposed **Project**($X, db|_X, k$) Algorithm

Input: X , a prefix itemset; $db|_X$, the projected database in which each transaction containing an itemset X ; k , the length of an itemset X .
Output: $PHAUBSPs$, the set of potentially high average-utility sequential patterns with a prefix X .
 1 Produce the $(k+1)$ -itemset X' consisting of the prefix itemset X and each $S \in PHAUBSPs^1$;
 2 put X' into the set of $(k+1)$ candidates C^{k+1} ;
 3 **for** each $(k+1)$ -itemset $X' \in C^{k+1}$ **do**
 4 scan $db|_{X'}$ to retrieve the projected sub-db $db|_{X'}$ of X' ;
 5 calculate $sp(X')$ and $suub(X')$;
 6 **if** $sp(X') \geq |D'| \times \mu$ and $suub(X') \geq TSU \times \delta$ **then**
 7 $PHAUBSPs^{k+1} \leftarrow PHAUBSPs^{k+1} \cup X'$;
 8 call **Project**($X', db|_{X'}, k + 1$);
 9 $PHAUBSPs \leftarrow \cup PHAUBSPs^{k+1}$;
 10 **return** $PHAUBSPs$ with the prefix X ;

Table 5 An example of the 1-sequences

Sequence	a	b	c	d	e	f
<i>suub</i>	18	22	22	8	6	4
<i>sp</i>	2.3	2.8	0.8	1.3	0.6	0.5

For the PHAUB algorithm, the k value is set to 2 and the set of 2-sequences are produced by the combinational procedure of the 1-sequences in the set of $PHAUBSPs^1$. The pruning Strategy 3 is used to remove unpromising candidates from the set of $PHAUBSPs^{k+1}$. The built AUCS was given in Table 3, which are used to produce the *suub* values of the 2-sequences. Therefore, many candidates extended from the 2-sequences are removed in advance. For example, the sequence $\langle (a), (c), (c) \rangle$ does not have to be evaluated as its *suub* value of 2-sequence $\langle (c), (c) \rangle$ fails to satisfy the condition. Recursively, this procedure is performed until there are no more candidates generated. After that, the database is required to be re-scanned to obtain the actual PHAUSPs, and the final results were given in Table 2.

For the improved PHAUP algorithm, a scan on the database is performed to obtain all $PHAUBSPs^1$ which will be put into the set of $PHAUBSPs^1$, i.e., $\langle (a) \rangle$, $\langle (b) \rangle$, and $\langle (c) \rangle$. For each sequence S^k in the set of $PHAUBSPs^k$ ($k = 1$), its projected database $D'|S^k$ is generated. Suppose that the sequence $\langle (a) \rangle$ is a $PHAUBSPs$ and its projected database is what have been shown in Table 4. After that, the *suub* and *sp* values of the extended sequences are then calculated. For example of the sequence $\langle (a) \rangle$, its extended sequences are $\langle (a), (a) \rangle$, $\langle (a), (b) \rangle$, $\langle (a), (c) \rangle$, and $\langle (a), (c) \rangle$. If their *suub* and *sp* values meet the conditions, they are then put into the set of $PHAUBSPs^{k+1}$. The actual average-utility value of the processed sequence is compared with the minimum average-utility count in order to obtain the qualified PHAUSPs. This algorithm is recursively executed until no more candidates are produced. The final results were given in Table 2.

6 Experimental evaluation

We have conducted substantial experiments to evaluate the performance of the proposed algorithms. More specifically, we evaluated the execution time, the total number of candidates

Table 6 Characteristics for the conducted datasets

Dataset	# D	# I	AvgLen	MaxLen	Type
SIGN	730	267	52	94	Sign language
FIFA	20,450	2990	36.2	100	Click-stream
BIBLE	36,369	13,905	21.6	100	Book
BMS	59,601	497	2.5	267	Click-stream

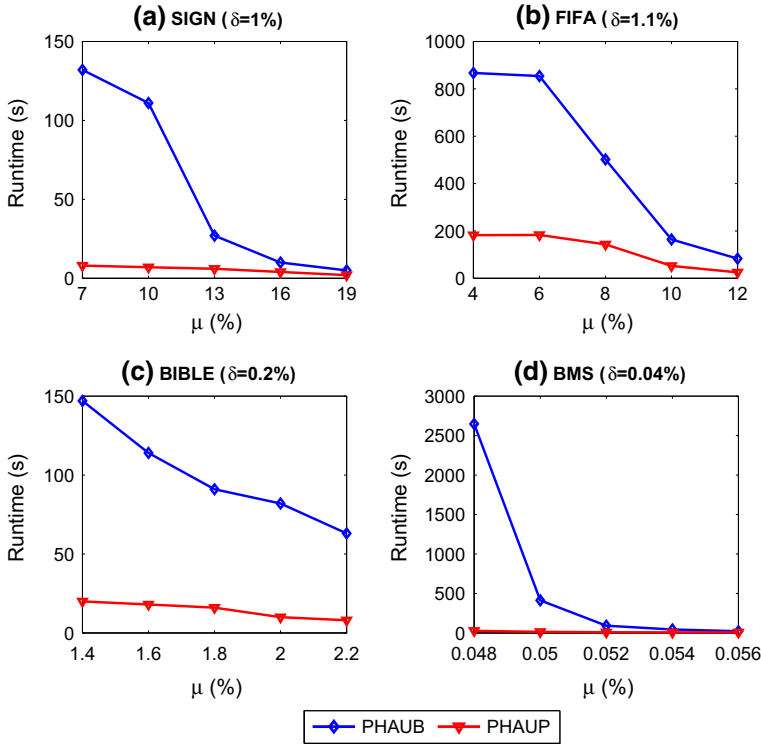


Fig. 1 Runtime w.r.t fixed μ with various δ

generated, space overhead, the number of discovered patterns and scalability toward large datasets on our algorithms. Note that there is currently a lack of existing works that are focused on high average-utility sequential pattern mining from uncertain database. As a result, only the two proposed approaches are reasonable compared in terms of speed performance, number of candidates, and memory usage. The HUSPM algorithm [42] is compared with the designed approaches in terms of number of discovered patterns and scalability to show the effectiveness of the designed patterns. All algorithms were implemented using Java on a Windows 7 desktop featuring an Intel i7-4790 CPU, clocked at 3.60GHz with 8GB of RAM. Four real-life and one databases obtained from SPMF [12] were considered for the experimental evaluation. The details of the parameters used for the datasets are: #|D| indicates the number of sequences in the database; #|I| indicates the number of distinct items in the database; AvgLen indicates the average length of transactions in the database; MaxLen indicates the maximal length of transactions in the database; and Type indicates the dataset type. The characteristics of the used datasets are given in Table 6.

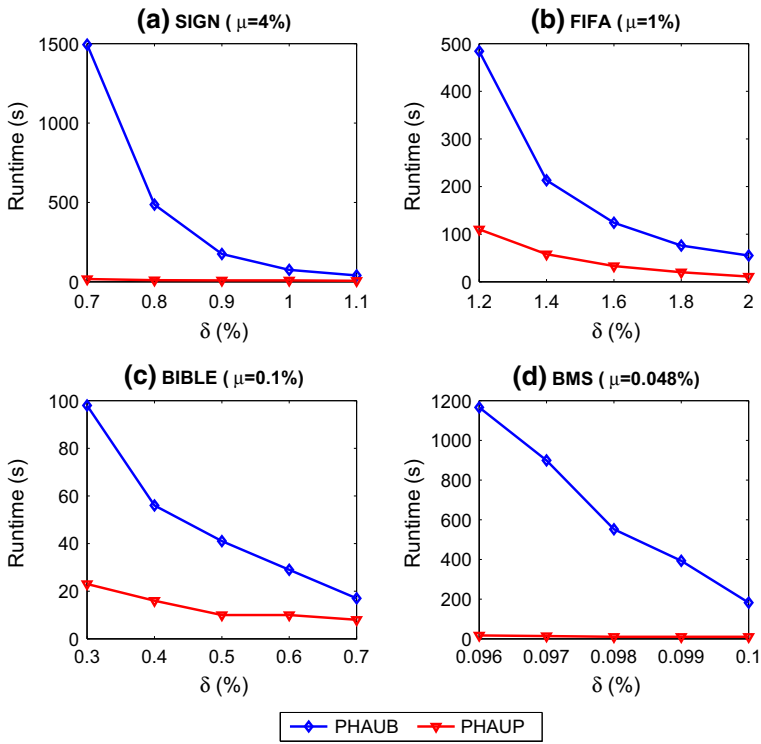


Fig. 2 Runtime w.r.t fixed δ with various μ

The BIBLE is moderately dense and contains many medium length sequences. This dataset is a conversion of the Bible into a sequence database (each word is an item). The SIGN dataset is a dense dataset containing very long sequences. It is the sign language containing approximately 730 sequences. The FIFA is moderately dense and contains many long sequences. It shows the click stream data from the website of FIFA World Cup 98. For the BMS dataset, it is a click-stream data from a webstore used in KDD-Cup 2000. For all datasets, external utilities of items are generated between 0 and 1000 by using a log-normal distribution and quantities of items are generated randomly between 1 and 5, which is similar to the existing works [44]. Also, the uncertainty factor is randomly assigned to each sequence in a range of (0, 1).

6.1 Speed performance

In this section, the speed performance for the proposed algorithms is then evaluated, which involves measuring the CPU and disk I/O times. Figures 1 and 2 represent the execution time of the proposed algorithms under different values of μ and δ , respectively.

From Figs. 1 and 2, we can see that the execution time of all the algorithms decreases when the minimum expected support threshold or minimum high average-utility threshold increases. Particularly, the PHAUP algorithm is much faster than the PHAUB algorithm in all cases since the projection mechanism can easily reduce the size of the processed sequence, thus the computational cost of PHAUP can be greatly reduced.

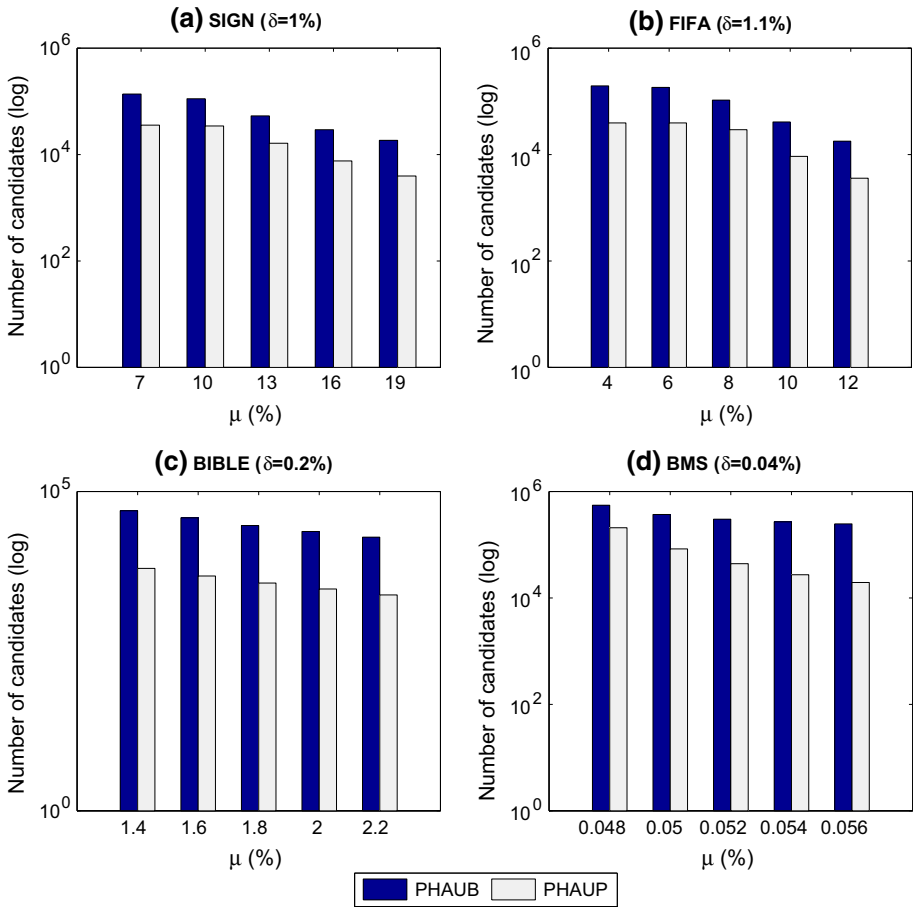


Fig. 3 Number of candidates w.r.t fixed μ with various δ

Specifically, for the SIGN dataset shown in Fig. 1a when the δ is set as 1% and μ was varied from 7 to 19%, the execution time of PHAUB decreases from 132 to 5 s, while that of PHAUP decreases from 8 to 2 s. The result is understandable since the PHAUB algorithm uses the generate-and-test mechanism, which has to generate large amounts of candidates to mine PHAUSPs. Also, performing multiple database scans while calculating the *suub* values of the candidates at each level requires a longer execution time. The projection-based PHAUP algorithm can gradually shrink the size of the database of the processed sequence, which leads to a decrease in the total number of candidates. The mining process can be accelerated by the increasing the size of sequences.

6.2 Number of candidates

Sequence candidates are generated by the proposed algorithms, and their sizes are analyzed in this section. A sequence becomes a candidate if it requires one more database scan in order to obtain its average-utility. The candidate number under different μ and δ values are presented in Figs. 3 and 4.

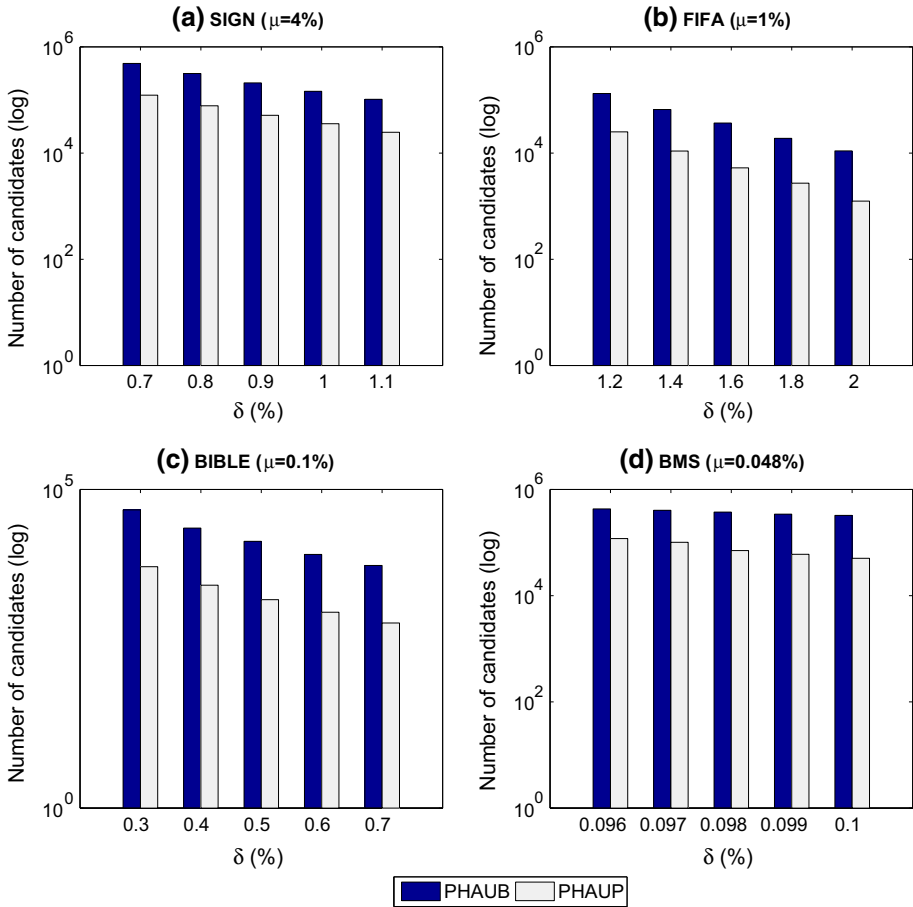


Fig. 4 Number of candidates w.r.t fixed δ with various μ

Figures 3 and 4 clearly illustrate that the PHAUP algorithm always has a lower number of candidates against the baseline method. For example in Fig. 3a, when μ is 7% and δ is 1%, the total number of candidates of PHAUB and PHAUP are, respectively, 136,479 and 35,577. This can be explained by the fact that the PHAUP algorithm is developed based on the projection strategy to reduce the size of the processed database, so the number of determined candidates is far smaller than that of the candidates in the whole database. Furthermore, the *sub* value of the candidate in the projected dataset is significantly smaller when compared with the candidates in the original database, contributing to fewer candidates being generated. Therefore, the projection mechanism plays a very important role in performance improvement by reducing the number of candidates for mining the PHAUSPs.

6.3 Memory overhead

The memory overhead of the two proposed algorithms is analyzed. The results of the memory usage under different μ and δ values are presented in Figs. 5 and 6, respectively.

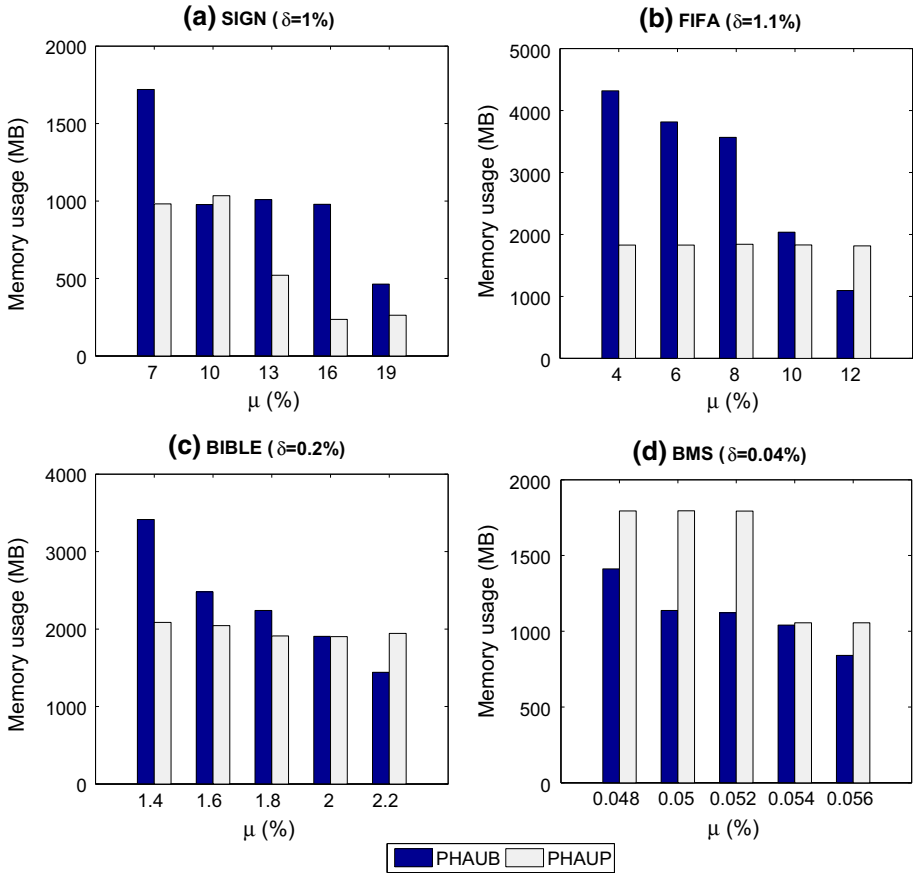


Fig. 5 Memory usage w.r.t fixed μ with various δ

We can see from Figs. 5 and 6 that the proposed PHAUP sometimes incurs a high memory consumption than that of the PHAUB algorithm. In Fig. 5d, when μ is 0.05% and δ is 0.04%, the memory usage of PHAUB and PHAUP are 1137 MB and 1795 MB, respectively. This is because the PHAUP algorithm needs to generate projection database iteratively, which typically requires more memory space. Yet, when the threshold is set lower, for example in Figs. 5a–c and 6b, c, the PHAUP requires lower memory consumption than that of the PHAUB because the PHAUB may generate a larger amount of candidates of PHAUSPs. Generally speaking, the PHAUP has a better space performance under lower thresholds compared to the PHAUB approach.

6.4 Number of discovered patterns

Although it is not reasonable to compare the designed algorithm with the HUSPM [42] in terms of speed, number of candidates, and memory overhead, it is possible to compare the numbers of discovered patterns of HUSPM and the designed algorithms to show the effectiveness of the designed pattern. The number of the discovered potential high-utility sequential patterns (PHUSPs) in HUSPM [42] and the number of mined potential high average-utility

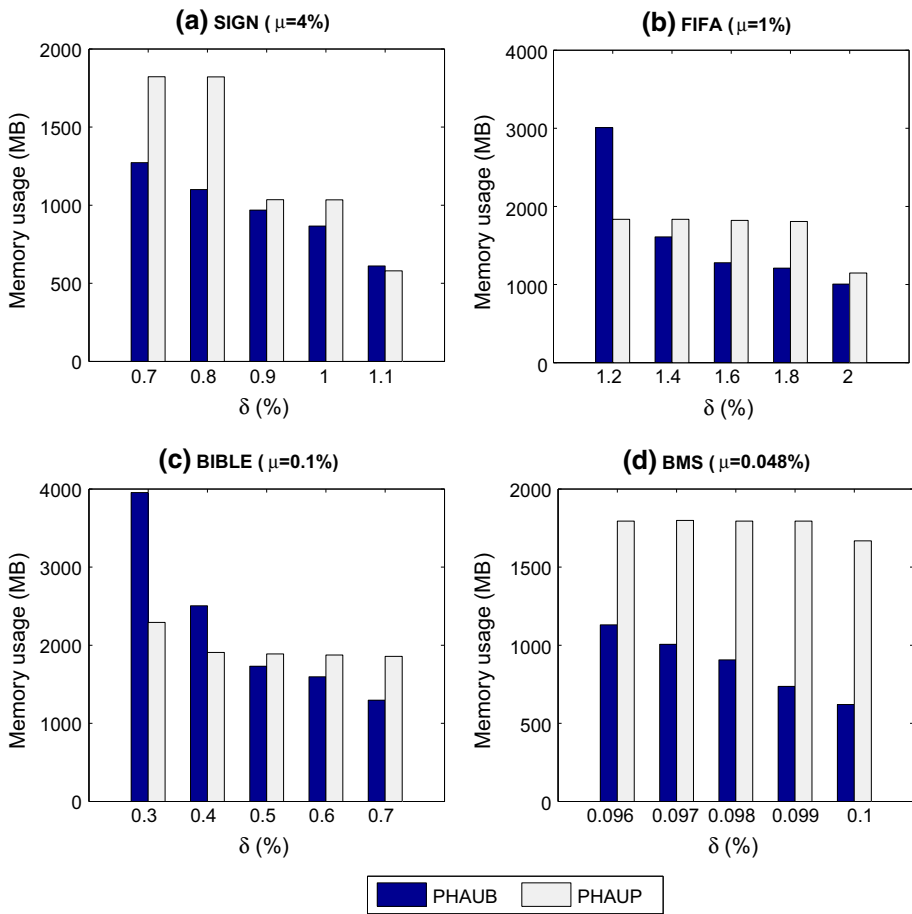


Fig. 6 Memory usage w.r.t fixed δ with various μ

sequential patterns (PHAUSPs) of the designed PHAUB and PHAUP algorithms are then evaluated and shown in Figs. 7 and 8. Note that the designed PHAUB and PHAUP generate the same number of the desired PHAUSPs.

From Figs. 7 and 8, we can observe that the designed algorithms generate much fewer patterns (PHAUSPs) than the HUSPM [42] (PHUSPs). This is reasonable since the average-utility concept is adopted in the designed framework, thus providing a fair measurement than that of the HUSPM because the utility of a discovered pattern will not increase along with its size. Thus, less information and fewer patterns are then discovered and the effectiveness for decision making can be achieved.

6.5 Scalability

In this section, we evaluated how well the two proposed algorithms scale under varying number of sequences in the input database. Also, the number of the discovered patterns with the HUSPM is then evaluated. Experiments were conducted on a series of artificial datasets called *S10I4N4KD|X|K*. Experiments of runtime, number of candidates, and memory

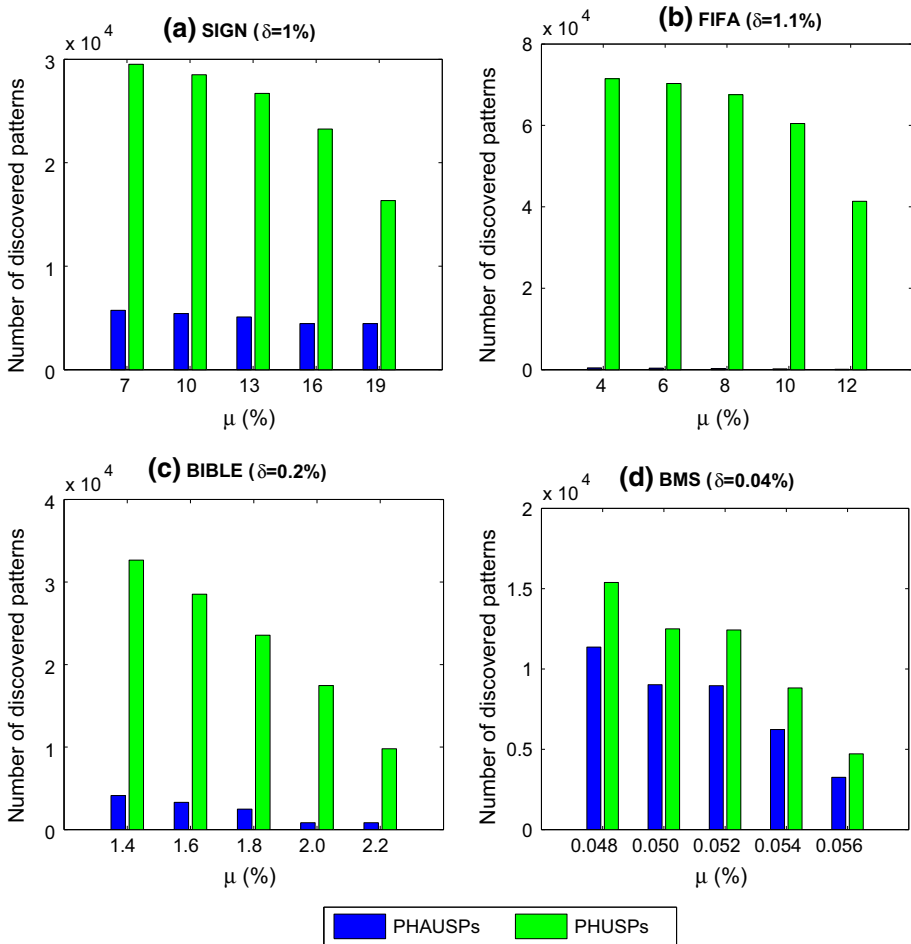


Fig. 7 Number of discovered patterns w.r.t fixed μ with various δ

usage are shown in Fig. 9. The minimum expected support threshold and the minimum high average-utility threshold are both set as 0.1%.

Figure 9 shows that the PHAUP algorithm enjoys a better scalability than that of the PHAUB algorithms in terms of the execution time and the size of the candidates. When the size of dataset increases, the projection method may require a higher memory consumption to project the sub-dataset of the processed sequence. Furthermore, when the size of dataset increases, it needs more memory space to maintain the *sub* and probability values of the candidates.

We can also see that, when the database size increases, the execution time and memory consumption of the two algorithms increase at the same time, but the number of candidates remain stable nevertheless. This is because the databases with different sizes are generated by the same dataset generator, which processes the minimum expected support value and the minimum high average-utility count with an increase in a right proportion to the database size. As a result, a sequence not considered as a PHAUUBSP in the small dataset may not be the PHAUUBSP in the large dataset. We also can observe

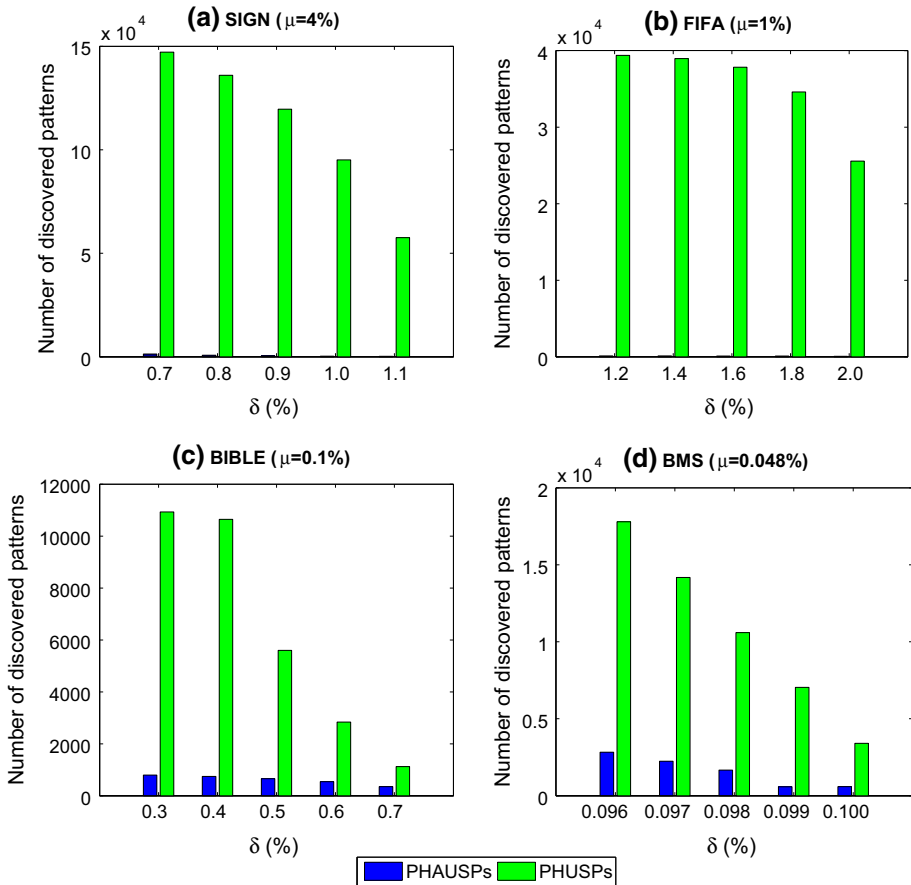


Fig. 8 Number of discovered patterns w.r.t fixed δ with various μ

that the number of PHUSPs is much more than that of the number of PHAUSPs, thus PHAUSPs hold the effectiveness for providing less but useful information for decision making.

7 Conclusion

We first present a novel algorithm framework called potentially high average-utility sequential pattern mining (PHAUSPM). This framework mines potentially high average-utility sequential patterns (PHAUSPs) from uncertain sequential databases. A new concept called potentially high average-utility upper-bound sequential pattern (PHAUBSP) was introduced to ensure the correctness and completeness of the discovered PHAUSPs. The PHAUB algorithm with three different pruning techniques is also proposed to accelerate the mining process, which mines the PHAUSPs at different levels. The second PHAUP algorithm is proposed based on the projection method to discover the PHAUSPs from the projected dataset. Extensive experiments were conducted for performance evaluation of the proposed algorithms from the perspectives of execution time, number of candi-

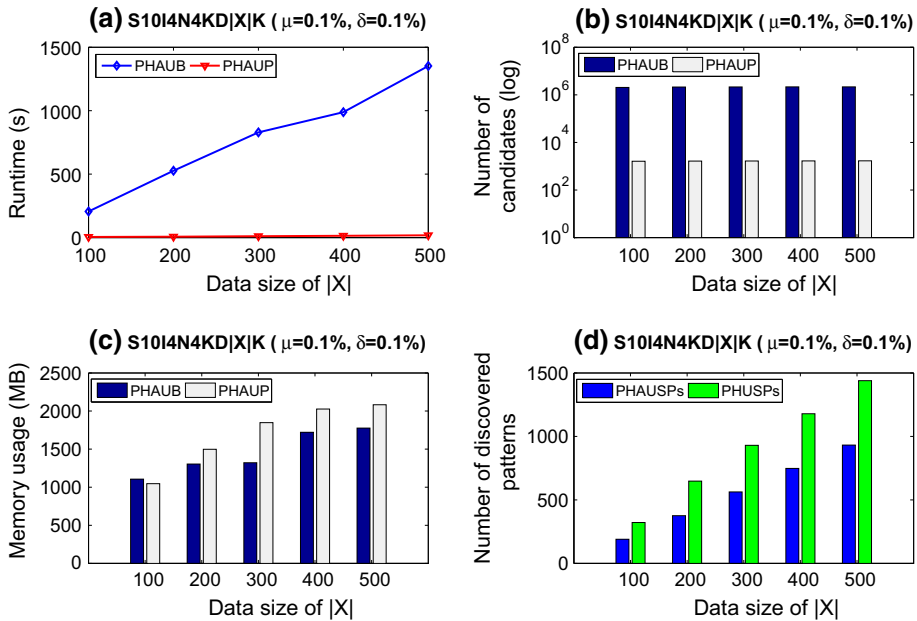


Fig. 9 Scalability of the compared approaches

dates, memory consumption, number of discovered patterns, and scalability. The results demonstrated that the proposed algorithms can efficiently and effectively mine the required patterns.

References

1. Aggarwal CC, Li Y, Wang J, Wang J (2009) Frequent pattern mining with uncertain data. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 29–38
2. Agrawal R, Imielinski T, Swami AA (1990) Database mining: a performance perspective. *IEEE Trans Knowl Data Eng* 5(6):914–925
3. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large database. In: ACM SIGMOD international conference on management of data, pp 207–216
4. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: International conference on very large data bases, pp 619–624
5. Agrawal R, Srikant R (1995) Mining sequential patterns. In: IEEE international conference on data engineering, pp 3–14
6. Ahmed CF, Tanbeer SK, Jeong BS (2010) A novel approach for mining high-utility sequential patterns in sequence databases. *ETRI J* 32(5):676–686
7. Alkan OK, Karagoz P (2015) CRoM and HuspExt: improving efficiency of high utility sequential pattern extraction. *IEEE Trans Knowl Data Eng* 27(10):2645–2657
8. Bernecker T, Kriegel HP, Renz M, Verhein F, Zue A (2009) Probabilistic frequent itemset mining in uncertain databases. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 119–128
9. Bui N, Vo B, Huynh VN, Lin CW, Nguyen LTT (2016) Mining closed high utility itemsets in uncertain databases. In: Proceedings of the 7th symposium on information and communication technology, pp 7–14
10. Chau M, Cheng R, Kao B (2005) Uncertain data mining: a new research direction. In: The workshop on the sciences of the artificial, pp 1–8
11. Chui CK, Kao B, Hung E (2007) Mining frequent itemsets from uncertain data. In: The Pacific-Asia conference on knowledge discovery and data mining, pp 47–58

12. Fournier-Viger P, Lin JCW, Gomariz A, Gueniche T, Soltani A, Deng Z, Lam HT (2016) The SPMF open-source data mining library version 2. In: European conference on principles of data mining and knowledge discovery, pp 36–40
13. Ge J, Xia Y, Wang J, Nadungodage CH, Prabhakar S (2017) Sequential pattern mining in databases with temporal uncertainty. *Knowl Inf Syst* 53(3):821–850
14. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. *Data Min Knowl Discov* 8(1):53–87
15. Hong TP, Lee CH, Wang SL (2011) Effective utility mining with the measure of average utility. *Expert Syst Appl* 38(7):8259–8265
16. Lan GC, Hong TP, Tseng VS (2012) Efficiently mining high average-utility itemsets with an improved upper-bound strategy. *Int J Inf Technol Decis Mak* 11:1009–1030
17. Lan GC, Hong TP, Tseng VS, Wang SL (2014) Applying the maximum utility measure in high utility sequential pattern mining. *Expert Syst Appl* 41(11):5071–5081
18. Lan Y, Wang Y, Wang Y, Yi S, Yu D (2015) Mining high utility itemsets over uncertain databases. In: International conference on cyber-enabled distributed computing and knowledge discovery, pp 235–238
19. Leung CKS, Mateo MAF, Brajczuk DA (2008) A tree-based approach for frequent pattern mining from uncertain data. In: The Pacific-Asia conference on knowledge discovery and data mining, pp 653–661
20. Lin CW, Hong TP, Lu WH (2010) Efficiently mining high average utility itemsets with a tree structure. *Lecture Notes Comput Sci* 5990:131–139
21. Lin CW, Hong TP, Lu WH (2011) An effective tree structure for mining high utility itemsets. *Expert Syst Appl* 38:7419–7424
22. Lin JCW, Li T, Fournier-Viger P, Hong TP, Zhan J, Voznak M (2016) An efficient algorithm to mine high average-utility itemsets. *Adv Eng Inform* 30(2):233–243
23. Lin JCW, Ren S, Fournier-Viger P, Hong TP, Su JH, Vo B (2017) A fast algorithm for mining high average-utility itemsets. *Appl Intell* 47(2):331–346
24. Lin JCW, Gan W, Fournier-Viger P, Hong TP (2017) Efficiently mining uncertain high-utility itemsets. *Soft Comput* 21(11):2801–2820
25. Lin JCW, Yang L, Fournier-Viger P, Hong TP (2019) Mining of skyline patterns by considering both frequent and utility constraints. *Eng Appl Artif Intell* 77:229–238
26. Liu Y, Liao W, Choudhary A (2005) A two-phase algorithm for fast discovery of high utility itemsets. In: The Pacific-Asia conference on knowledge discovery and data mining, pp 689–695
27. Liu M, Qu J (2012) Mining high utility itemsets without candidate generation. In: ACM international conference on information and knowledge management, pp 55–64
28. Lu T, Vo B, Nguyen HT, Hong TP (2014) A new method for mining high average utility itemsets. *Lecture Notes Comput Sci* 8838:33–42
29. Muzammal M, Gohar M, Rahman AU, Qu Q, Ahmad A, Jeon G (2018) Trajectory mining using uncertain sensor data. *IEEE Access* 6:4895–4903
30. Muzammal M, Rajeev (2015) Mining sequential patterns from probabilistic databases. In: The Pacific-Asia conference on knowledge discovery and data mining vol 44(2), pp 325–358
31. Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Trans Knowl Data Eng* 16(11):1424–1440
32. Sun L, Cheng R, Cheung DW, Cheng J (2010) Mining uncertain data with probabilistic guarantees. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 273–282
33. Tong Y, Chen L, Cheng Y, Yu PS (2012) Mining frequent itemsets over uncertain databases. *VLDB Endow* 5(11):1650–1661
34. Tseng VS, Shie BE, Wu CW, Yu PS (2013) Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Trans Knowl Data Eng* 25:1772–1786
35. Wang J, Huang J, Chen Y (2016) An efficiently mining high utility sequential patterns. *Knowl Inf Syst* 49(2):597–627
36. Wang L, Cheng R, Lee SD, Cheung D (2010) Accelerating probabilistic frequent itemset mining: a model-based approach. In: ACM international conference on information and knowledge management, pp 429–438
37. Wang J, Liu F, Jin C (2017) PHUIMUS: a potential high utility itemsets mining algorithm based on stream data with uncertainty. *Math Problems Eng*, vol 2017, Article ID 8576829, p 13
38. Yao H, Hamilton HJ (2006) Mining itemset utilities from transaction databases. *Data Knowl Eng* 59(3):603–626
39. Yao H, Hamilton HJ, Butz CJ (2004) A foundational approach to mining itemset utilities from databases. In: SIAM international conference on data mining, pp 211–225
40. Yin J, Zheng Z, Cao L (2012) USpan: an efficient algorithm for mining high utility sequential patterns. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 660–668

41. Yin J, Zheng Z, Cao L, Song Y, Wei W (2013) Efficiently mining top- k high utility sequential patterns. In: IEEE international conference on data mining, pp 1259–1264
42. Zhang B, Lin JCW, Fournier-Viger P, Li T (2017) Mining of high utility-probability sequential patterns from uncertain databases. PLoS One 12(7):1–21
43. Zhao Z, Yan D, Ng W (2014) Mining probabilistically frequent sequential patterns in large uncertain databases. IEEE Trans Knowl Data Eng 26(5):1171–1184
44. Zida S, Fournier-Viger P, Wu CW, Lin JCW, Tseng VS (2015) Efficient mining of high-utility sequential rules. In: International workshop on machine learning and data mining in pattern recognition, pp 157–171

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Jerry Chun-Wei Lin received his Ph.D. at Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan in 2010. He is currently an Associate Professor at the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He has published more than 250 research papers in refereed journals and international conferences. His research interests include data mining, soft computing, artificial intelligence, social computing, machine learning, and privacy-preserving and security technologies. He is the Editor-in-Chief of Data Science and Pattern Recognition (DSPR) journal, co-leader of the SPMF library, and also the founder and the leader of PPSF library. He is an IEEE Senior Member.



Ting Li received his Master degree at School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China in 2017. His research interests include data mining and big data analytics.



Matin Pirouz is a faculty member at the Computer Science Department in California State University, Fresno, USA. Her current research interests are algorithms, data mining, big data analytics, social network analysis, graph theory, and complex networks. She has received several national and international awards, and her research is published in various journal and conferences.




Ji Zhang is currently an Associate Professor in computing at the University of Southern Queensland (USQ), Australia. He is an IEEE senior member, ACM member, Australian Endeavour Fellow, Queensland Fellow (Australia) and Izaak Walton Killam Scholar (Canada). His research interests are Big data analytics, knowledge discovery and data mining (KDD), information privacy and security. He was a Post-doctoral Research Fellow in CSIRO ICT Center at Hobart, Australia, from 2008–2009. He received his degree of Ph.D. from the Faculty of Computer Science at Dalhousie University, Canada, in 2008, degree of M.Sc from Department of Computer Science at the National University of Singapore in 2002, and degree of B.E from Department of Information Management and Information Systems at the Southeast University, China in 2000. He has published over 140 papers in major peer-reviewed international journals and conferences.



Philippe Fournier-Viger is full professor at the Harbin Institute of Technology (Shenzhen). His research interests include data mining, pattern mining, sequence analysis and prediction, and their applications. He has published more than 220 research papers in refereed international conferences and journals, which have received more than 3600 citations. He is the founder of the popular SPMF open-source data mining library (<http://www.philippe-fournier-viger.com/spmf/>), which has been used in more than 600 research papers since 2010. He is editor-in-chief of the Data Mining and Pattern Recognition journal.

Affiliations

Jerry Chun-Wei Lin¹  · Ting Li² · Matin Pirouz³ · Ji Zhang⁴ ·
Philippe Fournier-Viger⁵

✉ Jerry Chun-Wei Lin
jerrylin@ieee.org

Ting Li
tingli@ikelab.net

Matin Pirouz
mpirouz@ieee.org

Ji Zhang
Ji.Zhang@usq.edu.au

Philippe Fournier-Viger
philfv8@yahoo.com

- ¹ Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway
- ² School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen, China
- ³ Department of Computer Science, California State University, Fresno, USA
- ⁴ School of Agricultural, Computational and Environmental Sciences, University of Southern Queensland, Toowoomba, QLD, Australia
- ⁵ School of Natural Sciences and Humanities, Harbin Institute of Technology (Shenzhen), Shenzhen, China