Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8816228, 12 pages
https://doi.org/10.1155/2020/8816228

Research Article

An Efficient Algorithm for Extracting High-Utility Hierarchical Sequential Patterns

Chunkai Zhang, Zilin Du, and Yiwen Zu

School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China

Correspondence should be addressed to Chunkai Zhang; ckzhang@hit.edu.cn

Received 18 March 2020; Revised 8 June 2020; Accepted 23 June 2020; Published 6 July 2020

Academic Editor: Huimin Lu

High-utility sequential pattern mining (HUSPM) is an emerging topic in data mining, where utility is used to measure the importance or weight of a sequence. However, the underlying informative knowledge of hierarchical relation between different items is ignored in HUSPM, which makes HUSPM unable to extract more interesting patterns. In this paper, we incorporate the hierarchical relation of items into HUSPM and propose a two-phase algorithm MHUH, the first algorithm for high-utility hierarchical sequential pattern mining (HUHSPM). In the first phase named Extension, we use the existing algorithm FHUSpan which we proposed earlier to efficiently mine the general high-utility sequences (g-sequences); in the second phase named Replacement, we mine the special high-utility sequences with the hierarchical relation (s-sequences) as high-utility hierarchical sequential patterns from g-sequences. For further improvements of efficiency, MHUH takes several strategies such as Reduction, FGS, and PBS and a novel upper bounder TSWU, which will be able to greatly reduce the search space. Substantial experiments were conducted on both real and synthetic datasets to assess the performance of the two-phase algorithm MHUH in terms of runtime, number of patterns, and scalability. Conclusion can be drawn from the experiment that MHUH extracts more interesting patterns with underlying informative knowledge efficiently in HUHSPM.

1. Introduction

Sequential pattern mining (SPM) [1–3] is an interesting and critical research area in data mining. According to the problem definition [4], a large database of customer transactions has three fields, i.e., customer-id, transaction-time, and the items bought. Each transaction corresponds to an itemset, and all the transactions from a customer are ordered by increasing transaction-time to form a sequence called customer sequence. The support of a sequence is the number of customer sequences that contains it. If the support of a sequence is larger than a user-specified minimum support, we call it a frequent sequence. The sequential pattern mining algorithm will discover the frequent sequences called sequential patterns among all sequences. In a word, the purpose of sequential pattern mining is to discover all frequent sequences as sequential patterns, which reflect the potential connections within items, from a sequence database under the given minimum support. An example of such a sequential pattern is that customers typically buy a phone, then a phone shell, and then a phone charger. Customers who buy some other commodities in between also support this sequential pattern. In the past decades, many algorithms [1, 5] have been proposed for sequential pattern mining, which makes it be widely applied in many realistic scenarios (e.g., consumer behavior analysis [6] and web usage mining [7]). However, sequential pattern mining has two apparent limitations.

Firstly, frequency does not fully reveal the importance (i.e., interest) in many situations [8–12]. In fact, many rare but important patterns may be missed under the frequency-based framework. For example, in retail selling, a phone usually brings more profit than selling a bottle of milk, while the quantity of phones sold is much lower than that of milk [9], and the decision-maker tends to emphasize the sequences consisting of high-profit commodities, instead of those frequent commodity sequences. This issue leads to the emergence of high-utility sequential pattern mining (HUSPM) [8, 12–15]. To represent the relative importance of patterns, each item in the database is associated with a value called external utility (e.g., indicating the unit profit of the item

purchased by a customer). In addition, each occurrence of the item is associated with a quantity called internal utility (e.g., indicating the number of units of the item purchased by a customer in a transaction). The utility of a sequence is computed by applying a utility function on all sequences in the database where it appears. The task of high-utility sequential pattern mining is to discover all high-utility sequential patterns (HUSPs, the sequences with high utility) from the quantitative sequence database with a predefined minimum utility threshold. Many high-utility sequential pattern mining algorithms have been proposed in the past decades [13, 16–20], and high-utility sequential patterns can be extracted more efficiently with a series of novel data structures and pruning strategies proposed. In addition, high-utility sequential pattern mining has many practical applications including web log data [21], mobile commerce environments [22], and gene regulation data.

Secondly, in sequential pattern mining, the hierarchical relation (e.g., product relation and semantic relation) between different items is ignored, so some underlying knowledge may be missed. In general, the individual items of the input sequences are naturally arranged in a hierarchy [23]. For example, suppose both the sequence S_1 : <phone, mobile power pack > and the sequence S_2 : <phone, bluetooth headset > are infrequent, then it seems that there is no association between the three commodities. However, we may find that the sequence S_3 : <phone, phone accessory > is frequent from the perspective of product hierarchy, indicating that customers usually buy a phone first, then buy a phone accessory (including "mobile power pack" and "bluetooth headset"). That is to say, products in sequences of customer transactions can be arranged in a product hierarchy, where mobile power pack and bluetooth headset can generalize phone accessory. Another example is that the individual word in a text can form a semantic hierarchy. The words drives and driven can generalize to their common lemma drive, which in turn generalize to their respective part-of-speech tag verb. The concept of hierarchy (is a taxonomy) provides the deciders with a different perspective to analyze sequential patterns. More informative patterns can be extracted through the hierarchy-based methodology. Besides, although the information reveled from the hierarchical perspective may be relatively fuzzy, it reduces the loss of underlying knowledge to a certain extent. Particularly, the hierarchical relation between different items is sometimes inherent to the application (e.g., hierarchies of directories or web pages) or they are constructed in a manual or automatic way (e.g., product relation) [23]. Figure 1 shows a simple example of a taxonomy of biology in the real application. Sequential pattern mining with hierarchical relation can be traced back to the article [6] where the hierarchy management was incorporated into sequential pattern mining, and the GSP algorithm was proposed to extract sequential patterns according to different levels of hierarchy. Later, sequential pattern mining with hierarchical relation has been studied extensively in the literature [24, 25]. Efficient algorithms were proposed in a wide range of real-world applications, such as customer behavior analysis [6, 26] and information extraction [27].

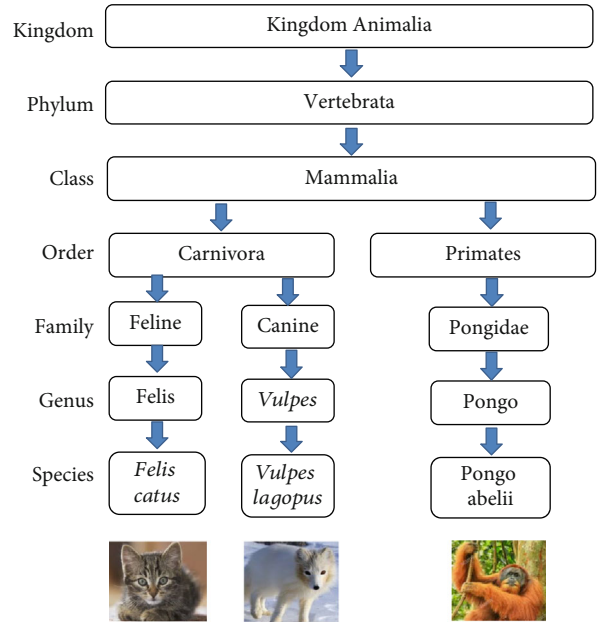


Figure 1: An example of a taxonomy of biology.

However, to the best of our knowledge, there is no related work taking consideration of both two limitations. In this paper, given a quantitative sequence database D (with an external utility table), a user-defined minimum utility threshold, and a series of taxonomies denoting the hierarchical relation, we are committed to finding all high-utility sequences consisting of items with the hierarchical relation (i.e., high-utility hierarchical sequential patterns). In fact, mining such patterns is more complicated than high-utility sequential pattern mining and sequential pattern mining with hierarchical relation. Firstly, compared with high-utility sequential pattern mining, the introduction of hierarchical relation leads to consumption of a large amount of memory and has long execution times due to the combinatorial explosion of the search space. Secondly, the methods of mining sequential pattern mining with hierarchical relation cannot be directly applied, for the download closure property (also known as the Apriori property) [28] is not held under the utility-based framework.

To address the above issues, we propose a new algorithm called MHUH (mining high-utility hierarchical sequential patterns) to mine high-utility hierarchical sequential patterns (to be defined later) by taking several strategies. The major contributions of this paper are as follows.

Firstly, we introduce the concepts of hierarchical relation into high-utility sequential pattern mining and formulate the problem of high-utility hierarchical sequential pattern mining (HUHSPM). Especially, important concepts and components of HUHSPM are defined.

Secondly, we propose a two-phase algorithm named MHUH (mining high-utility hierarchical sequential patterns), the first algorithm for high-utility hierarchical sequential pattern mining. So that the underlying informative knowledge of hierarchical relation between different items will not be missed and to improve efficiency of extracting

HUHPs, several strategies (i.e., FGS, PBS, and Reduction) and a novel upper bounder TSWU are proposed.

Thirdly, substantial experiments were conducted on both real and synthetic datasets to assess the performance of the two-phase algorithm MHUH in terms of runtime, number of patterns, and scalability. In particular, the experimental results demonstrate that MHUH can extract more interesting patterns with underlying informative knowledge efficiently in HUHSPM.

The rest of this paper is organized as follows. Related work is briefly reviewed in Section 2. We describe the related definitions and problem statement of HUHSPM in Section 3. The proposed algorithm is presented in Section 4, and an experimental evaluation assessing the performance of the proposed algorithm is shown in Section 5. Finally, a conclusion is drawn and future work is discussed in Section 6.

2. Related Work

In this section, related work is discussed. The section briefly reviews (1) the main approaches for sequential pattern mining, (2) the previous work of high-utility sequential pattern mining, and (3) state-of-the-art algorithms for sequential pattern mining with hierarchical relation.

2.1. Sequential Pattern Mining. Agrawal et al. [28] first presented a novel algorithm Apriori holding the download closure property for association rule mining. The proposed Apriori algorithm is based on a candidate generation approach that repeatedly scans the database to generate and count candidate sequential patterns and prunes those infrequent. They then defined the problem of sequential pattern mining over a large database of customer transactions and proposed the efficient algorithms AprioriSome and AprioriAll [4]. Srikant and Agrawal then proposed GSP, which is similar to AprioriAll in the execution process but greatly improves performance over AprioriAll. As Apriori's successor, adopting several technologies including time constraints, sliding time windows, and taxonomies, GSP uses a multiple-pass, candidate generation-and-test method to find sequential patterns [6]. Zaki [1] proposed the efficient SPADE which only needs three database scans. SPADE utilizes combinatorial properties to decompose the original problem into smaller subproblems, which can be independently solved in main memory using efficient lattice search techniques and using simple join operations. Later, SPAM was proposed by Ayres et al. [29], which applies to the situation that sequential patterns in the database are very long. In order to deal with the problems of large search spaces and the ineffectiveness in handling dense datasets, Yang et al. [30] proposed a novel algorithm LAPIN with a simple idea that the last position is the key to judging whether to extend the candidate sequential patterns or not. Then, they developed the LAPIN-SPAM algorithm by combining SPAM, which outperforms SPAM up to three times on all kinds of dataset in experiments. Notably, the property, used in SPAM, LAPIN, LAPIN-SPAM, and SPADE, that the support of superpatterns is always less than or equal to the support of its support patterns is different from the Apriori property used in GSP. Summarizing all algorithms mentioned above, they all belong to Apriori-based algorithms [2, 3].

It is known that database scans will be time-consuming when discovering sequential patterns. For this reason, a set of pattern growth sequential pattern mining algorithms that are able to avoid recursively scanning the input data were proposed. For example, Han et al. [31] proposed a novel, efficient algorithm FreeSpan that uses projected sequential databases to confine the search and growth of subsequences. The projected database can greatly reduce the size of a database. Pei et al. [7] designed a novel data structure called web access pattern tree, or WAP-tree in short, in their algorithm. WAP-tree stores highly compressed and critical information, and it makes mine access patterns from web logs efficiently. Then, Han et al. [32] proposed PrefixSpan with two kinds of database projections level-by-level projection and bilevel projection. PrefixSpan projects only their corresponding postfix subsequences into the projected database, so it runs considerably faster than both GSP and FreeSpan. Using a preorder linked, position-coded version of WAP-tree and providing a position code mechanism, the PLWAP algorithm was proposed by Ezeife et al. based on WAP-tree [33]. Recently, Sequence-Growth, the parallelized version of the PrefixSpan algorithm, was proposed by Liang and Wu [34], which adopts a lexicographical order to generate candidate sequences that avoid exhaustive search over the transaction databases.

There are some drawbacks to pattern growth sequential pattern mining algorithms. Obviously, it is time-consuming to build projected databases. Consequently, some algorithms with early pruning strategies were developed to improve efficiency. Chiu et al. [35] designed an efficient algorithm called DISC-all. Different with previous algorithms, DISC-all adopts the DISC strategy to prune the nonfrequent sequences according to the other sequences with the same length instead of the frequent sequences with shorter lengths. Recently, a more fast algorithm called CloFAST was proposed for mining closed sequential patterns using sparse and vertical id-lists. CloFAST combines a new data representation of the dataset, whose theoretical properties are studied in order to fast count the support of sequential patterns, with a novel one-step technique both to check sequence closure and to prune the search space [36]. It is more efficient than previous approaches. More details about the background of sequential pattern mining can be found in [2, 3].

2.2. High-Utility Sequential Pattern Mining. To address the problem that frequency does not fully reveal the importance in many situations, utility-oriented pattern mining frameworks, for example, high-utility itemset mining (HUIM), have been proposed and extensively studied [12, 37]. Although HUIM algorithms can extract interesting patterns in many real-life applications, they are not able to handle the sequence database where the timestamp is embedded in each item. Many high-utility sequential pattern mining algorithms have been proposed in the past decades [9, 13, 16, 18, 38], and high-utility sequential patterns can be extracted more efficiently with a series of novel data structures and

pruning strategies proposed. Ahmed et al. [13] first defined the problem of mining high-utility sequential patterns and proposed a novel framework for mining high-utility sequential patterns. They presented two new algorithms UL and US to find all high-utility sequential patterns. The UL algorithm, which is simpler and more straightforward, follows the candidate generation approach (based on breadth-first search), while the US algorithm follows the pattern growth approach (based on depth-first search). They can both be regarded as two-phase algorithms. In the first phase, they find a set of high-SWU sequences. In the second phase, they calculate the utility of sequences by scanning the sequence database to output high-SWU sequences, only those whose utility is no less than the threshold *minutil*.

The two-phase algorithms mentioned above have two important limitations, especially for low *minutil* values [8]. One limitation is that the set of high-SWU sequences discovered in the first phase needs a considerable amount of memory. The other one is that computing the utility of candidate sequences can be very time-consuming when scanning the sequence database. Instead of dividing the algorithm into two phases, Shie et al. [22] proposed a one-phase algorithm named UM-Span for high-utility sequential pattern mining. It improves efficiency by using a projected database-based approach to avoid additional scans of databases to check actual utilities of patterns. Similarly, a one-phase algorithm named PHUS was proposed by Lan and Hong [39], which adopted an effective upper bound model and an effective projection-based pruning strategy. Furthermore, the indexing strategy is also developed to quickly find the relevant sequences for prefixes in mining, and thus, unnecessary search time can be reduced.

Yin et al. then enriched the related definitions and concepts of high-utility sequential pattern mining. Two algorithms, USpan [9] and TUS [17], were proposed by Yin et al. for mining high-utility sequential patterns and top-k high-utility sequential patterns, respectively. In USpan, they introduced the lexicographic quantitative sequence tree to represent the search space and designed concatenation mechanisms for calculating the utility of a node and its children with two effective pruning strategies. The width pruning strategy avoids constructing unpromising patterns into the LP-Tree, while the depth pruning strategy stops USpan from going deeper by identifying the leaf nodes in the tree. Based on USpan, Alkan and Karagoz and Wang et al., respectively, proposed HuspExt [16] and HUS-Span [38] to increase efficiency of the mining process. Zhang et al. [18] proposed an efficient algorithm named FHUSpan (named HUS-UT in the paper), which adopts a novel data structure named Utility-Table to store the sequence database in the memory and the TRSU strategy to reduce search space. Recently, Gan et al. proposed two efficient algorithms named ProUM [40] and HUSP-ULL [41], respectively, to improve mining efficiency. The former utilizes the projection technique in generating utility array, while the latter adopts a lexicographic q-sequence- (LQS-) tree and a utility-linked-list structure to quickly discover HUSPs. More current development of HUSPM can be referred to in literature reviews [8, 14].

2.3. Sequential Pattern Mining with Hierarchical Relation. Sequential pattern mining with hierarchical relation can be traced back to article [6] where the hierarchies were incorporated into the mining process, and the GSP algorithm was proposed to extract sequential patterns according to different levels of hierarchy. There are two key strategies to improve efficiency in GSP. The first one is precomputing the ancestors of each item and dropping ancestors which are not in any of the candidates before making a pass over the data. The second strategy is to not count sequential patterns with an element containing both item and its ancestor. However, the depth of the hierarchy limits the efficiency of the algorithm because it increases the size of the sequence database. To represent the relationships among items in a more complete and natural manner, Chen and Huang [25] sketched the idea of fuzzy multilevel sequential patterns and presented the FMSM algorithm and the CROSS-FMSM algorithm based on GSP. Each item in hierarchies can have more than one parent with different degrees of confidence in their paper.

Plantevit et al. [24] incorporated the concept of hierarchy into a multidimensional database and proposed the two-phase algorithm HYPE extending their preceding M^2SP approach to extract multidimensional h-generalized sequential patterns. Firstly, the maximally specific items are extracted. Secondly, the multidimensional h-generalized sequences are mined in a further step. As the accessor of HYPE, they then proposed M^3SP [42] to extract multidimensional and multilevel sequential patterns based on M^2SP. The approaches are not incomplete; in other words, they do not mine all frequent sequences. Similarly applying fuzzy concepts to the hierarchy, Huang [43] later presented a divide-and-conquer strategy based on the pattern growth approaches to mine such fuzzy multilevel patterns. Recently, Egho then presented MMISP to extract heterogeneous multidimensional sequential patterns with hierarchical relation and applied it to analyze the trajectories of care for colorectal cancer. Beedkar et al. [23], who were inspired by MG-FSM, designed the first parallel algorithm named LASH for efficiently mining sequential patterns with hierarchical relation. MG-FSM first partitions the data and subsequently mines each partition independently and in parallel. Drawing lessons from the basic strategy of MG-FSM, Lash adopts a novel, hierarchy-aware variant of item-based partitioning, optimized partition construction techniques and an efficient special-purpose algorithm called pivot sequence miner (PSM) for mining each partition. As we know, the sequence database contains not only rich features (e.g., occurrence quantity, risk, and profit) but also multidimensional auxiliary information, which is partly associated with the concept of hierarchy. Recently, Gan et al. [44] proposed a novel framework named MDUS to extract multidimensional utility-oriented sequential useful patterns.

There are also several hierarchical frequent itemset mining algorithms, which are more or less similar to sequential pattern mining with hierarchical relations. For example, Kiran et al. [45] proposed a hierarchical clustering algorithm using closed frequent itemsets that use Wikipedia as an external knowledge to enhance the document representation. In Prajapati and Garg's research [46], the transactional dataset

is generated from a big sales dataset; then, the distributed multilevel frequent pattern mining algorithm (DMFPM) is implemented to generate level-crossing frequent itemset using the Hadoop Mapreduce framework. And then, the multilevel association rules are generated from frequent itemset.

3. Preliminaries and Problem Formulation

3.1. Definitions. Let I be a set of items. A nonempty subset $X \subseteq I$ is called an itemset, and the symbol $|X|$ denotes the size of X. A sequence $S: <X_1, X_2, \cdots, X_n>$ is an ordered list of itemsets, where $X_k \subseteq I$ $(1 \le k \le n)$. The length of S is $\sum_{k=1}^{n} |X_k|$, and the size of S is n. A sequence with the length of l is called an l-sequence. $T: <Z_1, Z_2, \cdots, Z_m>$ is the subsequence of S, if there exists m integers: $1 \le k_1 < k_2 < \cdots < k_m \le n$ so that $\forall 1 \le v \le m, Z_v \subseteq X_{k_v}$. For example, $<\{a\}\{b\,c\}>$ is the subsequence of $<\{a\,b\}\{a\,b\,c\}\{b\}>$.

A q-item (quantitative-item) is an ordered tuple (i, q), where $i \in I$ and q is a positive real number representing the quantity of i. A q-itemset with n q-items is denoted as $\{(i_1, q_1)(i_2, q_2) \cdots (i_n, q_n)\}$. A q-sequence, denoted as $<Y_1, Y_2, \cdots, Y_m>$, is an ordered list of m q-itemsets. A q-sequence database D (e.g., Figure 2(a)) consists of a collection of tuple $<\mathrm{ID}, Q>$, where ID is the identifier and Q is a q-sequence.

The hierarchical relation of different items is represented in the form of taxonomy which is a tree consisting of items in different abstraction levels. We assume that each item is only associated with one taxonomy. Figure 2(b) shows a simple example of taxonomies. In a taxonomy, if an item i is an ancestor of item j, we say that i is more general than j/j is more specific than i, denoted as $i \prec_I j / j \succ_I i$. We distinguish three different types of items: leaf items (most specific, no descendants), root items (most general, no ancestors), and intermediate items. The complete set consisting of descendants of item i is denoted as $\mathrm{down}(i)$. For example, in Figure 2(b), A is a root item, a_2 is an intermediate item, $a_{2,1}$ is a leaf item, and $\mathrm{down}(a_2) = \{a_{2,1}\, a_{2,2}\, a_{2,3}\}$. In this paper, we assume that different items belonging to the same itemset/q-itemset belong to different taxonomies.

Given two itemsets X and Y, we say that X is more specific than or equal to Y/Y is more general than or equal to X (denoted as $X \succcurlyeq_{\mathrm{IS}} Y / Y \preccurlyeq_{\mathrm{IS}} X$), iff $|X| = |Y|$ and $\forall i \in X, \exists j \in Y$ so that $i \succ_I j$ or $i = j$. For example, in Figure 2(b), $\{a_{1,1}\, b_1\, C\} \succcurlyeq_{\mathrm{IS}} \{a_1\, B\, C\}$; $\{A\, B\} \succcurlyeq_{\mathrm{IS}} \{A\, B\}$. Similarly, given two sequences with the size of m, S, and T, we say that S is more specific than or equal to T/T is more general than or equal to (denoted as $S \succcurlyeq_S T / T \preccurlyeq_S S$); if $\forall 1 \le k \le m$, we have $X \succcurlyeq_{\mathrm{IS}} Y$, where X is the kth itemset of S and Y is the kth itemset of T. In particular, if $S \succcurlyeq_S T$ and $S \ne T$, we say that S is more specific than T/T is more general than S, denoted as $S \succ_S T / T \prec_S S$. For example, in Figure 2(b), $<\{a_{1,1}\, b_1\, C\}\{A\, B\}> \succ_S <\{a_1\, B\, C\}\{A\, B\}>$.

3.2. Utility Calculation. Each item i is associated with an external utility (represented as $\mathrm{eu}(i)$) which is a positive real number representing the weight of i. For a nonleaf item i, it should meet the condition that $\mathrm{eu}(i) \ge \max\{\mathrm{eu}(j) \mid j \in \mathrm{down}(i)\}$. The external utility of each item $i \in I$ is recorded in an external utility table (e.g., Figure 2(c)).

The utility of a q-item (i, q) is defined as $q \times \mathrm{eu}(i)$. The utility of a q-itemset/q-sequence/q-sequence database is the sum of the utility of the q-items/q-itemset/q-sequence it contains. For example, in Figure 2, the utility of $a_{2,1}$ in the 1st itemset of Q_4 is 6 (1×6); the utility of the 1st itemset of Q_4 is 16 $(6 + 10)$; the utility of Q_4, represented as $u(Q_4)$, is 44 $(16 + 4 + 20 + 4)$; and the utility of D_E, represented as $u(D_E)$, is 228 $(74 + 39 + 71 + 44)$.

Given an itemset $X: \{i_1, i_2, \cdots, i_m\}$ and a q-itemset: $\{(j_1, q_1)(j_2, q_2) \cdots (j_n, q_n)\}(m \le n)$, we say that X occurs in Y, denoted as $X \subset_{\mathrm{IS}} Y$, iff there exist m distinct integers: $1 \le k_1, k_2, \cdots, k_m \le n$ so that $\forall 1 \le v \le m, i_v = j_{k_v}$ or $i_v \prec_I j_{k_v}$. The utility of X in Y is defined as $u(X, Y) = \sum_{v=1}^{m} q_{k_v} \times \mathrm{eu}(j_{k_v})$ if $X \subset_{\mathrm{IS}} Y$; otherwise, $u(X, Y) = 0$. For example, in Figure 2, let $Y_1 = \{(a_2, 2)(C, 2)\}$, $\{a_2\} \subset_{\mathrm{IS}} Y_1$; $\{A\, C\} \subset_{\mathrm{IS}} Y_1$; $\{A\, B\} \not\subset_{\mathrm{IS}} Y_1$; $u(\{A\}, Y_1) = 14(2 \times 7)$; $u(\{A\, C\}, Y_1) = 32(14 + 18)$; and $u(\{A\, B\}, Y_1) = 0$.

Given a sequence $S: <X_1, X_2, \cdots, X_m>$ and a q-sequence $Q: <Y_1, Y_2, \cdots, Y_n>$ where $m \le n$, we make the following definitions. We say that S occurs in Q (denoted as $S \subset_S Q$) at position p: $<k_1, k_2, \cdots, k_m>$ iff there exist m integers: $1 \le k_1 < k_2 < \cdots < k_m \le n$ so that $\forall 1 \le v \le m, X_v \subset_{\mathrm{IS}} Y_{k_v}$. The utility of S in Q at p, denoted as $u(S, p, Q)$, is defined as $u(S, p, Q) = \sum_{v=1}^{m} u(X_v, Y_{k_v})$. For example, in Figure 2, $S_1: <\{A\}\{A\, C\}>$ occurs in Q_1 at position $<1, 3>$; $u(S_1, <1, 3>, Q_1) = 8 + 32 = 40$.

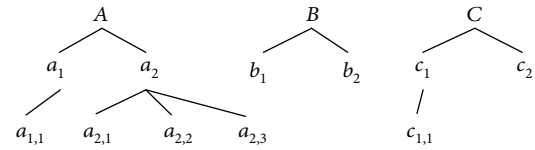
Obviously, S may occur in Q many times. The utility of S in Q, denoted as $u(S, Q)$, is defined as $u(S, Q) = \max\{u(S, p, Q) \mid p \in P(S, Q)\}$, where the symbol $P(S, Q)$ denotes the complete set containing all positions of S in Q. The utility of S in a q-sequence database D, denoted as $u(S)$, is defined as $u(S) = \sum_{Q \in D \wedge S \subset_S Q} u(S, Q)$. For example, in Figure 2, $S_2: <\{a_1\}\{C\}>$ occurs in Q_1 three times; $P(S_2, Q_1) = \{<1, 3>, <1, 4>, <1, 5>\}$; $u(S_2, <1, 3>, Q_1) = 8 + 18 = 26$; $u(S_2, Q_1) = \max\{26, 13, 16\} = 26$; $u(S_2) = 26 + 31 = 57$. More details about the methods of utility calculation can be found in [8].

Given a minimum utility ξ, we say that sequence W is high-utility if $u(W) \ge \xi$. In particular, W is the most specific pattern, denoted as s-sequence, iff $u(W) \ge \xi$ and $\forall T \succ_S W$, $u(T) < \xi$. Similarly, sequence S is the most general pattern, denoted as g-sequences, iff $u(S) \ge \xi$ and $\neg \exists T \prec_S S$. The s-sequences contain the underlying informative knowledge of hierarchical relations between different items, which cover less meaningless information compared with those sequences highly generalized. Therefore, we define these s-sequences as high-utility hierarchical sequential patterns (HUHSPs) to be extracted.

3.2.1. Problem Statement. Given a minimum utility ξ, a utility hierarchical sequence database including a quantitative sequential database D, a set of taxonomies, and an external utility table, the utility-driven mining problem of high-utility hierarchical sequential pattern mining (HUHSPM) consists of enumerating all HUHSPs whose overall utility

ID	q-sequence
Q_1	$<\{(a_{1,1}, 1)\}\ \{(a_2, 3)\}\ \{(a_2, 2)(C, 2)\}\ \{(c_{1,1}, 1)\}\ \{(c_2, 4)\}>$
Q_2	$<\{(a_{1,1}, 2)\}\ \{(a_2, 1)\}\ \{(b_2, 2)\}\ \{(b_1, 4)\}>$
Q_3	$<\{(a_{1,1}, 2)\}\ \{(c_{1,1}, 3)\}\ \{(A, 4)\}>$
Q_4	$<\{(a_{2,1}, 1)(c_{1,1}, 2)\}\ \{(a_{2,2}, 2)\}\{(c_{1,1}, 4)\}\ \{(a_{2,3}, 4)\}>$

(a) Quantitative-sequence database DE



(b) Taxonomies

Item	$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	a_1	a_2	A	b_1	b_2	B	$c_{1,1}$	c_1	c_2	C
eu	8	6	2	1	8	7	10	3	2	3	5	5	2	9

(c) External utility table

Figure 2: An example of a hierarchical utility sequence database.

values in this database are no less than the prespecified minimum utility account ξ.

4. Proposed HUHSPM Algorithm: MHUH

In this section, we present the proposed algorithm MHUH for HUHSPM. We incorporate the hierarchical relation of items into high-utility sequential pattern mining, which makes MHUH able to find the underlying informative knowledge of hierarchical relation between different items ignored in high-utility sequential pattern mining. In other words, MHUH can extract more interesting patterns. The mining process of MHUH mainly includes two phases named Extension and Replacement. MHUH finds high-utility sequences by the existing algorithm FHUSpan (also named HUS-UT) which we proposed earlier based on the prefix-extension approach in the first phase. For a g-sequence S, we then generate all sequences that are more specific than S by progressive replacement and store s-sequences with a collection G in the second phase. The work we need to do in the two phases can be observed visually from the two names. The mining process with two phases ensures that the underlying informative knowledge of hierarchical relation between different items will not be missed. At the same time, it can increase efficiency when discovering HUHSPs.

Without the loss of generality, in this section, we formalize the theorems under the context of a minimum utility ξ and a utility hierarchical sequence database (includes a q-sequence database D, taxonomies, and external utility table).

4.1. Reduction: Remove Useless Items. Before mining the sequential patterns, MHUH adopts the Reduction strategy in the data preprocessing procedure, which removes useless items to reduce search space in advance. It mainly consists of two points, removing the unpromising items from the q-sequence database and removing the redundant items from the taxonomies.

An item is unpromising if any sequence containing this item is not high-utility. Here, we propose a novel upper bound TSWU (Taxonomy Sequence-Weighted Utility) based on SWU [13] to filter out the unpromising items.

Definition 1. Given an item i, we define $\mathrm{TSWU}(i)$ as $\sum_{Q\in D\wedge<\{r\}>\subset_S Q} u(Q)$, where r is the root item of the taxonomy containing i.

For example, in Figure 2, $\mathrm{TSWU}(a_{1,1}) = \mathrm{TSWU}(A) = 228$;$\mathrm{TSWU}(b_1) = \mathrm{TSWU}(b_2) = \mathrm{TSWU}(B) = 39$; $\mathrm{TSWU}(c_2) = \mathrm{TSWU}(C) = 189$.

Theorem 2. *Given a q-sequence Q, two sequences S_1 and S_2, where $S_1 \succ_S S_2$, $P(S_1, Q) \subseteq P(S_2, Q)$.*

Proof. Let $S_1 : <X_1, X_2, \cdots, X_m>, S_2 : <Z_1, Z_2, \cdots, Z_m>$. We have $\forall 1 \le v \le m, Z_v \preccurlyeq_{\mathrm{IS}} X_v$. For a q-sequence $Q : <Y_1, Y_2, \cdots, Y_n> (m \le n)$, $\forall p : <k_1, k_2, \cdots, k_m> \in P(S_1, Q)$, $Z_v \preccurlyeq_{\mathrm{IS}} X_v \subset_{\mathrm{IS}} Y_{j_v}$ $(1 \le v \le m)$, so $p \in P(S_2, Q)$. Further, $P(S_1, Q) \subseteq P(S_2, Q)$.

Theorem 3. *For any sequence S that contains item i, $u(S) < \xi$ if $TSWU(i) < \xi$.*

Proof. From Theorem 3, we know that $P(<\{i\}>, Q) \subseteq P(<\{r\}>, Q)$, so $\{Q \mid Q \in D \wedge <\{i\} > \subset_S Q\} \subseteq \{Q \mid Q \in D \wedge <\{r\} > \subset_S Q\}$. If $\mathrm{TSWU}(i) < \xi$, $u(S) = \sum_{Q\in D\wedge S\subset_S Q} u(S, Q) \le \sum_{Q\in D\wedge S\subset_S Q} u(Q) \le \sum_{Q\in D\wedge<\{i\}>\subset_S Q} u(Q) \le \sum_{Q\in D\wedge<\{r\}>\subset_S Q} u(Q) = \mathrm{TSWU}(i) < \xi$.

For a given ξ, we can remove items satisfying $\mathrm{TSWU}(i) < \xi$ safely according to the above theorem. For example, in Figure 2, when $\xi = 40$, b_1 and b_2 can be safely removed from Figure 2(a).

We say that an item is redundant if it (1) appears in taxonomy but does not appear in q-sequence database and (2) has at most one child in taxonomy. For example, in Figure 2, a_1 and c_1 are redundant items. In terms of utility, removing these items has no effect on correctness, which will be proved in Subsection 4.3. Thus, we can safely remove these items.

4.2. Extension (Phase I): Find g-Sequences. In the first phase named Extension, we use the existing algorithm FHUSpan [18] which we proposed earlier to efficiently mine the general high-utility sequences (g-sequences). The main tasks of this phase are improving efficiency greatly of MUHU and extract g-sequences preparing for the next phase.

In fact, no s-sequences will be missed based on the FGS (From General to Special) strategy. To prove the correctness of this conclusion, we need to prove two points: (1) there does not exist s-sequence S that cannot be discovered by the FGS strategy and (2) the correctness of the algorithm that finds s-sequence is based on a given g-sequence. Here, we prove the correctness of (1), and the proof about (2) is illustrated in the next subsection.

Theorem 4. *Given two sequences S_1 and S_2 where $S_1 \succ_S S_2$, $u(S_1) \leq u(S_2)$.*

Proof. We first prove that $u(S_1, Q) \leq u(S_2, Q)$. From Section 4.2 ($\text{eu}(i) \geq \max\{\text{eu}(j) \mid j \in \text{down}(i)\}$) and Theorem 3, $u(S_2, Q) = \max\{u(S_2, p, Q) \mid p \in P(S_2, Q)\} \geq \max\{u(S_1, p, Q) \mid p \in P(S_1, Q)\} = u(S_1, Q)$. Then, we prove $u(S_1) \leq u(S_2)$. We have $D_1 \subseteq D_2$ if $S_1 \succ_S S_2$, where $D_k(k = 1, 2) = \{Q \mid Q \in D \wedge S_k \subset_S Q\}$. $u(S_2) = \sum_{Q \in D_2 \setminus D_1} u(S_2, Q) + \sum_{Q \in D_1} u(S_2, Q) \geq \sum_{Q \in D_1} u(S_2, Q) \geq \sum_{Q \in D_1} u(S_1, Q) = u(S_1)$.

Corollary 5. *Given a g-sequence S, $\forall T \succ_S S, u(T) \leq u(S)$.*

Theorem 4 and Corollary 5 reveal the correctness of (1). We assume that S is a s-sequence that cannot be discovered by the FGS strategy. In fact, we can always find (replace item with the item's ancestor) the sequence S_g where $S_g \preccurlyeq_S S$ and $\neg\exists T \prec_S S$. Because S_g is not g-sequence, $u(S_g) < \xi$ or $\exists T \prec_S S$. So, $u(S_g) < \xi$. We then draw a contraction that $S_g \preccurlyeq_S S \wedge u(S_g) < u(S)$. Therefore, the assumption does not hold, which ensures the correctness of (1).

Theorem 6. *All items contained in g-sequence are root items.*

Proof. Given a g-sequence S, we assume that the ith item of S is not the root item. Then, we can find a sequence T where $T \prec_S S$. We then draw a contraction that S is not g-sequence. Therefore, the theorem holds.

We then introduce how to find g-sequences. Theorem 6 shows that we merely need to consider the root items in the process of finding g-sequences. Thus, we can transform the q-sequences into another form so that we can ignore the hierarchical relation in this phase. We illustrate this transformation through an example. Consider Q_3 in Figure 2, we transform it into $<\{A[16]\}\{C[15]\}\{A[40]\}>$, where the value in the bracket is utility ($\text{eu} \times \text{iu}$). Obviously, with this transformation, mining g-sequence is equivalent to mining high-utility sequences. So, we use the existing high-utility sequential pattern mining algorithm FHUSpan [18], which we proposed earlier to find g-sequences.

Here, we briefly introduce the mining process of FHUSpan, which finds high-utility sequences based on the prefix-extension approach. It first finds all appropriate items (only the sequence starting with these items may be high-utility). Then, for each appropriate item, it constructs a sequence containing only this item and extends the sequence recursively until all sequences starting with the item are checked. In particular, two extension approaches are used, S-Extension (appending an itemset containing only one item to the end of the current sequence) and I-Extension (appending an item to the last itemset of the current sequence). It is based on the algorithm HUS-Span which uses two pruning strategies, PEU (Prefix Extension Utility) strategy and RSU (Reduced Sequence Utility) strategy to reduce the search space. The novel data structure named Utility-Table and the pruning are used to terminate extension in FHUSpan so that it can efficiently discover high-utility sequences.

4.3. Replacement (Phase II): Find s-Sequence. In the second phase named Replacement, we mine the special high-utility sequences with the hierarchical relation (s-sequences) from g-sequences by the PBS strategy. The main task of this phase is to extract s-sequences efficiently.

For a g-sequence S, we then generate all sequences that are more specific than S by progressive replacement and store s-sequences with a collection G. In particular, for each replacement, we replace the kth item of S with a child item. For example, in Figure 2, we replace the first item of $S_3 : <\{A\}\{A\}\{C\}>$ with the child items of A, and one specific sequence is $S_4 : <\{a_2\}\{A\}\{C\}>$.

Algorithm 1 shows the progressive replacement starting from the kth item of a sequence, which is based on DFS. Firstly, it checks if the current sequence S has been visited to avoid repeated utility calculation (line 1). If $u(S) < \xi$, we have $\forall S' \succ_S S, u(S') < \xi$ according to Theorem 4, so we terminate search (lines 2-3). Otherwise, it adds S into G and removes the sequences that are more general than S from G (line 5). Then, it generates the more specific sequences based on S, which follows the order from top to bottom (line 9), left to right (lines 10-12). In detail, it first finds $R(S, k)$ which is the set containing all child items for replacement. For each $r \in R(S, k)$, it replaces the ith item of S with r to generate S'. It then checks the sequences that are more specific than S' (line 9, from top to bottom). After that, it checks the sequences that are more specific than S' from left to right (lines 10-12), where l is the length of S'.

We also use a strategy, PBS (Pruning Before Search), to reduce search space before Algorithm 1. The main idea behind this strategy is considering only the items in the current index. In other words, we generate and check the more specific sequences in one direction (from top to bottom) to reduce the size of taxonomies.

We illustrate this strategy through an example under the context of Figure 2. Let $\xi = 45$, the sequence $S_3 : <\{A\}\{A\}\{C\}>$ is a g-sequence ($u(S_3) = 77 > \xi$). We construct copies of taxonomy, denoted as T_1, T_2, and T_3, for the kth ($k = 1, 2, 3$) item of S_3. Then, we reduce the size of the three taxonomies. For T_1, we have $R(S_3, 1) = \{a_{1,1}, a_2\}$ (a_1 is a redundant item and was removed). Then, we generate $S_5 : <\{a_{1,1}\}\{A\}\{C\}>$ by replacing the first A with $a_{1,1}$, and $u(S_5) = 47 > \xi$. So, we retain $a_{1,1}$. Because $R(S_5, 1) = \varnothing$, we then consider a_2 and generate $S_6 : <\{a_2\}\{A\}\{C\}>$. We also retain a_2, for $u(S_6) = 73 > \xi$. Then, we continue to check the child items of a_2. Such a procedure will continue until all items belonging to down(A) have been checked. Finally, we

```
Search for specific.
Input: S: the sequence, k: start index, visited, G
1:  if visitedisfalse then
2:    if u(S) < ξ then
3:      return
4:    end if
5:      G ⟵ Filter(G ∪ S)
6:  end if
7:  for all r ∈ R(S,k) do
8:      S' ⟵ Replace(S,k,r)
9:      SearchForSpecific(S ,k,false,G)
10:   for v = k +1 ⟶ l do
11:       SearchForSpecific(S ,v,true,G)
12:   end for
13: end for
```

Algorithm 1

remove $a_{2,1}, a_{2,2}, a_{2,3}$ from T_1, for $u(<\{a_{2,1}\}\{A\}\{C\}>) = 30 < \xi$, $u(<\{a_{2,2}\}\{A\}\{C\}>) = u(<\{a_{2,3}\}\{A\}\{C\}>) = 0 < \xi$. We then continue the above procedure for T_2 and T_3, and the processed taxonomies are shown in the right of Figure 3. In addition, note that in Algorithm 1, $R(S, k)$ is obtained from the processed taxonomies instead of the original taxonomies.

In the above example, the max count of sequences that are more specific than S reduces from 107 $(6 \times 6 \times 3 - 1)$ to 11 $(3 \times 2 \times 2 - 1)$. In fact, for a l-sequence S, this count reduces from $\prod_{k=1}^{l}(d_k^1 + 1) - 1$ to $\prod_{k=1}^{l}(d_k^2 + 1) - 1$, where d_k^1 and d_k^2 are the sizes of down(i_k) in the original and processed taxonomies, respectively, and i_k is the kth item of S.

In the rest of this subsection, we prove the conclusion left before. We first prove that removing redundant items has no effect on correctness.

Proof. For a sequence S, we assume that the kth item of S, i_k, is a redundant item. Firstly, if i_k is a leaf item, we can safely remove it, because for each sequence W that contains i_k, we have $u(W) = 0$. Secondly, if i_k has one child, we generate sequence W by replacing i_k with its child. Then, we have $u(W) = u(S)$ according to the related utility definition (the utility of X in Y). Therefore, removing redundant items does not change the utility of related sequences, which means that it has no effect on the correctness.

Then, we prove the conclusion the correctness of the algorithm which finds s-sequence based on a given g-sequence.

Proof. Firstly, the PBS strategy does not ignore the underlying s-sequences. Suppose we cannot find a s-sequence S from the taxonomies processed by PBS strategy, then we have $\exists S_g \prec_S S, u(S_g) < u(S)$, which violates Theorem 4. So, the assumption does not hold. Secondly, Algorithm 1 does not ignore any s-sequences. Algorithm 1 is based on the DFS framework, which ensures the completeness of the algorithm. Besides, Algorithm 1 terminates search in advance based on Theorem 4, so it does not ignore any s-sequences. In summary, the conclusion holds.

5. Experiments

We performed experiments to evaluate the proposed MHUH algorithm which was implemented in Java. All experiments were carried out on a computer with Intel Core i7 CPU of 3.2 GHz, 8 GB memory, and Windows 10.

5.1. Datasets. Five datasets, including three real datasets and two synthetic datasets, were used in the experiments. DS1 is the conversion of Bible where each word is an item. DS2 is the conversion of the classic novel called Leviathan. DS3 is a click-stream dataset called BMSWebView2. The three datasets can be obtained from the SPMF website [47]. DS4 and DS5 are two synthetic datasets. The characteristic of them is summarized in Table 1. The values of parameters in Table 1 are as follows: $\#S$ is the number of sequences, $\#I$ is the number of distinct items, M is the max length of sequences, and A is the average length of sequences.

Note that these datasets do not contain taxonomies. So, for each dataset, we generated taxonomies based on the items it contains. The max depth and degree of these taxonomies are 3, which indicates that the max number of leaf items contained in taxonomy is 27. The datasets and source code will be released at the author's Github after the acceptance for publication.

5.2. Performance Evaluation. We evaluated the performance of the proposed algorithm on different datasets when varying ξ. For the sake of simplicity, here, we calculate ξ as $\delta \times u(D)$, where δ is a decimal between 0 and 1, and $u(D)$ is the utility of the q-sequence database (see the concepts in Subsection 3.2). In addition, we also tested the effect of the PBS strategy, and the modified MHUH algorithm which does not take the PBS strategy is denoted as MHUH_base.

The execution times of MHUH and MHUH_base on DS1 to DS3 are shown in Figure 4. When δ/ξ increases, both of the two algorithms take less execution time since the search space reduces. The results prove that the PBS strategy effectively decreases the execution time, for it greatly reduces the search space on these datasets. Besides, the results also show that the MHUH algorithms can efficiently extract s-sequences under a low ξ.

Figure 5 shows the distribution of discovered patterns by MHUH on DS1 to DS3. It shows that the number of patterns per length increases with the decrease of ξ. In particular, it is interesting that some longer patterns may disappear as ξ increases, which indicates that the shorter patterns may have higher utility.

5.3. Utility Comparison with High-Utility Sequential Pattern Mining. We conducted this experiment to evaluate the utility difference between the patterns discovered by MHUH and that discovered by the existing algorithm FHUSpan [18] which we proposed earlier.

Figure 6 shows the sum utility of top # (depends on utility) patterns discovered by FHUSpan and MHUH from three datasets. The X-axis refers to the value of #, and the Y-axis

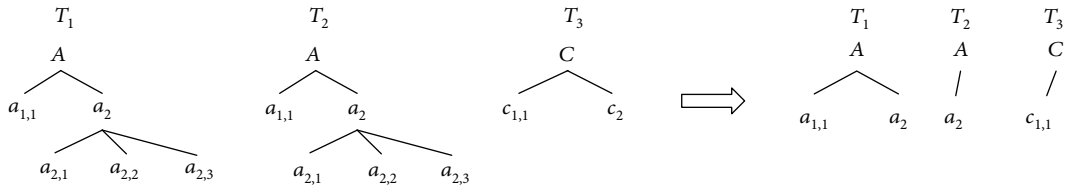


Figure 3: Reduce the size of T_1, T_2, and T_3 through the PBS strategy.

Table 1: Characteristic of datasets.

Dataset	#S	#I	M	A	Type
DS1	36,369	13,905	100	21.64	Real (text)
DS2	5,834	9,162	100	33.81	Real (text)
DS3	77,512	6,120	161	4.62	Real (click stream)
DS4	10,000	4,000	40	20.54	Synthetic
DS5	60,000	5,000	20	10.50	Synthetic

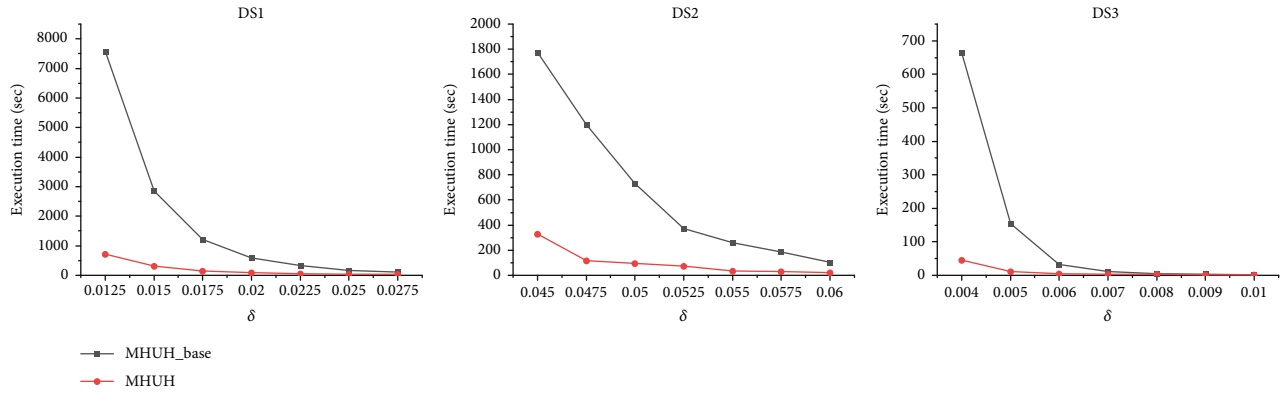


Figure 4: Execution time on three datasets when varying δ.

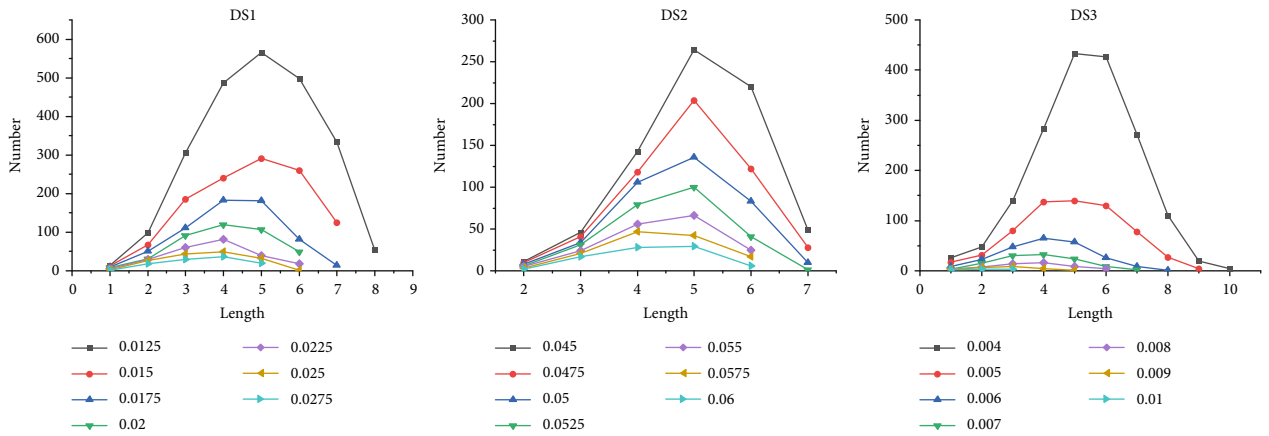


Figure 5: Distribution of discovered patterns on three datasets when varying δ.

represents the sum utility of top # patterns. For example, on DS1, the sum utility of top 1000 patterns extracted by MHUH is higher than the sum utility of that discovered by FHUSpan. Figure 7 shows that the average utility per length of top # patterns on DS1 to DS3 (# is set to 1000, 700, and 600, respectively). The X-axis refers to the length of patterns, and the Y-axis denotes the average utility of patterns with the same length. For example, on DS1, in terms of the top 1000 patterns, the average utility of patterns with length of 8 discovered by MHUH is higher than the average utility of that discovered by FHUSpan. From these two figures, we know that MHUH can discover higher utility patterns compared

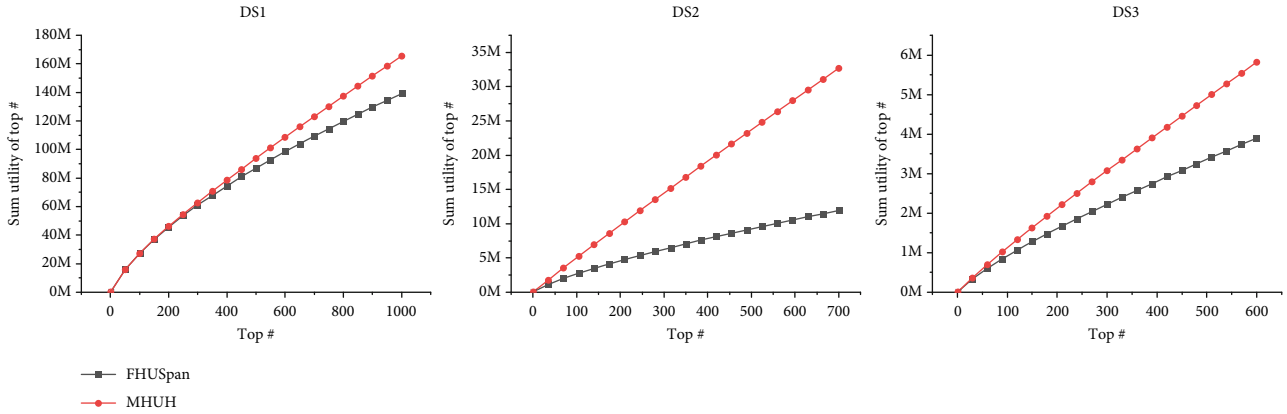


Figure 6: Sum utility of top # patterns discovered from three datasets.

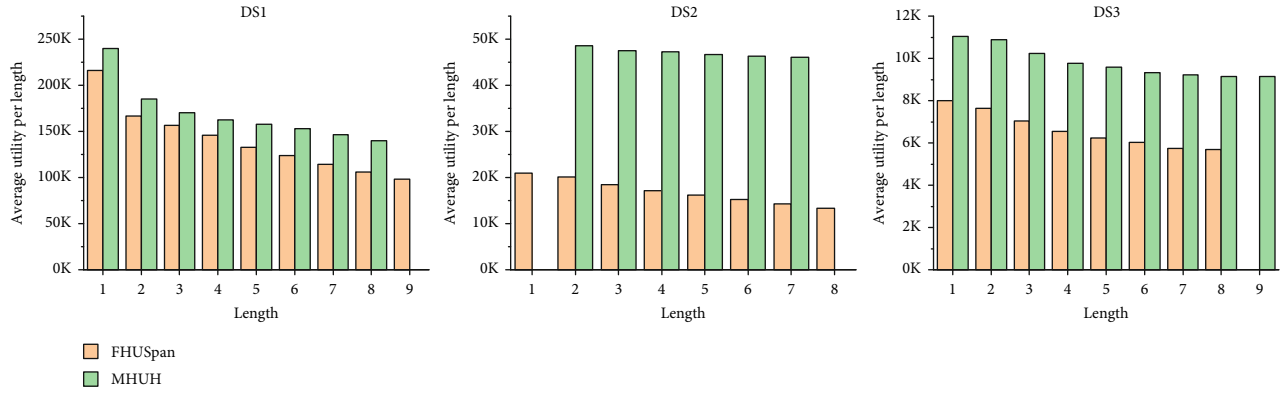


Figure 7: Average utility per length of top # patterns.

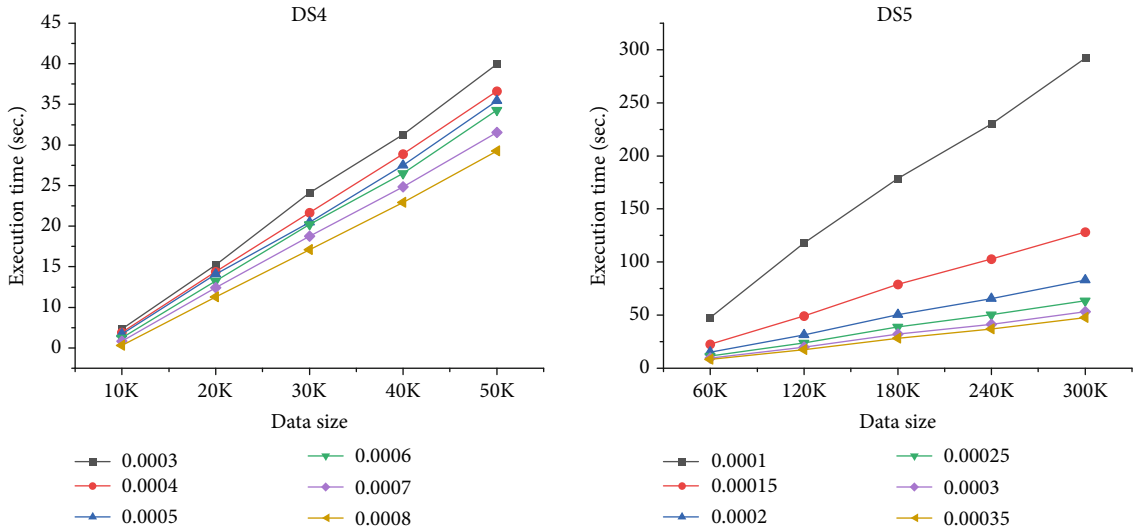


Figure 8: Scalability test on two datasets.

with FHUSpan, indicating that more informative knowledge can be found by MHUH.

5.4. Scalability. We conducted experiments to evaluate MHUH's performance on large-scale datasets. For each dataset, we increased its data size through duplication and performed the MHUH algorithm with different δ. Figure 8 shows the experimental results. We know from the figure that the MHUH algorithm has well scalability on the two datasets, for the execution time is almost linear with the data

size. For example, the execution time of MHUH ($\delta = 0.0003$) on DS4 almost linearly increases when the data size (the number of q-sequences it contains) changes from 10K to 50K. It also shows that MHUH can efficiently identify the desired patterns from the large-scale dataset with a low ξ. For example, in terms of DS5, MHUH costs 300 s when the data size is 300K and $\delta = 0.0001$.

6. Conclusion and Future Work

In this paper, we incorporate the hierarchical relation of items into high-utility sequential pattern mining and propose a two-phase algorithm MHUH, the first algorithm for high-utility hierarchical sequential pattern mining (HUHSPM). In the first phase named Extension, we use the existing algorithm FHUSpan which we proposed earlier to efficiently mine the general high-utility sequences (g-sequences); in the second phase named Replacement, we mine the special high-utility sequences with the hierarchical relation (s -sequences) from g-sequences. The proposed MHUH algorithm takes several novel strategies (e.g., Reduction, FGS, and PBS) and a new upper bound TSWU, so it will be able to greatly reduce the search space and discover the desired pattern HUHSPs efficiently. A conclusion can be drawn from the experiment that MHUH extracts more interesting patterns with underlying informative knowledge efficiently in HUHSPM.

In the future, we will generalize the proposed algorithm based on the more complete concepts. Besides, several extensions of the proposed MHUH algorithm can be considered such as improving the efficiency of the MHUH algorithm based on better pruning strategies, efficient data structures [40, 41], and the multithreading technology [2].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors have declared that no conflict of interest exists.

Acknowledgments

This work was supported by the Natural Science Foundation of Guangdong Province, China (Grant No. 2020A15150109 70) and Shenzhen Research Council (Grant No. GJHZ2018 0928155209705).

References

[1] M. J. Zaki, "Spade: an efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, no. 1/2, pp. 31–60, 2001.

[2] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "A survey of parallel sequential pattern mining," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, pp. 1–34, 2019.

[3] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54–77, 2017.

[4] R. Agrawal and R. Srikant, "Mining sequential patterns," in *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, vol. 95, pp. 3–14, Taipei, Taiwan, China, March 1995.

[5] H. Lu, Y. Li, S. Mu, D. Wang, H. Kim, and S. Serikawa, "Motor anomaly detection for unmanned aerial vehicles using reinforcement learning," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2315–2322, 2018.

[6] R. Srikant and R. Agrawal, "Mining sequential patterns: generalizations and performance improvements," in *International Conference on Extending Database Technology*, pp. 1–17, Springer, Berlin, Heidelberg, 1996.

[7] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu, "Mining access patterns efficiently from web logs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 396–407, Springer, Berlin, Heidelberg, 2000.

[8] T. Truong-Chi and P. Fournier-Viger, "A survey of high utility sequential pattern mining," in *High-Utility Pattern Mining: Theory, Algorithms and Applications*, pp. 97–129, Springer, 2019.

[9] J. Yin, Z. Zheng, and L. Cao, "Uspan: an efficient algorithm for mining high utility sequential patterns," in *18th ACM SIGKDD International conference on Knowledge discovery and data mining*, pp. 660–668, Beijing, China, August 2012, ACM.

[10] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain intelligence: go beyond artificial intelligence," *Mobile Networks and Applications*, vol. 23, no. 2, pp. 368–375, 2018.

[11] W. Gan, J. Chun-Wei, H.-C. Chao, S.-L. Wang, and S. Y. Philip, "Privacy preserving utility mining: a survey," in *2018 IEEE International Conference on Big Data*, pp. 2617–2626, Seattle, WA, USA, Nov 2018, IEEE.

[12] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, T.-P. Hong, and H. Fujita, "A survey of incremental high-utility itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, article e1242, 2018.

[13] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "A novel approach for mining high-utility sequential patterns in sequence databases," *ETRI Journal*, vol. 32, no. 5, pp. 676–686, 2010.

[14] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. S. Tseng, and P. S. Yu, "A survey of utility-oriented pattern mining," 2018, http://arxiv.org/abs/1805.10511.

[15] T. Wang, X. Ji, A. Song et al., "Output bounded and rbfnn-based position tracking and adaptive force control for security tele-surgery," *ACM Transactions on Multimedia Computing Communications and Applications*, ACM, 2020.

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

[17] J. Yin, Z. Zheng, L. Cao, Y. Song, and W. Wei, "Efficiently mining top-k high utility sequential patterns," in *2013 IEEE 13th International conference on data mining*, pp. 1259–1264, Dallas, TX, USA,, 2013, IEEE.

[18] C. Zhang, Z. Yiwen, J. Nie, and D. Zilin, "Two efficient algorithms for mining high utility sequential patterns," in *17th*

IEEE International Symposium on Parallel and Distributed Processing with Applications, Xiamen, China, 2019IEEE.

[19] H. Lu, D. Wang, Y. Li et al., "Conet: a cognitive ocean network," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 90–96, 2019.

[20] X. Yang, H. Wu, Y. Li et al., "Dynamics and isotropic control of parallel mechanisms for vibration isolation," in *IEEE/ASME Transactions on Mechatronics*, IEEE, 2020.

[21] C. F. Ahmed, S. K. Tanbeer, and B.-S. Jeong, "Mining high utility web access sequences in dynamic web log data," in *2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 76–81, London, UK, June 2010, IEEE.

[22] B.-E. Shie, J.-H. Cheng, K.-T. Chuang, and V. S. Tseng, "A one-phase method for mining high utility mobile sequential patterns in mobile commerce environments," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pp. 616–626, Berlin, Heidelberg, 2012.

[23] K. Beedkar and R. Gemulla, "Lash: large-scale sequence mining with hierarchies," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 491–503, Melbourne, Australia, May 2015, ACM.

[24] M. Plantevit, A. Laurent, and M. Teisseire, "Hype: mining hierarchical sequential patterns," in *Proceedings of the 9th ACM International workshop on Data warehousing and OLAP*, pp. 19–26, Arlington, Virginia, USA, November 2006, ACM.

[25] Y.-L. Chen and T. C.-K. Huang, "A novel knowledge discovering model for mining fuzzy multilevel sequential patterns in sequence databases," *Data & Knowledge Engineering*, vol. 66, no. 3, pp. 349–367, 2008.

[26] R. Srikant and R. Agrawal, "Mining generalized association rules," *Future Generation Computer Systems*, vol. 13, no. 2-3, pp. 161–180, 1997.

[27] L. V. Q. Anh and M. Gertz, "Mining spatio-temporal patterns in the presence of concept hierarchies," in *2012 IEEE 12th International Conference on Data Mining Workshops*, pp. 765–772, Brussels, Belgium, Dec. 2012, IEEE.

[28] R. Agrawal R. Srikant et al., "Fast algorithms for mining association rules," in *Proceedings of the 20th VLDB Conference*, vol. 1215, pp. 487–499, Santiago, 1994.

[29] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *Proceedings of the eighth ACM SIGKDD International conference on Knowledge discovery and data mining*, pp. 429–435, Edmonton Alberta, Canada, July 2002, ACM.

[30] Z. Yang, Y. Wang, and M. Kitsuregawa, "Lapin: effective sequential pattern mining algorithms by last position induction for dense databases," in *Advances in Databases: Concepts, Systems and Applications*, pp. 1020–1023, Springer, 2007.

[31] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu, "Freespan: frequent pattern-projected sequential pattern mining," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 355–359, Boston Massachusetts USA, August 2000.

[32] J. Han, J. Pei, B. Mortazavi-Asl et al., "Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth," in *Proceedings of the 17th international conference on data engineering*, pp. 215–224, Heidelberg, Germany, Germany, April 2001.

[33] C. I. Ezeife, Y. Lu, and Y. Liu, "Plwap sequential mining: open source code," in *Proceedings of the 1st International workshop on open source data mining: frequent pattern mining implementations*, pp. 26–35, Chicago Illinois USA, August 2005.

[34] Y.-h. Liang and W. Shiow-yang, "Sequence-growth: a scalable and effective frequent itemset mining algorithm for big data based on mapreduce framework," in *2015 IEEE International Congress on Big Data*, pp. 393–400, New York, NY, USA, Jun 2015.

[35] D.-Y. Chiu, Y.-H. Wu, and A. L. P. Chen, "An efficient algorithm for mining frequent sequences by a new strategy without support counting," in *Proceedings of the 20th International Conference on Data Engineering*, pp. 375–386, Boston, MA, USA, USA, April 2004.

[36] F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba, "Clofast: closed sequential pattern mining using sparse and vertical id-lists," *Knowledge and Information Systems*, vol. 48, no. 2, pp. 429–463, 2016.

[37] C. F. Ahmed, S. K. Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

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

[39] G.-C. Lan, T.-P. Hong, V. S. Tseng, and S.-L. Wang, "Applying the maximum utility measure in high utility sequential pattern mining," *Expert Systems with Applications*, vol. 41, no. 11, pp. 5071–5081, 2014.

[40] W. Gan, J. C.-W. Lin, J. Zhang, H.-C. Chao, H. Fujita, and P. S. Yu, "Proum: projection-based utility mining on sequence data," *Information Sciences*, vol. 513, pp. 222–240, 2020.

[41] W. Gan, J. C.-W. Lin, J. Zhang, P. Fournier-Viger, H.-C. Chao, and P. S. Yu, "Fast utility mining on sequence data," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[42] M. Plantevit, A. Laurent, D. Laurent, M. Teisseire, and Y. W. E. I. Choong, "Mining multidimensional and multilevel sequential patterns," *ACM Transactions on Knowledge Discovery from Data*, vol. 4, no. 1, pp. 1–37, 2010.

[43] T. C.-K. Huang, "Developing an efficient knowledge discovering model for mining fuzzy multi-level sequential patterns in sequence databases," *Fuzzy Sets and Systems*, vol. 160, no. 23, pp. 3359–3381, 2009.

[44] W. Gan, J. C.-W. Lin, J. Zhang et al., "Utility mining across multi-dimensional sequences," 2019, http://arxiv.org/abs/1902.09582.

[45] G. V. R. Kiran, R. Shankar, and V. Pudi, "Frequent itemset based hierarchical document clustering using Wikipedia as external knowledge," in *International Conference on Knowledge-based and Intelligent Information and Engineering Systems*, pp. 11–20, Cardiff, Wales, UK, September 2010.

[46] D. J. Prajapati and S. Garg, "Map reduce based multilevel association rule mining from concept hierarchical sales data," in *International Conference on Advances in Computing and Data Sciences*, pp. 624–636, Springer, Singapore, July 2017.

[47] P. Fournier-Viger, "SPMF: an open-source data mining library," June 2019, http://www.philippe-fournier-viger.com/spmf/.