symmetry

MDPI

Article

An Efficient Algorithm for Mining Stable Periodic High-Utility Sequential Patterns

Shiyong Xie and Long Zhao *

Department of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China
* Correspondence: zhaolong@qlu.edu.cn

Abstract: Periodic high-utility sequential pattern mining (PHUSPM) is used to extract periodically occurring high-utility sequential patterns (HUSPs) from a quantitative sequence database according to a user-specified minimum utility threshold (*minutil*). A sequential pattern's periodicity is determined by measuring when the frequency of its periods (the time between two consecutive happenings of the sequential pattern) exceed a user-specified maximum periodicity threshold (*maxPer*). However, due to the strict judgment threshold, the traditional PHUSPM method has the problem that some useful sequential patterns are discarded and the periodic values of some sequential patterns fluctuate greatly (i.e., are unstable). In frequent itemset mining (FIM), some researchers put forward some strategies to solve these problems. Because of the symmetry of frequent itemset pattern (FIPs), these strategies cannot be directly applied to PHUSPM. In order to address these issues, this work proposes the stable periodic high-utility sequential pattern mining (SPHUSPM) algorithm. The contributions made by this paper are as follows. First, we introduce the concept of stability to overcome the abovementioned problems, mine sequential patterns with stable periodic behavior, and propose the concept of stable periodic high-utility sequential patterns (SPHUSPs) for the first time. Secondly, we design a new data structure named the PUL-list to record the periodic information of sequential patterns, thereby improving the mining efficiency. Thirdly, we propose the maximum lability pruning strategy in sequential pattern (MLPS), which can prune a large number of unstable sequential patterns in advance. To assess the algorithm's effectiveness, we perform many experiments. It turns out that the algorithm can not only mine patterns that are ignored by traditional algorithms, but also ensure that the discovered patterns have stable periodic behavior. In addition, after using the MLPS pruning strategy, the algorithm can prune 46.5% of candidates in advance on average in six datasets. Pruning a large number of candidates in advance not only speeds up the mining process, but also greatly reduces memory usage.

Keywords: data mining; high-utility sequential pattern mining; stable periodic high-utility sequential pattern mining

check for updates

Citation: Xie, S.; Zhao, L. An Efficient Algorithm for Mining Stable Periodic High-Utility Sequential Patterns. *Symmetry* **2022**, *14*, 2032. https://doi.org/10.3390/sym14102032

Academic Editor: Mihai Postolache

Received: 30 August 2022
Accepted: 22 September 2022
Published: 28 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

High-utility sequential pattern mining (HUSPM) [1–11] is a significant area of knowledge discovery and data mining that has been the subject of a great deal of research. HUSPM has been applied in many applications, such as mining high-utility sequential patterns (HUSPs) in online dynamic log data [12], mobile commerce data [13], and gene regulation data [14]. Large numbers of HUSPs are mined, but some are redundant in some special scenarios. In the marketing example, the merchant needs to consider which product combinations are both highly profitable and can be sold regularly. However, some product combinations that are highly profitable but not frequently sold are considered HUSPs, in which case they are redundant. In recent years, some researchers have added the time constraint problem to HUSPM [15,16]. Considering the periodicity of HUSPs in the quantitative sequence database, some researchers proposed the periodic high-utility

sequential pattern mining (PHUSPM) to mining periodic high-utility sequential patterns (PHUSPs) [15,16]. The PHUSPM is also widely used in pattern discovery and knowledge discovery-related fields, such as research on consumer habits, website click-through rate data, financial market analysis, biomedical applications, and mobile computing. The PHUSPM defines the interval of the same pattern in different sequences as a period. The maximum periods of a sequential pattern are generally used to define the pattern's period. If a sequential pattern's period is below the user-defined upper limit ($maxPer$), it will be regarded as periodic. However, $maxPer$ is set too strictly, because if a sequential pattern exceeds the $maxPer$ threshold for only one period, it will be discarded. For example, in the market basket analysis, $maxPer$ is assumed to be one week and there are customers buying eggs and milk every weekend. This pattern will be periodic, but it will not be considered periodic if the customer skips a week. Therefore, traditional PHUSPM discards some useful and interesting PHUSPs. In addition, when the $maxPer$ value of PHUSPM is set too large, the PHUSPs mined will also vary. Obviously, these sequential patterns are not suitable for most practical applications. To sum up, the traditional PHUSPM method suffers from problem that some useful sequential patterns are discarded and some sequential patterns have large periodic fluctuations.

These problems also exist in periodic frequency pattern mining (PFPM) [17–19]. In order to provide greater flexibility, Kiran et al. proposed the partial periodic frequency pattern mining (PPFPM) algorithm [20]. This algorithm relaxes the $maxPer$ threshold constraint, allowing a specific amount of periods beyond it. In brief, if a pattern is considered periodic, it has no more than x (user-defined) periods that exceed the $maxPer$ threshold. Obviously, PFPM is a special situation of $x = 0$. Although PPFPM is more adaptable than PFPM, there is still an important problem in that it only verifies whether each period exceeds the $maxPer$ threshold. At the same time, PFPM and PPFPM ignore the amount by which each period exceeds the $maxPer$ threshold. For example, if the $maxPer$ threshold is set to a week, it makes no difference whether some products are discontinued by customers for two weeks or a year. Additionally, none of the models discussed above consider how closely spaced the values of the periods meeting the $maxPer$ threshold are. As a result, a pattern can be considered periodic, even though this pattern's several periods often alternate between values greater or smaller than the $maxPer$ threshold. In order to solve the above problems, Fournier–Viger et al. proposed stable periodic frequent pattern mining (SPFPM) [21] and top-k stable periodic frequent pattern mining (TSPIN) [22]. SPFPM and TSPIN propose the concept of stability in PFPM, mining stable periodic patterns that satisfy periodicity while maintaining similar period lengths and thus are more predictable than unstable patterns. However, these algorithms are not suitable for PHUSPM, because they do not consider the utility of items, and PHUSPM has no symmetry compared to these algorithms. Briefly, an item could have several utility values in a sequence, and a corresponding sequence will also have multiple utility values.

This paper suggests the stable periodic high-utility sequential pattern mining (SPHUSPM) algorithm for mining stable periodic high-utility sequential patterns (SPHUSPs). The SPHUSPM has three important contributions.

- The SPHUSPM provides a brand-new stability method and utilizes the time period information to a greater extent to mine more useful patterns. In the HUSPM research field, the algorithm provides a new research strategy. The addition of multiple methods also makes the mined patterns more interesting and more in line with user requirements. At the same time, in the field of practical application, the algorithm considers the maximum profit and the time period information at the same time, giving decision makers more accurate and efficient decision-making methods.
- We design a new data structure named PUL-list and a maximum stability pruning strategy in HUSPM (MLPS) to increase the effectiveness of mining. Experiments show that these two methods greatly improve the efficiency of the algorithm.

- We perform some experiments on six different datasets, which are guaranteed to be able to mine the desired SPHUSPs, while also showing excellent performance in operational efficiency and memory usage efficiency.

The remainder of this paper is structured as follows. Section 2 discusses related work. The SPHUSPM's preliminaries and problem definitions are introduced in Section 3. In Section 4 the suggested SPHUSPM algorithm is described. Comparative experiments are then presented in Section 5. Lastly, the conclusion of the paper is given.

2. Related Work

2.1. High-Utility Sequential Pattern Mining

High-utility itemset mining (HUIM) [23], which takes into account the quantity of items bought and unit profit, aims to find interesting patterns. Although a growing number of researchers have proposed several HUIM algorithms [24–29], they cannot be directly applied to the HUSPM because they did not consider the order of the itemsets.

HUSPM [2–5] is a research that combines HUIM and SPM, and its goal is to mine HUSPs in quantitative sequence databases. HUSPM was first proposed by Zhou et al., who added the concept of high utility to web log sequence pattern mining [1]. Ahmed et al. proposed a horizontal method called UL and a mode growth method called US [6]. Yin et al. proposed the concept of maximum utility value, the efficient USpan algorithm [3], which used a lexicographic sequence tree (LS-tree) structure to store sequence and utility information and used width and depth pruning strategies. On this basis, the top-k strategy was introduced, and the TUS algorithm [7] was proposed to mine HUSPs. The HUS-span algorithm [4], which made use of the same LS-tree structure as USpan, was put forth by Wang et al. In addition, pruning strategies such as the prefix extension utility (PEU) strategy and reduced sequence utility (RSU) strategy were also used in this algorithm. Lan et al. devised the PHUS algorithm [8], which used a sequential utility table structure to store sequential utility values, and employed a projection-based pruning strategy, and an indexing strategy to reduce search time. Alkan et al. proposed the HuspExt algorithm [9], which used a data matrix storage structure and a PBCG pruning strategy, which was based on a more compact overestimation strategy named CRoM, which could delete an abundance of unpromising candidates, greatly improving the mining efficiency. Recently, Gan et al. devised ProUM [10], an innovative projection-based mining algorithm. To increase mining speed, the algorithm makes use of the utility array structure and the sequence extended utility (SEU) pruning strategy. On this basis, Gan et al. devised the HUSP-ULL algorithm [11], which used a data structure called the utility-linked (UL)-list, which could efficiently record utility and location. To get strict upper constraints on the utility of candidate sequences, the algorithm also suggested the pruning methods irrelevant item pruning strategy (IPS) and look-ahead strategy (LAS).

Although the HUSPM could discover a large number of HUSPs, in some application scenarios, some of the sequential patterns were redundant and useless. The traditional HUSPM method ignored the time constraint problem, so some researchers proposed the PHUSPM algorithm. The next section will review the related work of PHUSPM.

2.2. Periodic High-Utility Sequential Pattern Mining

Some researchers have developed algorithms to mine periodic frequent patterns (PFPs) in transaction databases in the area of frequent pattern mining (FPM) [20,30–33]. Most of these algorithms relied on the excellent tree-based data structures to produce an entire collection of periodic–frequent patterns in a transactional database. Researchers have simultaneously suggested certain algorithms to exploit periodic–frequent sequential patterns (PFSPs) in sequence databases [34,35]. Because the above two types of algorithms used the support of the patterns to mine the corresponding patterns, ignoring the utility problem of the items, these algorithms could not find high-utility and periodic patterns.

In the field of HUIM, researchers have devised some methods for mining periodic high-utility itemsets (PHUIs) in the quantitative transaction database [36–38]. PHM [36]

is an approach that Fournier–Viger et al. suggested for mining PHUIs in quantitative transaction datasets. The algorithm created a new class of pattern called as periodic high-utility itemsets by fusing the ideas of periodic itemsets and high-utility itemsets. They proposed the minimum periodicity and the average periodicity as two new measurements to more precisely assess periodic behavior. For finding short-period high-utility itemsets (SPHUIs) [37,38] in quantitative transaction databases, Lin et al. suggested two techniques.

In the field of HUSPM, there are currently only two algorithms to mine PHUSPs in the quantitative sequence database. Dinh et al. proposed an algorithm named PHUSPM [15] by adding the method of periodicity to HUSPM for the first time. However, this algorithm did not design the special data structure and pruning strategy, so the algorithm was not efficient. After that, Dinh et al. [16] suggested the PUSOM algorithm based on the original algorithm, which designed a data structure called PUSP and used the maximum periodic pruning (MPP) strategy. However, the maxPer threshold set by this algorithm was too strict, and some useful sequential patterns were discarded and some patterns had large periodic fluctuation (unstable) problems. These problems also exist in PFPM [18–20], and some researchers propose the concept of stability to solve this problem. The following sections will review related work on SPFPM.

2.3. Stable Periodic Frequent Pattern Mining

Most research about PFP mining has evaluated the periodic behavior of patterns by comparing them with a *maxPer* threshold, but ignored the extent to which these periods exceed *maxPer* [18–20,36]. In order to find patterns with stable periodic behavior, Fournier–Viger et al. [21] proposed to mine a novel class of periodic frequent patterns in transaction databases, named stable periodic frequent patterns (SPFPs). The algorithm was called stable periodic frequent pattern mining (SPFPM). On this basis, in order to address the issue that the minimum support threshold is difficult to set, Fournier–Viger et al. [22] suggested an algorithm named top-K stable periodic patterns (TSPIN). Although the above algorithms could mine patterns with stable periodic behavior in transaction databases, they could not be directly applied to HUSPM. Because they do not take into account the order of itemsets in a sequence, that is, asymmetry, nor the utility of patterns.

- In light of the above, we list the limitations of the previously proposed work.
- In the PHUSPM algorithm, it is difficult to set the *maxPer* threshold accurately. Some patterns have a few periodic fluctuations. However, this situation has little impact on the decision, and they are still useful patterns. If *maxPer* is set too small, these interesting patterns will be ignored. If *maxPer* is set too large, the mined patterns will have unstable periods.
- Because the SPFPM algorithm is designed specifically for FPM, it cannot be directly applied to HUSPM. In short, it did not take into account the order between itemsets and the utility values of the items.

To resolve the aforementioned issues, we suggest a new stabilization method to discover stable periodic high-utility sequential patterns (SPHUSPs). We will define the SPHUSPM problem definitions and provide its preliminary information in the next part.

3. Preliminaries and Problem Definitions

To help the reader better understand the topic, we summarized symbols that appear in the definition section into Table 1.

Let $I = \{i_1, i_2, \ldots, i_M\}$ be a finite set containing M unique items. A q-item is denoted as (i_k, q_k), which represents the item $i_k \in I (1 \leq k \leq M)$ and its purchase quantity (internal utility). Each item has a weight to represent importance or profit per unit, which is called external utility and is denoted by $p(i_k)$. A q-itemset $X = [(i_1, q_1)(i_2, q_2)\ldots(i_m, q_m)]$ is a set of q-items. Without loss of generality, the order of q-items in a q-itemset is in lexicographic order($\prec$). A q-sequence $s = \ < X_1 X_2 \ldots X_n >$ is an order list of itemsets. A quantitative sequence database $S = \{s_1, s_2, \ldots, s_N\}$ is a set of q-sequences wherein each q-sequence has a unique identifier called *sid*. Table 2 is a quantitative sequence database. The external

utility (profit) of each item in I is shown in Table 3. All the examples in this article are from this quantitative sequence database.

Table 1. Symbols.

i	item
X	q-itemset
t	sequence
s	q-sequence
S, D	a quantitative sequence database
sid	the identifier of sequence
$q(i,s)$	the quantity of a q-item i in a q-sequence s
$p(i_k)$	the unit profit or importance (external utility) of i_k
$\prec$	the lexicographical order
$u(i,q)$	the utility of a q-item (i,q) in a q-sequence s
$u(X)$	the utility of a q-itemset X in a q-sequence s
$u(s)$	the utility of a q-sequence s
$t \sim s$	sequence t matches q-sequence s
$v(t,s)$	the sequence utility of a sequence t in a q-sequence s
$v(t)$	the utility of t in a q-sequence database S
$u_{max}(t,s)$	the maximum utility of a sequence t in a q-sequence s
$u_{max}(t)$	the maximum utility of a sequence t in a q-sequence database S
$< s-t >_{rest}$	the extension of a sequence t in a q-sequence s
$I(t)_{rest}$	the set of extension items of a sequence t in a quantitative sequential database D
$ru(t,s)$	the remaining utility of a sequence t in a q-sequence s
$S(t)$	the set of q-sequences containing the sequence t
$pe(s_\alpha, s_\beta)$	the period of two consecutive q-sequence s_α and s_β
$pes(t)$	periods of the sequence t
$la(t)$	the lability of the sequence t
$< t \oplus i_j >$	the concatenation of t with i_j

Table 2. A quantitative sequence database.

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

Table 3. A utility table.

Item	*a*	*b*	*c*	*d*	*e*	*f*	*g*
Profit	1	3	4	2	1	6	2

Definition 1. *Let* $X_a = [(i_{a_1}, q_{a_1})(i_{a_2}, q_{a_2}) \ldots (i_{a_m}, q_{a_m})]$ *and* $X_b = [(i_{b_1}, q_{b_1})(i_{b_2}, q_{b_2}) \ldots (i_{b_{m'}}, q_{b_{m'}})]$ *be two q-itemsets, where* $i_{a_k} \in I (1 \leq k \leq m)$ *and* $i_{b_{k'}} \in I (1 \leq k' \leq m')$*. If there exist positive integers* $1 \leq j_1 \leq j_2 \leq \ldots \leq j_m \leq m'$*, such that* $i_{a_1} = i_{b_{j1}} \wedge q_{a_1} = q_{b_{j1}}$, $i_{a_2} = i_{b_{j2}} \wedge q_{a_2} = q_{b_{j2}}$, $\ldots$, $i_{a_m} = i_{b_{jm}} \wedge q_{a_m} = q_{b_{jm}}$*, then* X_b *is said to contain* X_a*, which is denoted as* $X_a \subseteq X_b$*.*

For example, the q-itemset $[(b,3)(d,2)(e,2)]$ in q-sequence s_2 contains q-itemsets $(b,3), (d,2), (e,2), [(b,3)(d,2)], [(b,3)(e,2)], [(d,2)(e,2)]$, and $[(b,3)(d,2)(e,2)]$.

Definition 2. *Let* $A = < A_1 A_2 \ldots A_n >$ *and* $B = < B_1 B_2 \ldots B_{n'} > (n \leq n')$ *be the two q-sequences, where* A_α, B_β *are q-itemsets*$(1 \leq \alpha \leq n, 1 \leq \beta \leq n')$*. If there exists positive integers*

$1 \leq j_1 \leq j_2 \leq \ldots \leq j_n \leq n'$, *such that* $A_1 \subseteq B_{j_1}, A_2 \subseteq B_{j_2}, \ldots, A_n \subseteq B_{j_n}$, *then A is a q-subsequence of B and B is a q-supersequence of A, denoted as* $A \subseteq B$.

For example, the q-sequences $< [(a,3)(f,2)],[(a,5)],[(e,2)] >$ and $< [(a,3)(b,1)],[(a,5)(c,2)(g,5)],[(b,3)(d,2)] >$ are two q-subsequences of s_2.

Definition 3. *The utility of a q-item (i, q) in a q-sequence s is denoted and defined as*

$$u(i,q) = p(i) \times q(i). \tag{1}$$

The utility of a q-itemset X in a q-sequence s is denoted and defined as

$$u(X) = \sum_{k=1}^{m} u(i_k, q_k). \tag{2}$$

The utility of a q-sequence s is denoted and defined as

$$u(s) = \sum_{j=1}^{n} u(X_j). \tag{3}$$

For example, the utility of the q-item c in q-sequence s_4 (e.g., c_1) is $u(c,3) = 4 \times 3 = 12$. The utility of the q-itemset $[(b,2)(c,3)]$ in q-sequence s_4 is $u([(b,2)(c,3)]) = u(b,2) + u(c,3) = 3 \times 2 + 4 \times 3 = 18$. The utility of the q-sequence s_4 is $u(s_4) = u([(b,2)(c,3)] + u([(a,5)(e,1)]) + u([(b,4)(d,3)(e,5)]) = 18 + 6 + 23 = 47$.

Definition 4. *Given a q-sequence* $s = < (i_1,q_1)(i_2,q_2)\ldots(i_n,q_n) >$ *and a sequence* $t = < t_1 t_2 \ldots t_m >$, *s is said to match t if n = m and* $i_k = t_k$ *for* $1 \subseteq k \subseteq n$, *denoted as* $t \sim s$.

For example, $t = < (abcf)(acg)(bde) > \sim s_2$.

Definition 5. *The sequence utility of a sequence* $t = < t_1 t_2 \ldots t_m >$ *in a q-sequence* $s = < X_1 X_2 \ldots X_n >$ *is denoted and defined as*

$$v(t,s) = \bigcup_{s' \sim t \wedge s' \subseteq s} u(s'). \tag{4}$$

The utility of t in a q-sequence database S is denoted as

$$v(t) = \bigcup_{s \in S} v(t,s). \tag{5}$$

For example, the utility of the sequence $t = < gb >$ in the q-sequence s_1 is calculated as $v(t,s_1) = \{u(< (g,3)(b,2) >)\} = \{12\}$. The utility of t shown in Table 2 is $v(t) = \{v(t,s_1), v(t,s_2), v(t,s_3), v(t,s_4), v(t,s_5), v(t,s_6)\} = \{12, 19, 16, 16\}$.

Definition 6. *The maximum utility of a sequence t in a q-sequence s is denoted and defined as*

$$u_{max}(t,s) = max\{v(t,s)\}. \tag{6}$$

The maximum utility of a sequence t in a q-sequence database S is denoted and defined as

$$u_{max}(t) = \sum u_{max}(t,s), \forall s \in S. \tag{7}$$

For example, the maximum utility of the sequence $t =< gb >$ in the sequence database S shown in Table 2 is $u_{max}(t) = u_{max}(< gb >, s_1) + u_{max}(< gb >, s_2) + u_{max}(< gb >, s_3) = 12 + 19 + 16 = 47$.

Definition 7. *Given two q-sequences s and s′, if* $s \subseteq s'$*, the extension of s in s′ is said to be the rest of s′ after s, and is denoted as* $< s' - s >_{rest}$*. Given a sequence t and a q-sequence s, if* $t \sim s_k \wedge s_k \subseteq s (t \subseteq s)$*, the extension of t in s is the rest of s after* s_k*, which is denoted as* $< s - t >_{rest}$*, where* s_k *is the first match of t in s.*

For example, given a sequence $t = \ < [ac] >$. There exist two matches of t in s_2. The first one is $< [(a,3)(c,3)]] >$. Thus, $< s - t >_{rest} = \ < [(f,2)], [(a,5)(c,2)(g,5)], [(b,3)(d,2)(e,2)] >$.

Definition 8. *The set of extension items of a sequence t in a quantitative sequential database D is denoted as* $I(t)_{rest}$ *and defined as*

$$I(t)_{rest} = \{i_j | i_j \in < s - t >_{rest} \wedge t \subseteq s \wedge s \subseteq D\}. \tag{8}$$

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

Definition 9. *The remaining utility of a sequence t in a q-sequence s is denoted as* $ru(t)$ *and defined as*

$$ru(t,s) = u < s - t >_{rest} (t \subseteq s) = \sum_{i_j \in < s-t >_{rest}} u(i_j). \tag{9}$$

For example, given a sequence $t = \ < ab >$ and a q-sequence s_1 in Table 2, the extension of t in s_1 is $rest = \ < [(e,1)], [(d,3)] >$. The remaining utility is $ru(< ab >, s_1) = u < s_1 - < ab >>_{rest} = u(e,1) + u(d,3) = 1 + 6 = 7$.

Definition 10. *A sequence t is said to be a high-utility sequential pattern if* $u_{max}(t) \geq minutil(or\ \xi)$*, where* $minutil(or\ \xi)$ *is a given a user-specified minimum utility threshold.*

Definition 11. *Let there be a q-sequence database* $S = \{s_1, s_2, \ldots, s_n\}$ *and a sequence t. The set of q-sequences containing t is denoted as*

$$S(t) = \{s_{\alpha_1}, s_{\alpha_2}, \ldots, s_{\alpha_k}\},\ 1 \leq \alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k \leq n. \tag{10}$$

For example, Table 4 shows the occurrences of items in the q-sequence database S (Table 2). The list of q-sequences containing the sequences $< ab >$ and $< (ab) >$ are respectively $S(< ab >) = \{s_1, s_2, s_3, s_4, s_5\}$ and $S(< (ab) >) = \{s_1, s_2, s_3, s_5, s_6\}$.

Table 4. The occurrences of all items.

Sequnence ID	1				2			3			4			5		6		
Transaction ID	1	2	3	4	1	2	3	1	2	3	1	2	3	1	2	1	2	3
Items	a				a	a			a			a		a	a		a	a
	b		b		b		b	b	b	b	b		b		b		b	
		c			c	c		c		c	c			c	c			
		d		d			d						d		d			d
	e		e				e	e	e	e		e	e		e			
					f				f							f		
		g				g		g										

Definition 12. *Let there be two q-sequences* s_α*,* s_β *and a sequence t, such that* $t \sim s' \wedge s' \subseteq s_\alpha \wedge s_\alpha \in S(t)$ *and* $t \sim s'' \wedge s'' \subseteq s_\beta \wedge s_\beta \in S(t)$*.* s_α *and* s_β *are said to be consecutive with respect to t if there is not a q-sequence* $s_\gamma \in S(t)$*, such that* $\alpha < \gamma < \beta$*.*

The period of two consecutive q-sequence s_α *and* s_β *is denoted and defined as*

$$pe(s_\alpha, s_\beta) = \beta - \alpha. \tag{11}$$

In a word, $pe(s_\alpha, s_\beta)$ is the number of q-sequences between s_α and s_β.

For example, The sequence $< (ab) >$ appears in s_1, s_2, s_3, s_5 and s_6. Hence, $pe(s_1, s_2) = 2 - 1 = 1, pe(s_2, s_3) = 3 - 2 = 1, pe(s_3, s_5) = 5 - 3 = 2, pe(s_5, s_6) = 6 - 5 = 1$.

Definition 13. *Let there be a sequence t and $S(t) = \{s_{\alpha_1}, s_{\alpha_2}, \ldots, s_{\alpha_k}\}$, where $1 \leq \alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k \leq n$. The periods of a sequence t is a list of periods denoted and defined as*

$$pes(t) = \bigcup_{1 \leq \rho \leq k+1} pe(s_{\alpha_{\rho-1}}, s_{\alpha_\rho}) = \{pes(t,0), pes(t,1), \ldots pes(t, |S(t)|)\}, \alpha_0 = 0, \alpha_{k+1} = n. \quad (12)$$

For example, the sequence $< ab >$ has $pes(< ab >) = \{1, 1, 1, 1, 1, 1\}$. The sequence $< (ab) >$ has $pes(< (ab) >) = \{1, 1, 1, 2, 1, 0\}$.

In PHUSPs, the mining algorithm PHUSPM and PUSOM proposed by Dinh et al. [15,16], three periodicity measures are used to assess the periodicity of HUSPs in sequence databases.

Definition 14. *The maximum periodicity, minimum periodicity, and average periodicity of a sequence t are denoted and defined respectively as*

$$maxper(t) = max(pes(t)), \quad (13)$$

$$minper(t) = min(pes(t)), \quad (14)$$

$$avgper(t) = \sum x \in pes(t)/|pes(t)|. \quad (15)$$

For example, the periods of $< (ab) >$ are $pes(< (ab) >) = \{1, 1, 1, 2, 1, 0\}$. Thus, $maxper(< (ab) >) = 2$, $minper(< (ab) >) = 0$, and $avgper(< (ab) >) = 6/6 = 1$.

Definition 15. *Let there be five positive user-specified thresholds: $minutil(or\ \xi)$, $minAvg$, $maxAvg$, $minPer$, and $maxPer$. A sequence t is a periodic high-utility sequential pattern if t is a HUSP (it satisfies Definition 8) and $minAvg \leq avgper(t) \leq maxAug, minper(t) \geq minPer$ and $maxper(t) \leq maxPer$.*

Fournier–Viger et al. introduced a novel model based on the cumulative sum in order to find frequent patterns having a stable periodic behavior in PFP mining [21,22]. The main function of the model is to determine whether all periods of patterns are stable or not. Experiments on SPP [21] and TSPIN [22] algorithms show that this model is flexible and practical. However, this model is not designed for mining patterns on sequential databases. For the problem of this paper, we have modified this model to accommodate sequential databases. This model evaluates the stability of a pattern by calculating the cumulative sum of the difference between each period of the pattern and the *maxPer*. We define a method called lability to determine the periodic behavior of patterns.

Definition 16. *The lability of a sequence t is a list of values denoted as*

$$la(t) = \ < la(t,0), la(t,1), \ldots, la(t, |S(t)|) > . \quad (16)$$

The $la(t)$ list contains $|S(t)| + 1$ values. In other words, $|la(t)| = |S(t)| + 1 = |pes(t)|$. Each lability value in $la(t)$ is no less than zero. The first lability value of t is defined as $la(t,0) = max(0, pes(t,0) - maxper)$. Then, the i-th lability value of t for $i > 0$ is defined based on the the previous lability value as $la(t,i) = max(0, la(t,i-1) + pes(t,i) - maxper)$. Thus, lability values are calculated as a cumulative sum. Note that the above definition of lability can also be rewritten more concisely as follows:

$$la(t,i) = max(0, la(t,i-1) + pes(t,i) - maxper), la(t,-1) = 0. \quad (17)$$

For example, for the database of the running example and $maxPer = 1$, the periods of $t =< (ab) >$ are $pes(< (ab) >) = \{1, 1, 1, 2, 1, 0\}$. Because the t has six periods, it also has six lability values. The first lability value of t is $la(t, 0) = max(0, pes(t, 0) - maxper) = max(0, 1 - 1) = 0$. Then, the following lability values are $la(t, 1) = 0, la(t, 2) = 0, la(t, 3) = 1, la(t, 4) = 1$, and $la(t, 5) = 0$. Thus, the lability of itemset $\{d\}$ is $la(< (ab) >) = \{0, 0, 0, 1, 1, 0\}$.

For a sequence t, its $la(t)$ corresponds one-to-one to the value in $pes(t)$. If the value in $pes(t)$ is smaller, the calculated value in $la(t)$ will also show a smaller value. Conversely, the calculated value in $la(t)$ will also be relatively large. In addition, if a large value appears in $la(t)$, the value after that value may also become large. If the value of $la(t)$ is 0 or tends to 0, it indicates that the sequence t has good periodic stability. Conversely, the sequence t has unstable periodic behaviors.

Definition 17. *The maximum lability of a sequence t is defined as*

$$maxla(t) = max(la(t)). \tag{18}$$

The maximum lability of a sequence t is also called the stability of t. For example, as the lability values of sequence $t =< (ab) >$ *is* $la(< (ab) >) = \{0, 0, 0, 1, 1, 0\}$, *then* $maxla(< (ab) >) = 1$.

Definition 18. *Let there be a sequence database D, a sequence t, three user-defined thresholds minimum utility threshold (minutil or* ξ*) > 0, maximum periods (maxPer) > 0, and maximum lability threshold (maxLa)* ≥ 0. *The problem of mining the stable periodic high-utility sequential patterns in D consists of enumerating each sequence t in D such that* $maxla(t) \leq maxLa$ *and* $u(t) \geq \xi$.

To better understand the above definitions, we will give a example. At the same time, the pattern in this example will provide a case that is ignored by other algorithms. Assume that $maxPer = 2$ and $maxLa = 2$ are the limiting conditions. Given a sequence $t = \ < (cg) >$, it appears in sequences s_1, s_2 and s_3 in the quantitative sequence database shown in Table 2. Consequently, $S(< (cg) >) = \{s_1, s_2, s_3\}$. We get $pes(< (cg) >) = \{1, 1, 1, 3\}$ by Definition 13. So, $maxpes(< (cg) >) = 3$. In the traditional PHUSPM algorithm, the sequence $t = \ < (cg) >$ will not be considered periodic according to the constraint $maxPer = 2$. Obviously, this pattern appears periodically in the first half of the database. Therefore, this pattern is useful for some applications. However, due to strict restrictions, this pattern is ignored by the traditional algorithm. In this article, we use the stability strategy that solves this problem. By Definition 16, we get that $la(< (cg) >) = \{0, 0, 0, 1\}$. According to $maxla(< (cg) >) = 1 < maxLa = 2$, we get that $t = \ < (cg) >$ is a useful pattern. It is clear from this example that this method can find interesting patterns that traditional methods miss.

4. Proposed Algorithms

4.1. The Data Structure

In 2019, Gan et al. [11] proposed the HUSP-ULL algorithm, which utilized a utility-linked (UL)-list structure and a lexicographic sequence (LS)-tree for mining HUSPs. This paper also uses LS-tree and designs a new data structure based on the UL-list, namely period utility-linked (PUL)-list structure. This structure can quickly access period and utility information, which greatly improves the operating efficiency of the algorithm.

4.1.1. Lexicographic Sequence Tree and Concatenations

Each node in the lexicographic sequence tree [39] represents a potential SPHUSP candidate. To identify an SPHUSP candidate, the utility value in the node can be evaluated to the minimal utility threshold and the stability value to the maximum lability threshold.

In the LS-tree nodes, all of the original database's sequences are converted to UL-lists. The designed algorithm utilizes two common sequence mining operations named I-concatenation and S-concatenation to create new sequences (child nodes) in the LS-tree.

Definition 19. *Given a sequence t and an item i_j, the I-concatenation of t with i_j consists of appending i_j to the last itemset of t, denoted as $< t \oplus i_j >_{I-concatenation}$. The S-concatenation of t with an item i_j consists of adding i_j to a new itemset appended after the last itemset of t, denoted as $< t \oplus i_j >_{S-concatenation}$.*

For example, given a sequence $t = < [b], [c] >$ and a item a, $< t \oplus a >_{I-concatenation} = < [b], [ac] >$ and $< t \oplus a >_{S-concatenation} = < [b], [c], [a] >$.

It is clear that after executing the I-concatenation operation, the sequence's itemsets count stays the same; however, after doing the S-concatenation action, the itemsets count rises by one. All potential sequence patterns in the search space for SPHUSPM can be constructed based on these two methods.

The algorithm's search procedure can be compared to the procedure of gradually constructing a LS-tree. The method first searches the database for a set of 1-sequences that meet the minimum utility and maximum lability thresholds. The LS-tree is then explored by using a depth-first search approach, starting from the 1-sequence. We use the I-concatenation and S-concatenation operations to obtain the child nodes of each node. Finally, the complete sequence database set is obtained.

4.1.2. The Period-Utility-Linked List Structure

The burden increases during the mining process because the program must repeatedly scan the original database. To solve this problem, this study uses the same UL-list structure as the HUSP-ULL algorithm [11] to store the relevant information of each q-sequence in the q-sequence database, and then constructs the PUL-list to discover SPHUSPs. The PUL-list contains the SID of each q-sequence where the candidate sequence is located and the suffix sequence information in the q-sequence. In order to facilitate candidate sequence expansion and utility calculation, the PUL-list also stores the first occurrence position of each different item of the suffix sequence in the q-sequence, that is, index information. In addition, in order to easily measure the stability of the candidate pattern, the order of the q-sequence (period information) where the candidate sequence is located in the original database is also included in the PUL-list.

In the UL-list, the utility and position (UP) information and header table of the q-sequence are included. Among them, the UP information records the item name, the utility of the item, the remaining utility of the item, and the position where the item appears next in the sequence. The heard table stores the name of each item in the q-sequence and the position where the item first appears in the sequence.

For example, Table 5 shows the UL-list of s_1. In UP information, the first item in s_1 is a, the utility is 1, the remaining utility is 41, and the next position is empty, which means that it no longer appears in this sequence. In the heard table, the first occurrence location of the different items in s_1 is stored. The information $(a, 1)$ indicates that the position where item a first appears in s_1 is the first position.

Table 5. The utility-linked (UL)-list structure of s_1.

UP Information of s_1	$[(a,1,41,-)(b,3,38,7)(e,3,35,8)]$, $[(c,12,23,-)(d,4,19,9)(g,6,13,-)]$, $[(b,6,7,-)(e,1,6,-)]$, $[(d,6,0,-)]$
Header Table of s_1	$(a,1)(b,2)(c,4)(d,5)(e,3)(g,6)$

The PUL-list is a projection database of candidate sequences in the mining process. It contains UP information, heard table and periodic information of candidate sequences of multiple sequences. Its purpose is to facilitate the calculation of periodicity and stability.

The PUL-list takes $< (ab) >$ as an example, as shown in Table 6. As indicated in Table 6, the PUL-list takes $< (ab) >$ as an example. In the PUL-list of $< (ab) >$, the projection information of sequence $< (ab) >$ in each q-sequence is stored; for example, the projection of sequence $< (ab) >$ in s_2 is $< [(c,3)(f,2)], [(a,5)(c,2)(g,5)], [(b,3)(d,2)(e,2)] >$. The periodic information in this table records the index of the sequence containing the sequence $< (ab) >$ in original databases (i.e., the quantitative sequence database), and $< 1,2,3,5,6 >$ means the sequence $< (ab) >$ appears in the first, second, third, fifth and sixth sequences.

The SPHUSPM algorithm uses the PUL-list structure to simply access utility, location, and period information instead of repeatedly scanning the original database. The access efficiency and mining efficiency of the algorithm are both significantly increased by this structure.

Table 6. The period-utility-linked (PUL)-list structure of $< (ab) >$.

UP Information of s_1	$[(e,3,35,6)]$, $[(c,12,23,-)(d,4,19,7)(g,6,13,-)]$, $[(b,6,7,-)(e,1,6,-)]]$, $[(d,6,0,-)]$
Header Table of s_1	$(b,5)(c,2)(d,3)(e,6)(g,4)$
UP Information of s_2	$[(c,12,50,4)(f,12,38,-)]$, $[(a,5,33,-)(c,8,25,-)(g,10,15,-)]$, $[(b,9,6,-)(d,4,2,-)(e,2,0,-)]]$
Header Table of s_2	$(a,3)(b,6)(c,1)(d,7)(e,8)(f,2)(g,5)$
UP Information of s_3	$[(e,4,24,5)(f,12,12,-)]$, $[(b,6,6,-)(c,4,2,-)(e,2,0,-)]$
Header Table of s_3	$(b,3)(c,4)(e,1)(f,2)$
UP Information of s_5	$[(c,8,11,-)(d,8,3,-)(e,3,0,-)]$
Header Table of s_5	$(c,1)(d,2)(e,3)$
UP Information of s_6	$[(a,3,8,-)(d,8,0,-)]$
Header Table of s_6	$(a,3)(d,2)$
The Periodic Information of $< (ab) >$	$< 1,2,3,5,6 >$

4.2. Pruning Strategy

Numerous candidates are produced during the mining process. Therefore, this will increase the chance of combinatorial explosion, which in turn will cause the algorithm to run slower, so we need to introduce several pruning strategies to solve this problem.

4.2.1. The Downward Closure Property of Upper Bound

Definition 20. *The sequence-weighted utilization (SWU) [3] of a sequence t in a quantitative sequential database D is denoted as SWU(t) and defined as*

$$SWU(t) = \sum_{s' \sim t \wedge s' \subseteq s \wedge s \subseteq D} u(s). \tag{19}$$

For example, $SWU(< a >) = u(s_1) + u(s_2) + (s_3) + (s_4) + (s_5) + (s_6) = 42 + 68 + 56 + 47 + 52 + 49 = 314$ and $SWU(< f >) = u(s_2) + u(s_3) + (s_6) = 68 + 56 + 49 = 173$.

Theorem 1. *Given a quantitative sequential database D and two sequences t and t′ . If $t \subseteq t'$, then*

$$SWU(t') \leq SWU(t). \tag{20}$$

Proof. Because $t \subseteq t'$, $SWU(t') = \sum_{s' \sim t' \wedge s' \subseteq s \wedge s \subseteq D} u(s) \leq \sum_{s'' \sim t \wedge s'' \subseteq s \wedge s \subseteq D} u(s) = SWU(t)$. □

Theorem 2. *Given a quantitative sequential database D and a sequence t, it can be obtained that*

$$u(t) \leq SWU(t). \tag{21}$$

Proof. Because $u(t,s) \leq u(s)$, we can obtain that $u(t) = \bigcup_{s \in D} v(t,s) \leq \sum_{s' \sim t \wedge s' \subseteq s \wedge s \subseteq D} u(s)$ $= SWU(t)$. □

From the above SWU definition and theorems, the utility of sequence t must be less than the minimum utility threshold if the SWU value of t is less than that threshold. When this happens, the utility of any supersequence of t will also be below this threshold. SWU is able to eliminate many candidates that are unqualified as a result. In actuality, the utility of a sequence t is typically significantly smaller than its SWU value, so a tighter upper bound also needs to be introduced.

We introduce two strategies from the HUSP-ULL algorithm [11] to generate a stricter upper bound, which is built on the prefix extension utility (PEU) model.

Definition 21. *The PEU of a sequence t in a q-sequence s is denoted as $PEU(t,s)$ and defined as*

$$PEU(t,s) = max\{u(s_k) + u(< s - s_k >)_{rest} \| t \sim s_k \wedge s_k \subseteq s\}. \tag{22}$$

Definition 22. *The PEU of a sequence t in D is denoted as $PEU(t)$ and defined as*

$$PEU(t) = \sum_{s \in D} \{PEU(t,s) | t \subseteq s\}. \tag{23}$$

Theorem 3. *Given a quantitative sequential database D, and two sequences t and t′. If $t \subseteq t'$, we obtain*

$$PEU(t') \leq PEU(t). \tag{24}$$

Theorem 4. *Given a quantitative sequential database D and a sequence t, we can obtain*

$$u(t) \leq PEU(t). \tag{25}$$

The proofs of Theorems 3 and 4 can be found in [11]. Theorems 3 and 4 demonstrate that if the sequence t's PEU value is below the minimal utility threshold, then the supersequence of t's PEU value is similarly below the minimum utility threshold. The utility of t and the utility of the supersequence of t are both less than the minimum utility threshold if the PEU value of the sequence t is less than the minimum utility threshold.

4.2.2. Pruning Strategies

The candidate sequence t can produce a lot of candidate sequences when doing I-concatenations and S-concatenations. In order to reduce a mount of candidate sequences, this research introduces the look-ahead strategy (LAS) and the irrelevant item pruning strategy (IPS) in the HUSP-ULL algorithm [11] to eliminate hopeless candidates in advance.

Theorem 5. *Given a sequence t and a quantitative sequential database D, two situations are considered to generate a supersequence.*

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

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

The proof of Theorem 5 can be referred to [11].

Look-Ahead Strategy (LAS):

(1) If i_j is an I-concatenation candidate item of t and

$\sum_{s \in D}\{PEU(t,s)| < t \oplus i_j >_{I-concatenation} \subseteq s\}$ is less than the minimum utility threshold, i_j should be removed from C^I (the set of candidate items for I-concatenation with t).

(2) If i_j is a S-concatenation candidate item of t and $\sum_{s \in D}\{PEU(t,s)| < t \oplus i_j >_{S-concatenation} \subseteq s\}$ is less than the minimum utility threshold, i_j should be removed from C^S (the set of candidate items for S-concatenation with t).

Theorem 6. *For any sequence t and item* $i_j \in I(t)_{rest}$*, the maximal utility of* $< t \oplus i_j >_{I-concatenation} \subseteq s$ *or* $< t \oplus i_j >_{S-concatenation} \subseteq s$ *is no more than* $\sum_{s \in D}\{PEU(t,s)|(< t \oplus i_j >_{I-concatenation} \subseteq s) \vee (< t \oplus i_j >_{S-concatenation} \subseteq s)\}$.

Proof of Theorem 6 can be referred to [11].

Irrelevant Item Pruning Strategy (IPS):

Given a sequence t and an item $i_j \in I(t)_{rest}$, if $\sum_{s \in D}\{PEU(t,s)|(< t \oplus i_j >_{I-concatenation} \subseteq s) \vee (< t \oplus i_j >_{S-concatenation} \subseteq s)\}$ is less than the minimum utility threshold, i_j is called an irrelevant item of t and should be removed from the utility linked lists of t and t's supersets.

This algorithm uses LAS and IPS pruning strategies to remove a large number of candidates, which greatly improves the running efficiency of the algorithm. The places where these two strategies are used will be marked in the algorithm section.

Because the pruning strategies mentioned above are all pruning in terms of utility, in order to further improve the mining speed of the algorithm, we introduced the maximum lability pruning (MLP) strategy in the TSPIN algorithm [22]. However, this strategy is only applicable to transactional databases, so we modified it to make it suitable for quantitative sequence databases.

Theorem 7. *For any two sequences* $t \subset t' \subseteq D$*, the relationship* $maxla(t') \geq maxla(t)$ *holds.*

Proof. Because $t \subset t'$, it follows that $S(t') \subseteq S(t)$. In the case where, $S(t') = S(t)$, the periods of t and t' are the same. Hence, $la(t) = la(t')$ and $maxla(t) = maxla(t')$. In the case where $S(t') \subset S(t)$, then for each sequence $\{s_z | s_z \in S(t) \wedge s_z \notin S(t')\}$, the corresponding $pes(t)$ will have smaller numbers. Thus, $maxla(t') \geq maxla(t)$.

Theorem 8. *For a sequence database D, if* $maxla(t) > maxLA$ *for an sequence t,then t and its supersequences don't have stability.*

Hence, the part of the search space containing t and its supersets can be ignored.

Proof. According to the definition of SPHUSPM, if $maxla(t) > maxLa$, then t is not an SPHUSP. Then, any supersequence t' of t is also not a SPHUSPM based on Theorem 7. □

4.3. The SPHUSPM Algorithm

Based on the proposed problem definition, the PUL-list structure and SWU, LAS, IPS, and MLPS strategies, this section proposes the stable periodic high-utility sequential pattern mining (SPHUSPM) algorithm. The framework of the proposed SPHUSPM algorithm is shown in Figure 1, which shows the main parts of the method and their interworking. This algorithm is the first in the field of HUSPM to mine sequential patterns with stable periods. This work is challenging because SPHUSPs have not been defined before, and stability methods cannot be directly applied to quantitative sequence databases.

The pseudo-code of the Algorithm 1: SPHUSPM will be given below .

First, scan the quantitative sequence database D to calculate $u(s)$ and $u(D)$, and construct the PUL-list for each q-sequence $s \in D$ (Line 1). Initialize the set of SPHUSPs (Line 2). For each item $i_j \in D$, the algorithm builds a projection database $PD(i_j)$ to store the transformed PUL-list (Lines 3 to 4). The utility and SWU values for each $1-sequence$ are computed by using the corresponding projection database (Line 5). A $1-sequence$ whose SWU value is not less than the minimum utility threshold will be considered as the candidate SPHUSP (Lines 5 to 12). Therefore, those $1-sequences$ with lower SWU

values will be considered unpromising, and they will be deleted at this step. Then use the PUL-list of (i_j) to calculate the set of *pes* and the set of *la*. Then judge the stability of (i_j). If not satisfied, both it and its supersequence are considered not to be SPHUSPs, and the MLPS pruning strategy is applied here. The $1-sequence$ utility if not less than the minimum utility threshold will be output as SPHUSPs (Lines 6 to 10). Next, the PGrowth algorithm treats candidates as prefixes for mining more SPHUSPs (Line 12). The program is cited from [11]. In this part, we acquire the projection database $PD(prefix)$ of the $prefix$ and C^I (candidate item set for I-concatenation), C^S (candidate item set for S-concatenation) through IPS and LAS strategies, respectively. Then, we use the items in C^I and C^S to perform I-concatenation or S-concatenation with the $prefix$ respectively, and then enter the Judge-SPHUSPs program (Lines 13 to 18).

Algorithm 1: SPHUSPM

Input: D, a quantitative sequential database; *utable*, a utility table containing the unit profit of each item; *minutil*, the minimum utility threshold; *maxPer*, the maximum periods; *maxLa*, the maximum lability threshold.
Output: The complete set of SPHUSPs.

1 scan D to: (1). calculate $u(s)$ for each $s \in D$and calculate $u(D)$; (2). build the PUL-list of each $s \in D$;
2 SPHUSPs $\leftarrow \emptyset$;
3 **for** *each* $i_j \in D$ **do**
4 PD$(i_j) \leftarrow \{thePUL-list\ of\ i_j\}$;
5 calculate $SWU(i_j)$ and $u(i_j)$;
6 **if** $SWU(< i_j >) \geq minutil \times u(D)$ **then**
7 calculate $pes(< i_j >)$ and $la(< i_j >)$;
8 **if** $maxla(i_j) \leq maxLa$ **then**
9 **if** $u(< i_j >) \geq minutil \times u(D)$ **then**
10 SPHUSPs $\leftarrow$ SPHUSPs $\cup < i_j >$
11 **end**
12 **PGrowth**$(< i_j >, PD(i_j), SPHUSPs)$;
13 **for** *each* $i_j \in C^I$ **do**
14 **Judge-SPHUSPs**$(< prefix \oplus i_j >_{I-Concatetion}, PD(prefix),$
15 $SPHUSPs)$;
16 **end**
17 **for** *each* $i_j \in C^S$ **do**
18 **Judge-SPHUSPs**$(< prefix \oplus i_j >_{S-Concatetion}, PD(prefix),$
19 $SPHUSPs)$;
20 **end**
21 **end**
22 **end**
23 **end**
24 **return** SPHUSPs

The judge-SPHUSPs process (Algorithm 2) first constructs the PUL-list of $prefix'$ through the projection database of $prefix$, and obtains the projection database of $prefix'$ (Line 1). Then calculate the PEU value, utility value, the set of *pes*, and the set of *la* of $prefix'$ by $PD(prefix')$ (Line 2). If the utility of $prefix'$ is not less than the minimum utility threshold and the *maxla* value of $prefix'$ is not greater than the *maxLa* value, $prefix'$ is determined to be a SPHUSP (Lines 4 to 6). If the PEU value of $prefix'$ is not less than the minimum utility threshold and the *maxla* value of $prefix'$ is not greater than the *maxLa* value, $prefix'$ will enter the PGrowth process (Line 8). If no candidate sequence is generated, the process ends. Finally, the designed algorithm returns a list of mined SPHUSPs.

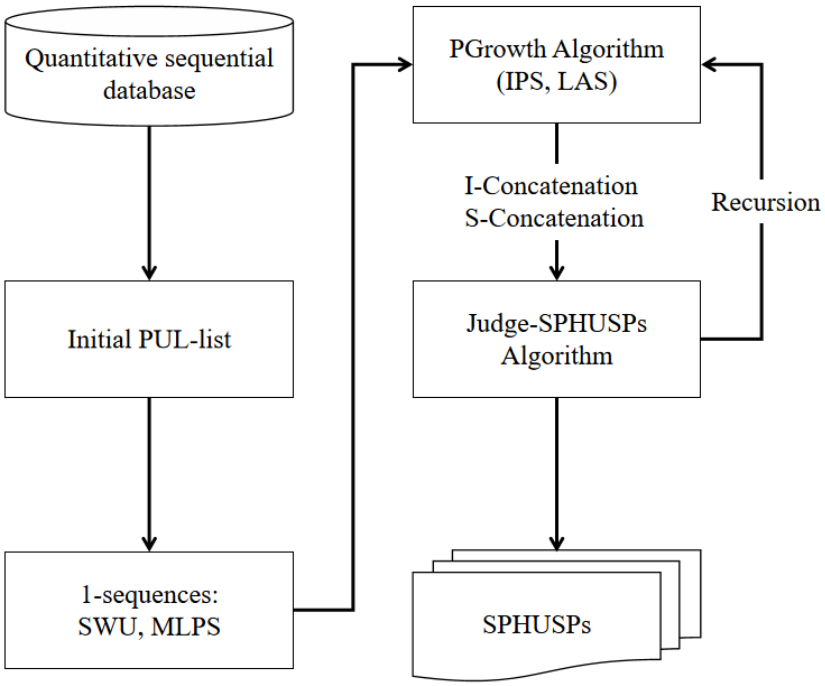

Figure 1. Framework of the SPHUSPM algorithm.

Algorithm 2: Judge-SPHUSPs

Input: $prefix'$, $PD(prefix)$, $SPHUSPs$

$PD(prefix') \leftarrow \{PUL - list\ of\ prefix'\}$;
calculate $u(prefix')$ and $PEU(prefix')$, calculate $pes(prefix')$ and $la(prefix')$;
if $PEU(prefix') \geq \delta \times u(D)$ **then**
  if $maxla(prefix') \leq maxLa$ **then**
    if $u(prefix') \geq \delta \times u(D)$ **then**
      SPHUSPs $\leftarrow$ SPHUSPs $\cup prefix'$;
    end
    PGrowth$(prefix', PD(prefix'), SPHUSPs)$
  end
end

4.4. Total Computational Complexity

In order to further understand the proposed algorithm, we will analyze the time complexity of the algorithm in the following. N_1 is the number of sequences in the quantitative sequence database. N_2 is the average number of items in a sequence in the quantitative sequence database. N_3 is the number of all candidates produced by the algorithm. N_4 is the average number of occurrences of all candidates.

The SPHUSPM algorithm must scan the original database once when building the projection database and performing width pruning. Therefore, the time complexity of this operation is $O(N_1 \times N_2)$. This algorithm must build LS-tree by expanding the candidates, so the time complexity of this process is $O(N_3 \times N_4)$. All candidates must enter the judge-SPHUSPs algorithm to determine whether they are real SPHUSPs. Therefore, the time complexity of this operation is $O(N_3)$.

In summary, the time complexity of the SPHUSPM algorithm is $O(N_1 \times N_2 + N_3 \times N_4 + N_3)$.

5. Experimental Evaluation

The performance of SPHUSPM is not compared to that of other algorithms because it is the first algorithm for mining SPHUSPs. SPHUSPM is implemented in Java. To evaluate the SPHUSPM algorithm, extensive experiments have been done on a workstation running Windows 10, equipped with an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz 3.41 GHz, and 32 GB of RAM.

5.1. Datasets

Six real datasets [40] are used in the experiment to evaluate the performance of the algorithm.

Sign. The National Center for Sign Language and Gesture Resources at Boston University developed Sign, a real-world dataset of sign language utterance sequences. Every utterance in the dataset corresponds to a video segment that has been meticulously transcribed.

Bible. By converting the Bible into a collection of item sequences, the Bible can be viewed as a real-world dataset.

Kosarak10k. Kosarak10k is a subset of the original Kosarak dataset. This is a real-world dataset made up of click-stream data from a news portal in Hungary.

Leviathan. Leviathan is a conversion of Thomas Hobbes' Leviathan novel (1651) to a sequence of items (words).

yoochoose-buys. YOOCHOOSE GmbH created the yoochoose-buys commercial dataset to assist RecSys Challenge 20,151 participants.

MSNBC. The click-stream data from the MSNBC website has been transformed from the original data from the UCI repository to create the MSNBC dataset. Only 31,790 sequences remain after the smallest ones were eliminated.

Tables 7 and 8, which show the parameters and properties of these datasets, respectively. The above datasets can be downloaded from [40].

Table 7. Parameters of the datasets.

$\|D\|$	Number of sequences
$\|I\|$	Number of distinct items
C	Average number of itemsets per sequence
T	Average number of items per itemset
$MaxLen$	Maximum number of items per sequence

Table 8. Characteristics of the datasets.

Dataset	$\|D\|$	$\|I\|$	C	T	$MaxLen$
Sign	730	267	52.0	1	94
Bible	36,369	13,905	21.6	1	100
Kosarak10k	10,000	10,094	8.14	1	608
Leviathan	5834	9025	33.8	1	100
yoochoose-buys	234,300	16,004	1.13	1.97	21
MSNBC	31,790	423,776	13.33	1	86

5.2. Execution Time

Figure 2 shows the experimental results of the execution time of the SPHUSPM algorithm. The experiment was conducted on the six real datasets mentioned above. We observe the performance of the algorithm on different datasets with different thresholds by setting the values of $minutil$, $maxPer$, and $maxLa$. In Figure 2, the x axis represents $minutil$ and the y axis represents execution time. $P - L$ indicates $maxPer = P$ and $maxLa = L$. The following observations can be drawn from Figure 2.

When the execution time of the algorithm is high, the value of $maxLa$ is usually set high. The reason is that with the increase of $maxLa$, the stability value of sequence patterns also decreases. In this case, SPHUSPM must consider more sequential patterns in mining.

At the same time, decreasing the value of $minutil$ and increasing the value of $maxLa$ will relax the threshold. Therefore, in these cases, SPHUSPM must consider more sequence patterns and increase the execution time of the algorithm.

On the sparse dataset, the SPHUSPM method performs better than it does on the dense dataset. Because the occurrence probability of the same sequence pattern in sparse datasets is low, the sequence pattern stability is low. Thus, a large number of unstable patterns will not be considered.

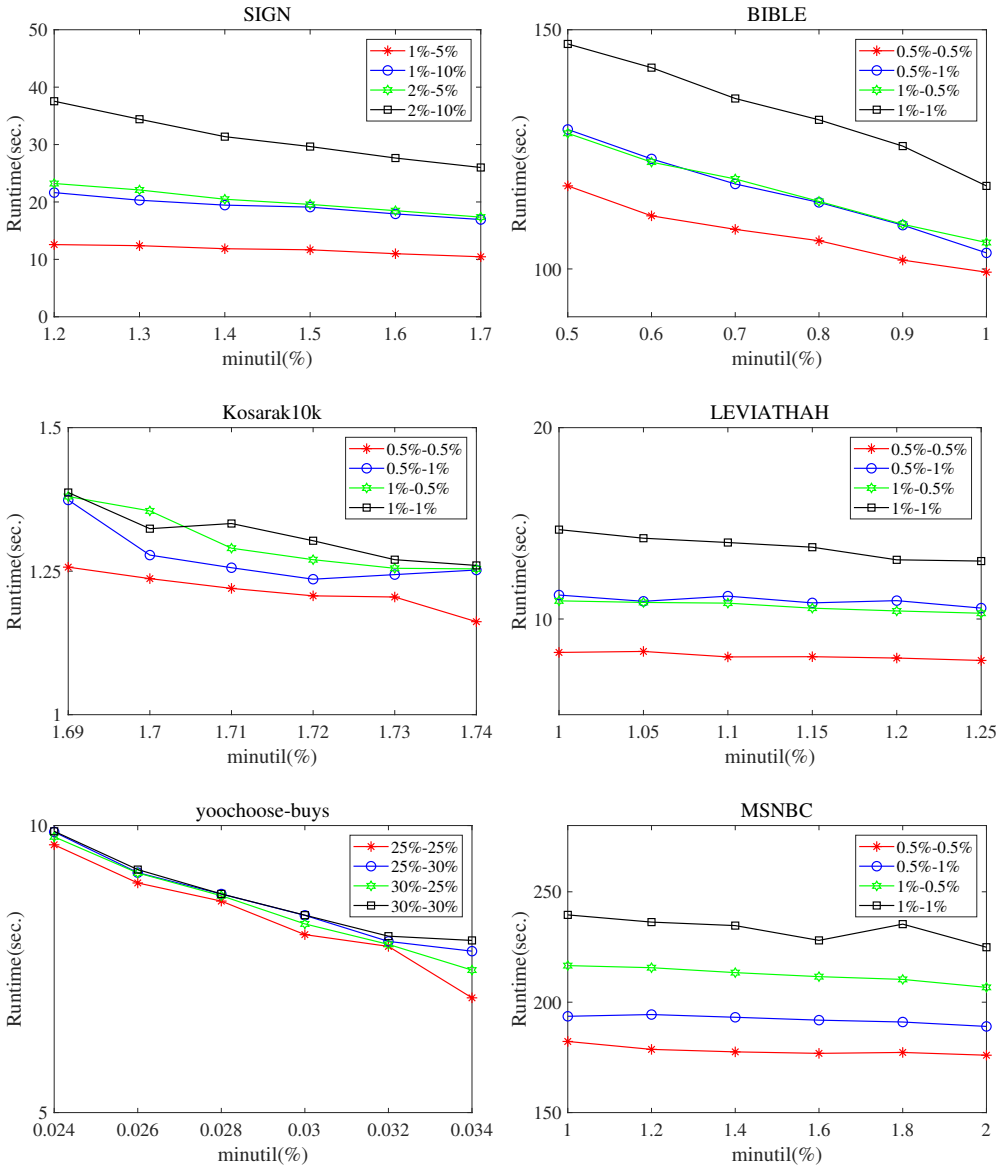

Figure 2. Execution times for different parameter values. ($maxPer - maxLa(P - L)$).

5.3. Pattern Count

Figure 3 shows the experimental results of the pattern count of SPHUSPM algorithm. The experiment was conducted on the six real datasets mentioned above. We observe the performance of the algorithm on different datasets with different thresholds by setting the values of $minutil$, $maxPer$, and $maxLa$. In this figure, the x axis represents the minimum utility threshold ($minutil$), and the y axis represents the number of SPHUSPs generated.

$P - L$ means $maxPer = P, maxLa = L$. The following observations can be drawn from Figure 3.

Fix the *maxLa* value and the *maxPer* value, and increase the *minutil* value. Alternatively, fix the *minutil* value and the *maxPer* value, and decrease the *maxLa* value. In both cases, the number of sequential patterns is reduced because the criteria for the threshold is increased.

When *minutil* remains unchanged, the number of SPHUSPs increases with *maxPer* and *maxLa*, indicating a large number of potentially stable patterns in the HUSPs in datasets.

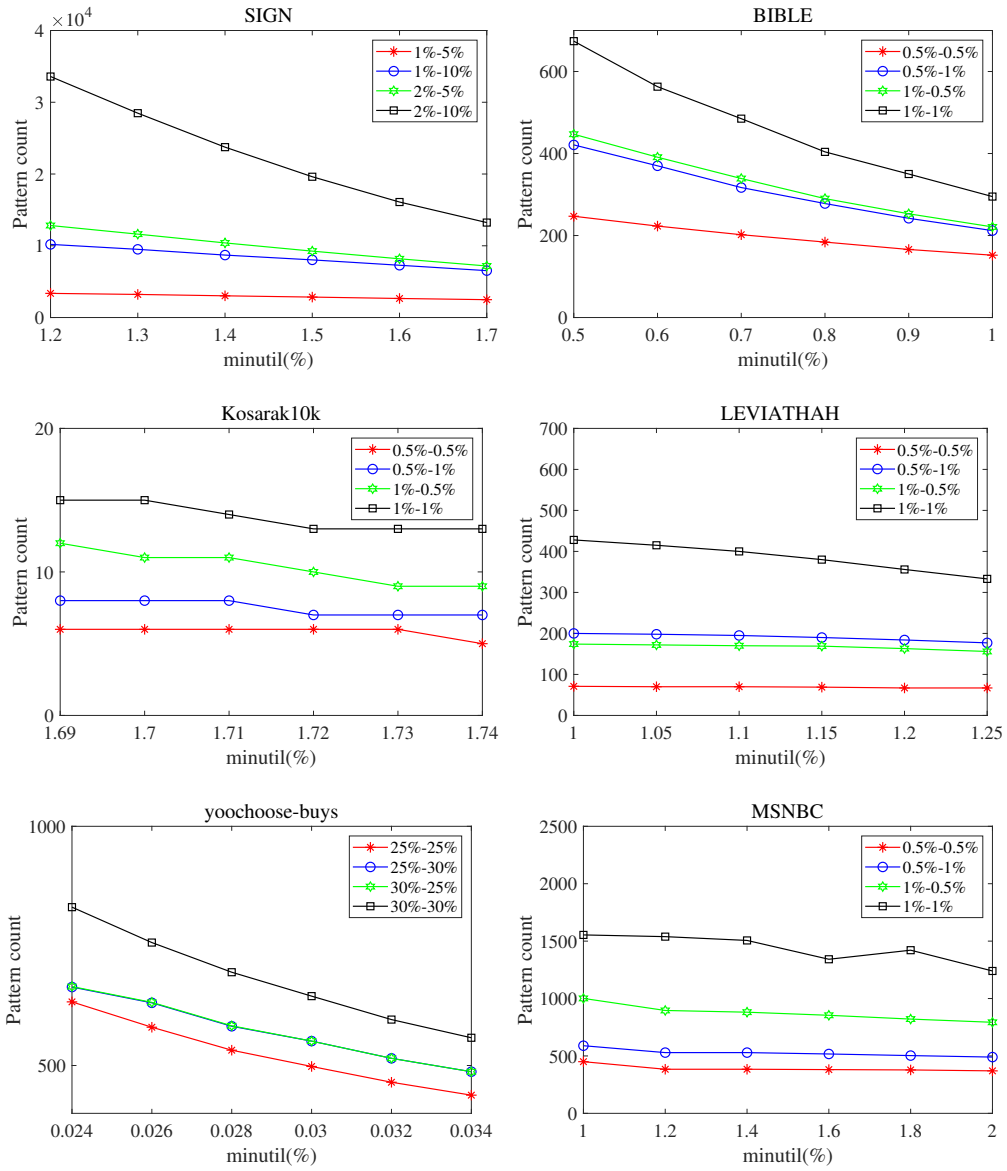

Figure 3. Pattern count for different parameter values. ($maxPer - maxLa(P - L)$).

5.4. Memory Usage

Table 9 shows the memory usage of the SPHUSPM algorithm with different *minutil*, *maxPer*, and *maxLa* values set on six datasets. The results show that the SPHUSPM algorithm uses less memory when the high *minutil* value, the low *maxPer* value and the low *maxLa* value are set. This is reasonable because the algorithm can prune more sequential patterns in these cases.

Table 9. Memory usage for different parameter values.

SIGN				**BIBLE**			
maxPer	*maxLa*	*minutil*	*Maxmemory*	*maxPer*	*maxLa*	*minutil*	*Maxmemory*
1%	5%	1.2%	306.42	0.5%	0.5%	0.5%	1106.55
1%	5%	1.7%	306.32	0.5%	0.5%	1%	1112.40
1%	10%	1.2%	307.49	0.5%	1%	0.5%	1118.63
1%	10%	1.7%	306.68	0.5%	1%	1%	1125.90
2%	5%	1.2%	307.60	1%	0.5%	0.5%	1113.62
2%	5%	1.7%	306.81	1%	0.5%	1%	1133.95
2%	10%	1.2%	310.12	1%	1%	0.5%	1140.22
2%	10%	1.7%	307.75	1%	1%	1%	1134.50
Kosarak10k				**LEVIATHAN**			
maxPer	*maxLa*	*minutil*	*Max memory*	*maxPer*	*maxLa*	*minutil*	*Max memory*
0.5%	0.5%	1.69%	238.70	0.5%	0.5%	1%	666.37
0.5%	0.5%	1.74%	236.87	0.5%	0.5%	1.25%	648.24
0.5%	1%	1.69%	248.66	0.5%	1%	1%	674.32
0.5%	1%	1.74%	242.95	0.5%	1%	1.25%	662.90
1%	0.5%	1.69%	248.30	1%	0.5%	1%	676.01
1%	0.5%	1.74%	247.64	1%	0.5%	1.25%	666.23
1%	1%	1.69%	250.92	1%	1%	1%	681.99
1%	1%	1.74%	248.94	1%	1%	1.25%	678.90
yoochoose-buys				**MSNBC**			
maxPer	*maxLa*	*minutil*	*Max memory*	*maxPer*	*maxLa*	*minutil*	*Max memory*
25%	25%	0.024%	549.85	0.5%	0.5%	1%	636.57
25%	25%	0.034%	536.48	0.5%	0.5%	2%	620.43
25%	30%	0.024%	579.85	0.5%	1%	1%	640.75
25%	30%	0.034%	560.78	0.5%	1%	2%	626.33
30%	25%	0.024%	582.66	1%	0.5%	1%	641.33
30%	25%	0.034%	567.38	1%	0.5%	2%	634.43
30%	30%	0.024%	586.02	1%	1%	1%	651.61
30%	30%	0.034%	561.23	1%	1%	2%	637.15

5.5. Effectiveness of Pruning Strategies

To test the performance of the improved maximum lability pruning strategy in sequential pattern mining (MLPS), we conduct experiments on six datasets. Figure 4 shows the number of candidates generated by the SPHUSPM when setting different values of *minutil*, *maxPer*, and *maxLa* on six datasets. In this figure, the x axis represents the minimum utility threshold (*minutil*), and the y axis represents the number of generated candidates. $P - L$ means $maxPer = P$, $maxLa = L$. In addition, $MLP\ P - L$ indicates that the *MLPS* strategy is used. The following observations can be drawn from Figure 4.

MLPS can eliminate a large number of candidates in advance and show excellent performance on all datasets. The reduction of the number of candidates makes the execution time of the algorithm greatly reduced and the search space is greatly reduced. Moreover, with low *maxPer* and low *maxLa*, fewer candidates will be generated.

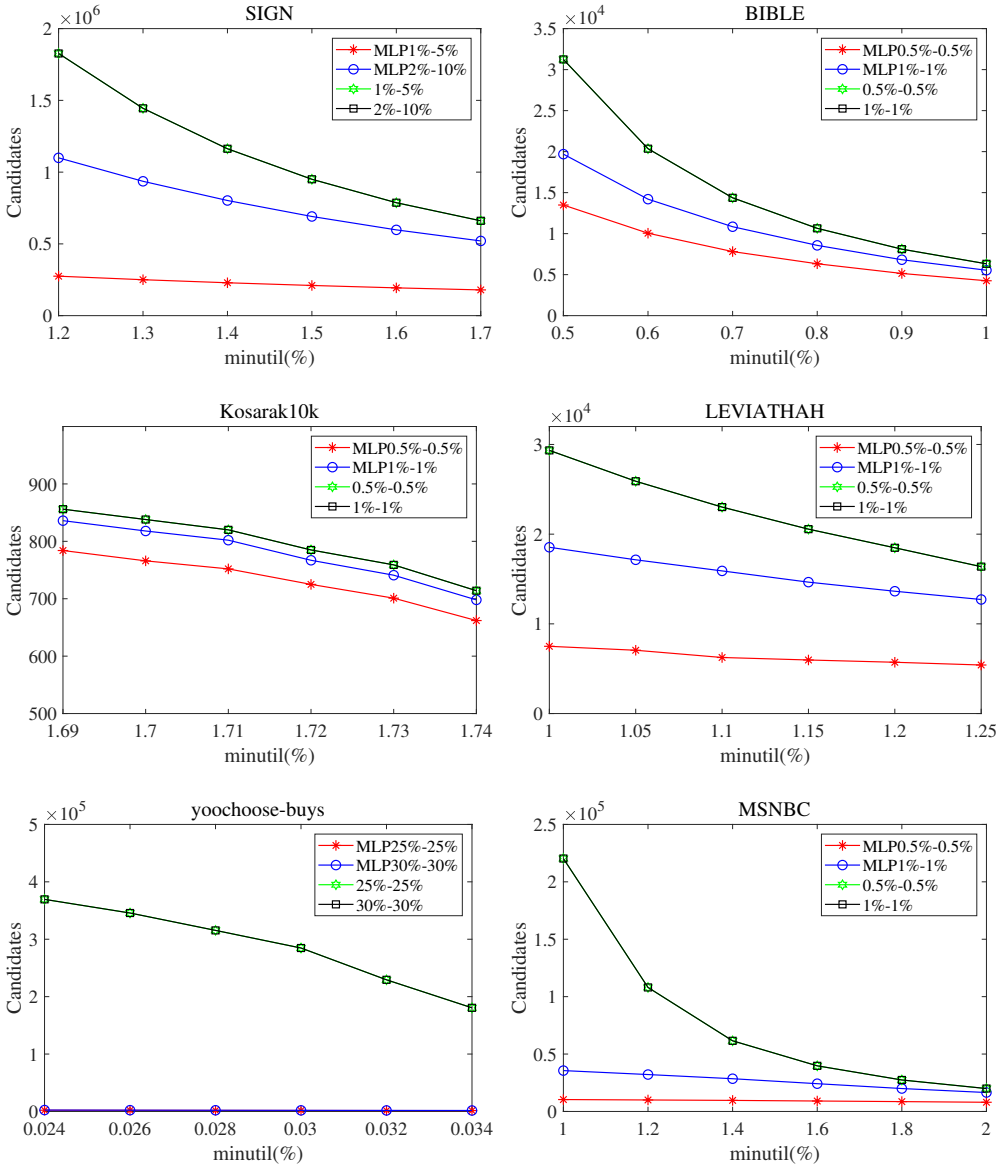

Figure 4. Candidates for different parameter values. ($maxPer - maxLa(P - L)$).

6. Conclusions

In this paper, the problem of mining stable periodic high-utility sequential patterns is defined, and the stability method is proposed for the first time in the mining of high-utility sequential patterns. In the HUSPM research area, this method provides a more flexible, precise decision-making mining strategy. In application areas, this method can be widely used in pattern discovery and knowledge discovery-related fields, such as research on consumer habits, website click-through rate data analysis, financial market analysis, biomedical applications, and mobile computing. To efficiently discover all SPHUSPs, an efficient SPHUSPM algorithm is designed. After experimental verification, the SPHUSPM algorithm can not only mine the sequence patterns ignored by the traditional algorithm, but also ensure that the mined sequence patterns show stable periodic characteristics in databases. At the same time, the addition of the PUL-list structure and the MLPS strategy accelerates the access speed and reduces the search space, so that the algorithm can ensure that the required sequence pattern can be mined, while also improving the mining efficiency and memory usage efficiency.

In future work, we will introduce the non-redundant strategy [17] and the negative utility strategy [41] based on this algorithm. These strategies are used to improve the accuracy of decision making and expand application scenarios.

Author Contributions: Conceptualization, S.X.; methodology, S.X.; software, S.X.; validation, L.Z.; formal analysis, L.Z.; investigation, L.Z.; resources, L.Z.; data curation, S.X.; writing—original draft preparation, S.X.; writing—review and editing, L.Z.; visualization, S.X.; supervision, L.Z.; project administration, L.Z.; funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was partly supported by the National Natural Science Foundation of China (61806105, 62076143, and 61906104) and the Natural Science Foundation of the Shandong Province (ZR2019BF018, ZR2021QF059).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PFPM	Periodic Frequency Pattern Mining
PPFPM	Partial Periodic Frequency Pattern Mining
SPFPM	Stable Periodic Frequent Pattern Mining
HUIM	High-Utility Itemset Mining
HUSPM	High-Utility Sequential Pattern Mining
HUSPs	High Utility Sequential Patterns
PHUSPM	Periodic High Utility Sequential Pattern Mining
PHUSPs	Periodic High Utility Sequential Patterns
SPHUSPM	Stable Periodic High Utility Sequential Pattern Mining
SPHUSPs	Stable Periodic High Utility Sequential Patterns
minutil	minimum utility threshold
maxPer	maximum periodicity threshold
minPer	minimum periodicity threshold
avgPer	average periodicity threshold
maxLa	maximum lability threshold
UL-list	utility-linked-list
PUL-list	period-utility-linked-list
LS-tree	Lexicographic Sequence Tree
SWU	Sequence Weighted Utilization
PEU	Prefix Extension Utility
RSU	Reduced Sequence Utility
SEU	Sequence Extended Utility
MPP	Maximum Periodic Pruning
MLP	Maximum Lability Pruning
MLPS	Maximum Lability Pruning in sequential pattern mining
IPS	Irrelevant Item Pruning Strategy
LAS	Look Ahead Strategy

References

1. Zhou, L.; Liu, Y.; Wang, J.; Shi, Y. Utility-based web path traversal pattern mining. In Proceedings of the Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), Omaha, NE, USA, 28–31 October 2007; pp. 373–380.
2. Truong-Chi, T.; Fournier-Viger, P. A survey of high utility sequential pattern mining. In *High-Utility Pattern Mining*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 97–129.

3. Yin, J.; Zheng, Z.; Cao, L. USpan: An efficient algorithm for mining high utility sequential patterns. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Beijing, China, 12–16 August 2012; pp. 660–668.
4. Wang, J.Z.; Huang, J.L.; Chen, Y.C. On efficiently mining high utility sequential patterns. *Knowl. Inf. Syst.* **2016**, *49*, 597–627. [CrossRef]
5. Ishita, S.Z.; Ahmed, C.F.; Leung, C.K. New approaches for mining regular high utility sequential patterns. *Appl. Intell.* **2022**, *52*, 3781–3806. [CrossRef]
6. Ahmed, C.F.; Tanbeer, S.K.; Jeong, B.S. A Novel Approach for Mining High-Utility Sequential Patterns in Sequence Databases. *ETRI J.* **2010**, *32*, 676–686. [CrossRef]
7. Yin, J.; Zheng, Z.; Cao, L.; Song, Y.; Wei, W. Efficiently mining top-k high utility sequential patterns. In Proceedings of the 2013 IEEE 13th international Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 1259–1264.
8. Lan, G.C.; Hong, T.P.; Tseng, V.S.; Wang, S.L. Applying the maximum utility measure in high utility sequential pattern mining. *Expert Syst. Appl.* **2014**, *41*, 5071–5081. [CrossRef]
9. Alkan, O.K.; Karagoz, P. CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2645–2657. [CrossRef]
10. Gan, W.; Lin, J.C.W.; Zhang, J.; Chao, H.C.; Fujita, H.; Philip, S.Y. ProUM: High utility sequential pattern mining. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 767–773.
11. Gan, W.; Lin, J.C.W.; Zhang, J.; Fournier-Viger, P.; Chao, H.C.; Philip, S.Y. Fast utility mining on sequence data. *IEEE Trans. Cybern.* **2020**, *51*, 487–500. [CrossRef]
12. Ahmed, C.F.; Tanbeer, S.K.; Jeong, B.S. Mining high utility web access sequences in dynamic web log data. In Proceedings of the 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, London, UK, 9–11 June 2010; pp. 76–81.
13. Shie, B.E.; Yu, P.S.; Tseng, V.S. Mining interesting user behavior patterns in mobile commerce environments. *Appl. Intell.* **2013**, *38*, 418–435. [CrossRef]
14. Zihayat, M.; Davoudi, H.; An, A. Top-k utility-based gene regulation sequential pattern discovery. In Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Shenzhen, China, 15–18 December 2016; pp. 266–273.
15. Dinh, T.; Huynh, V.N.; Le, B. Mining periodic high utility sequential patterns. In Proceedings of the Asian Conference on Intelligent Information and Database Systems, Kanazawa, Japan, 3–5 April 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 545–555.
16. Dinh, D.T.; Le, B.; Fournier-Viger, P.; Huynh, V.N. An efficient algorithm for mining periodic high-utility sequential patterns. *Appl. Intell.* **2018**, *48*, 4694–4714. [CrossRef]
17. Afriyie, M.K.; Nofong, V.M.; Wondoh, J.; Abdel-Fatao, H. Mining non-redundant periodic frequent patterns. In Proceedings of the Asian Conference on Intelligent Information and Database Systems, Phuket, Thailand, 23–26 March 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 321–331.
18. Amphawan, K.; Surarerks, A.; Lenca, P. Mining periodic-frequent itemsets with approximate periodicity using interval transaction-ids list tree. In Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining, Phuket, Thailand, 9–10 January 2010; pp. 245–248.
19. Fournier-Viger, P.; Lin, C.W.; Duong, Q.H.; Dam, T.L.; Ševčík, L.; Uhrin, D.; Voznak, M. PFPM: Discovering periodic frequent patterns with novel periodicity measures. In Proceedings of the 2nd Czech-China Scientific Conference 2016, Ostrava, Czech Republic, 7 June 2016; IntechOpen: London, UK, 2017.
20. Kiran, R.U.; Venkatesh, J.; Fournier-Viger, P.; Toyoda, M.; Reddy, P.K.; Kitsuregawa, M. Discovering periodic patterns in non-uniform temporal databases. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Chengdu, China, 16–19 May 2022; Springer: Berlin/Heidelberg, Germany, 2017; pp. 604–617.
21. Fournier-Viger, P.; Yang, P.; Lin, J.C.W.; Kiran, R.U. Discovering stable periodic-frequent patterns in transactional data. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Kitakyushu, Japan, 19–22 July; Springer: Berlin/Heidelberg, Germany, 2019; pp. 230–244.
22. Fournier-Viger, P.; Wang, Y.; Yang, P.; Lin, J.C.W.; Yun, U.; Kiran, R.U. Tspin: Mining top-k stable periodic patterns. *Appl. Intell.* **2022**, *52*, 6917–6938. [CrossRef]
23. Gan, W.; Lin, J.C.W.; Fournier-Viger, P.; Chao, H.C.; Hong, T.P.; Fujita, H. A survey of incremental high-utility itemset mining. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1242. [CrossRef]
24. Fournier-Viger, P.; Wu, C.W.; Zida, S.; Tseng, V.S. FHM: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In Proceedings of the International Symposium on Methodologies for Intelligent Systems, Limassol, Cyprus, 29–31 October 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 83–92.
25. Lin, C.W.; Hong, T.P.; Lu, W.H. An effective tree structure for mining high utility itemsets. *Expert Syst. Appl.* **2011**, *38*, 7419–7424. [CrossRef]
26. Lin, Y.C.; Wu, C.W.; Tseng, V.S. Mining high utility itemsets in big data. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Chengdu, China, 16–19 May 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 649–661.

27. Liu, M.; Qu, J. Mining high utility itemsets without candidate generation. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, 29 October–2 November 2012; pp. 55–64.
28. Yun, U.; Ryang, H.; Ryu, K.H. High utility itemset mining with techniques for reducing overestimated utilities and pruning candidates. *Expert Syst. Appl.* **2014**, *41*, 3861–3878. [CrossRef]
29. Zida, S.; Fournier-Viger, P.; Lin, J.C.W.; Wu, C.W.; Tseng, V.S. EFIM: A highly efficient algorithm for high-utility itemset mining. In Proceedings of the Mexican International Conference on Artificial Intelligence, Mexico City, Mexico, 25–30 October 2015; Springer: Berlin/Heidelberg, Germany, 2015; pp. 530–546.
30. Amphawan, K.; Lenca, P.; Surarerks, A. Mining top-k periodic-frequent pattern from transactional databases without support threshold. In Proceedings of the International Conference on Advances in Information Technology, Bangkok, Thailand, 1–5 December 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 18–29.
31. Kiran, R.U.; Kitsuregawa, M.; Reddy, P.K. Efficient discovery of periodic-frequent patterns in very large databases. *J. Syst. Softw.* **2016**, *112*, 110–121. [CrossRef]
32. Surana, A.; Kiran, R.U.; Reddy, P.K. An efficient approach to mine periodic-frequent patterns in transactional databases. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Shenzhen, China, 24–27 May 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 254–266.
33. Tanbeer, S.K.; Ahmed, C.F.; Jeong, B.S.; Lee, Y.K. Discovering periodic-frequent patterns in transactional databases. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Bangkok, Thailand, 27–30 April 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 242–253.
34. Han, J.; Dong, G.; Yin, Y. Efficient mining of partial periodic patterns in time series database. In Proceedings of the 15th International Conference on Data Engineering (Cat. No. 99CB36337), Sydney, NSW, Australia, 23–26 March 1999; pp. 106–115.
35. Yu, X.; Yu, H. An asynchronous periodic sequential patterns mining algorithm with multiple minimum item supports. In Proceedings of the 2014 Ninth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Guangzhou, China, 8–11 November 2014; pp. 274–281.
36. Fournier-Viger, P.; Lin, J.C.W.; Duong, Q.H.; Dam, T.L. PHM: Mining periodic high-utility itemsets. In Proceedings of the Industrial Conference on Data Mining, New York, NY, USA, 13–17 July 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 64–79.
37. Lin, J.C.W.; Zhang, J.; Fournier-Viger, P. High-utility sequential pattern mining with multiple minimum utility thresholds. In Proceedings of the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data, Guangzhou, China, 23–25 August 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 215–229.
38. Lin, J.C.W.; Zhang, J.; Fournier-Viger, P.; Hong, T.P.; Zhang, J. A two-phase approach to mine short-period high-utility itemsets in transactional databases. *Adv. Eng. Inform.* **2017**, *33*, 29–43. [CrossRef]
39. Ayres, J.; Flannick, J.; Gehrke, J.; Yiu, T. Sequential pattern mining using a bitmap representation. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, 23–26 July 2002; pp. 429–435.
40. Fournier-Viger, P.; Lin, J.C.W.; Gomariz, A.; Gueniche, T.; Soltani, A.; Deng, Z.; Lam, H.T. The SPMF open-source data mining library version 2. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Riva del Garda, Italy, 19–23 September 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 36–40.
41. Dong, X.; Gong, Y.; Cao, L. e-RNSP: An efficient method for mining repetition negative sequential patterns. *IEEE Trans. Cybern.* **2018**, *50*, 2084–2096. [CrossRef] [PubMed]