ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20220205155952
Original scientific paper

Sequential Pattern Mining with Multidimensional Interval Items

Bob CHEN, Weiming PENG, Jihua SONG*

Abstract: In real sequence pattern mining scenarios, the interval information between two item sets is very important. However, although existing algorithms can effectively mine frequent subsequence sets, the interval information is ignored. This paper aims to mine sequential patterns with multidimensional interval items in sequence databases. In order to address this problem, this paper defines and specifies the interval event problem in the sequential pattern mining task. Then, the interval event items framework is proposed to handle the multidimensional interval event items. Moreover, the MII-Prefixspan algorithm is introduced for the sequential pattern with multidimensional interval event items mining tasks. This algorithm adds the processing of interval event items in the mining process. We can get richer and more in line with actual needs information from mined sequence patterns through these methods. This scheme is applied to the actual website behaviour analysis task to obtain more valuable information for web optimization and provide more valuable sequence pattern information for practical problems. This work also opens a new pathway toward more efficient sequential pattern mining tasks.

Keywords: data mining; item intervals; prefixspan; sequential pattern mining

1 INTRODUCTION

Sequential pattern mining is a very important and common research topic in pattern recognition tasks. The goal of sequential pattern mining is to mine frequently occurring subsequences in a given sequence database. Such problems can be solved effectively using the most commonly used algorithms, such as GSP [1], Prefixspan [2], SPADE [3], and VGEN [4], and the resulting FAST [5] and VMSP [6] algorithms. Recent studies have also extended the original sequential pattern mining problem with more mining problems, such as Rule Mining [7-9] and High-Utility Pattern Mining [10].

These algorithms can be employed to mine sequential patterns in real-world scenarios, such as the most common shopping sequence mining problem. The most frequent shopping sequence is mined through many shopping records. Sellers can provide product recommendation services based on this result. Take the shopping sequence <{A}, {B}, {C}, {DE}, {FG}> as an example. Assuming that {A}{B}{C}{D}{E}{F}{G} denote different products, the customer has gone on to purchase products B, C, and DE after purchasing product A. It can be concluded that customers who have purchased product A may be interested in product B and have a higher probability of purchasing product B.

However, in real scenarios, the sequential patterns found by existing algorithms are very coarse and miss much information about interval events. The simplest interval events are time intervals. For example, two customers have a sequence of <{A}, {B}> shopping records. The difference is that there is only one month between customer 1's purchases of A and B, while customer 2 has a year between purchases of A and B. In this case, customers 1 and 2 behave differently. These two scenarios should be considered as two different sequential patterns. In a more complex scenario, many intervening events may occur between the two shopping events. For example, the customer may have experienced a week-long promotion between the two shopping events {A}{B}, or they both have seen advertisements for the product in question more than 2 times, or they have both had a subsequent purchase after some product upgrade. None of these scenarios is related to a simple time interval. These interval events should be considered while generating sequential patterns.

Some sequential pattern mining algorithms have also incorporated conditional restrictions on gaps [11]. For example, in the sequential pattern mining algorithms, gap restrictions are employed to filter sets of sequences with excessively long gaps [12]. The gaps between itemsets in the sequential pattern results are not too significant. The difference between the previously-mentioned interval events and the gap is that an interval event is a set of attributes in the original sequence rather than simply a count of gaps between itemsets. Besides, the frequent itemsets are not simply filtered based on the gap counts during mining but as part of the frequent itemsets. For example, the final mined sequential pattern should be <{A}, [5], {B}, [1], {C}>, <{A}, [1], {B}, [4], {C}>. The counts in square brackets indicate the interval items between two itemsets.

In the algorithm proposed by Yu et al. [12], time intervals have been processed. Although sequential patterns with time intervals can be generated using this algorithm, its processing method only considers the time interval between two itemsets. More itemset interval events mining sequential patterns should be considered in practical situations instead of simply connecting two itemsets by a time interval. This interval item is also counted as a factor in sequential pattern mining. For example, we expect that the mined sequential patterns should be <{A}, [one month, one promotion], {B}> or <{A}, [1-year, product upgrade], {B}>.

Take web behavior analysis as another example. After the user clicks button A, the user clicks button B five minutes later. We cannot just focus on the 5-minute interval. We should focus more on what happens after clicking button A and clicking button B. The user may watch a web video or leave the computer and do something else. The sequential patterns represented in these two cases are also different.

In order to address the mentioned challenges, the sequential pattern is mined using multidimensional interval items. Specifically, the following works are performed: 1) The interval event problem is defined and clarified in sequential pattern mining tasks. 2) The interval item event framework is proposed, which enables the processing of multidimensional interval event items. We do not confuse interval events with user behavior events because these two events have different dimensions. It can also be understood that user behavior events are the objectives, while interval events are methods. 3) A sequential pattern mining algorithm containing multi-dimensional interval event items is proposed to mine sequential patterns through more interval event information for the existing sequence database. Finally, the task of sequential pattern mining containing multi-dimensional interval events is achieved. We can solve the mining requirements for interval information in practical scenarios based on the proposed methods. In the implementation process, the extensibility of the scheme is considered, and an easily extensible framework is provided. This scheme is applied to the actual website behavior analysis task, and more detailed and valuable information is obtained. It provides strong support for website product optimization and product problem discovery.

2 RELATED WORK

2.1 Basic Algorithms

Various stable and efficient sequential pattern mining algorithms have been proposed recently. These sequential pattern mining algorithms can be divided into three categories: the Breadth-first search, the Depth-first search, and the Pattern-growth algorithms. Most sequential pattern algorithms are extensions of these three algorithms[13].

The Breadth-first search algorithm represented by AprioriAll [14] and GSP [1] sequentially scans the sequence database for frequent subsequences such as 1-sequences, 2-sequences, and 3-sequences, until no more frequent subsequences of higher order can be generated. This type of algorithm is logically simple, but has a large search space. In the worst case, the efficiency of the algorithm is also significantly reduced. Based on such algorithms, some extensions use various pruning strategies to reduce their search space.

The Depth-first search algorithms, represented by algorithms such as Spade [3], Spam [15], CM-Spam [16], and Fast [5], are explored using a different search order than the Breadth-first search algorithm. Sequences containing single items in the sequence database, are first searched and then larger sequences are generated by i-extensions and s-extensions operations [13]. This is performed recursively until the extension operation cannot be continued. This type of algorithm also suffers from the problem of large search space, so there are more pruning schemes to optimize the original algorithm.

The Pattren-growth algorithm represented by algorithms such as FreeSpan [17] and PrefixSpan [2] introduces the concept of a projection databases to avoid recursive scanning of the original sequence database [11]. However, creating a large number of projection databases may cause excessive memory usage. In order to solve this problem, the researchers further proposed "pseudo-projection" [18], which does not create a new projection database, but only searches the projection database by keeping the original database pointer.

2.2 Extended Algorithms

Yu et al. solved the problem of sequential pattern mining by using time interval information [12]. By adding time intervals to pseudo items, pattern mining is performed with both item gaps and time intervals. This scheme can partially solve the item interval problem proposed in this paper, but it is still relatively simple to measure the interval of two itemsets by the time interval. In real scenarios, more complex and diverse interval events should be addressed.

Extended algorithms adapt the existing algorithm to improve their performance. For example, the concepts of closed sequential patterns [19-22] and maximal sequential patterns [23-26] have been proposed to reduce the output of too many sequential pattern sets. Various constraints have been added to the search based on practical needs, such as gaps and length restrictions.

There are also some extensions on topics related to the context of the problem presented in this paper, such as weighted sequential pattern mining [27-28] and high-utility sequential pattern mining [10]. Such concepts are introduced to identify more valuable information from the original sequence database in real scenarios. Weighted sequential pattern mining indicates that the itemset in a sequence database should contain different weights and should not be generalized [13]. The efficient use of sequential patterns proposes that the set of items in a sequence should have a notion of quantity [13]. For example, in a shopping analysis scenario, some products are purchased in large quantities and others in small quantities, influencing the final sequential pattern output. Although the specific treatments and problems differ, similar to the problem presented in this paper, the context of these algorithms is in the application of sequential pattern mining algorithms to the analysis of real scenarios. Simple sequence mining is rough, and many details are left out. Various improvements should be made based on specific scenarios to meet the practical requirements.

In recent years, there are still many related researches. Fournier-Viger et al. proposed the MRCPPS algorithm [31] for periodic-frequent pattern mining. The ProUM algorithm [32] was proposed to improve the mining efficiency based on the projection technique. Van et al. proposed MSPIC-DBV algorithm [33] to solve the problem of mining sequential patterns with itemset constraints. DSPM-MTC [34] try to solve false positives problem in discriminative sequential patterns mining task. HANP-Miner algorithm [35] involves support calculation and candidate pattern reduction steps for nonoverlapping sequential pattern mining task. Gan et al. proposed a method named HUSP-ULL [36] to solve High-utility sequential pattern task more efficiently. Huynh et al. proposed pseudo-IDLists structure for clickstream pattern mining scenario [37]. Kim et al. proposed RF-MINER [38] an effective prediction method for recency-based sequential patterns mining task.

2.3 PrefixSpan Algorithm

The algorithm proposed in this paper is based on the Prefix-Projected Pattern Growth (PrefixSpan algorithm) [2]. We first describe the PrefixSpan algorithm here. This algorithm introduces the two concepts of prefix and projection [2]. The prefix refers to the subsequence of the previous part of the sequence data. For both sequence A = {a1, a2, ..., an} and sequence B = {b1, b2, ..., bm}, n ≤ m, which satisfies a1 = b1, a2 = b2, ..., an – 1 = bn − 1, and an⊆bn, then A is said to be a prefix of B. In the sequence data B = <{BG}{A}{DG}{AC}>, A = <{G}{A}{D}>, then A is a prefix of B. The Prefix projection refers to the remaining part of the sequence after removing the prefix, as shown in Tab. 1:

Table 1 Example of sequence projection database

Prefix	Prefix projection
<A>	<{DG}{AC}>
<AD>	<{G}{AC}>
<AG>	<{D}{AC}>

Where "_" is a placeholder for the prefix corresponding to that projection. Since the PrefixSpan algorithm does not need to generate candidate sequences, the projection database converges quickly, and the memory consumption is relatively stable, so it significantly improves the efficiency of performing sequential pattern pattern mining tasks. Fig. 1 shows the flowchart of the Prefixspan algorithm in sequential pattern mining.

In some extensions of Prefixspan algorithms, conditional constraints such as gap filtering and sequence length filtering are added. However, the algorithm flow of Prefixspan shows that a frequent subsequence, such as <{ab}c>, can be obtained in the mining process of sequential patterns. And this sequential pattern also hides some information, such as the interval information between the itemsets to be solved in this paper. For example, there is a period of 1-month between {ab} and {c}, or there is a specific event. Corresponding to these two cases, they should belong to different sequential patterns. The gap constraint introduces a gap count between two item sets, but the gap between two item sets is only counted and filtered when the projection database is generated, and it is not possible to generate different sequential patterns for different interval items. In addition, the information about the interval items is not reflected in the final sequential pattern results. As a result, this problem cannot be solved by simple restrictions and extensions of other algorithms. The overall flow of the PrefixSpan algorithm is shown in Fig. 2:

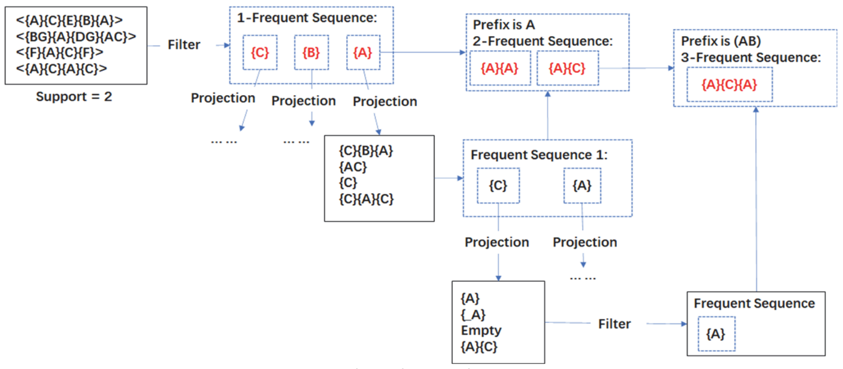

Figure 1 The flow of the Prefixspan algorithm

```
Input: A sequence database S, and the minimum support threshold min_sup
Output: The complete set of sequential patterns
Method: Call PrefixSpan(‹›, 0, S)
SubroutinePrefixSpan($\alpha$, $l$, $S|_\alpha$)
Parameters:$\alpha$: a sequential pattern; $l$ the length of $\alpha$; $S|_\alpha$: α-projected database, $\alpha \neq$ ‹›; otherwise, the sequence database $S$.
Method:
    1. Scan $S|_\alpha$ once, find the set of frequent items $b$ such that
    (a) $b$ can be combined with the last element of $\alpha$ to form a sequential pattern; or
    (b) ‹$b$› can be attached to $\alpha$ to form a sequential pattern.
    2. For each frequent item $b$, append it to $\alpha$ to form a sequential pattern $\alpha'$, and output $\alpha'$;
    3. For each $\alpha'$, construct the $\alpha'$-projected database $S|\alpha'$, and call PrefixSpan($\alpha'$, $l+1$, $S|\alpha'$)
```

Figure 2 Pseudocode for PrefixSpan [29]

3 METHOD

In order to solve the above problem of missing interval information, we first need to define a data format that contains interval items. Secondly, a new algorithm needs to be proposed to process the input sequence data based on the existing sequential pattern mining algorithms. Finally, a sequential pattern format containing interval items is defined and the corresponding sequential patterns are generated by a pre-defined threshold value.

3.1 Preliminaries

Definition 1: Sequence

A sequence is a complete information flow, which can be a user's historical purchase record, web browsing record, or other data source with sequential correlation.

Definition 2: Subsequence

For the sequence A ={a1, a2, ..., an} and the sequence B = {b1, b2, ..., bm}, n ≤ m, if there is a sequence of numbers 1 ≤ j1 ≤ j2 ≤, …, ≤ jn ≤ m , satisfying a1⊆bj1, a2⊆bj2, ..., an⊆bjn, then A is a subsequence of B.

Definition 3: Sequence Database

It contains a complete set of sequence data. It should be noted that all sequences in the database need to be generated in the same context and under the same conditions.

Definition 4: Support

The support of sequence α is the proportion of α in the sequence database, which is denoted as Support(α).

Definition 5: Sequential pattern

Sequential patterns are subsequences that frequently appear in the sequence database. Given a minimum support threshold ξ, a sequence α is called a sequential pattern if the support of the sequence α in the sequence database is not lower than ξ.

Definition 6: Sequential pattern mining

To find out all frequent subsequences, i.e., the subsequence whose frequency of occurrence in the set of sequences is less than the minimum support threshold.

Definition 7: Sequence item, interval item

In order to distinguish between ordinary and interval items in a sequence, we propose the concept of sequence items and interval items where a "sequence item" is an ordinary item in a regular sequence mining task." An "interval item" is used to represent the event interval between two item sets. For example, the items purchased in a shopping sequence are sequence items, while other events between two shopping actions are defined as interval items. Interval items can be time intervals, promotional actions and advertising actions.

3.2 Problem Statement

Our goal is to address the following questions:

The input is a sequence database (including interval items), with minimum support. In the process of sequential pattern mining, the information about interval items between sets of items is integrated to eventually produce a sequential pattern containing interval items.

For example, the given sequence database is shown in Tab. 2. The letters in curly brackets represent the sequence items which are processed in the traditional sequential pattern mining task. The letters in square brackets represent the interval items proposed in this paper. The minimum support is 2. Then it produces a sequential pattern like <{a}[AA]{c}>, where "AA" in square brackets represents the two "A" events between item {a} and item {C}. Although the sequence <{a}{b}> appears many times in the database, because the interval items count between {a} and {b} do not reach the minimum support threshold, <{a}{b}> is an infrequent subsequence.

Table 2 Example of Sequence Database with interval items

SID	Sequence
1	<{a}[A]{b}[AB]{c}{d}>
2	<{ac}{b}[A]{e}[A]{c}>
3	<{b}{a}[AA]{c}{b}{d}>
4	<{e}{b}{a}{d}{b}>

3.3 Input and Output Data Format Definition

The current general format for storing sequences is: a –1 b c – 1 d – 2. Each item is separated by a space, with –1 representing the separation of each itemset, and –2 representing the end of the entire sequence [31]. The sequence corresponding to the above example is represented in parentheses as <{a}{bc}{d}>. In this paper, the interval item between each sequence item set needs to be added to the input sequence. We still use the format box above and add the interval items to item –1. We use –3 to separate the interval items. When an item value equals –1, if the immediately following items are surrounded by –3, then those items are interval items. For example: a –1 –3 A B C – 3 b c – 1 2 d, this means that the interval items between a and bc are {ABC}. There is no itemset surrounded by –3 between {bc} and {d}, which means that there are no interval items set between {bc} and {d}. If represented in parentheses, it is <{a}[ABC]{bc}{d}>, where the items in the square brackets is the interval item.

We use a letter here to represent the interval items. They will be uniformly converted to positive integer ids when processed by the algorithm to improve the algorithm's processing speed. Like sequence items, each interval item is an index of real events. For example, we can assume that "A" represents a one-day interval, "B" represents a 1-year interval, "C" represents a promotion, and "D" represents a product upgrade.

For the output data, we also use a similar format definition of the input data. In particular, it is important to note that the interval items in the output data are cumulative of the interval items in the meta-sequence. For example, suppose the original sequence <{a}[ABC]{bc}[A]{d}>. The pattern of extracted sequence is <{a}[AABC]{d}>, where [AA] means that between {a} and {d}, are two A events. These two A events are accumulated as each sequence item is removed during the mining process.

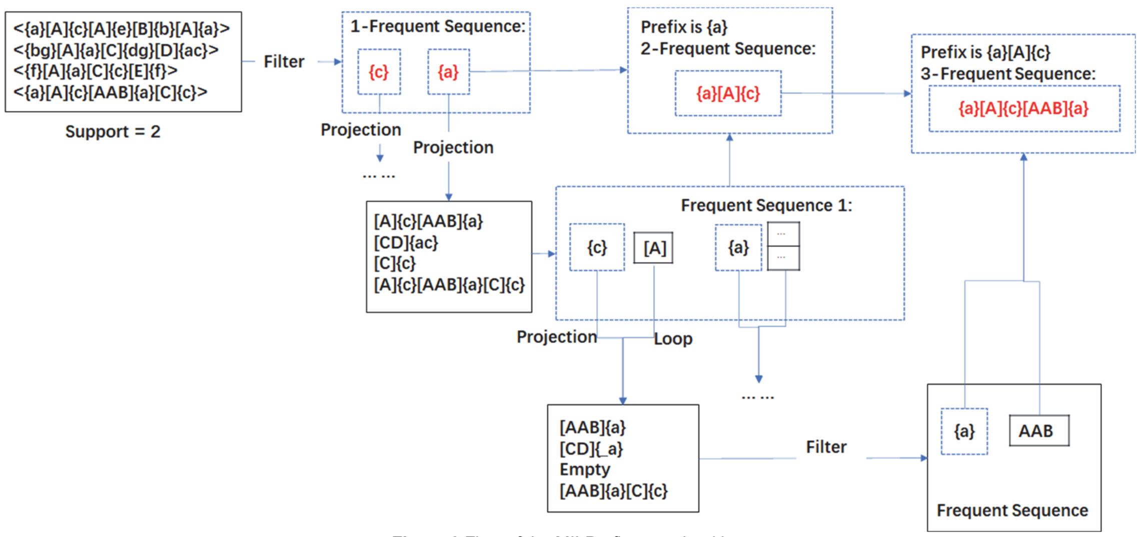

Figure 3 Flow of the MII-Prefixspan algorithm

3.4 MII-Prefixspan

To solve the problem mentioned in this paper, we improve on the existing algorithm by adding the processing of interval items. Prefixspan is a typical Pattern-growth algorithm which has three features: 1) a projection database is used in the algorithm for recursive level-by-level queries; 2) the traversal of the algorithm is essentially a depth-first search. The projection database is generated based on the sequence database, so the sequential patterns generated by the algorithm are present in the sequence database [13].

3) the algorithm is logically clear, simple and easy to extend. Based on the above three features, this paper modifies the prefixspan algorithm based on the following: 1) being capable of identifying and processing interval items in the source sequence database; 2) considering interval items in the process of generating sequential patterns; and 3) adding interval items to the generated sequential patterns. Fig. 3 shows the flowchart of the MII-Prefixspan algorithm in sequential pattern mining.

1) Input parameters

The original Prefixspan algorithm contains two parameters: the original sequence database and the minimum support. There are also extended algorithms that add c. gap constraints; d. sequential pattern length constraints.

At this stage, we do not consider the extended restriction parameter. The difference with the prefixspan algorithm is that the original sequence as input should contain interval items. Further restriction parameters are discussed in section 4.3.

2) Interval item record in projection database

One of the innovations of the Prefixspan algorithm is the introduction of the concept of a projection database. By constructing a projection database with different prefixes, the purpose of projection search for sequences is achieved. This can effectively reduce the search scale of the sequence database. An extended algorithm introduces pseudo-projections to reduce the memory usage during the search.

In order to record and process the interval items, we need to record interval items when building the projection database. Therefore, in the pseudo-projection structure, an interval item variable of type List<String> is added to record interval items.

3) Interval items processing

In the first step of the Prefixspan algorithm, prefix items of length 1 that occur more than the minimum support are first selected. Then the original sequences in the sequence database are processed according to the selected prefix items: 1) removing the sequences that do not contain frequent prefixes, and 2) removing the infrequently occurring items in the sequence.

In the MII-Prefixspan algorithm, when we scan the sequence and infrequently occurring items, we record the interval items to the deleted items. Finally, these interval items are accumulated. The true interval item between two items will then be obtained. For example, the original sequence is <{a}[AB]{b}[A]{cd}[A]{e}> where both b and cd are infrequently occurring items, and the processed sequence is: <{a}[AAAB]{e}>. Similarly, when constructing the projection database, it is also necessary to add the corresponding interval items of the infrequent items that are removed and store them.

After obtaining the frequent prefixes, a projection database needs to be constructed for each frequent prefix. The projection database is processed recursively for each prefix. During each recursion, a new projection database of frequent prefixes is generated, and each frequent prefix generated recursively is combined to produce the final sequential pattern.

We need to add a processing step for the interval items before each recursion. We first calculate the interval items between the items to be processed this time before each recursion, and then loop through the values of each interval items. In addition to passing the projection database and prefix information to the next-level of the recursive function, we also need to pass the interval item corresponding to that projection database.

For example, for a sequence database shown in Tab. 3 item {a} is a frequent item. When we construct the projection database for {a}, we get the projection database shown in Tab. 4.

Table 3 Example of sequence database with interval items

SID	Sequence
1	<{a}[A]{c}[A]{e}[B]{b}[A]{a}>
2	<{bg}[A]{a}[C]{dg}[D]{ac}>
3	<{f}[A]{a}[C]{c}[E]{f}>
4	<{a}[A]{c}[E]{a}[C]{c}>

Table 4 Projection database of {a}

Prefix	Prefix projection
{a}	[A]{c}[AAB]{a}
{a}	[CD]{ac}
{a}	[C]{c}
{a}	[A]{c}[E]{a}[C]{c}

At this time, there are three interval item pairs: [A], [CD] and [C]. Assuming a minimum support degree of 50%, the interval items [A] are obtained respectively. At this time, we continue the recursive process using the interval item [A].

```
Input: A sequence database $S$, and the minimum support threshold $min\_sup$
Output: The complete set of sequential patterns
Method: Call MII-PrefixSpan(‹›, 0, $S$, ‹›)
Subroutine MII-PrefixSpan($\alpha$, $l$, $S|_\alpha$, $i$)
Parameters: $\alpha$: a sequential pattern; $l$ the length of $\alpha$; $S|_\alpha$: α-projected database, $\alpha \neq$ ‹›; otherwise, the sequence database $S$. $i$:the interval item before $\alpha$, $\alpha \neq$ ‹›;
Method:
    1. Scan $S|_\alpha$ once, find the set of frequent items $b$ such that
    (a) $b$ can be assembled to the last element of $\alpha$ to form a sequential pattern; or
    (b) ‹$b$› can be appended to $\alpha$ to form a sequential pattern.
    2. For each frequent item $b$ and interval items $i$ , append them to $\alpha$ to form a sequential pattern $\alpha'$ , and output $\alpha'$;
    3. For each frequent item $b$, scan $S|_\alpha$ for finding all interval items which the count larger than $min\_sup$. Build frequent interval items $f$;
    4. For each $f$ and α', construct $\alpha'$-projected database $S|\alpha'$, and call MII-PrefixSpan($\alpha'$, $l+1$, $S|\alpha'$, $f$)
```

Figure 4 MII-PrefixSpan pseudocode

4) Output

During the traversal recursion, a frequent prefix is generated each time, and this prefix is combined with each previously generated prefix to form a new sequential pattern. During generation, the recursively passed interval items are output together, finally forming a sequential pattern containing the interval items.

5) Algorithm description

The flow of the overall MII-PrefixSpan algorithm is shown in Fig. 4.

There are some differences between the Prefixspan and MII-Prefixspan algorithms: whenever an item is removed, the interval items between the corresponding items need to be updated. And in the SavePattern function, the interval items should be saved as part of the frequent pattern.

3.5 Pruning and Optimization

When the Prefixspan algorithm produces sequential patterns, there have been some extensions to filter sequential patterns by limiting the gap length. The specific count of gaps is the maximum count of gaps between the two itemsets [13]. In the GSPM [12] proposed by Yu et al., there are also four restriction solutions for the item interval. On the one hand, this can produce more meaningful sequential patterns. This is because even if the two itemsets are adjacent, if there are too many gaps, then the two itemsets may be irrelevant. On the other hand, by restricting the gap length, it also plays a role in pruning. The MaxGap or Max interval value is set by human experience and is passed into the algorithm for execution at the beginning of the algorithm. However, a simple gap count or interval restriction is often too coarse. In many cases, it is difficult to filter the set of unrelated events. For example, without detailed information about the interval events, the true interval represented by the gap or interval cannot be determined. Then it is impossible to determine whether the two events are related.

In the MII-Prefixspan algorithm, we need to further expand the restriction scheme for interval items. We designed an independent limit threshold mechanism for each interval item. At the beginning of the algorithm, we add a threshold parameter of type Map<String, int>. The value is similar to {<A, 5><B, 3>}. This means that the maximum count of interval item A between two sequence itemsets is 5, and the maximum count of occurrences of B is 3. When we process the third step of the above algorithm, in addition to comparing the frequency of each item with the minimum support, we also need to add a condition, i.e., whether the interval items count is greater than the corresponding preset MaxIntervalItem value, if it is greater than the preset value, it is returned directly. In this way, the purpose of filtering the sequential pattern with the IntervalItem count threshold is achieved in the process of generating the sequential pattern.

4 EXPERIMENTAL RESULTS

Several experiments were performed on the proposed MII-Prefixspan algorithm. These experiments were performed to evaluate the algorithm's effectiveness and performance. The algorithm effectiveness test focuses on the ability of the algorithm to find sequential patterns and tests the count of output sequential patterns under different input parameters. The algorithm performance test evaluates the algorithm's execution speed and memory consumption under different input conditions. The experiments were implemented on a PC with an Intel Core i7 processor and 16 GB of RAM.

4.1 Experiment 1: Algorithm Effect Test

The application was evaluated against a sequence of user behaviors from an internally used application for the validity assessment. This data was generated from the tracking logs on the application. It mainly records the main actions of the user while using the application, such as page scrolling, button clicks, and page jumps. The sequential pattern mining algorithm can find out the user's behavior sequences employed to optimize the application's ease of use.

Table 5 Sequential patterns from App tracking records

{ButtonAClick}[{20 sec}{Play Video 220}{10sec}]{ButtonB Click}[{5sec}]{ButtonC Click}
[{Image 2392 Load}{18sec}]{ButtonC Click}
[{PageA Load}{3sec}{PageA scroll}{16sec}]{ButtonC Click}

Tab. 5 shows the sequential patterns that were mined. {Button C Click} is a paid button. Our optimization goal is to increase the count of Button C Clicks. Based on the results, playing Video 220, Image 2392, and Page A display are all conducive to the click of Button C. Accordingly, the operation process of the app can be further optimized.

4.2 Experiment 2: The Impact of Minimum Support

Since our main goal is not to improve the execution efficiency of the algorithm, we do not compare the operational efficiency metrics too much with existing algorithms. Because the MII-Prefixspan algorithm is mainly an extension of PrefixSpan algorithm, we compare its performance with that of the PrefixSpan algorithm. We mainly test the performance difference between the two algorithms under different support values and interval item restriction.

We selected Kosarak, BMSWebView1, BMSWebView2 and Bible sequence databases for comparison experiments. The data was chosen on the basis of testing the operation of the MII-Prefixspan algorithm on sequence databases with different sizes of data and different contexts. The test dataset was downloaded from the SPMF website (http://www.philippe-fournier-viger.com/spmf/). The contents of the sequence database are as follows.

Table 6 Datasets for Sequential Pattern Mining [30]

	Sequence count	Item count	Sequence length	Description
Kosarak	990000	41270	8.1	This is a very large dataset containing 990 000 sequences of click-stream data from an hungarian news portal. The dataset was converted in SPMF format using the original data from: http://fimi.ua.ac.be/data/.
BMSWebView1	59601	497	2.42	This dataset was used in KDD CUP 2000. It contains clickstream data from an e-commerce In this dataset, there are some long sequences. For example, 318 sequences contain more than 20 items.
BMSWebView2	77512	3340	4.62	This dataset was used in KDD CUP 2000. It contains clickstream data from an e-commerce
Bible	36369	13905	21.6	This dataset is a conversion of the Bible into a sequence database (each word is an item).

(http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php)

Since none of the published sequence databases contain interval items in order to test the performance of the algorithm, we need to process the existing sequence database to simulate a sequence database containing interval items. The simulation is done by adding interval items randomly between different itemsets. The value of interval item is randomly assigned within the interval [0, 20].

We first test the effect on the execution time of the algorithm and the count of sequential patterns produced for different support thresholds. The experimental results are shown in Fig. 5 and Fig. 6. Because the sequential pattern algorithm with interval items identifies valuable sequential patterns more accurately and adds more filtering restrictions the execution time is significantly reduced. As the minimum support increases, the impact becomes smaller. Fig. 5 shows the variation in the generated sequential pattern counts. Due to the increased processing of interval item values, the number of sequential patterns is reduced compared to the prefixspan algorithm. As the value of the minimum support setting increases, the number of sequential patterns decreases.

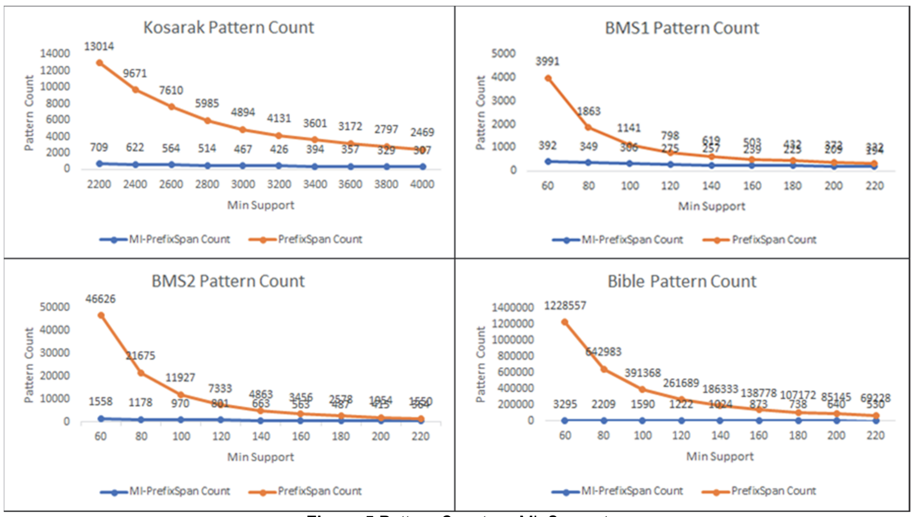

Figure 5 Pattern Count vs. MinSupport

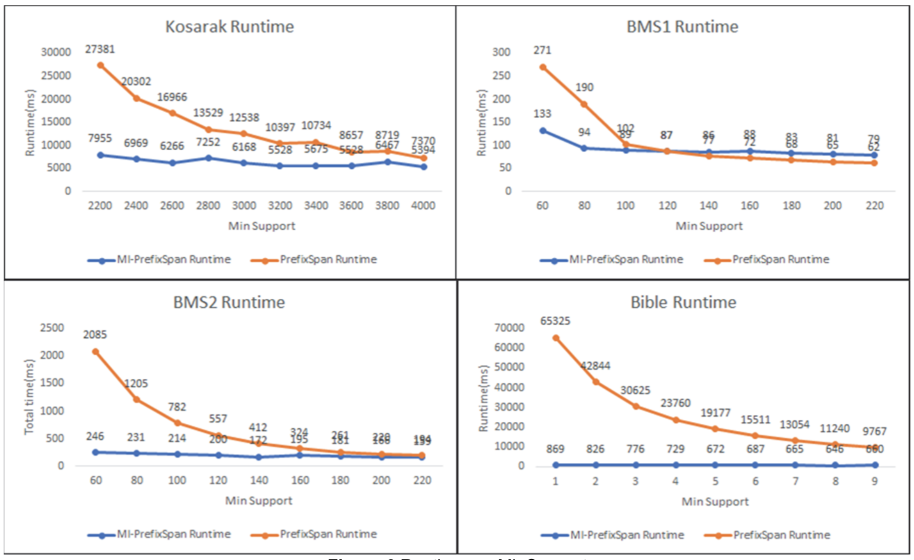

Figure 6 Runtime vs. MinSupport

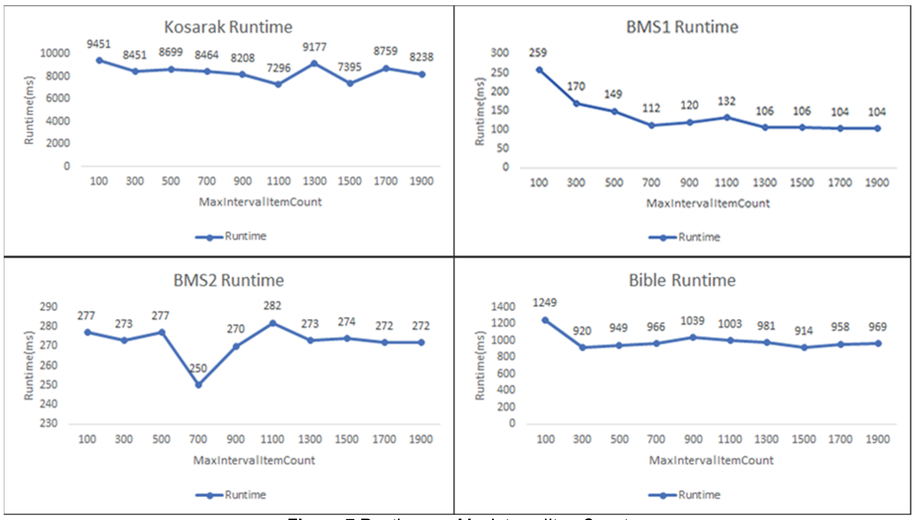

Figure 7 Runtime vs. MaxIntervalItemCount

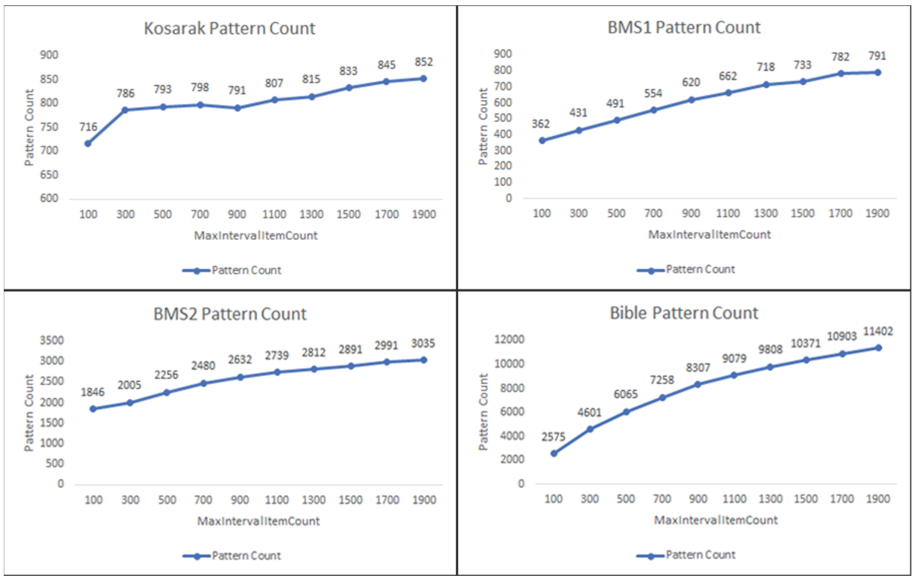

Figure 8 Pattern count vs. MaxIntervalItem count

4.3 Experiment 3: The Impact of Interval Item Constraints

In this experiment, we tested the effect of item constraints on the performance and effectiveness of the algorithm. We compared different values of maxIntervalItemCount. The results of the experiments are shown in Fig. 7 and Fig. 8. It can be seen that this constraint has a large impact on the count of sequential patterns. Since the processing is not continued without satisfying the sequential interval item count condition, the number of traversals and recursions is greatly reduced and the execution performance is improved.

Table 7 Memory consumption for Kosarak database

Min Support	Max Memory(MB)
2200	1347.33
2400	1510.69
2600	1229.73
2800	1241.15
3000	1394.27
3200	1248.05
3400	993.311
3600	1266.45
3800	1078.64
4000	1428.61

In addition, Tab. 7 records the memory consumption corresponding to each experiment. This is because the pseudo-projection method is used in the algorithm to process the projection database. The interval item values are also recorded and processed based on the pseudo-projection method, so the memory consumption is only related to the size of the original sequence database, and the memory consumption is independent of the experimental parameters and constraints.

5 CONCLUSION

This paper first defines the sequential pattern mining problem containing interval items to address the demand for sequential pattern mining under realistic conditions. The sequence representation and sequential pattern output form containing interval items are defined, and the MII-Prefixspan algorithm is proposed. A filtering restriction is also added on interval items to more efficiently and accurately mine the hidden information in sequential patterns. Finally, we simulate a sequence database containing interval items. The effectiveness and efficiency of the MII-Prefixspan algorithm are evaluated through experiments with different sequence data. These methods can be easily extended to other fields and only need to provide data in the desired format to obtain sequential patterns with interval information.

In the future, this study can be extended in the following directions: 1) improving the algorithm's efficiency using other sequential pattern mining algorithms, and 2) combining different interval event weights to generate sequential patterns.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grants No. 61877004 and No. 62007004), the Major Program of National Social Science Foundation of China (Grant No. 18ZDA295).

6 REFERENCES

[1] Srikant, R. & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology — EDBT '96. EDBT 1996. Lecture Notes in Computer Science. Apers, P., Bouzeghoub, M., & Gardarin, G., Eds., Springer, Berlin, Heidelberg. 1057*. https://doi.org/10.1007/BFb0014140

[2] Jian, P., Jiawei, H., Mortazavi-Asl, B., Jianyong, W., Pinto, H., Qiming, C., Dayal, U., & Mei-Chun, H. (2004). Mining sequential patterns by pattern-growth: the PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, *16*(11), 1424-1440. https://doi.org/10.1109/TKDE.2004.77

[3] Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, *42*(1), 31-60. https://doi.org/10.1023/A:1007652502315

[4] Fournier-Viger, P., Gomariz, A., Šebek, M., & Hlosta, M. (2014). VGEN: Fast Vertical Mining of Sequential Generator Patterns. *Data Warehousing and Knowledge Discovery*, 476-488. https://doi.org/10.1007/978-3-319-10160-6_42

[5] Salvemini, E., Fumarola, F., Malerba, D., & Han, J. (2011). FAST Sequence Mining Based on Sparse Id-Lists. *Foundations of Intelligent Systems*, 316-325. https://doi.org/10.1007/978-3-642-21916-0_35

[6] Fournier Viger, P., Wu, C.-W., Gomariz, A., & Tseng, V. *VMSP: Efficient Vertical Mining of Maximal Sequential Patterns*, 2014. https://doi.org/10.1007/978-3-319-06483-3_8

[7] Agrawal, R., & Srikant, R. (2000). Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. Very Large Data Bases VLDB*, *1215*.

[8] Fournier-Viger, P., Wu, C.-W., & Tseng, V. S. (2012). Mining Top-K Association Rules. *Advances in Artificial Intelligence*, 61-73. https://doi.org/10.1007/978-3-642-30353-1_6

[9] Lenca, P., Vaillant, B., Meyer, P., & Lallich, S. (2007). Association Rule Interestingness Measures. *Experimental and Theoretical Studies*. *43*, 51-76. https://doi.org/10.1007/978-3-540-44918-8_3

[10] Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J. C.-W., & Tseng, V. S. (2015). Efficient Mining of High-Utility Sequential Rules. *Machine Learning and Data Mining in Pattern Recognition*, 157-171. https://doi.org/10.1007/978-3-319-21024-7_11

[11] Ho, J., Lukov, L., & Chawla, S. (2008). Sequential Pattern Mining with Constraints on Large Protein Databases. *The International Conference on Management of Data*, 89-100.

[12] Hirate, Y., & Yamana, H. (2006). Generalized Sequential Pattern Mining with Item Intervals. *Journal of Computers*, *1*. https://doi.org/10.4304/jcp.1.3.51-60

[13] Fournier Viger, P., Lin, C.-W., Rage, U., Koh, Y. S., & Thomas, R. (2017). A Survey of Sequential Pattern Mining. *Data Science and Pattern Recognition*, *1*, 54-77.

[14] Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Proceedings of the Eleventh International Conference on Data Engineering*, 3-14. https://doi.org/10.1109/ICDE.1995.380415

[15] Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. *Sequential PAttern mining using a bitmap representation*, 2002. https://doi.org/10.1145/775047.775109

[16] Fournier-Viger, P., Gomariz, A., Campos, M., & Thomas, R. (2014). Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information. *Advances in Knowledge Discovery and Data Mining*, 40-52. https://doi.org/10.1007/978-3-319-06608-0_4

[17] Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M. *FreeSpan: Frequent pattern-projected sequential pattern mining*, 2000. https://doi.org/10.1145/347090.347167

[18] Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., & Yang, D. *H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases*, 2001.

[19] Huang, K.-Y., Chang, C.-H., Tung, J.-H., & Ho, C.-T. (2006). COBRA: Closed Sequential Pattern Mining Using Bi-phase Reduction Approach. *Data Warehousing and Knowledge Discovery*, 280-291. https://doi.org/10.1007/11823728_27

[20] Gomariz, A., Campos, M., Marin, R., & Goethals, B. (2013). ClaSP: An Efficient Algorithm for Mining Frequent Closed Sequences. *Advances in Knowledge Discovery and Data Mining*, 50-61. https://doi.org/10.1007/978-3-642-37453-1_5

[21] Wang, J., Han, J., & Li, C. (2007). Frequent Closed Sequence Mining without Candidate Maintenance. *Knowledge and Data Engineering, IEEE Transactions on*, *19*, 1042-1056. https://doi.org/10.1109/TKDE.2007.1043

[22] Yan, X., Han, J., & Afshar, R. (2003). CloSpan: Mining: Closed Sequential Patterns in Large Datasets. *SIAM International Conference on Data Mining*, 166-177. https://doi.org/10.1137/1.9781611972733.15

[23] Fournier-Viger, P., Wu, C.-W., & Tseng, V. S. (2013). Mining Maximal Sequential Patterns without Candidate Maintenance. *Advanced Data Mining and Applications*, pp. 169-180. https://doi.org/10.1007/978-3-642-53914-5_15

[24] García-Hernández, R. A., Martínez-Trinidad, J. F., & Carrasco-Ochoa, J. A. (2006). A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection. *Computational Linguistics and Intelligent Text Processing*, 514-523. https://doi.org/10.1007/11671299_53

[25] En-Zheng, G., Xiao-Yu, C., Zhe, W., & Chun-Guang, Z. (2005). Mining Maximal Sequential Patterns. *2005 International Conference on Neural Networks and Brain*, pp. 525-528. https://doi.org/10.1109/ICNNB.2005.1614668

[26] Lin, N., Hao, W.-H., Chen, H.-J., Chueh, H.-E., & Chang, C.-I. (2007). Fast Mining Maximal Sequential Patterns. *Proc. 7th WSEAS Int. Conf. Simulation, Model. Optim.*

[27] Chang, J. (2011). Mining weighted sequential patterns in a sequence database with a time-interval weight. *Knowledge-Based Systems*, *24*, 1-9. https://doi.org/10.1016/j.knosys.2010.03.003

[28] Yun, U. & Leggett, J. *WSpan: Weighted Sequential pattern mining in large sequence databases*, 2006. https://doi.org/10.1109/IS.2006.348472

[29] Odyński, P. (2017). Using Frequent Pattern Mining Algorithms In Text Analysis. *Information System in Management*, *6*(3), 213-222. https://doi.org/10.22630/ISIM.2017.6.3.5

[30] Fournier Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., & Tseng, V. (2015). SPMF: A java open-source pattern mining library. *Journal of Machine Learning Research*, *15*, 3389-3393.

[31] Fournier-Viger, P., Yang, P., Li, Z., Lin, J. C. W., & Kiran, R. U. (2020). Discovering rare correlated periodic patterns in multiple sequences. *Data & Knowledge Engineering*, *126*, 101733. https://doi.org/10.1016/j.datak.2019.101733

[32] Gan, W., Lin, J. C. W., Zhang, J., Chao, H. C., Fujita, H., &

Philip, S. Y. (2020). ProUM: Projection-based utility mining on sequence data. *Information Sciences*, *513*, 222-240. https://doi.org/10.1016/j.ins.2019.10.033

[33] Van, T., Vo, B., & Le, B. (2018). Mining sequential patterns with itemset constraints. *Knowledge and Information Systems*, *57*(2), 311-330. https://doi.org/10.1007/s10115-018-1161-6

[34] He, Z., Zhang, S., & Wu, J. (2019). Significance-based discriminative sequential pattern mining. *Expert Systems with Applications*, *122*, 54-64. https://doi.org/10.1016/j.eswa.2018.12.046

[35] Wu, Y., Geng, M., Li, Y., Guo, L., Li, Z., Fournier-Viger, P., & Wu, X. (2021). HANP-Miner: High average utility nonoverlapping sequential pattern mining. *Knowledge-Based Systems*, *229*, 107361. https://doi.org/10.1016/j.knosys.2021.107361

[36] Gan, W., Lin, J. C. W., Zhang, J., Fournier-Viger, P., Chao, H. C., & Philip, S. Y. (2020). Fast utility mining on sequence data. *IEEE transactions on cybernetics*, *51*(2), 487-500. https://doi.org/10.1109/TCYB.2020.2970176

[37] Huynh, H. M., Nguyen, L. T., Vo, B., Yun, U., Oplatková, Z. K., & Hong, T. P. (2020). Efficient algorithms for mining clickstream patterns using pseudo-IDLists. *Future Generation Computer Systems*, *107*, 18-30. https://doi.org/10.1016/j.future.2020.01.034

[38] Kim, H. & Choi, D. W. (2021). Recency-based sequential pattern mining in multiple event sequences. *Data Mining and Knowledge Discovery*, *35*(1), 127-157. https://doi.org/10.1007/s10618-020-00715-7

Contact information:

Bob CHEN, Postrgraduate Student
School of Artificial Intelligence,
Beijing Normal University,
No.19, Xinjiekouwai St, Haidian District, Beijing, 100875
E-mail: bochen@mail.bnu.edu.cn

Weiming PENG, Lecturer
School of Artificial Intelligence,
Beijing Normal University,
No.19, Xinjiekouwai St, Haidian District, Beijing, 100875
E-mail: pengweiming@bnu.edu.cn

Jihua SONG, Professor
(Corresponding author)
School of Artificial Intelligence,
Beijing Normal University,
No.19, Xinjiekouwai St, Haidian District, Beijing, 100875
E-mail: songjh@bnu.edu.cn