e-RNSP: An Efficient Method for Mining Repetition Negative Sequential Patterns

Xiangjun. Dong, Yongshun. Gong*, and Longbing. Cao*, Senior Member, *IEEE*

Abstract*—Negative sequential patterns (NSP), which capture both frequent occurring and non-occurring behaviors, become increasingly important and sometimes play a role irreplaceable by analyzing occurring behaviors only. Repetition sequential patterns (RSP) capture repetitions of patterns in different sequences as well as within a sequence and are very important to understand the repetition relations between behaviors. Though some methods are available for mining NSP and repetition positive sequential patterns (RPSP), we have not found any methods for mining repetition NSP (RNSP). RNSP can help analysts to further understand the repetition relationships between items and capture more comprehensive information with repetition properties. However, mining RNSP is much more difficult than mining NSP due to the intrinsic challenges of non-occurring items. To address the above issues, we first propose a formal definition of repetition negative containment. Then we propose a method to convert repetition negative containment to repetition positive containment, which fast calculates the repetition supports only using the corresponding RPSP's information without re-scanning databases. Finally, we propose an efficient algorithm, called e-RNSP, to mine RNSP efficiently. To the best of our knowledge, e-RNSP is the first algorithm to efficiently mine RNSP. Intensive experimental results on the first four real and synthetic datasets clearly show that e-RNSP can efficiently discover the repetition negative patterns; results on the fifth dataset prove the effectiveness of RNSP which are captured by the proposed method; the results on the rest 16 datasets analyze the impacts of data characteristics on mining process.

Index Terms*— sequence analysis; repetition patterns; negative sequential patterns; repetition negative sequential patterns.

This paper is submitted in 19 Mar 2018. It is supported in part by the National Natural Science Foundation of China (71271125, 61502260)"

Xiangjun Dong is with the School of information, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, (e-mail: d-xj@163.com).

*Corresponding author. Yongshun Gong is now the Ph.D. student in University of Technology Sydney, New South Wales 2007, Australia. (e-mail: yongshun.gong@student.uts.edu.au).

*Corresponding author. Longbing Cao is with the Faculty of Engineering and Information Technology, University of Technology Sydney, New South Wales 2007, Australia (e-mail: longbing.cao@uts.edu.au).

I. Introduction

SEQUENTIAL data is widely seen in real-life applications in particular behaviors, such as high-impact behavior analysis [1], group behavior analysis [2], contrast behavior analysis [3], abnormal behavior detection [4], and so forth. As an important means for behavior analysis [7-9], sequence analysis, in particular, sequential pattern mining has been increasingly explored to discover frequent subsequences in a sequence database [27-31,35]. Since the first proposal of sequential pattern mining, many algorithms, such as GSP [10], FreeSpan [11], PrefixSpan [12], SPADE [13], and SPAM [14], have been successfully proposed to enhance the algorithm efficiency. The patterns mined by these algorithms, focusing only on occurring items, are called positive sequential patterns (PSP). But limited research has been conducted on analyzing non-occurring behavior sequences [46], e.g., mining negative sequential patterns (NSP) [5, 6, 40]. NSP, which contains both occurring and non-occurring [46] items, such as $<ab\neg c>$, sometimes play an irreplaceable role in many intelligent systems and applications, such as intelligent transport systems (ITS), health and medical management systems, bioinformatics, biomedical systems, risk management, counter-terrorism, and security [15,40]. For instance, assume s_1=$<abcX>$ is a PSP; s_2=$<ab\neg cY>$ is a NSP, where a, b and c stand for medical service codes that a patient receives in health care, and X and Y stand for disease states. s_1 shows that a patient who usually receives medical services a, b and then c is likely to have disease status X, whereas s_2 indicates that patients receiving treatments of a and b but NOT c have a high probability of having status Y [15].

Although many algorithms can be used to discover PSP, NSP cannot be described or discovered by these algorithms. This is because mining NSP is much more difficult than mining PSP, particularly due to the following three intrinsic complexities: *hidden nature of non-occurring items*, *high computational complexity* and *large negative sequential candidates (NSC) search space* [15,40]. In fact, research on NSP mining is at an early stage, and has seen only limited progress in recent years [5, 40]. All existing methods are very inefficient and are too specific for mining NSP, except e-NSP [40]. e-NSP proposes a method to fast calculate the support of NSC only using the corresponding PSP's information, without database rescanning. By this way, e-NSP obtains high time efficiency.

e-NSP, however, does not consider the repetition sequential patterns (RSP) mining problem. RSP is important as they represent repetition behaviors, and can capture repetitions of a pattern in different sequences as well as within a sequence, in which the same item(s) can occur more than once in a sequence [20-26,48]. It is helpful for deeply understanding the relations between items in many applications, such as network attack detection, DNA periodic analysis [21,51], outlier pattern detection [34], and so on [18,36-39,52]. For example, suppose a dataset contains two sequences below: {10: <*abababab c*>; 20: <*ac*>} and a given minimum support threshold *min_sup* =2. RSP mining algorithms can find pattern <*ab*> occurring at least 4 times and thus mark it as a frequent pattern. If <*ababababc*> represents the behavior that a hacker attacks a server in a short time period, mining RSPs like <*ab*> can help analysts to capture more useful information about a pattern's appearance within or between sequences. Some RSP mining algorithms have also been proposed to mine such patterns [19-30]. Unfortunately, all existing RSP mining algorithms we have found only consider repetition PSP (RPSP).

Repetition NSP (RNSP) combines the respective information of NSP and RPSP, representing non-occurring repetition behaviors. It can help analysts to further understand the relationships between items and capture more comprehensive information with repetition properties. For example, in auto insurance fraud detection, s_3=<*xy¬zW*> denotes a customer's collision-payment sequence, where *x* denotes the event of a vehicle collision caused by a customer's own reason, *y* denotes the event that the insurance company assesses the damage, *z* denotes the event of repairing car in the garages that the insurance company suggests, and *W* denotes the event of the payment to customer by the insurance company. s_3 denotes that a customer gets the payment, but s/he doesn't repair her/his car in the garages that insurance company suggests. This case is normal because the insurance company doesn't force their customers to repair car in their suggested garages. However, sequence s_4=< *xy¬zW xy¬zW xy¬zW*> should be highly abnormal, since it indicates that the same events repetitively occur to the same customer which is likely a fraud. In fact, such suspicions happen sometimes in real life. Hence, mining such RNSP is very important in real applications.

However, RNSP mining is more difficult than NSP mining and RSP mining, particularly because of the following two intrinsic complexities.

(1) *Repetition negative containment problem*. In NSP mining, there is not a unified definition about negative containment [15-18] so far because the hidden nature of non-occurring items [46] makes it complicated in defining the negative containment problem. For example, for a sequence s_5=<*ababababc*>, in PSP mining, the support of <*ab*> in s_5 is 1; in RPSP mining, the repetition support of <*ab*> in s_5 is 4 (this value may be different in different RSP mining methods). But in NSP mining, whether s_5 contains <*ab¬d*> is inconsistent in different papers [15-18]. In RNSP mining, does s_5 contain <*ab¬d*>? If yes, how many repetition times that s_5 contains <*ab¬d*>? Therefore, how to define repetition negative containment is a challenging problem unsolved.

(2) *High computational complexity*. Most of existing methods are very inefficient because they calculate the support of NSC by additionally scanning the database after identifying PSP. If we use the same way to obtain the repetition supports, it will bring enormous consumption both on running time and space. Therefore, how to fast calculate the repetition support of RNSP is a significant yet difficult problem.

In order to address the above critical challenges and make RNSP running feasible in real-life applications, this paper proposes an efficient algorithm, called e-RNSP, to mine RNSP efficiently. To the best of our knowledge, e-RNSP is the first algorithm to mine RNSP. The main contributions are as follows.

First, we propose a definition to formally define repetition negative containment.

Second, we propose a method to convert the problem of repetition negative containment to the problem of repetition positive containment, which lets us fast calculate the support of NSC by only using the corresponding RPSP's information and avoid database rescanning.

Further, a hash table is proposed to store the corresponding information of RPSP and propose an efficient algorithm, called e-RNSP, to mine RNSP efficiently.

Lastly, experiments are conducted on real and synthetic datasets to compare e-RNSP with three available NSP mining methods, e-NSP [40], NegGSP [17] and PNSP [16] in terms of the number of patterns and their running time. Particularly, based on a basic dataset, we generate 15 additional datasets in terms of different data factors, to access the runtime and pattern number of e-RNSP and e-NSP respectively. Intensive experiments clearly show that e-RNSP can efficiently discover repetition negative patterns.

The rest of this paper is organized as follows. The related work is discussed in Section 2. In Section 3, we introduce some basic concepts of PSP mining. In Section 4, we define the definition of negative containment. The e-RNSP algorithm is explained in Section 5, and Section 6 displays the experimental outcomes. Section 7 includes the conclusions and future work.

II. Related Work

In this section, we first introduce some available methods of mining NSP. Further, we introduce the state-of-the-art research of mining RSP.

In [17], a GSP-like way was introduced to mine for NSP, called NegGSP. Chen et al. designed a negative NSP mining approach PNSP [16]. Only the form of $(\neg X, Y)$, $(X, \neg Y)$ and $(\neg X, \neg Y)$ are suitable for the method in [31], which is similar to mine negative association rules. Lin et al. designed an algorithm NSPM [18] for mining negative sequential patterns, in which only the last element can be negative. They then extended their algorithm to NFSPM for mining negative fuzzy sequential patterns [32] and PNSPM for mining strong positive and negative sequential patterns [33]. In our previous work, we proposed an efficient NSP mining method e-NSP in [15,40]. E-NSP calculates NSC's supports only by using the corresponding PSP information without re-scanning database and can handle large-scale NSP. A NSP mining method based on multiple minimum supports, named e-msNSP, was proposed in [41]. [47] utilized the bitmap structure with a self-adaptive data storage strategy to improve the efficiency of e-NSP. A method mining NSP from both frequent and infrequent positive sequence, named, was proposed in [42]. Xu et al. considered utility when mining NSP [5].

Very limited work has been reported on how to identify RPSP from sequence datasets. The authors in [34] proposed a stable and efficient suffix tree-based approach for detecting the periodicity of outlier patterns in a time series. Meanwhile, the methods in [20,23,25] follow the unified definition of repetition sequences. The work in [20] faces the overlap issue when calculating the repetition times. For example, given a data sequence *ds*=< *AXYABXYXA*>, <XYX> appears twice in *ds* at <2,3,4> and <6,7,8> respectively, where 2,3,4 and 6,7,8 are the element ID in *ds*. Authors of [23] compressed repetition gapped sequential patterns and proposed an algorithm CRGSgrow. A navigation pattern clustering method was proposed in [25] based on closed repetition gapped subsequences. An RB-EZH2 Complex Mediates Silencing of Repetition DNA Sequences is proposed in [43].

There are some other algorithms which take different definitions. RptGSP was proposed in [19] to mine RPSP, it uses the way similar to GSP to find sequential patterns, but calculates repetition supports in data sequences. Repetition expansion was introduced in [21] for DNA replication. The gap requirement was discussed in [22] when mining repetition patterns from DNA sequences. The definition of gap weight for sub-sequences was discussed in [24]. Different events have different gaps, and their paper put forward an approach EWM to mine repetition patterns with gap weight. However, their method does not discriminate overlapping subsequences and non-overlapping ones. Mannila et al. performed an approach of mining episode to catch frequent episodes within a sequence [25]. An episode is defined as a series of events occurring relatively close to one another. An episode is supported by a window if it is a sub-sequence of the series of events appearing in the window. In [29], a sequence is divided into non-overlapping windows. A pattern is frequent if it appears in at least a certain number of windows. With this definition, it is shown that the Apriori property applies. It simplifies the design of the mining algorithm by segmenting a sequence into windows and counting the number of windows in which a pattern frequently occurs. However, patterns that span multiple windows cannot be discovered, and in some cases, a suitable window width is difficult to determine. Yang et al. studied asynchronous periodic patterns in time series data [30]. In their model, shifts in the occurrence of patterns are permitted to filter out random noises. They also considered a range of periods instead of those used in [29], although there is still a limit of the maximum length of a period.

A method was proposed in [27] for identifying iterative patterns, which captures occurrences in the semantics of Message Sequence Chart/Live Sequence Chart, a standard in software modeling. Iterative pattern is known as a series of events which repeat within and across sequences. Both work in [20] and [27] mine repetition closed subsequences with different underlying target formalism and semantics. Different search space pruning strategies and mining algorithms are used to efficiently mine recurrent rules. The work in [28] uses the definition of iterative patterns similar to [27]. It proposed an approach to find generators of iterative patterns and investigate catching of iterative generators from program execution traces. Generators are the minimal members of an equivalence class, while closed patterns are the maximal members. An equivalence class in turn is a set of frequent patterns with the same support and corresponding pattern instances.

Other papers discussed research on sequences, but they didn't consider negative sequences. The authors in [44] proposed a characteristic-based framework for multiple sequence aligners. The work in [45] includes a new initialization technique, which is a heuristic space-filling approach based on both functions to be optimized and a search space. In [49], a novel approach rep-PrefixSpan for mining RSP with multiple minimum item repetition support was proposed and authors of [50] utilized the cyclic model to predict likely consumer behavior within a certain time frame. Fan et al, proposed an efficient Apriori algorithm for frequent tri-patterns discovery [53]. [54] designed two novel algorithms for mining inter-sequence patterns with item constraint and [55] proposed an efficient way to discover maximal frequent patterns in transactional databases and dynamic data streams.

In summary, existing methods were not designed to identify RNSP, and there are inconsistencies in defining and extracting repetition patterns. RNSP is thus proposed to address this gap.

TABLE I. NOTATION DESCRIPTION

Symbol	Description
I	A set of items, $I=\{i_1, i_2,\dots, i_n\}$, consisting of n items $i_k (1\le k\le n)$
s	A sequence, $s=<s_1, \dots, s_l>$, consisting of l elements s_j $(1\le j\le l)$
min_sup	Minimum support threshold
ns	A negative sequence
length(*s*)	Length of sequence *s*, referring to the total number of items in all elements in *s*
size(*s*)	Size of a sequence *s*, referring to the total number of elements in *s*
sup(*s*)	The support of *s*
p(*ns*)	*ns*'s positive partner
MPS(*s*)	Maximum positive sub-sequence of *ns*
1-negMS	1-neg-length maximum subsequence of *ns*
1-negMSSns	1-neg-length maximum subsequence set of *ns*
LCSP	The left containment subsequence position

III. PRELIMINARIES

Assume a set of *items* $I=\{i_1, i_2,\dots, i_n\}$, an *itemset* is a subset of I. A *sequence* is an ordered list of itemsets. A sequence s is described by $< s_1, s_2,\dots, s_l >$, where $s_j \subseteq I$ $(1\le j \le l)$. s_j is also named a sequence's *element*, labelled as $(x_1, x_2,\dots, x_m)$, where x_k is an item, $x_k \in I$ $(1\le k \le m)$, j is the *id* of the element. For simplicity, if an element only contains one item, the bracket is omitted, i.e., (x_1) is equal to x_1. An item in a sequence can appear at most once in an element, but can occur multiple times in different elements.

Length(s) is the *length* of sequence s, which is the total number of items in all elements in s. *Size*(s) is the *size* of s, coded as *size*(s), which is the total number of elements in s. For example, sequence $<a(ad)de>$ is comprised of 4 elements a, (ad), d and e; meanwhile, it is also comprised of 3 items a, d and e. It is a *4-size* and *5-length* sequence.

Sequence $s_\alpha=<\alpha_1,\alpha_2,...,\alpha_n>$ is named a *sub-sequence* of sequence $s_\beta=<\beta_1, \beta_2,..., \beta_m>$ and s_β is a *super-sequence* of s_α, denoted as $s_\alpha \subseteq s_\beta$, if there exists $1\le j_1 < j_2<\ldots< j_n \le m$ such that $\alpha_1 \subseteq \beta_{j1}$, $\alpha_2 \subseteq \beta_{j2},\ldots, \alpha_n \subseteq \beta_{jn}$. We also call s_β contains s_α. For example, $<c>$, $<ac>$ and $<(ab)\ d>$ are sub-sequences of $<(ab)\ c\ d>$.

A set of tuples $<sid,ds>$ is used to represent a sequence dataset D (see Table III about an example dataset for details), where ds is the data sequence and sid is the number of sequence. $|D|$ is the number of tuples in D. The set of tuples containing sequence s is described as $\{<s>\}$. *Sup*(s) refers to the support of s, it is the frequency of $\{<s>\}$, i.e., $sup(s)=|\{<s>\}|=|\{<sid,ds>, <sid,ds>\in D \wedge (s\subseteq ds)\}|$. *min_sup* is a minimum support threshold, denoted as *min_sup*. If $sup(s)\ge$ *min_sup*, then we call the sequence s is frequent. By contrast, s is infrequent if $sup(s)<$ *min_sup*.

PSP mining aims to discover all positive sequences that satisfy the minimum support. For simplicity, we often omit "positive" when discussing positive items, positive elements and positive sequences in mining PSP.

The main symbols used in this paper are listed in Table I.

IV. The Definitions of Negative Containment

In this section, we first introduce the constraints to negative sequence, then discuss the definitions of negative containment in e-NSP, finally propose the definitions of repetition negative containment.

A. Constraints to Negative Sequences

In real-life applications, the number of NSC and the identified negative sequences are usually in an enormous scale, and most of which are meaningless [40]. The number of NSC may be huge or even infinite if no constraints are added. This makes NSP mining very challenging. In order to solve this problem, some available constraints are introduced in the existing methods. This paper involves three constraints the same as e-NSP. Here we only introduce these constraints because of page limitation, please refer to [40] for the feasibility and rationality if interested. We first introduce the definition *positive partner*, which is used in the constraints.

Definition 1. Positive Partner. Given a negative element $\neg b$, its *positive partner* is b, described as $p(\neg b)$, i.e., $p(\neg b)=b$.

A positive element b's *positive partner* is b itself, i.e., $p(b)=b$. Suppose $ns=<s_1...s_k>$ is a negative sequence, its positive partner can be obtained by converting all negative elements in ns to their positive partners, denoted as $p(ns)$, i.e., $p(ns)=\{<s_1'\ldots s_k'>|\ s_i'=p(s_i),\ s_i \in ns\}$. For example, $p(<\neg(cd)a\neg c>)=<(cd)ac>$.

Constraint 1. Frequency Constraint. We only focus on those negative sequences ns whose $p(ns)$ is frequent, i.e., $sup(p(ns)) >$ *min_sup*.

Constraint 2. Formation Constraint. Continuous negative elements are not allowed in a NSC, because we cannot tell the right order of two continuous negative elements if there is no positive element between them.

Example 1. $<a\ \neg(ab)\ c\ a\ \neg c>$ satisfies *Constraint* 2, but $<a\ \neg(ab)\ c\ \neg a\ \neg c>$ does not.

Constraint 3. Element Negative Constraint. An element is the minimum negative unit in a NSC. If an element includes more than one item, it is not permitted that certain items in the element are negative while others are not.

Example 2. $<a\ \neg(ab)\ c\ a\ \neg c>$ satisfies this constraint, but $<a\ (\neg ab)\ c\ a\ \neg c>$ doesn't because only $\neg a$ is negative in element ($\neg a\ b$), while b is not.

Definition 2. Negative Sequential Pattern (NSP). The support of a negative sequential pattern (NSP) is not less than *min_sup*.

B. Negative Containment

The definition of negative containment is very important to the efficiency of a NSP mining algorithm because it affects the efficiency of calculating the support of NSC. In e-NSP, a definition of negative containment that is consistent with the set theory was proposed. In order to fast calculate the support of NSC, e-NSP converts the negative containment problems to positive containment problems such that the support of NSC is fast calculated by only using the information of PSP. In order to do so, e-NSP defines a series of strict definitions which are not easily understood. This paper also uses the same definitions, but we simplify them in an easily understandable way: we only use the converted definitions and omit those preparatory definitions. Interested readers can refer to [40] to understand these definitions from negative containment angle. We use an example to explain them first.

Given $ds=<a(bc)d(cde)>$ and $ns=<a\neg bb\neg a(cde)>$, ds contains ns if and only if ds contains $<ab(cde)>$ and ds doesn't contain $<abb(cde)>$ (i.e., $p(<a\neg bb(cde)>)$) and $<aba(cde)>$ (i.e., $p(<ab\neg a(cde)>)$), where $<ab(cde)>$ is the sub-sequence that contains all positive elements with the same order as ns, called *Maximum Positive Sub-sequence* and denoted by *MPS*(ns); $<a\neg bb(cde)>$ (or $<ab\neg a(cde)>$) is the sub-sequence that contains all positive elements and only one negative element with the same order as ns, called 1−*neg*−*size* maximum sub-sequences and denoted by 1-*negMS*. The set consisting of all 1-*negMS* in ns is called 1-neg-size maximum sub-sequence set, denoted as 1-$negMSS_{ns}$. For example, 1-$negMSS_{<a\neg bb\neg a(cde)>}=\{<a\neg bb\ (cde)>, <ab\neg a(cde)>\}$.

Now we formally define *negative containment.*

Definition 3. Negative containment. Given a data sequence ds and a negative sequence ns, ds contains ns if and only if the two conditions hold: (1) $MPS(ns) \subseteq ds$; and (2) $\forall 1-negMS\in 1-negMSS_{ns}, p(1-negMS) \not\subseteq ds$.

Example 5. Assume $ds=<(ab)c(de)f>$ and (1) $ns=<a\ c\neg d>$, 1-$negMSS_{ns}=\{<ac\neg d>\}$, ds does not contain ns because $p(<ac\neg d>)=<acd> \subseteq ds$; (2) $ns'=<a\neg bc\neg g>$, *1-negMSS'*$_{ns}$={<*a* ¬*bc*>, <*ac* ¬*g*>}, *ds* contains *ns* because *MPS*(*ns'*)=<*ac*> $\subseteq$ *ds* $\wedge$ *p*(<*a* ¬*b c*>) $\not\subset$ *ds* $\wedge$ *p*(<*a c* ¬*g*>) $\not\subset$ *ds*.

From Definition 3 we can see that the negative containment now is converted to positive containment: a data sequence contains a positive sequence but does not contain some other related positive sequences. In this way, we can calculate the support of negative sequences by only using the information of corresponding positive sequences.

C. Repetition Negative Containment

As a data sequence *ds* may contain a negative sequence *ns* more than once without overlap, we need to know the positions that *ds* contains *ns* from the left side of *ds*. This is very important to give a cutting point in *ds* and define the repetition negative containment problem.

Definition 4. Left Containment Subsequence Position. For a data sequence *ds* = <$e_1e_2 \ldots e_n$>, and *ns* as a negative sequence, if *ns*$\subseteq$*ds* and $\exists i$ ($1< i \leq n$), s.t. *MPS*(*ns*)$\subseteq$<$e_1 \ldots e_i$> $\wedge$ *MPS*(*ns*) $\not\subset$<$e_1 \ldots e_{i-1}$>, then the *id* of the element e_i, *i*, is the *left containment subsequence position*, denoted by *LCSP*(*ns*, *ds*)=*i;* if *ns*$\not\subset$*ds, then* *LCSP*(*ns*, *ds*)=0. In particular, if *ns* is a *1-size* negative sequence, such as <¬*e*> and <¬(*ab*)>, *ns* is not repetition, hence its support can be calculated per the traditional way of valuing support.

Example 6. Given ns_1=<*a*¬*db*>, ns_2=<*a*¬*dc*>, ds_1=<*ac*(*bc*)*a*(*ab*)cb> and ds_2=<*aca*(*ab*)cb>. According to definition 4, *MPS*(ns_1)=<*ab*> and the leftmost subsequence in ds_1 that contains <*ab*> is <*ac*(*bc*)>. The *id* of element (*bc*) is 3, thus, *LCSP*(ns_1, ds_1)=3. Similarly, *LCSP*(ns_2, ds_2)=2.

Definition 4 tells us the following two facts.

(1) The negative containment problem (whether *ds* contains *ns*) is converted to the positive containment problem (whether *ds* contains *MPS*(*ns*)). So the repetition negative containment problem is consequently converted to the repetition positive containment problem.

(2) *LCSP*(*ns*, *ds*) gives the position of the leftmost subsequence that *ds* contains *ns*, identifying this position as a cutting point to calculate the repetition negative containment times subsequently.

Algorithm 1 presents how to calculate the repetition times when a *ns* crossing over a *ds*.

Algorithm 1: Calculate ***RptTimes*(*ns*, *ds*).**
Input: *ns*: a negative sequence;
ds =<$e_1e_2 \ldots e_n$>: data sequence;
Output: repetition containment times;
(1) *t* = 0;
(2) If *ns* $\subseteq$ *ds* {
(3) **Until** (*MPS*(*ns*) $\not\subset$*ds*) **Do {**
(4) *t*++;
(5) *m* = *LCSP*(*ns*, *ds*)
(6) *ds* = <$e_{m+1} \ldots e_n$>;
(7) **}**
(8) **Return** *t*;

$$RptTimes\,(ns,ds) = RptTimes(MPS(ns),ds),\ \text{if } ns \subseteq ds \quad (1)$$

According to Eq. (1), the repetition negative containment problem is converted to the repetition positive containment problem, i.e. the repetition times of any NSC in a data sequence can be converted to a calculation of its *Maximum Positive Sub-sequence*. For example, given *ns*=<*a*¬*dc*>; ds_1=<*aca*(*ab*)cb>, ds_2=<*abababd*>. As the progress shown in Fig.1, *LCSP*(*ns*, ds_1)=2, *LCSP*(*ns*, ds_2) does not exist because *ns*$\not\subset$$ds_2$, *RptTimes* (*ns*,$ds_1$)=*RptTimes* (*MPS*(*ns*),ds_1)=2. Fig. 1 shows this process. Furthermore, the repetition times of any PSP can be easily got by a non-overlapping RPSP mining method [19]. The demonstration of Eq. (1) is shown in Section V (*F*).

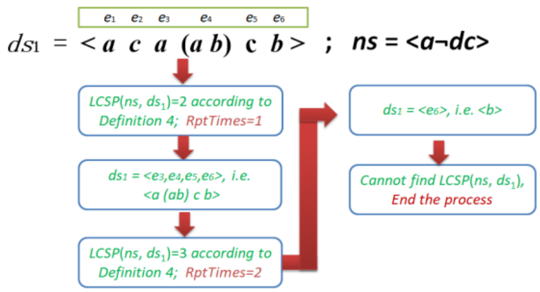


Fig.1 Repetition times

V. E-RNSP ALGORITHM

A. E-RNSP Candidate Generation

In order to generate all non-redundant NSC from PSP, we use the efficient method e-NSP to generate NSC. The key process of generating a NSC is to convert non-contiguous elements in a positive pattern to their negative partners.

The further explanation is that, to generate NSC, the algorithm changes any *m* non-contiguous elements in a RPSP to their negative partners. For a *j* size RPSP, *m*=1,2,…,$\lceil j/2 \rceil$.

For example, the NSC of <(*xy*) *a b c* > include:

m=1, <¬(*xy*) *a b c* >, <(*xy*) ¬*a b c* >, <(*xy*) *a* ¬*b c*>, <(*xy*) *a b* ¬*c*>;

m=2, <¬(*xy*) *a* ¬*b c*>, <¬(*xy*) *a b* ¬*c*>, <(*xy*) ¬*a b* ¬*c*>.

Obviously, we can use the above strategy to generate NSC that meet the condition of the three constraints described in Section 3.2.

TABLE II. E-RNSP DATA STRUCTURE

RPSP	*sup*	*rsup*	sidHash					
<*a*>	5	12	*sid*	10	20	30	40	50
			rt	2	1	4	3	2
<*a b*>	3	5	*sid*	10	20	30		
			rt	2	1	2		
…		…	…					

B. Calculate the Repetition Support of NSC

Let *ns* be a *n-neg-size* and *m-size* negative sequence, for $\forall$1-*negMS*$_i$ $\in$ 1-*negMSS*$_{ns}$ (*1*$\leq$*i*$\leq$*n*), the repetition support (*rsup*) of *ns* can be calculated by the following three equations.

$$\{ns\} = \{MPS(ns)\} - \{\cup_{i=1}^{n}\{p(1 - negMS_i)\}\} \quad (2)$$

Eq. (*2*) is used to obtain a *sid* set of data sequences which contains *ns*, where {*MPS*(*ns*)} is a *sid* set of sequences which contains *MPS*(*ns*), and $\{\cup_{i=1}^{n}\{p(1\text{-}negMS_i)\}\}$ is a *sid*'s union set from {*p*(*1-negMS*$_i$)} based on the corresponding RPSP.

The ordinary negative support of *ns* following the traditional support definition can be calculated by $|\{ns\}|$, where $|\{ns\}|$ is the number of *sid* in $\{ns\}$. To calculate the repetition support of *ns*, we have to know the repetition times that *ns* occurs in each $\{ns\}$. Accordingly, the repetition support of *ns* is shown below.

$$rsup(ns) = \sum_{i=1}^{|\{ns\}|} RptTimes(ns, ds_i)(\forall ds_i \in \{ns\}) \quad (3)$$

where ds_i is a data sequence and its position in $\{ns\}$ is *i*. Then we can get $RptTimes(ns,ds_i)$ in terms of Eq. (*1*) without re-scanning the sequence database.

In particular, if the size of *ns* is 1, i.e., it has only one negative element, such as $<\neg e>$ and $<\neg(ab)>$, the repetition support of *ns* is the same as its ordinary support, as shown in Eq. (*4*):

$$rsup(ns) = sup(ns) = |D| - sup(p(ns)) \quad (4)$$

For example, given a negative sequence $ns=< a \neg b\ c \neg d >$, then $MPS(ns)=<a\ c>$, $p(<a \neg b\ c>)=<a\ b\ c>$, $p(<a\ c \neg d >)=<a\ c\ d>$. We assume that the *sid* set of $<a\ c>$ is {10, 20, 30, 40, 50}, i.e., data sequences "10", "20", "30", "40", "50" contain $< a\ c >$. The repetition times of $<a\ c>$ in the corresponding data sequences are 2, 2, 3, 1 and 4, respectively. The *sid* set of $<a\ b\ c>$ is {10, 20}; {20, 40} is the *sid* set of $<a\ c\ d>$. Subsequently, $\{<a \neg b\ c \neg d>\} = \{<a\ c>\}-\{\{<a\ b\ c>\} \cup \{<a\ c\ d>\}\} =\{10,20,30,40,50\}-\{\{10,20\}\cup\{20,40\}\} =\{30,50\}$;

$rsup(<a \neg b\ c \neg d>)=RptTimes(<a \neg b\ c \neg d>,30)+ RptTimes(<a \neg b\ c \neg d >,50)=3+4=7$.

C. Data Structure and Hash Table in e-RNSP

In order to efficiently calculate the repetition support of negative sequences, we design a data structure to store the e-RNSP related data. The data structure is shown in Table II. Column one stores RPSP mined by RptGSP [19]. Column two holds the regular support of RPSP. Column three saves their repetition support. Column four encloses a hash table *sidHash* <*sid, rt*>. The *sid*s of data sequences contain the corresponding RPSP and the repetition times (*rt*) of the RPSP occurring in the corresponding data sequence.

For example, Table II shows that, for a RPSP $<a\ b>$, its corresponding hash table consists of {<10, 2>, <20, 1>, <30, 2>}, meaning that $<a\ b>$ is contained in the sequences 10, 20 and 30. The repetition times of $<a\ b>$ are 2, 1 and 2, respectively.

In order to identify PSP and IPS efficiently, we use the hash table to store the e-RNSP data, as shown in Algorithm 2.

Algorithm 2: Hash table creation process in e-RNSP
Input: All RPSP and their related information;
Output: RPSP's hash table;
(1) CreateHash(*RPSP)*{
(2) Create *RPSPHash* ;
(3) **For** (each pattern *p* in *RPSP*){
(4) Create *sidHash*;
(5) **For** (each data sequence *ds*){
(6) **If** (*ds* contains *p*){
(7) *rt* = *RptTimes*(*p, ds*);
(8) *sidHash.put*(*p*.sid, *rt*);
(9) }
(10) }
(11) *PSPHash*.put(*p, sidHash*);
(12) **Return** *RPSPHash*;
(13) } }

D. The e-RNSP Algorithm

The e-RNSP algorithm mines for RNSP by only using the identified RPSP.

Algorithm 3: e-RNSP
Input: D: Sequence Dataset; *min_sup*;
Output: RNSP;
(1) RPSP = RptGSP(D);
(2) CreateHash(RPSP)
(3) For (each *rpsp* in RPSP) {
(4) INT *rsup* = 0;
(5) Generate NSC by Section 5.1;
(6) For (per *nsc* in NSC) {
(7) If (the size of *nsc* is one){
(8) Calculate *rsup* by Eq. (4);
(9) }Else{
(10) Calculate *rsup* by Eq. (2) and (3);
(11) }
(12) If (*rsup* >= *min_sup*)
(13) RNSP.add(*nsc*);
(14) } // END OF (6)
(15) } // END OF (3)
(16) Return RNSP;

Below is the explanation of the **Algorithm 3**. In Section V(*F*), we provide a brief theoretical analysis of the working mechanism of the e-RNSP algorithm.

(1) Line (1) finds all *RPSP* from the sequence database using the RptGSP algorithm. Meanwhile, all *RPSP* are saved in the e-RNSP data structure, as detailed in Section 5.4 (Lines (2,3));

(2) For each *RPSP*, generate *NSC*(*s*) by the Candidate Generation method in Section 5.2 (Line (6));

(3) The repetition support for each *nsc* in *NSC*(*s*) can be easily calculated by Eq. (1-4) (Lines (7~24)) and then we determine whether they are *RNSP* (Lines (25~27)).

We calculate the repetition support of *1-size nsc* by using Eq. (4) (Lines (8~10)). Further, in lines (12) to (17), we calculate $\{\cup_{i=1}^{n}\{p(1\text{-}negMS_i)\}\}$, and obtain the *sid* set of *ns* by $\{MPS(ns)\}-\{\cup_{i=1}^{n}\{p(1\text{-}negMS_i)\}\}$ (Lines (18~21)). Lines (22~24) calculate the repetition support of *nsc* by Eq. (3). If $rsup(nsc) >= min\_sup$, then *nsc* is inserted into *RNSP* (lines (25~27)).

(4) Obtain the results (Line (29)).

E. An Example

The above sections introduce key concepts and components as well as the e-RNSP algorithm for RNSP mining. This section uses an example to illustrate how to mine for RNSP. The datasets are shown in Table III. In the example, we set *min_sup*=2.

TABLE III. EXAMPLE DATASET

sid	ds
10	$<a\ b\ (bc)>$
20	$<a\ b\ e\ a\ b\ e>$

30	<(*bc*) *f*>
40	<*a* (*bc*) *c*>
50	<*d e*>

The process is as follows.

(1) Mining repetition positive sequential patterns (RPSP) using RptGSP, and storing the results in terms of the e-RNSP data structures (see Section 5.4), which are detailed in Table IV.

TABLE IV. EXEMPLARY RESULTS – REPETITION POSITIVE PATTERNS

RPSP	Sup	Rsup	SidHash
<*a*>	3	4	**10 20 40** 1 2 1
<*b*>	4	6	**10 20 30 40** 2 2 1 1
<*c*>	3	4	**10 30 40** 1 1 2
<*e*>	2	3	**20 50** 2 1
<(*bc*)>	3	3	**10 30 40** 1 1 1
<*a b*>	3	4	**10 20 40** 1 2 1
<*a c*>	2	2	**10 40** 1 1
<*a e*>	1	2	**20** 2
<*b b*>	2	2	**10 20** 1 1
<*b c*>	2	2	**10 40** 1 1
<*b e*>	1	2	**20** 2
<*a b e*>	1	2	**20** 2
<*a b c*>	2	2	**10 40** 1 1
<*a* (*bc*)>	2	2	**10 40** 1 1

(2) Using the e-RNSP generation approach to get all negative sequential candidates (NSC).

(3) Computing these NSC repetition support values based on Eq. (1-4). Table V shows the results, and the final RNSP are marked in bold.

Among the RNSP, <*ab*¬*c*>, <*a*¬*c*> and <*a* ¬(*bc*)> are three special ones because they are mined as RNSP, but they are not mined as patterns in e-NSP. Obviously, not all of RNSP are actionable for supporting decision-making [40], especially those patterns with only one positive element, such as <*b* ¬*e*> and <¬*a b* ¬*e*>, their repetition supports are high but misleading. How to catch those actionable RNSP is our future task.

F. The theoretical analysis of the working

Here, we discuss the theoretical soundness of e-RNSP from its working mechanism perspective.

The mining process of e-RNSP could be mainly divided into four stages. The first stage mines all RPSP and uses them to generate negative sequential candidates (NSC). For a certain NSC (*nsc*), the second stage is to identify that whether this *nsc* is contained by a data sequence based on the negative containment as discussed in Section 4.2. The third is to catch repetition times when *nsc* crossing the above data sequence. The last stage is to achieve its *rsup* utilizing *Eq.* (3) or *Eq.* (4).

For the first stage, this paper utilizes RptGSP[11] to capture all RPSP and generates NSC based on the strategy in [32]. This generation method converts non-contiguous elements in a positive pattern to their negative partners, which means for each NSC, $MPS(NSC)\in\{\text{RPSP}\}$, where {RPSP} means the set of RPSP. Accordingly, this strategy ensures that the supports of all generated NSC could be then calculated based only on the corresponding RPSP.

In second stage, this paper uses the same definitions of negative containment in e-NSP, which converts the negative containment problem to positive containment problem in terms of set theory. We introduce briefly the conversion process as follows, please find detailed proof in [40].

$\{< a >\}$, $\{< b >\}$ mean the set of tuples that respectively contain sequences $< a >$, $< b >$ in a sequence database. The intersection of sequences $< a >$ and $< b >$ will generate four disjointed sets: $\{< (ab) >^{only}\}$, $\{< ab >^{only}\}$, $\{< ba >^{only}\}$ and $\{< ab >\}$ $\{< ba >\}$, representing the sets of tuples that contain sequences $< (ab) >^{only}$, $< ab >^{only}$, $< ba >^{only}$, and both $< ab >$ and $< ba >$ respectively, as shown in Fig. 2.

For simplicity, let us take $\{<a\neg b>\}$ as an example, we have:

$$\{a\neg b\} = (\{< a >\} - \{< b >\}) \cup \{<\ (ab) >^{\text{only}}\} \cup \{< ba >^{\text{only}}\}$$
$$= \{< a >\} - \{< ab >^{\text{only}}\} \cup (\{< ab >\} \cap \{< ba >\})$$
$$= \{< a >\} - \{< ab >\}$$

This result illustrates the strategy of conversion process, i.e. data sequences that contain $< a\neg b >$ are the same sequences that contain $< a >$ but do not contain $< ab >$.

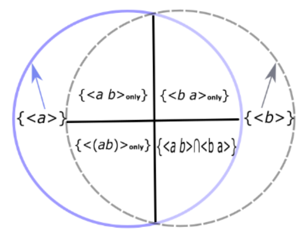


Fig. 2. The intersection of {< a >} and {< b >}

To address the repetition containment problem, we extend the above conversion strategy to a cyclic conversion strategy in third stage. We will demonstrate that the repetition negative containment can also be converted into the repetition positive containment.

Corollary 1. Repetition Negative Conversion Strategy.

For a data sequence *ds*, and a negative sequence *ns*, the repetition negative containment can be converted to the following problem: if $ns \subseteq ds$, the repetition times that *ns* crosses through *ds* equal to times that MPS(*ns*) occurs in *ds*.

Proof of Corollary 1.

Given a data sequence $ds =< d_1 d_2 \ldots d_l >$, and *ns* is a negative sequence. According to the negative containment in Section 4.2, if $ns \subseteq ds$, satisfying (1) $MPS(ns) \subseteq ds$; and (2) $\forall 1-negMS \in 1-negMSS_{ns}, p(1-negMS) \not\subseteq ds$. Assume $LCSP(ns, ds)=i$, for the sub-sequence $<d_{i+1}\ d_{i+2}\ldots d_l>$ of *ds*, denote as ds^i, $\forall 1-negMS \in 1-negMSS_{ns}$, $p(1-negMS) \not\subseteq ds^i$. Thus, we only need to determine whether $MPS(ns) \subseteq ds^i$.

Intrinsically, the last stage is a combination process which incorporates the above three processes to calculate the *rsup* of a NSC crossing all the data sequences based on a set theory in [40].

VI. Experiments and Evaluation

The experiments on 15 synthetic and real databases have been conducted to compare with three available NSP mining methods, e-NSP [40], NegGSP [17] and PNSP [16] from two aspects: the number of patterns and their running time for identifying negative patterns. To compare their performance, we make PNSP and NegGSP to follow the same constraints and definitions in e-NSP. All algorithms are coded in Java and executed in a Windows 7 Professional PC with Intel Core i5 CPU of 3.2GHz, 4GB memory. In the experiments, all supports (and minimum supports) are calculated in terms of the percentage of the frequency $|< s >|$ of a pattern s compared to the number of sequences $|D|$ in the database.

TABLE V. EXAMPLE RESULTS – NSC AND REPETITION SUPPORTS (MIN_SUP=2)

RPSP	NSC	Related RPSP	sup	rsup
<a>	<¬***a***>	*<a>*	2	**2**
<b>	<¬*b*>	*<b>*	1	1
<c>	<¬***c***>	*<c>*	2	**2**
<e>	<¬***e***>	*<e>*	3	**3**
<(bc)>	<¬***(bc)***>	*<(bc)>*	2	**2**
<a b>	*<¬a b>* <***a*** ¬***b***>	*<b>, <a b>* *<a>, <a b>*	1 2	1 **2**
<a c>	*<¬a c>* <***a*** ¬***c***>	*<c>, <a c>* *<a>, <a c>*	1 1	1 **2**
<a e>	*<¬a e>* <***a*** ¬***e***>	*<e>, <a e>* *<a>, <a e>*	1 2	1 **2**
<b b>	<¬***b b***> <***b*** ¬***b***>	*<b>, <b b>* *<b>, <b b>*	2 2	**2** **2**
<b c>	*<¬b c>* <***b*** ¬***c***>	*<c>, <b c>* *<b>, <b c>*	1 2	1 **3**
<b e>	*<¬b e>* <***b*** ¬***e***>	*<e>, <b e>* *<b>, <b e>*	1 3	1 **4**
<a b e>	*<¬a b e>* *<a ¬b e>* <***a b*** ¬***e***> *<¬a b ¬e>*	*<b e>, <a b e>* *<a e>, <a b e>* *<a b>, <a b e>* *<b>, <a b>, <b e>*	0 0 2 1	0 0 **2** 1
<a b c>	*<¬a b c>* *<a ¬b c>* <***a b*** ¬***c***> *<¬a b ¬c>*	*<b c>, <a b c>* *<a c>, <a b c>* *<a b>, <a b c>* *<b>, <a b>, <b c>*	0 0 1 1	0 0 **2** 1
<a (bc)>	*<¬a (bc)>* <***a*** ¬***(bc)***>	*<(bc)>, <a (bc)>* <a>, *<a (bc)>*	0 1	0 **2**

A. Datasets

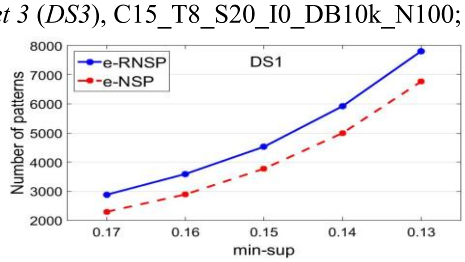
We use the following data factors: *C, T, S, I, DB* and *N* to describe and observe the effect of data characteristics on algorithm performance, which are defined to describe characteristics of sequence data [40]. C: Average number of elements per sequence; T: Average number of items per element; S: Average size of maximal potentially large sequences; I: Average size of items per element in maximal potentially large sequences; DB: The number of sequences; N: The number of items.

Four source databases are applied in this experiment. The synthetic databases are generated by IBM data generator.

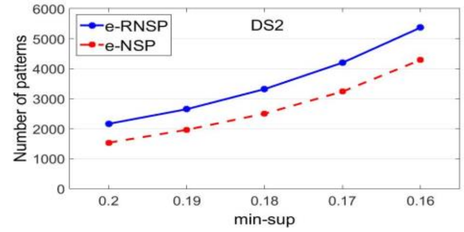
Dataset 1 (*DS1*), C8_T6_S6_I6_DB10k_N100;

Dataset 2 (*DS2*), C12_T4_S6_I6_DB10k_N100;

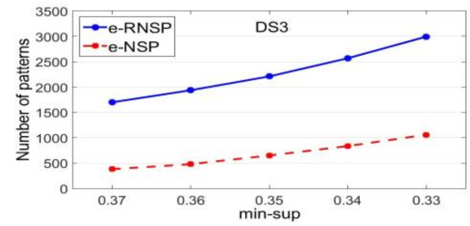
Dataset 3 (*DS3*), C15_T8_S20_I0_DB10k_N100;



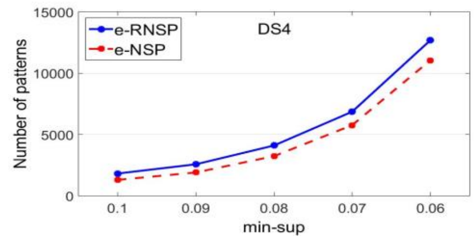
(a) Results on DS1



(b) Results on DS2



(c) Results on DS3



(d) Results on DS4

Fig.3 The Number of Patterns Comparison

Dataset 4 (*DS4*) is the real application dataset about health insurance claim sequences. This data contains 5,269 customers, each sequence stands for one customer. The average size in a sequence is 21. The maximum size of a sequence is 144, and the minimum size is 1. The size of this dataset is around 5M. We use the above four datasets to evaluate the mining performance of e-RNSP.

Dataset 5 (*DS5*) is a real dataset which contains 9 sets of sanitized user data drawn from the command histories of 8 UNIX computer users at Purdue over the course of up to 2 years. Due to the confidentiality of DS4, we choose this real-life dataset to present the patterns mined by our approach.

We further create Dataset 6 (*DS6*: C12_T10_S20_I10_DB 1k_N100). Based on it, we generate 15 additional datasets in terms of different data factors, denoted as *DS 6*.x (x = 1…15), to access the runtime and pattern number of e-RNSP and e-NSP influenced by different data factors. For instance, *DS6* = **C12**_T10_S20_I10_DB1k_N100, DS6.1=**C13**_T10_S20_I10_DB1k_N100, and DS6.2=**C14**_T10_S20_I10_DB1k_N100 are different on factor **C**, which means they have different average numbers of elements in a sequence, while the other factors are fixed. These datasets are listed in Table VII.

B. The Ability of Mining Patterns

The number of negative patterns mined by e-NSP, Neg-GSP and PNSP respectively are the same because we use a unified negative containment definition for all of them. Therefore, here we just need to compare e-RNSP with e-NSP, and the results are shown in Fig. 3. e-RNSP has the ability of mining more negative patterns than e-NSP at the same *min_sup*, because it caters for the repetition negative patterns when calculating the NSC support.

The number of RNSP is greatly affected by the distribution of a dataset. The more repetition items in a dataset are, the more the number of RNSP are. The repetition items in DS3 are more than the other datasets, so the gap between the two lines on DS3 is larger than the other. More details about the pattern number impacted by data factors are discussed in Sections 6.4 and 6.5.

To reveal the strength of e-RNSP, we choose two real-life results mined from *DS5*, shown in Table VI. It is clear that these two RNSP have the higher repetition supports but the lower traditional support, which might be ignored if setting a small support threshold. The first e-RNSP means if an operator uses '*is*' to list the catalogue, he will not use the instruction '*finger*' to search user's information but often utilize '*cd*' to change other catalogues. The second RNSP presents that if the operator did not use '*rm*' to delete files after listing the catalogue, it has a high probability of changing and showing the next catalogue subsequently.

TABLE VI. EXAMPLE RESULTS OF DS5

	RNSP	*sup*	*rsup*
1	*<is, ¬finger, cd>*	46	122
2	*<is, ¬rm, cd, is>*	40	72

C. Computational Cost

For observing the efficiency of e-RNSP, we conduct experiments on DS1 and DS2 with four algorithms and just run e-RNSP and e-NSP on DS3 and DS4. In the following comparisons, all positive patterns are identified by RptGSP, negative patterns are further mined by e-RNSP, e-NSP, NegGSP and PNSP. So their runtime of mining PSP are the same. In order to show their difference, we just need to compare their runtime on mining negative patterns. Fig. 4 and Fig. 5 show the comparisons.

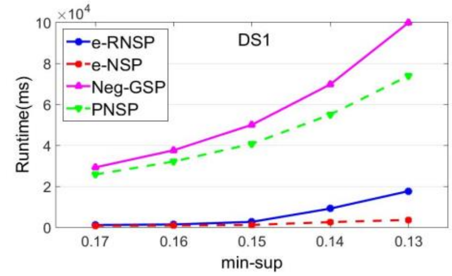
From fig. 4 we can see that e-RNSP and e-NSP are much faster than the other algorithms. E-RNSP spends 3% to 20% of the running time of PNSP and NegGSP on DS1 and DS2. For example, e-RNSP spends 3.7% to 17.6% of Neg-GSP running time on *DS1* when *min_sup* decreases from 0.17 to 0.13. E-RNSP and e-NSP are both efficient, because they only need to calculate the NSC support based on identified positive partners, while Neg-GSP and PNSP have to re-scan the whole datasets.

However, from Fig. 5 we can see that the running time of e-RNSP is also higher than e-NSP, especially when *min_sup* decreases. The reasons are as follows.

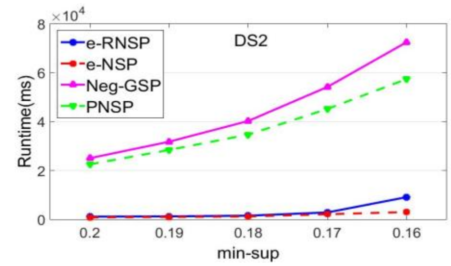
(1) In order to calculate the repetition support, e-RNSP has to count the number of times that a NSC repetition occurs in the database, whereas e-NSP does not need to do so.

(2) The number of NSC generated from e-RNSP is larger than that in e-NSP, because e-RNSP needs to consider the RSP problem when it mines PSP, but e-NSP mines PSP only.

In our future work, we will further study the method to increase the efficiency of e-RNSP.

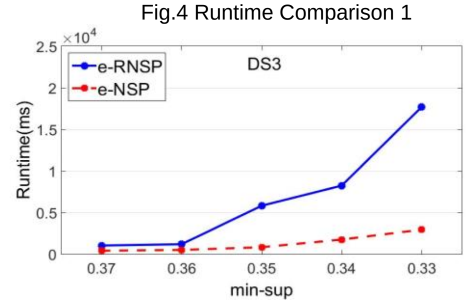


(a) Results on DS1

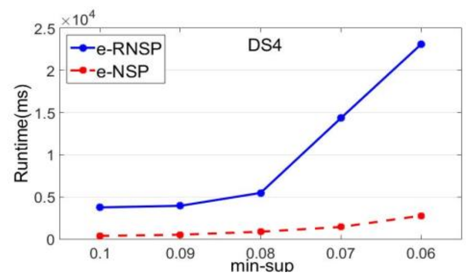


(b) Results on DS2

Fig.4 Runtime Comparison 1



(a) Results on DS3



(b) Results on DS4

Fig.5 Runtime Comparison 2

D. Performance Analysis of the Impact of Different Data Factors

1) Effect of C on Pattern Number

Here we analyze the impact of tuning data factor C on the pattern number of e-RNSP and e-NSP while fixing other factors T, S, I, DB and N. C is the size of data sequence, and its increase directly causes the increase of *rsup* in e-RNSP. So the number of RNSP increases quickly with the increase of C (the maximum number of RNSP can be mined when setting C to 15). Although the number of NSP also increases with the increase of C, its increasing speed is slower than that in e-RNSP.

2) Effect of T on Pattern Number

This is to adjust data factor T while fixing others to observe its impact on the pattern number. The increase of T will increase the number of RNSP and NSP (the maximum number of NSP can be mined when setting T to 14). This is because, with T increasing, i.e., the average number of items per element increasing, the number of NSC increases. Hence, the number of RNSP and NSP increase.

3) Effect of S on Pattern Number

This is to adjust data factor S while fixing others to observe its impact on the pattern number. The increase of S will decrease the number of RNSP and NSP (the maximum number of RNSP can be mined when setting S to 14). This is because, with S increasing, i.e., the average size of maximal potentially large sequences increasing, the number of NSC decreases. Hence, the number of RNSP and NSP decrease.

4) Effect of I on Pattern Number

This is to tune the factor I to observe its impact on the pattern number. With I increasing, the numbers of RNSP and NSP increase too (the maximum number of RNSP can be mined when setting I to 16). But e-RNSP increases proportionally faster than e-NSP, and the gap thus increases too.

5) Effect of DB on Pattern Number

The effect of DB on Pattern Number will be discussed in Section 6.5 (scalability test).

6) Effect of N on Pattern Number

Similarly, we adjust N while fixing all other data factors. Increasing N will decrease repetition items in data sequence, which further decrease the support of sequences (the maximum number of RNSP can be mined when setting N to 200). Hence, the numbers of RNSP and NSP decrease with the increase of N.

In summary, e-RNSP can perform efficiently from the various data factor perspectives.

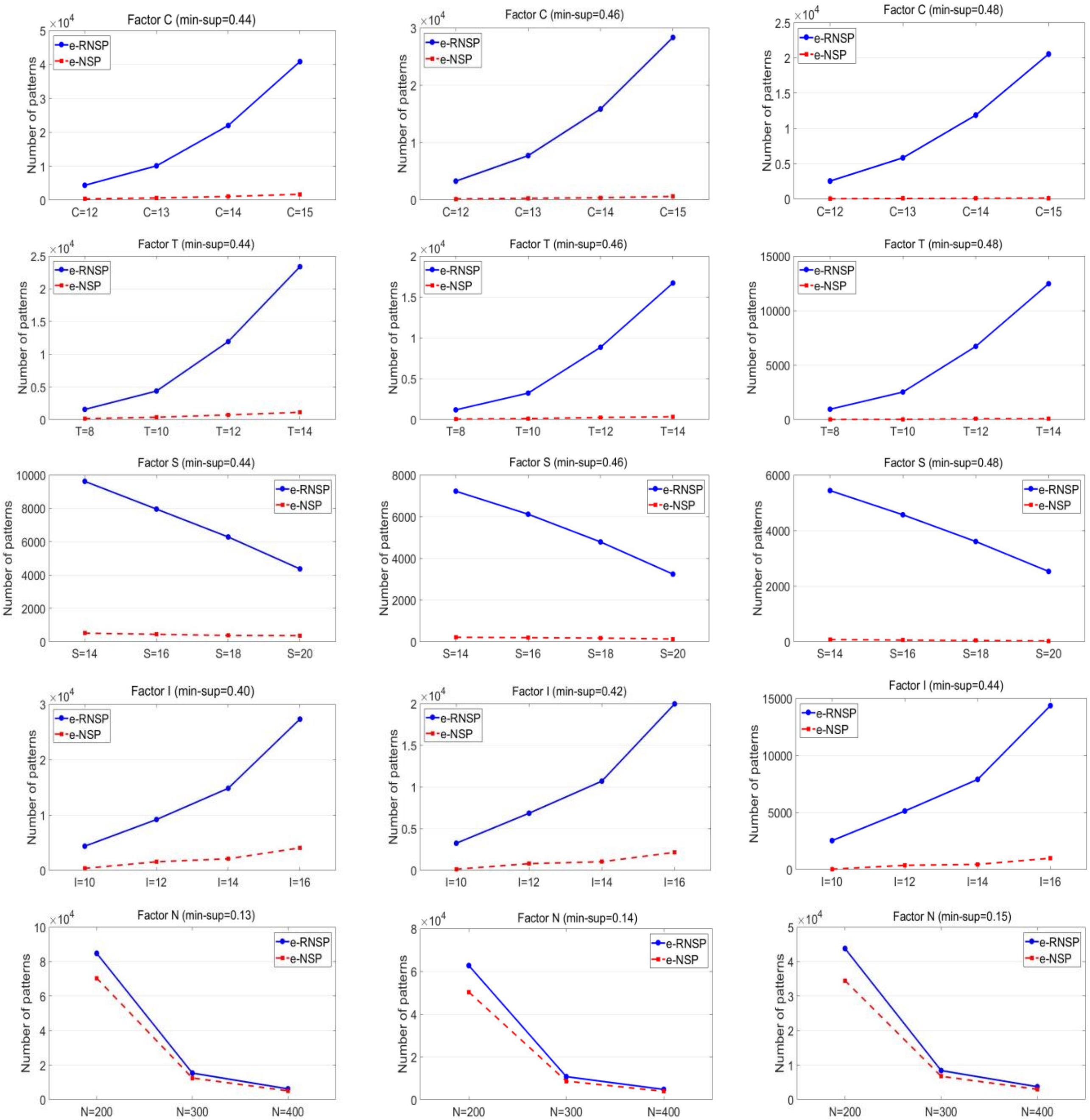


Fig.6 Pattern Number Comparison on Various Factors

TABLE VII. DATASET CHARACTERISTICS ANALYSIS RESULTS

Data factors	Dataset ID	*min_sup*	RNSP number by e-RNSP (n_1)	NSP number by e-NSP (n_2)	NSC number by e-RNSP (n_3)	NSC number by e-NSP (n_4)	RNSP time by e-RNSP (t_1, ms)	NSP time by e-NSP (t_2, ms)	t_1/n_3 *1000 (ns)	t_2/n_4 *1000 (ns)	$(t_1/n_3)/(t_2/n_4)$
C=12	DS6=C12_T10_S20_I10_DB1k_N100	0.44	4365	364	25786	9740	889	219	34.48	22.48	1.53
		0.46	3242	129	19765	6811	717	140	36.28	20.55	1.76
		0.48	2528	26	15598	5018	593	125	38.02	24.91	1.53
C=13	DS6.1=C13_T10_S20_I10_DB1k_N100	0.44	10102	645	78909	33385	2528	780	32.04	23.36	1.37
		0.46	7698	251	61133	23909	1966	561	32.16	23.46	1.37
		0.48	5827	59	47149	17127	1622	406	34.40	23.71	1.45
C=14	DS6.2=C14_T10_S20_I10_DB1k_N100	0.44	21999	1074	255651	86606	8174	2262	31.97	26.12	1.22
		0.46	15847	344	187575	58213	6084	1544	32.44	26.52	1.22
		0.48	11885	84	142612	41012	4696	1124	32.93	27.41	1.20
C=15	DS6.3=C15_T10_S20_I10_DB1k_N100	0.44	**40850**	1724	657109	258587	21263	6973	32.36	26.97	1.20
		0.46	**28341**	573	467934	168421	14368	4617	30.71	27.41	1.12
		0.48	**20545**	111	347521	113561	11263	3151	32.41	27.75	1.17
T=8	DS6.4=C12_T8_S20_I10_DB1k_N100	0.44	1558	152	6967	2554	250	62	35.88	24.28	1.48
		0.46	1194	62	5530	1883	250	47	45.21	24.96	1.81
		0.48	951	18	4405	1373	187	47	42.45	34.23	1.24
T=10	DS6=C12_T10_S20_I10_DB1k_N100	0.44	4365	364	25786	9740	889	219	34.48	22.48	1.53
		0.46	3242	129	19765	6811	717	140	36.28	20.55	1.76
		0.48	2528	26	15598	5018	593	125	38.02	24.91	1.53
T=12	DS6.5=C12_T12_S20_I10_DB1k_N100	0.44	11928	719	94430	35777	3010	843	31.88	23.56	1.35
		0.46	8848	264	71475	25132	2278	609	31.87	24.23	1.32
		0.48	6716	83	54793	17754	1841	499	33.60	28.11	1.20
T=14	DS6.6=C12_T14_S20_I10_DB1k_N100	0.44	**23374**	1122	260022	82539	8268	2060	31.80	24.96	1.27
		0.46	**16699**	351	187848	55017	6037	1419	32.14	25.79	1.25
		0.48	**12462**	78	142026	38485	5445	982	38.34	25.52	1.50
S=14	DS6.7=C12_T10_S14_I10_DB1k_N100	0.44	**9618**	512	72176	22129	2403	530	33.29	23.95	1.39
		0.46	**7221**	211	54539	15362	1950	421	35.75	27.41	1.30
		0.48	**5438**	78	42056	10548	1560	265	37.09	25.12	1.48
S=16	DS6.8=C12_T10_S16_I10_DB1k_N100	0.44	7952	442	59822	20620	1950	484	32.60	23.47	1.39
		0.46	6116	193	46341	14626	1544	357	33.32	24.41	1.37
		0.48	4567	57	35074	10056	1217	281	34.70	27.94	1.24
S=18	DS6.9=C12_T10_S18_I10_DB1k_N100	0.44	6281	372	40975	15662	1389	358	33.90	22.86	1.48
		0.46	4786	174	31905	11231	1154	265	36.17	23.60	1.53
		0.48	3605	43	24235	7751	920	188	37.96	24.25	1.57
S=20	DS6=C12_T10_S20_I10_DB1k_N100	0.44	4365	364	25786	9740	889	219	34.48	22.48	1.53
		0.46	3242	129	19765	6811	717	140	36.28	20.55	1.76
		0.48	2528	26	15598	5018	593	125	38.02	24.91	1.53
I=10	DS6=C12_T10_S20_I10_DB1k_N100	0.40	4365	364	25786	9740	889	219	34.48	22.48	1.53
		0.42	3242	129	19765	6811	717	140	36.28	20.55	1.76
		0.44	2528	26	15598	5018	593	125	38.02	24.91	1.53
I=12	DS6.10=C12_T10_S20_I12_DB1k_N100	0.40	9185	1550	44352	18521	1466	343	33.05	18.52	1.78
		0.42	6842	792	33306	13033	1107	249	33.24	19.11	1.74
		0.44	5120	364	25435	9157	889	187	34.95	20.42	1.71
I=14	DS6.11=C12_T10_S20_I14_DB1k_N100	0.40	14822	2092	81319	28823	2386	515	29.34	17.87	1.64
		0.42	10691	1020	59718	19286	1825	343	30.56	17.78	1.72
		0.44	7888	445	44615	13181	1435	234	32.16	17.75	1.81
I=16	DS6.12=C12_T10_S20_I16_DB1k_N100	0.40	**27318**	4069	146127	52033	4040	874	27.65	16.80	1.65
		0.42	**19958**	2152	108510	36353	3058	624	28.18	17.17	1.64
		0.44	**14354**	981	79329	24510	2293	421	28.90	17.18	1.68
N=200	DS6.13=C12_T10_S20_I10_DB1k_N200	0.13	**84754**	70363	108177	92026	2122	764	19.62	8.30	2.36
		0.14	**62707**	50293	80757	67090	1639	562	20.30	8.38	2.42
		0.15	**43780**	34476	57287	46690	1264	421	22.06	9.02	2.45
N=300	DS6.14=C12_T10_S20_I10_DB1k_N300	0.13	15326	12458	16401	13449	374	93	22.80	6.92	3.30
		0.14	10780	8695	11511	9460	280	94	24.32	9.94	2.45
		0.15	8393	6727	9001	7374	218	78	24.22	10.58	2.29
N=400	DS6.15=C12_T10_S20_I10_DB1k_N400	0.13	6159	4992	5104	5104	156	47	30.56	9.21	3.32
		0.14	4807	3923	4002	4002	141	31	35.23	7.75	4.55
		0.15	3703	2956	3028	3028	125	21	41.28	6.94	5.95

E. Scalability Test

e-RNSP calculates support based on calculation not on re-scanning database, thus its performance is sensitive to the size of data sequence. If a dataset is huge, it produces a large number of data sequences. The scalability test is conducted to evaluate the e-RNSP performance on large datasets. Fig. 7 shows the results of e-RNSP on datasets *DS*6 in terms of different data sizes: from 5 times (see the results corresponding to label 'X6') of its original size to 25 times, with minimum supports 0.4 and 0.46 respectively.

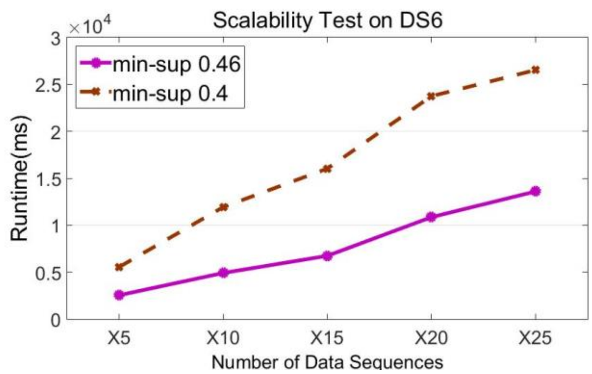


Fig. 7. Scalability Test on Data Factor DB on DS6

Fig. 7 shows that the growth of running time of e-RNSP follows a roughly linear relationship with the data size increase on different minimum supports.

VII. Conclusion and Future Work

Repetition sequential patterns (RSP) are usually used to understand those special behaviors with repetition sequences and thus have attracted increasing attention in recent years. We have not found any work to identify repetition negative sequential patterns (RNSP), which can capture non-occurring repetition behavioral patterns. RNSP can play a role irreplaceable by RSP to understand such issues that a lung cancer patient iteratively avoiding certain treatment combinations may cause a lower survival rate. In this paper, we define the repetition negative containment problem and propose an efficient RNSP mining algorithm, named e-RNSP. e-RNSP has been tested on both real-world and synthetic databases and compared with three available NSP methods: e-NSP, NegGSP and PNSP. The experiments and comparisons on 15 databases have clearly demonstrated that e-RNSP could efficiently capture interesting repetition negative patterns.

Not all of patterns mined by e-RNSP are actionable. We will consider constraints on RNSP to enhance the actionability of RNSP findings, and improve the mining efficiency by using bitmap strategy. In addition, in pattern mining, it is an open issue to verify the correctness and completeness of patterns discovered by a pattern mining algorithm. We will explore this further with the NSP research.

Acknowledgment

This work was partially supported by National Natural Science Foundation of China (71271125), and Natural Science Foundation of Shandong Province, China (ZR2018MF011), and Australian Research Council Linkage grant (DP130102691).

References

[1] L. Cao., Y. Zhao, and C. Zhang, "Mining Impact-Targeted Activity Patterns in Imbalanced Data", *IEEE Trans. on Knowledge and Data Engineering*, vol.20, no. 8, pp. 1053-1066, 2008.

[2] L. Cao, Y. Ou and P.S Yu, "Coupled Behavior Analysis with Applications", *IEEE Trans. on Knowledge and Data Engineering*, vol. 24, no. 8, pp. 1378-1392, 2012.

[3] Z. Zheng, W. Wei, C. Liu, W. Cao and L. Cao, Maninder Bhatia. "An effective contrast sequential pattern mining approach to taxpayer behavior analysis", *World Wide Web*, vol. 19, no. 4, pp. 633-651, 2016.

[4] Y. Song, L. Cao, X. Wu, G. Wei, W. Ye and W. Ding, "Coupled Behavior Analysis for Capturing Coupling Relationships in Group-based Market Manipulation", *In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp.976-984, 2012.

[5] T. Xu, T. Li, and X. Dong, "Efficient High Utility Negative Sequential Patterns Mining in Smart Campus". *IEEE Access*, vol. 6, pp. 23839 - 23847, 2018.

[6] Y. Gong, C. Liu and X. Dong, "Research on Typical Algorithms in Negative Sequential Pattern Mining". *The Open Automation and Control Systems Journal*, vol.7, pp. 934-941, 2015.

[7] L. Cao. and P.S. Yu, "Behavior Computing: Modeling, Analysis, Mining and Decision", 2012.

[8] L. Cao, "Behavior informatics: A new perspective". *IEEE Intelligent Systems*, vol. 29, no. 4, pp. 62-80, 2014.

[9] L. Cao, "In-depth Behavior Understanding and Use: The Behavior Informatics Approach", *Information Science*, vol. 180, no. 17, pp. 3067-3085, 2010.

[10] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements", *International Conference on Extending Database Technology*, pp. 1-17, 1996.

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

[12] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M.C. Hsu, "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth", *International Conference on Data Engineering*, pp.215-226, 2001.

[13] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences", *Machine Learning*, vol. 42, no. 1-2, pp. 21-60, 2001

[14] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," *In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 429-435, 2002.

[15] X. Dong, Z. Zheng, L. Cao, Y. Zhao, C.Q. Zhang, J.J. Li, W. Wei and Y.M. Ou, "E-NSP: Efficient negative sequential pattern mining based on identified positive patterns without database rescanning." *International Conference on Information and Knowledge Management, Proceedings*, pp. 825-830, 2011

[16] S. Hsueh, M. Lin and C. Chen, "Mining Negative Sequential Patterns for E-commerce Recommendations." *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference*, pp.1213-1218, 2008.

[17] Z. Zheng, Y. Zhao, Z. Zuo and L. Cao, "Negative-GSP: An Efficient Method for Mining Negative Sequential Patterns." *The 8th Australian Data Mining Conference*. vol.101, pp. 63-67, 2009.

[18] N. Lin, H. Chen, and W. Hao, "Mining negative sequential patterns." *In process. of the 6th WSEAS International Conference on Applied Computer Science*, Hangzhou, China, pp. 654–658, 2007.

[19] Y. Gong, X. Dong, X. Han and R. Hou, "Mining Non-overlapping Repetitive Sequential Patterns by Improving GSP Algorithm", *The Open Cybernetics & Systemics Journal*, vol. 9, pp. 473-477, 2015.

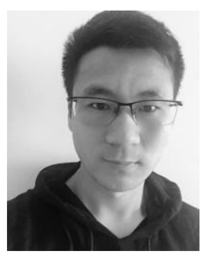
[20] B. Ding, D. Lo and J. Han, "Efficient Mining of Closed Repetition Gapped Subsequences from a Sequence Database." *International Conference on Data Engineering*, pp. 1024-1035, 2009.

[21] L. Brooke. Heidenfelder and D. Michael. Topal, "Effects of sequence on repeat expansion during DNA replication." *Nucleic Acids Research*, vol. 31, no. 24, pp. 7159-7164, 2003.

[22] M. Zhang, B. Kao, D. Cheung and K. Yip, "Mining periodic patterns with gap requirement from sequences," *ACM Transactions on Knowledge*

Discovery from Data, vol.1, issue. 2, no.7, pp. 1-40, 2007.
[23] Y. Tong, L. Zhao, D. Yu, S. Ma, Z. Cheng and K. Xu, “Mining Compressed Repetition Gapped Sequential Patterns Efficiently,” *In International Conference on Advanced Data Mining and Applications*, pp. 652-660, 2009.
[24] E. Lee, W. Kim, J. Ryu and U. Kim, “Efficient Weighted Mining of Repetition Subsequences,” In *Web Society, 2009. SWS'09. 1st IEEE Symposium on*, pp. 66-70. IEEE, 2009.
[25] C. Ma and W. Shen, “Clustering Navigation Patterns using Closed Repetition Gapped Subsequence,” *Logistics Systems and Intelligent Management*, vol.3, pp. 1660 -1663, 2010.
[26] H. Mannila, H. Toivonen and A.I. Verkamo, “Discovery of frequent episodes in event sequences,” *Data mining and knowledge discovery*, vol. 1, no.3, pp.259-289, 1997.
[27] D. Lo, S.-C. Khoo and C. Liu, “Efficient mining of iterative patterns for software specification discovery,” in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 460-469, 2007.
[28] D. Lo, J. Li, L. Wong and S.-C. Khoo, “Mining Iterative Generators and Representative Rules for Software Specification Discovery,” *IEEE Transactions on Knowledge and Data Engineering*, vol.32, no.2, pp. 282-296, 2011.
[29] J. Han, G. Dong and Y. Yin, “Efficient mining of partial periodic patterns in time series database,” *In Data Engineering, Proceedings., 15th International Conference on*, pp. 106-115, 1999.
[30] J. Yang, W. Wang, and P.S. Yu, “Mining asynchronous periodic patterns in time series data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no.3, pp. 613-628, 2003.
[31] W. Ouyang and Q. Huang, “Mining negative sequential patterns in transaction databases.” I*n Proc. of International Conference on Machine Learning and Cybernetics*, pp. 830-834, 2007.
[32] N. Lin, H. Chen, W. Hao, H. Chueh and C. Chang, “Mining Negative Fuzzy Sequential Patterns.” *In Proceedings of the 7th WSEAS international conference on simulation, modelling and optimization*, pp.52-57, 2007.
[33] N. Lin, H. Chen, W. Hao, H. Chueh and C. Chang, “Mining Strong Positive and Negative Sequential Patterns.” *WSEAS Transactions on Computers*, vol. 7, no. 3, pp. 119-124, 2008.
[34] F. Rasheed and R. Alhajj, “A Framework for Periodic Outlier Pattern Detection in Time-Series Sequences.” *Cybernetics, IEEE Transactions on*, vol. 44, no. 5, pp. 569 - 582, 2014.
[35] S. Zhang, Z. Du and J. Wang, “New Techniques for Mining Frequent Patterns in Unordered Trees.” *Cybernetics, IEEE Transactions on*, vol. 45, no. 6, pp. 1113 – 1125, 2014.
[36] J. Luna, J. Romero, C. Romero and S. Ventura, “On the Use of Genetic Programming for Mining Comprehensible Rules in Subgroup Discovery.” *Cybernetics, IEEE Transactions on*, vol. 44, no. 12, pp. 2329 -2341, 2014.
[37] Y. Chen and T.K. Huang, “Discovering fuzzy time-interval sequential patterns in sequence databases.” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*. Vol. 35, no. 5, pp. 959 - 972, 2005.
[38] I. Traore, I. Woungang, Y. Nakkabi, M.S. Obaidat. A.A.E. Ahmed and B. Khalilian, “Dynamic Sample Size Detection in Learning Command Line Sequence for Continuous Authentication.” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 5, pp.1343 - 1356, 2012.
[39] Z. Liu; L. Bruton, J. Bezdek, J. Keller, S. Dance, N. Bartley and C. Zhang, “Dynamic image sequence analysis using fuzzy measures.” *Systems, Man, and Cybernetic*, vol. 31, no. 4, pp.557 - 572, 2001.
[40] L. Cao, X. Dong and Z. Zheng, “e-NSP: Efficient negative sequential pattern mining.” *Artificial Intelligence*, vol. 235, pp. 156-182, 2016.
[41] T. Xu, X Dong, J. Xu and Y. Gong, “E-msNSP: Efficient negative sequential patterns mining based on multiple minimum supports.” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 31, no. 2, pp.1-17, 2017.
[42] Y. Gong, T. Xu, X. Dong and G. Lv, “e-NSPFI: Efficient Mining Negative Sequential Pattern from both Frequent and Infrequent Positive Sequential Patterns.” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 31, no.2, pp.1-20, 2017.
[43] C. Ishak, A. Marshall, D. Passos, et al., “An RB-EZH2 Complex Mediates Silencing of Repetition DNA Sequences.” *Molecular Cell*, vol. 64, no.6, pp. 1074-1087, 2016.
[44] R.L. Alvaro, V. Leonardo, C. Mauro and M. A. Vega-Rodr ǵuez, “A Characteristic-Based Framework for Multiple Sequence Aligners.” *IEEE Transactions on Cybernetics*, vol. 48, no.1, pp. 41-51, 2018.
[45] S. Elsayed, R. Sarker, C. Coello and A. Carlos, “Sequence-Based Deterministic Initialization for Evolutionary Algorithms.” *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2911-2923, 2017.
[46] L. Cao, P. Yu and V. Kumar, “Nonoccurring Behavior Analytics.” *IEEE Intelligent Systems*, vol. 30, no. 6, pp. 4-11, 2015.
[47] X. Dong, Y. Gong and L. Cao, “F-NSP+: A fast negative sequential patterns mining method with self-adaptive data storage.” *Pattern Recognition*, vol. 84, pp. 13-27, 2018.
[48] I. H. Toroslu, “Repetition support and mining cyclic patterns.” *Expert Systems with Applications*, vol. 25, no. 3, pp.303-311, 2003.
[49] Y. H. Hu, C.F Tsai, C.T Tai and I. C Chiang, “A novel approach for mining cyclically repeated patterns with multiple minimum supports.” *Applied Soft Computing*, vol. 28, pp.90-99, 2015.
[50] D.A Chiang., C.T Wang, S.P Chen and C.C Chen, “The cyclic model analysis on sequential patterns.” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1617-1628, 2009.
[51] Y. Li, Y. Zhao, G. Wang, et al., “ELM-Based Large-Scale Genetic Association Study via Statistically Significant Pattern.” *IEEE Transactions on Systems Man and Cybernetics Systems*, issue. 99, pp.1-14, 2017.
[52] D. Fradkin and F. Mrchen, “Mining sequential patterns for classification.” *Knowledge and Information Systems*, vol. 45, no. 3, pp. 731-749, 2015.
[53] M. Fan, Z. Zhang, W. Zhai and R. Shen, "Frequent pattern discovery with tri-partition alphabets." *Information Sciences*, pp. 1-18, 2018.
[54] T. Le, A. Nguyen, B. Huynh, B. Vo, and W. Pedrycz, “Mining constrained inter-sequence patterns: a novel approach to cope with item constraints.” *Applied Intelligence*, vol. 48, no. 5, pp. 1327-1343, 2018.
[55] M.R, Karim, M. Cochez, O.D. Beyan, C.F. Ahmed and S. Decker, “Mining maximal frequent patterns in transactional databases and dynamic data streams: A spark-based approach”. *Information Sciences*, vol. 432, pp. 278-300, 2018.

Xiangjun Dong received the Ph.D degrees in computer applications from Beijing Institute of Technology in 2005. From 2007 to 2009, he was a postdoctoral position in School of Management and Economics, Beijing Institute of Technology. He is currently the Professor and master’s tutor of School of Information, Qilu University of Technology (Shandong Academy of Sciences) in Jinan, China. He is the author of more than 70 academic papers. His research interests include data mining, association rules, sequential pattern mining and negative sequential pattern mining. He is a director of Shandong Computer Federation.



Yongshun Gong is now working toward the Ph.D. degree in the Faculty of Engineering and IT, University of Technology Sydney, Australia. His principal research interest covers the data science and machine learning, in particular, the following areas: traffic analysis; crowd flow prediction; matrix factorization and sequential pattern mining. He has published 5 journal papers and 3 international conference publications, including Pattern Recognition and CIKM.

Longbing Cao (SM’06) received the PhD degree in pattern recognition and intelligent systems from the Chinese Academy of Science, and the PhD degree in computing sciences from the University of Technology Sydney. He is a professor and the founding director of the UTS Advanced Analytics Institute. His current research interests include data science, artificial intelligence, behavior informatics, and their enterprise applications. He is a senior member of the IEEE