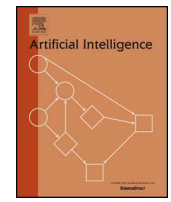
Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



e-NSP: Efficient negative sequential pattern mining ☆

Longbing Cao [a,*], Xiangjun Dong [b], Zhigang Zheng [c]

[a] University of Technology Sydney, Australia
[b] Qilu University of Technology, Jinan, China
[c] University of Technology Sydney, Australia

ARTICLE INFO

Article history:
Received 13 January 2015
Received in revised form 3 March 2016
Accepted 7 March 2016
Available online xxxx

Keywords:
Negative sequence analysis
Sequence analysis
Behavior analytics
Non-occurring behavior
Behavior informatics
Behavior computing
Pattern mining

ABSTRACT

As an important tool for behavior informatics, negative sequential patterns (NSP) (such as missing medical treatments) are critical and sometimes much more informative than positive sequential patterns (PSP) (e.g. using a medical service) in many intelligent systems and applications such as intelligent transport systems, healthcare and risk management, as they often involve non-occurring but interesting behaviors. However, discovering NSP is much more difficult than identifying PSP due to the significant problem complexity caused by non-occurring elements, high computational cost and huge search space in calculating negative sequential candidates (NSC). So far, the problem has not been formalized well, and very few approaches have been proposed to mine for specific types of NSP, which rely on database re-scans after identifying PSP in order to calculate the NSC supports. This has been shown to be very inefficient or even impractical, since the NSC search space is usually huge. This paper proposes a very innovative and efficient theoretical framework: set theory-based NSP mining (ST-NSP), and a corresponding algorithm, e-NSP, to efficiently identify NSP by involving only the identified PSP, without re-scanning the database. Accordingly, negative containment is first defined to determine whether a data sequence contains a negative sequence based on set theory. Second, an efficient approach is proposed to convert the negative containment problem to a positive containment problem. The NSC supports are then calculated based only on the corresponding PSP. This not only avoids the need for additional database scans, but also enables the use of existing PSP mining algorithms to mine for NSP. Finally, a simple but efficient strategy is proposed to generate NSC. Theoretical analyses show that e-NSP performs particularly well on datasets with a small number of elements in a sequence, a large number of itemsets and low minimum support. e-NSP is compared with two currently available NSP mining algorithms via intensive experiments on three synthetic and six real-life datasets from aspects including data characteristics, computational costs and scalability. e-NSP is tens to thousands of times faster than baseline approaches, and offers a sound and effective approach for efficient mining of NSP in large scale datasets by directly using existing PSP mining algorithms.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

UNCORRECTED PROOF

☆ The source codes of e-NSP are available from http://www-staff.it.uts.edu.au/~lbcao/.

* Corresponding author.
E-mail address: longbing.cao@gmail.com (L. Cao).

http://dx.doi.org/10.1016/j.artint.2016.03.001
0004-3702/© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

Behavior is widely seen in our daily study, work, living and entertainment [7]. A critical issue in understanding behavior from the informatics perspective, namely behavior informatics [6,9], is to understand the complexities, dynamics and impact of non-occurring behaviors (NOB) [8]. Mining *Negative sequential patterns* (NSP) [43] is one of few approaches available for understanding NOB. NSP refer to frequent sequences with non-occurring and occurring behaviors (also called *negative* and *positive* behaviors in behavior and sequence analysis), such as a driver failing to stop before driving through an intersection with a stop sign or a missing treatment in medical service.

Discovering NSP is becoming increasingly important, and sometimes play a role that cannot be replaced by analyzing occurring behaviors alone 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. For example, in ITS, negative driving behavior patterns result in drivers failing to follow certain traffic rules could cause serious traffic problems or even disasters. In healthcare, a patient missing an important medical appointment might result in serious health issues. In gene sequencing, the non-occurrence of certain genes may be associated with particular diseases. Such problems cannot be handled by the identification of occurring behavior patterns alone.

Formally, a NSP in healthcare may appear as follows. Assume $p_1 =< abc\ X >$ is a *positive sequential pattern* (PSP); $p_2 = < ab\neg c\ Y >$ is a NSP, where a, b and c stand for medical service codes indicating the services a patient has received in health care, and X and Y stand for disease status. p_1 shows that patients who usually receive medical services a, b and then c are likely to have disease status X, whereas p_2 indicates that patients receiving treatments of a and b but NOT c have a high probability of having disease status Y.

Although intensive efforts have been made to develop PSP (such as p_1) mining algorithms such as GSP [32], FreeSpan [15], SPADE [34], PrefixSpan [30], and SPAM [4], NSP (such as p_2) cannot be described or discovered by these algorithms. This is because mining NSP is much more difficult than mining PSP [8], particularly due to the following three intrinsic complexities.

- *Problem complexity*. The hidden nature of non-occurring items makes definition of the NSP mining problem complicated, particularly the NSP format and negative containment. This is why researchers present different and even inconsistent definitions and constraints in their identification of NSP. As research into NSP is at a very early stage, it is important to formalize the problem properly and comprehensively.
- *High computational complexity*. Existing methods calculate the support of negative sequential candidates (NSC) by additionally scanning the database after identifying PSP. This leads to additional costs and low efficiency in mining NSP. It is thus essential to develop efficient NSP mining methods without database re-scanning.
- *Large NSC search space*. The existing approaches generate k-size NSC by conducting a joining operation on (k-*1*)-size NSP. This results in a huge number of NSC [11,24,26,38], which makes it difficult to search for meaningful outputs. Further, NSC does not satisfy the Apriori principle [38]. It is a challenge to prune the large proportion of meaningless NSC, and it is thus important to develop efficient approaches for generating a limited number of truly useful NSC.

NSP mining is at an early stage, and has seen only very limited progress in recent years [3,11,12,14,16–18,20,21,38–40, 43]. All existing methods are very inefficient and are too specific for mining NSP. As NSP analysis is very complex, challenging and immature, the addition of appropriate constraints makes the problem solvable to some degree. All the reported work in NSP analysis therefore incorporates constraints on format, frequency and/or negative elements from respective aspects (see more discussion in Section 3.2.1) to reduce the number of NSC, discover specific NSP of particular interest, and enhance computational efficiency. More importantly, there are different definitions of the most important concept in NSP mining – negative containment – which defines whether a data sequence contains a negative sequence (see more details in Section 4.2). Some definitions are more generic and typical [11,12,38,39] than others which either incorporate additional constraints on negation format and containment [21–23,25–29] or offer no clear definition [18,31].

To address the above intrinsic complexities in NSP mining and make it efficient for real-life applications, this paper proposes an innovative, flexible and efficient framework: *the set theory-based NSP* mining framework (ST-NSP) and a comprehensive algorithm called *e-NSP* to instantiate the ST-NSP framework. We first formalize the NSP mining problem by defining some important concepts in NSP, including negative containment. The formalization draws a clear boundary between whether a data sequence set contains a NSC or not. Building on the set theory, e-NSP then calculates the support of NSC based on the support of the corresponding PSP, without additional database scans. We convert the negative containment problem to a positive containment problem. The NSC supports are then calculated by only using a NSC's corresponding PSP information. In this way, there is no need to re-scan the database after discovering PSP. More importantly, any existing PSP algorithms can then be directly used or slightly changed to discover NSP.

We specify the frequency, format and negative element constraints in e-NSP to make the problem consistent with set theory (frequency constraint), reduce confusion, ambiguity and uncertainty (format constraint), and to control computational complexity (negative element constraint). Our definitions and specifications of such constraints and negative containment form the ST-NSP framework and make e-NSP consistent with set theory which thus enables e-NSP to be much more efficient and flexible than existing methods.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Below, we summarize the main differences between our work and the related works in terms of the constraints imposed on NSP, key definitions and concepts, as well as the significant contributions made in this work:

- An innovative and very efficient framework ST-NSP and its corresponding theoretical design and algorithm e-NSP are proposed to discover NSP within one database scan; e-NSP is built on set theory and is the only work reported so far that uses set theory for sequence analysis, which opens a new paradigm for NSP analysis.
- e-NSP is built on a systematic and comprehensive statement and formalization of the NSP problem, and incorporates new concepts, specifications on constraints and effective technical design that make it efficient and flexible, in addition to offering substantial theoretical analysis in terms of data characteristics represented by data factors and computational costs.
- e-NSP introduces three constraints on frequency, format and negative elements respectively, which not only make e-NSP consistent with typical existing work but also support the proposed framework of set theory-based NSP study.
- Theoretical analysis shows that e-NSP is much less sensitive to data factors and is much more efficient on datasets that have a small number of elements in a sequence and a large number of itemsets, compared to available baseline algorithms we can find. This advantage of e-NSP is especially clear when minimum support is low. It is thus very suitable for large scale data.

Unfortunately, it is not easy to find baseline datasets and NSP algorithms that satisfy similar NSP settings. We compare e-NSP with two modified NSP algorithms in the literature using intensive experiments on three synthetic and six real-world datasets from many perspectives, including computational complexity against different minimum supports on 8 distinct datasets, data characteristics analysis on 48 combinations of various data factors on 16 subsets, and scalability tests on two scalable datasets with low minimum support. The experimental results verify the theoretical analyses, showing that e-NSP is much more flexible and efficient and is tens to thousands of times faster than the baseline methods, thus highly suitable for large scale datasets. To the best of our knowledge, this is the first approach that efficiently discovers NSP by involving PSP only without rescanning databases, directly applies existing PSP discovery algorithms, and is particularly effective for very large datasets. This demonstrates the significant value of the proposed ST-NSP framework.

The remainder of the paper is organized as follows. Section 2 discusses the related work and gaps in the current knowledge. In Section 3, we formalize the problem of mining PSP and NSP, providing corresponding definitions and constraints. The ST-NSP framework and the e-NSP algorithm are detailed in Section 4. The theoretical analyses of e-NSP compared to baseline are presented in Section 5 from the perspective of data factors. Section 6 presents substantial experimental results and a real-life case study. Discussions on several critical issues are offered in Section 7, followed by conclusions and future work in Section 8.

2. Related work

In this section, we first discuss the related work on a fundamental concept in NSP; that is, *negative containment*. Different researchers present inconsistent definitions and explanations of negative containment for respective interests and purposes. In [11], a data sequence $ds =< dc >$ cannot contain negative sequence $ns =< \neg(ab)c\neg d >$ since $size(ns) > size(ds)$; while the work in [38] allows that ds contains ns. Another critical issue is how to deal with a non-occurring element. Chen et al. [11] argued that $ds =< dc >$ cannot contain $< \neg cd >$ because $< d >$ in ds has no antecedent itemset; ds cannot contain $< c\neg d >$ because $< c >$ in ds has no successor. However, Zheng et al. [38] allowed that ds contains $< c\neg d >$. Furthermore, the containment position of each element is very tricky. Chen et al. [11] proposed that a data sequence $ds =< aacbc >$ does not contain a negative sequence $ns =< a\neg bc >$, since opposite evidence of $< abc >$ can be found in ds. However, Zheng et al. [38] presented a divided opinion since $ds =< aacbc >$ matches a and c; his algorithm finds the corresponding positive element in ds for each negative element of ns, such as the second a for $\neg b$. According to our understanding, since $< e >$ means that e occurs, no element (including element e) occurs before or after e. Accordingly, $< e >$ contains $< e\neg? >$, $< \neg?\ e >$, $< \neg?e\neg? >$, where "?" represents any element.

Second, we summarize the status of the NSP research. Unlike PSP mining, which has been widely explored, very limited research outcomes are available in the literature on mining NSP. We briefly introduce what we have been able to find. Zheng et al. [38] proposed a negative version of the GSP algorithm, i.e. NegGSP, to mine for NSP. This algorithm first discovers PSP by GSP, then generates and prunes NSC. It then counts the support of NSC by re-scanning the database to generate negative patterns. Chen et al. [11] proposed a PNSP approach for mining positive and negative sequential patterns in the form of $< (abc)\neg(de)(ijk) >$. This approach is broken into three stages. PSP are mined by traditional algorithms and all positive itemsets are derived from these PSP. All negative itemsets are then derived from these positive itemsets. Lastly, both positive and negative itemsets are joined to generate NSC, which are in turn joined iteratively to generate longer NSC in an Apriori-like way. This approach calculates the support of NSC by re-scanning the database. In [22–24], the authors only handled NSP in which the last element was negative. An algorithm NSPM in [24] mines such NSP. The extended versions of [24] are in [22,23], which add fuzzy and strong constraints respectively to NSPM. In [39], a genetic algorithm is proposed to mine NSP. It generates candidates by crossover and mutation by involving a dynamic fitness function to generate as many candidates as possible and avoid population stagnation. In [26], only NSP are identified in the form of $(\neg A, B)$, $(A, \neg B)$ and $(\neg A, \neg B)$, which is similar to mining negative association rules [13,33]. The work in [26] requires $A \bigcap B = \emptyset$, which is a

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Table 1
Notation description.

Symbol	Description
I	A set of items, $I = \{x_1, \ldots, x_n\}$, consisting of n items x_k $(1 \leqslant k \geqslant n)$
s	A sequence, $s =< s_1, \ldots, s_l >$, consisting of l elements s_j $(1 \leqslant j \geqslant l)$
min_sup	Minimum support threshold
ns	A negative sequence
$length(s)$	Length of sequence s, referring to the 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
$PP(ns)$	ns's positive partner
$EidS_s$	Elements id set of sequence s
$OPS(EidS_s)$	Order preserving sequence with $EidS_s$
$MPS(s)$	Maximum positive sub-sequence of s
1-$negMS_{ns}$	1-neg-size maximum subsequence of ns
1-$negMSS_{ns}$	1-neg-size maximum subsequence set of ns
FSE or fse	First subsequence ending position
LSB or lsb	Last subsequence beginning position

usual constraint in association rule mining but is a very strict constraint in sequential pattern mining. It generates frequent itemsets first, then generates frequent and infrequent sequences, and lastly derives NSP from the infrequent sequences. Three extended versions of [26] can be found in [27–29] in which conditions are added to fuzzy, multiple level and multiple minimum supports, respectively. Although the authors of [31] mentioned the question of mining NSP, they did not propose how to mine them. In [20], NSP are mined in the same form as [26] in incremental transaction databases.

Zhao et al. [35] proposed an approach to mining event-oriented negative sequential rules from infrequent sequences in the form of $< A >\Rightarrow< \neg B >$, $< \neg A >\Rightarrow< B >$, $< \neg A >\Rightarrow< \neg B >$. Based on the work in [35], Zhao et al. [36] also presented an approach for discovering both positive and negative impact-oriented sequential rules. Issues about sequence classification using positive and negative patterns were discussed in [25,37]. Positive and negative usage patterns are used in [19] to filter Web recommendation lists. None of these papers involve NSP mining directly.

The above discussions about negative containment and existing NSP research involve the key issue of various *constraints* applied to NSP mining. As detailed in Section 3.2.1, constraints are generally empowered according to the *frequency* of elements, patterns or positive partners, the *format* of continuous negative elements, and the *negation* of elements or items.

Another relevant research topic is negative association rule mining [5,33]. However, as the ordering relationship between items and elements in a sequence is inbuilt in NSP, it is much more challenging to discover NSP than negative associations and patterns. In fact, the ordinal nature of NSP means that algorithms for negative association rule mining and negative pattern mining cannot be directly used to mine NSP.

In summary, the above discussions show that NSP research presents the following strong early-stage characteristics:

- Significant inconsistency in key concepts and settings. In particular, there is no consolidated concept of negative containment in the literature. In Section 4.2, we will present a generic definition of negative containment and formalize the issue.
- Different constraints are incorporated into NSP, leading to varied settings and even divided assumptions about NSP. In Section 3.2.1, we will provide formal and generic definitions of frequency constraint, format constraint, and negative element constraint.
- NSP mining is embedded with specific constraints and requires rescanning databases.
- Existing NSP approaches either re-scan a database as a result of inefficient computational design or do not directly address the problem of NSP mining. In Section 4, e-NSP is introduced which scans a database only once.

Our proposed ST-NSP (see Section 4.1) and e-NSP (more details in Section 4) directly address the above fundamental issues and the substantial complexities of NSP mining by building an innovative, formal, comprehensive and generic design, together with theoretical analysis, and experiment evaluation.

3. Problem statement

In a sequence, a non-occurring item is called a *negative item* and an occurring item is called a *positive item*. Sequences that consist of at least one negative item are called *negative sequences*. The sequences in source data are called *data sequences* [32]. Classic sequential pattern mining handles occurring items only, and generates frequent *positive sequences* [11,24,26,38]. Below, we formally define PSP and NSP. The main symbols used in this paper are listed in Table 1.

3.1. Positive Sequential Patterns – PSP

Let $I = \{x_1, x_2, \ldots, x_n\}$ be a set of *items*. An *itemset* is a subset of I. A *sequence* is an ordered list of itemsets. A sequence s is denoted by $< s_1 s_2 \ldots s_l >$, where $s_j \subseteq I (1 \leqslant j \leqslant l)$. s_j is also called an *element* of the sequence, and denoted as

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

$(x_1x_2 \ldots x_m)$, where x_k is an item, $x_k \in I$ $(1 \leqslant k \leqslant m)$. For simplicity, the brackets are omitted if an element only has one item, i.e., element (x) is coded x. To reduce complexity, we assume that an item occurs at most once in an element, but can appear multiple times in different elements of a sequence.

The *length* of sequence s, denoted as $length(s)$, is the total number of items in all elements in s. s is a k-length sequence if $length(s) = k$. The *size* of sequence s, denoted as $size(s)$, is the total number of elements in s. s is a k-size sequence if $size(s) = k$. For example, a given sequence $s =< (ab)cd >$ is composed of 3 elements (ab), c and d, or 4 items a, b, c and d. Therefore s is a 4-length and 3-size sequence.

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

A *sequence database* D is a set of tuples $< sid, ds >$, where sid is the *sequence_id* and ds is the *data sequence*. The number of tuples in D is denoted as $\mid D \mid$. The set of tuples containing sequence s is denoted as $\{< s >\}$. The support of s, denoted as $sup(s)$, is the number of $\{< s >\}$, i.e., $sup(s) =\mid \{< s >\} \mid=\mid \{< sid, ds >, < sid, ds >\in D \wedge (s \subseteq ds)\} \mid$. *min_sup* is a minimum support threshold predefined by users. Sequence s is called a frequent *(positive) sequential pattern* if $sup(s) \geqslant min\_sup$. By contrast, s is *infrequent* if $sup(s) < min\_sup$.

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

3.2. Negative Sequential Patterns – NSP

3.2.1. Constraints on NSP

In real applications such as health and medical business, the number of NSC and the identified negative sequences are often large, but many of them are not actionable [10]. The number of NSC may be huge or even infinite if no constraints are added. This makes NSP mining very challenging. For example, for dataset with 10 distinct items, the total number of potential itemsets is 1023 $(= C(10, 1) + C(10, 2) + \ldots + C(10, 10))$, where $C(m, n)$ represents the number of combinations created by choosing n items from m distinct items. Assuming that 10 of the 1023 are frequent, if the maximum size of data sequences in DB is 3, the total number of NSC could be $1033^1 + 1033^2 + 1033^3$. Many of the combinations are uninteresting or meaningless in reality.

Various constraints have been added to existing NSP mining, such as those detailed in [11,38,39] and [24], to deal with the fundamental challenges embedded in NSP mining and the current immature situation in NSP study. Although they are not consistent nor generic, these constraints aim to reduce problem complexity and the number of NSC, and to efficiently discover actionable NSP. For example, in health insurance, a common business rule says if a prosthesis *(a)* has been charged for, there should be a charge for a prior corresponding procedure *(b)*. Accordingly, if a customer claims a but does not claim b, which is represented as a NSP: $< \neg ba >$, then the claim behavior could be potentially suspicious. More complicated NSP may be found, which contain more negative items, such as $< a\neg ba\neg c >$ (c is another procedure).

In this work, we incorporate three constraints in e-NSP: *frequency constraint*, *format constraint*, and *negative element constraint*. The reasons for introducing these three constraints are as follows. First, as explained above, NSP mining is at an early stage and is too complicated to conduct without the imposition of specific constraints. Second, we specify constraints on frequency, format and negative elements in order to reduce problem complexity, and in particular, to build a framework of NSP mining on set theory to extract negative sequential patterns from identified positive patterns (frequency constraint), reduce the number of NSC (format constraint), and substantially reduce the complexity in handling partially negative elements (negative element constraint).

Below, we define key concepts for these three constraints for mining NSP, and further explain why each of them is introduced.

Definition 1 *(Positive Partner).* The *positive partner* of a negative element $\neg e$ is e, denoted as $p(\neg e)$, i.e., $p(\neg e) = e$. The positive partner of positive element e is e itself, i.e., $p(e) = e$. The positive partner of a negative sequence $ns =< s_1 \ldots s_k >$ changes all negative elements in ns to their positive partners, denoted as $p(ns)$, i.e., $p(ns) = \{< s'_1 \ldots s'_k > \mid s'_i = p(s_i), s_i \in ns\}$. For example, $p(< \neg(ab)c\neg d >) =< (ab)cd >$.

Constraint 1 *(Frequency constraint).* For simplicity, this paper only focuses on the negative sequences ns whose positive partners are frequent, i.e., $sup(p(ns)) \geq min\_sup$. In contrast, the authors in [11] and [38] only require that the positive partner of each element in ns is frequent.

Although there may be many negative sequences that can be mined from infrequent positive partner sequences (just as many useful negative association rules can be mined from infrequent itemsets [33]), requiring positive partners to be frequent serves multiple purposes: (1) users are often interested in the absence of certain "frequent" (positive) itemsets; the positive partners of those negative itemsets appearing in negative sequential patterns should therefore satisfy a certain frequency. (2) If we do not enforce this constraint, the number of NSC may be huge or even infinite, which would lead to very low efficiency NSP mining.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

A similar constraint has been adopted by other researchers, although it is specified differently for particular purposes. For example, in [11], only the positive partner of each element in negative sequences *ns* is required to be frequent. In our work, we require that the positive partner, not only each element, of *ns* should be frequent, i.e., $sup(p(ns)) \geq min\_sup$. This is because in this work we build a new framework that allows to only negative sequential patterns to be minded from positive sequential patterns by using the set theory to rapidly "calculate" the support of NSC based only on the support of their corresponding PSP without additional database scans.

Constraint 2 *(Format constraint).* Continuous negative elements in a NSC are not allowed.

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

This is similar to the settings in [11,38].

This constraint is introduced for three reasons. (1) If two or more continuous negative elements are allowed, more and more negative sequential candidates can be generated, which would result in a very challenging and sophisticated task. (2) In practice, it would be difficult to ascertain the correct order of two continuous negative elements if there were no positive elements between them. (3) As argued in [11], the order of consecutive negative itemsets is trifling for many applications. As a result, adding this constraint will substantially reduce the number of NSC.

Constraint 3 *(Negative element constraint).* The smallest negative unit in a NSC is an element. If an element consists of more than one item, either all or none of the items are allowed to be negative.

Example 2. $< \neg(ab)cd >$ satisfies Constraint 3, but $< (\neg ab)cd >$ does not because, in element $(\neg ab)$, only $\neg a$ is negative while b is not.

Although negative sequential patterns with negative items (such as in $< (\neg ab)cd >$) may be also interesting in real applications, the complexity could be too high to solve the problem. Introducing this constraint avoids the complexity of handling partially negative elements, and substantially reduces the number of NSC. This is because the number of NSC will increase exponentially if negative items are allowed. For example, given positive pattern $ps =< (ab)(cde)(fghi) >$, the number of NSC is only 4 if this constraint is satisfied; they are $< \neg(ab)(cde)(fghi) >$, $< (ab)\neg(cde)(fghi) >$, $< (ab)(cde)\neg(fghi) >$ and $< \neg(ab)(cde)\neg(fghi) >$. If negative items are allowed, the number of NSC generated from $< (ab)(cde)(fghi) >$ is 1260 $(= 4 * (C(2,1) + C(2,2)) * (C(3,1) + C(3,2) + C(3,3)) * (C(4,1) + C(4,2) + C(4,3) + C(4,4)) = 4 * 3 * 7 * 15)$. It would be very inefficient and complex to generate all NSC. When the size and length of data is huge, it is unrealistic to generate all NSC. Therefore, similar to the settings in [11,38], we also apply this constraint to avoid the complexity of handling partially negative elements.

In summary, the above three constraints introduced into e-NSP not only maintain a certain consistency with existing work, but also enable the efficient generation of NSC based on their positive partners and enable the calculation of NSC support. This leads to a fundamentally new and efficient framework for efficient discovery of NSP in large scale data.

In practice, with the development of more efficient learning frameworks, data structures, and NSP mining algorithms, these constraints may be progressively relaxed. Given the current stage of maturity of NSP mining, we only work on those negative sequences that satisfy the above three constraints in this work.

3.2.2. NSP concepts

According to Constraint 3, the definition of sub-sequences in a positive sequence is not applicable to negative sequences. Below, we define *sub-sequence* and *super-sequence* for negative sequences, in addition to the concepts of *element-id set* and *order-preserving sequence*.

Definition 2 *(Positive/Negative Element-id Set). Element-id* is the order number of an element in a sequence. Given a sequence $s =< s_1 s_2 \dots s_m >$, $id(s_i) = i$ is the element identifier of element s_i. Element-id set $EidS_s$ of s is the set that includes all elements and their ids in s, i.e., $EidS_s = \{(s_i, id(s_i)) \mid s_i \in s = (s_1, 1), (s_2, 2), \dots, (s_m, m)\}$ $(1 \leqslant i \leqslant m)$.

The set including all positive and negative element-ids of a sequence s is called *positive and negative element-id set* of s, denoted as $EidS_s^+$, $EidS_s^-$, respectively. For example, $s =< \neg(ab)c\neg d >$, $EidS_s^+ = \{(c, 2)\}$, $EidS_s^- = \{(\neg(ab), 1), (\neg d, 3)\}$.

Definition 3 *(Order-preserving Sequence).* For any subset $EidS'_s = \{(\alpha_1, id_1), (\alpha_2, id_2), \dots, (\alpha_p, id_p)\}$ $(1 < p \leqslant m)$ of $EidS_s$, $\alpha = < \alpha_1 \alpha_2 \dots \alpha_p >$, if $\forall \alpha_i, \alpha_{i+1} \in \alpha (1 \leqslant i < p)$, there exists $id_i < id_{i+1}$, then α is called an order-preserving sequence of $EidS'_s$, denoted as $\alpha = OPS(EidS'_s)$.

Example 3. Given $s =< \neg(ab)c\neg d >$, its $EidS_s = \{(\neg(ab), 1), (c, 2), (\neg d, 3)\}$, $EidS_s^+ = \{(c, 2)\}$, $EidS_s^- = \{(\neg(ab), 1), (\neg d, 3)\}$. We can obtain $OPS(EidS_s^+) =< c >$. Also, if $EidS'_s = \{(\neg(ab), 1), (c, 2)\}$, we can create a sequence $OPS(EidS'_s) =< \neg(ab)c >$.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

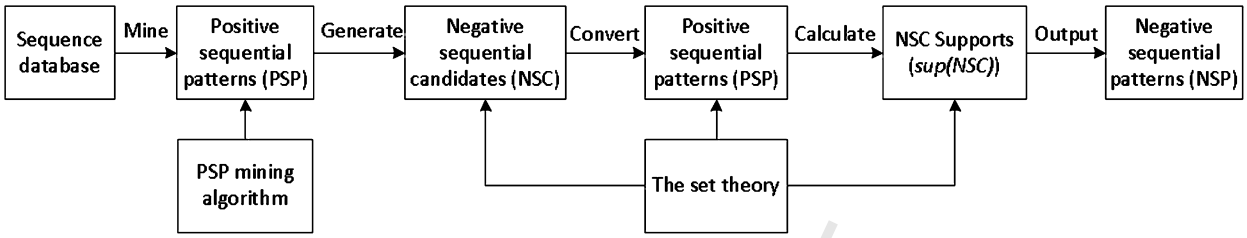


Fig. 1. The set theory-based NSP mining framework: CT-NSP.

Definition 4 *(Sub-sequence and Super-sequence of A Negative Sequence).* Sequence s_α is called a *sub-sequence* of a negative sequence s_β, and s_β is a *super-sequence* of s_α, if $\forall EidS'_{s_\beta}$, $EidS'_{s_\beta}$ is a subset of $EidS_{s_\beta,s_\alpha} = OPS(EidS'_{s_\beta})$, denoted as $s_\alpha \subseteq s_\beta$. If s_α is a negative sequence, it is required to satisfy Constraint 2, which means that there must not be continuous negative elements in s_α.

Example 4. Given $s_\beta =< \neg(ab)cd >$ and $s_\alpha =< \neg(ab)d >$, $EidS_{s_\beta} = \{(\neg(ab), 1), (c, 2), (d, 3)\}$, $EidS'_{s_\beta} = \{(\neg(ab), 1), (d, 3)\}$ is a subset of $EidS_{s_\beta.s_\alpha}$ is a sub-sequence of s_β since $s_\alpha = OPS(EidS'_{s_\beta})$.

Definition 5 *(Maximum Positive Sub-sequence).* Let $ns =< s_1 s_2 \dots s_m >$ be a *m-size* and *n-neg-size* negative sequence $(m - n > 0)$, $OPS(EidS^+_{ns})$ is called the maximum positive sub-sequence of ns, denoted as $MPS(ns)$.

Example 5. Given a negative sequence $s =< \neg(ab)cd >$, $EidS^+ = \{(c, 2), (d, 3)\}$, its maximum positive sub-sequence is $MPS(s) =< cd >$.

Definition 6 *(Negative Sequential Pattern).* A negative sequence s is a negative sequential pattern (NSP) if its support is not less than the threshold *min_sup*.

4. The set theory-based NSP framework and e-NSP algorithm

4.1. The set theory-based NSP mining framework

Here we propose an innovative NSP mining framework, based on set theory. The framework and working mechanism of the proposed set theory-based NSP mining framework (ST-NSP for short) is illustrated in Fig. 1. We also propose an efficient NSP mining algorithm, called e-NSP, to instantiate the ST-NSP framework. The e-NSP algorithm is summarized in Section 4.7, and an example is given in Section 4.8.

A NSP algorithm instantiating the ST-NSP framework consists of several components:

(1) *negative containment* to define how a data sequence contains a negative sequence, which will be discussed in Section 4.2;
(2) a *negative conversion strategy* to convert the negative containment to positive containment, and then use the information of corresponding PSP to calculate the support of a NSC, which will be discussed in Section 4.3;
(3) *NSC support calculation* to calculate the support of NSC, which will be discussed in Section 4.4;
(4) *NSC generation* to generate NSC, which will be discussed in Section 4.5; and
(5) an *e-NSP data structure and optimization strategy* to calculate the union set, which will be discussed in Section 4.6.

Given a sequence database, algorithms like e-NSP built on the ST-NSP framework work on the following process to discover NSP, in which steps (2) to (4) rely on the set theory.

(1) All PSP are mined by a traditional PSP algorithm (with slight changes if necessary) or new PSP mining algorithm;
(2) NSC are generated based on the identified PSP in terms of the three constraints proposed in Section 3.2.1;
(3) The generated NSC are converted to their corresponding PSP in terms of the negative conversion strategy;
(4) The NSC support is calculated based on the corresponding PSP support in terms of negative containment, relevant constraints and the proposed e-NSP data structure and optimization strategy;
(5) Lastly, NSP are identified from the NSC to satisfy certain support criteria.

4.2. Negative containment

As a sub-sequence (e.g., $s_1 =< d >$) may occur more than once in its super-sequence (e.g., $s_2 =< a\ (bc)\ d\ (cde) >$), we need to know the exact positions of s_2 containing s_1 from the left and right sides of s_2. We therefore define the fundamental concept of *negative containment* below.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Definition 7 *(First Sub-sequence Ending Position/Last Sub-sequence Beginning Position).* Given a data sequence $ds =< d_1 d_2 \dots d_t >$ and a positive sequence α,

(1) if $\exists p(1 < p \leqslant t)$, $\alpha \subseteq < d_1 \dots d_p > \wedge \alpha \not\subseteq < d_1 \dots d_{p-1} >$, then p is called the *First Sub-sequence Ending Position*, denoted as $FSE(\alpha, ds)$; if $\alpha \subseteq < d_1 >$ then $FSE(\alpha, ds) = 1$;
(2) if $\exists q(1 \leqslant q < t)$, $\alpha \subseteq < d_q \dots d_t > \wedge \alpha \not\subseteq < d_{q+1} \dots d_t >$, then q is called the *Last Sub-sequence Beginning Position*, denoted as $LSB(\alpha, ds)$; if $\alpha \subseteq < d_t >$ then $LSB(\alpha, ds) = t$;
(3) if $\alpha \not\subseteq ds$, then $FSE(\alpha, ds) = 0$, $LSB(\alpha, ds) = 0$.

Example 6. Given $ds =< a(bc)d(cde) >$. $FSE(< a >, ds) = 1$, $FSE(< c >, ds) = 2$, $FSE(< cd >, ds) = 3$, $LSB(< a >, ds) = 1$, $LSB(< c >, ds) = 4$, $LSB(< cd >, ds) = 2$, $LSB(< (cd) >, ds) = 4$.

Our definition of a data sequence containing a negative sequence is as follows. We use $n - neg - size$ to denote a negative sequence containing n negative elements.

Definition 8 *(Negative Containment).* Let $ds =< d_1 d_2 \dots d_t >$ be a data sequence, $ns =< s_1 s_2 \dots s_m >$ be an $m - size$ and *n-neg-size* negative sequence, (1) if $m > 2t + 1$, then *ds does not contain ns*; (2) if $m = 1$ and $n = 1$, then *ds contains ns* when $p(ns) \not\subseteq ds$; (3) otherwise, *ds contains ns* if, $\forall (s_i, id(s_i)) \in EidS_{ns}^{-}(1 \leqslant i \leqslant m)$, one of the following three cases holds:

(a) $(lsb = 1)$ or $(lsb > 1) \wedge p(s_1) \not\subseteq < d_1 \dots d_{lsb-1} >$, when $i = 1$;
(b) $(fse = t)$ or $(0 < fse < t) \wedge p(s_m) \not\subseteq < d_{fse+1} \dots d_t >$, when $i = m$;
(c) $(fse > 0 \wedge lsb = fse + 1)$ or $(fse > 0 \wedge lsb > fse + 1) \wedge p(s_i) \not\subseteq < d_{fse+1} \dots d_{lsb-1} >$, when $1 < i < m$, where $fse = FSE(MPS(< s_1 s_2 \dots s_{i-1} >), ds)$, $lsb = LSB(MPS(< s_{i+1} \dots s_m >), ds)$.

In the above definition, Case (a) indicates that the first element in *ns* is negative. "$(lsb > 1) \wedge p(s_1) \not\subseteq < d_1 \dots d_{lsb-1} >$" means that $< d_{lsb} \dots d_t >$ contains $MPS(< s_2 \dots s_m >)$ but $< d_1 \dots d_{lsb-1} >$ does not contain $p(s_1)$. "$lsb = 1$" means that the last sub-sequence's beginning position is 1, so $p(s_1)$ cannot be contained by *ds*. Case (b) indicates that the last element in *ns* is negative. Case (c) indicates that the negative element is between the first and last element in *ns*. "$lsb > fse + 1$" ensures there is at least one element in "$< d_{fse+1} \dots d_{lsb-1} >$". "$fse > 0 \wedge lsb = fse + 1$" means that d_{fse} and d_{lsb} are contiguous elements, so $p(s_i)$ cannot be contained within them.

Example 7. Given $ds =< a(bc)d(cde) >$, we have

(1) $ns =< \neg ac >$. $EidS_{ns}^{-} = \{(\neg a, 1)\}$. *ds* does not contain *ns*. $lsb = 4 > 0$, but $p(s_1) =< a > \subseteq < d_1 \dots d_3 > =< a(bc)d >$ (Case (a)).
(2) $ns =< \neg aac >$. $EidS_{ns}^{-} = \{(\neg a, 1)\}$. *ds* contains *ns* because $lsb = 1$ (Case (a)).
(3) $ns =< (ab)\neg(cd) >$. $EidS_{ns}^{-} = \{(\neg(cd), 2)\}$. *ds* does not contain *ns* because $fse = 0$ (Case (b)).
(4) $ns =< (de)\neg(cd) >$. $EidS_{ns}^{-} = \{(\neg(cd), 2)\}$. *ds* contains *ns* because $fse = 4$ $(t = 4)$ (Case (b)).
(5) $ns =< a\neg dd\neg d >$. $EidS_{ns}^{-} = \{(\neg d, 2), (\neg d, 4)\}$. *ds* does not contain *ns*. For $(\neg d, 2)$, $fse = 1$, $lsb = 4$, but $p(\neg d) \subseteq < d_2 \dots d_3 > =< (bc)d >$ (Case (c)). If one negative element does not satisfy the condition, we do not need to consider other negative elements.
(6) $ns =< a\neg bb\neg a(cde) >$. $EidS_{ns}^{-} = \{(\neg b, 2), (\neg a, 4)\}$. *ds* contains *ns*. For $(\neg b, 1)$, $fse = 1$, $lsb = 2$, $fse > 0 \wedge lsb = fse + 1$ (Case (c)); For $(\neg a, 4)$, $fse = 2$, $lsb = 4$, $p(\neg a) \not\subseteq < d_3 > =< d >$ (Case (c)).

Note that negative sequences do not satisfy the Apriori property. As shown in Example 7, $sup(< \neg ac >) = 0$, $sup(< \neg aac >) = 1$, $sup(< \neg ac >) < sup(< \neg aac >)$, though $< ac > \subseteq < \neg aac >$. Accordingly, we cannot simply apply or change the pattern pruning strategies available in PSP mining to NSP mining, and new NSP filtering methods need to be developed.

4.3. Negative conversion

e-NSP is built on the strategy of converting negative containment to positive containment; PSP mining can then be used to mine NSP. For this, we define a special subsequence: 1-neg-size Maximum Sub-sequence.

Definition 9 *(1-neg-size Maximum Sub-sequence).* For a negative sequence *ns*, its sub-sequences that include $MPS(ns)$ and one negative element e are called $1 - neg - size$ maximum sub-sequences, denoted as $1 - negMS = OPS(EidS_{ns}^{+}, e)$, where $e \in EidS_{ns}^{-}$. The sub-sequence set including all 1-neg-size maximum sub-sequences of *ns* is called 1-neg-size maximum sub-sequence set, denoted as $1 - negMSS_{ns}$, $1 - negMSS_{ns} = \{OPS(EidS_{ns}^{+}, e) \mid \forall e \in EidS_{ns}^{-}\}$.

Example 8. 1) $ns =< \neg(ab)c\neg d >$, $1 - negMSS_{ns} = \{< \neg(ab)c >, < c\neg d >\}$; 2) $ns' =< \neg a(bc)d\neg(cde) >$, $1 - negMSS_{ns'} = \{< \neg a(bc)d >, < (bc)d\neg(cde) >\}$.

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

Corollary 1. *Negative Conversion Strategy.*

Given a data sequence $ds =< d_1 d_2 \dots d_t >$, and $ns =< s_1 s_2 \dots s_m >$, which is an *m-size* and *n-neg-size* negative sequence, the negative containment problem can be converted to the following problem: data sequence ds contains negative sequence ns if and only if the two conditions hold: (1) $MPS(ns) \subseteq ds$; and (2) $\forall 1 - negMS \in 1 - negMSS_{ns}$, $p(1 - negMS) \nsubseteq ds$.

Example 9. Given $ds =< a(bc)d(cde) >$, 1) if $ns =< a\neg dd\neg d >$, $1 - negMSS_{ns} = \{< a\neg dd >, < ad\neg d >\}$, then ds does not contain ns because $p(< a\neg dd >) =< add > \subseteq ds$; 2) if $ns' =< a\neg bb\neg a(cde) >$, $1 - negMSS'_{ns} = \{< a\neg bb(cde) >, < ab\neg a(cde) >\}$, then ds contains ns because $MPS(ns) =< ab(cde) > \subseteq ds \wedge p(< a\neg bb(cde) > \nsubseteq ds \wedge p(< ab\neg a(cde) >) \nsubseteq ds$.

Corollary 1 proves that the problem *whether a data sequence contains a negative sequence* can be converted to the problem *whether a data sequence does not contain other positive sequences*. This lays a foundation for calculating the support of negative sequences by using only the information of corresponding positive sequences. Its proof is given below.

Proof of Corollary 1. Here we only prove that Case (c) in the negative containment definition is equivalent to the negative converting strategy, because Cases (a) and (b) can be proved in the same way. In Case (c), condition "$(fse > 0 \wedge lsb = fse + 1)$" indicates that d_{fse} and d_{lsb-1} are contiguous elements, so $p(s_i)$ cannot be contained within them. It is a special case of another condition "$(fse > 0 \wedge lsb > fse + 1) \wedge p(s_i) \nsubseteq < d_{fse+1} \dots d_{lsb-1} >$". Hence, we only need to prove that "$(fse > 0 \wedge lsb > fse + 1) \wedge p(s_i) \nsubseteq < d_{fse+1} \dots d_{lsb-1} >$" is equivalent to negative conversion strategy.

For $(s_i, id(s_i)) \in EidS^{-}_{ns}(1 \leqslant i \leqslant m)$, "$0 < fse$" means $MPS(< s_1 s_2 \dots s_{i-1} >) \subseteq < d_1 \dots d_{fse} >$, and "$0 < lsb$" means $MPS(< s_{i+1} \dots s_m >) \subseteq < d_{lsb} \dots d_t >$. Since $fse < lsb$, $MPS(< s_1 s_2 \dots s_{i-1} s_{i+1} \dots s_m >) \subseteq < d_1 \dots d_{fse} d_{lsb} \dots d_t >$, i.e., $MPS(ns) \subseteq ds$. On the other hand, if $MPS(ns) \subseteq ds$, for $\forall (s_i, id(s_i)) \in EidS^{-}_{ns}$, there must exist $0 < fse < lsb$ s.t. $MPS(< s_1 s_2 \dots s_{i-1} >) \subseteq < d_1 \dots d_{fse} >$ and $MPS(< s_{i+1} \dots s_m >) \subseteq < d_{lsb} \dots d_t >$.

In addition, according to the definition of 1-neg-size maximum sub-sequence, $MPS(< s_1 s_2 \dots s_{i-1} >)$, $MPS(< s_{i+1} \dots s_m >)$ and s_i only construct a 1-neg-size maximum sub-sequence 1-negMS of s_i, so "$(fse > 0 \wedge lsb > fse + 1) \wedge p(s_i) \nsubseteq < d_{fse+1} \dots d_{lsb-1} >$" also means $p(1 - negMS) \nsubseteq ds$. For $\forall (s_i, id(s_i)) \in EidS^{-}_{ns}$, "$(fse > 0 \wedge lsb > fse + 1) \wedge p(s_i) \nsubseteq < d_{fse+1} \dots d_{lsb-1} >$" can be converted to: $\forall 1 - negMS \in 1 - negMSS_{ns}$, $p(1 - negMS) \nsubseteq ds$, and vice versa. □

4.4. Supports of negative sequences

To calculate the support of negative sequences, we first present Corollary 2.

Corollary 2. *Support of negative sequences.*

Given a *m-size* and *n-neg-size* negative sequence ns, for $\forall 1 - negMS_i \in 1 - negMSS_{ns}(1 \leqslant i \leqslant n)$, the support of ns in sequence database D is:

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

This can be easily derived from Corollary 1. Because $\cup_{i=1}^{n} \{p(1 - negMS_i)\} \subseteq \{MPS(ns)\}$, Equation (1) can be rewritten as:

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

Example 10. $sup(< \neg a(bc)d\neg(cde) >) = sup(< (bc)d >) - |\{< a(bc)d >\} \cup \{< (bc)d(cde) >\}|$; $sup(< \neg(ab)c\neg d >) = sup(< c >) - |\{< (ab)c >\} \cup \{< cd >\}|$.

If ns only contains a negative element, the support of ns is:

$$sup(ns) = sup(MPS(ns)) - sup(p(ns)) \quad (3)$$

Example 11. $sup(< (ab)\neg cd >) = sup(< (ab)d >) - sup(< (ab)cd >)$

In particular, for negative sequence $< \neg e >$,

$$sup(< \neg e >) = |D| - sup(< e >) \quad (4)$$

In the following, we illustrate negative containment in terms of set theory. Fig. 2 shows the intersection of sequences $< a >$ and $< b >$. $\{< a >\}$, $\{< b >\}$ mean the set of tuples that respectively contain sequences $< a >$,$< b >$ in a sequence database. There are three 2-length sequences: $< ab >$, $< ba >$ and $< (ab) >$, and four disjointed sets: $\{< (ab) >^{only}\}$, $\{< ab >^{only}\}$, $\{< ba >^{only}\}$ and $\{< ab >\} \cap \{< ba >\}$, which are the sets of tuples that contain sequences $< (ab) >$ only, $< ab >$ only, $< ba >$ only, and both $< ab >$ and $< ba >$ respectively.

UNCORRECTED PROOF

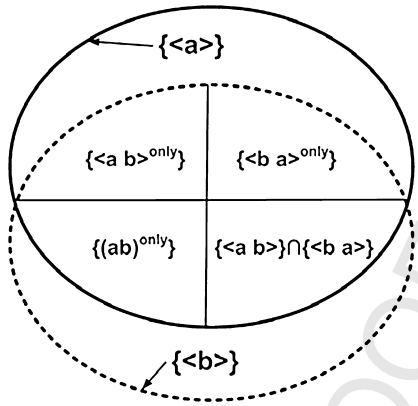


Fig. 2. The intersection of $\{< a >\}$ and $\{< b >\}$.

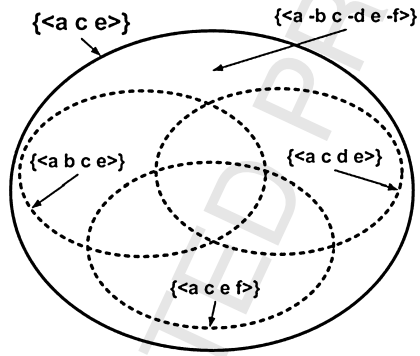


Fig. 3. The interpretation of $sup(< a\neg bc\neg de\neg f >)$ in terms of the set theory.

Taking $\{< a\neg b >\}$ as an example, as seen in Fig. 2, we have:

$$\begin{aligned}
&\{< a\neg b >\}\\
&= (\{< a >\} - \{< b >\}) \cup \{< (ab) >^{only}\} \cup \{< ba >^{only}\}\\
&= \{< a >\} - \{< ab >^{only}\} \cup (\{< ab >\} \cap \{< ba >\})\\
&= \{< a >\} - \{< ab >\}
\end{aligned}$$

This result is consistent with the negative containment definition, by which data sequences containing $\{< a\neg b >\}$ are the same sequences that contain $\{< a >\}$ but do not contain $\{< ab >\}$.

Subsequently, we obtain:

$$sup(< a\neg b >) = sup(< a >) - sup(< ab >).$$

Similarly,

$$sup(< \neg ab >) = sup(< b >) - sup(< ab >);$$

$$sup(< b\neg a >) = sup(< b >) - sup(< ba >);\ \text{and}$$

$$sup(< \neg ba >) = sup(< a >) - sup(< ba >).$$

Fig. 3 illustrates the meaning of $sup(ns)$ in terms of set theory. Given $ns =< a\neg bc\neg def >$, $1 - negMSS_{ns} = \{< a\neg bc\ e >, < ac\neg de >, < ace\neg f >\}$, $sup(ns) = |\{< ace >\}| - |< abce > \cup < acde > \cup < acef >|$.

The above examples show that our proposed negative containment definition is consistent with set theory. Therefore, set properties are applicable for calculating $sup(ns)$. From Equation (2), we can see that $sup(ns)$ can be easily calculated if we know $sup(MPS(ns))$ and $|\cup_{i=1}^{n} \{p(1 - negMS_i)\}|$. According to Constraint 3 and the negative candidate generation approach discussed in Section 4.5, $MPS(ns)$ and $p(1 - negMS_i)$ are frequent. $sup(MPS(ns))$ can be directly obtained by applying traditional PSP mining algorithms.

We further explain why the positive partners $p(1 - negMS_i)$ are frequent. Constraint 3 ensures that the smallest negative unit in a NSC is an element. Hence, the positive partner of each NSC generated from a PSP is the PSP itself. $p(1 - negMS_i)$ is

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

Table 2
e-NSP data structure & an example.

PSP	Support	{*sid*}
$< (ab) >$	2	{20, 30}
$< (ab)c >$	1	{30}
...	...	...

a sub-sequence of the PSP. Both $p(1-negMS_i)$ and the PSP are positive sequences and meet the downward closure property, therefore $p(1-negMS_i)$ are frequent.

Now the problem is how to calculate $|\cup_{i=1}^{n}\{p(1-negMS_i)\}|$. Our approach is as follows. We store the *sid* of the tuples containing $p(1-negMS_i)$ into set $\{p(1-negMS_i)\}$, then calculate the union set of $\{p(1-negMS_i)\}$. Because $p(1-negMS_i)$ are frequent, the *sid* of the tuples containing $p(1-negMS_i)$ can be easily obtained by those well-known algorithms with minor modifications. For instance, we store the *sid* of the tuples containing $p(1-negMS_i)$ to $\{p(1-negMS_i)\}$.

4.5. Negative sequential candidate generation

With the above proposed components, the way to generate the negative sequential candidates (NSC) is to change any non-contiguous elements (note: not items) in a PSP to their corresponding negative elements.

Definition 10 *(e-NSP Candidate Generation).* For a k-size PSP, its NSC are generated by changing any m non-contiguous elements to their negative elements, $m = 1, 2, \ldots, \ulcorner k/2 \urcorner$, where $\ulcorner k/2 \urcorner$ is a minimum integer that is not less than $k/2$.

Example 12. The NSC based on $< (ab)cd >$ include:

$$m = 1,\ < \neg(ab)cd >,\ < (ab)\neg cd >,\ < (ab)\ c\neg d >;$$

$$m = 2, < \neg(ab)c\neg d > .$$

Clearly, we can generate all NSC that satisfy the three constraints for all PSP in a sequence database, as described in Section 3.2.1.

4.6. e-NSP data structure and optimization

To efficiently calculate the union set, we design a data structure to store the e-NSP related data. The data structure is shown in Table 2. Column one stores positive sequential patterns, column two holds their support values, and column three encloses {*sid*} which forms the set of the tuples that contain the corresponding PSP.

The e-NSP data is stored in a hash table to identify PSP efficiently, as shown in the following pseudocode.

Algorithm 1 PSP Hash Table Generation.

```
Input: All PSP and their related information;
Output: PSP hash table;
CreatePSPHashTable(PSP){
  HashTable PSPHash = new HashTable();
  for (each pattern p in PSP)
    PSPHash.put(p.HashCode(),p.support,p.size,p.sidSet);
  return PSPHash;
}
```

To calculate the union set efficiently, we propose two optimization methods.

(1) When we calculate the support of a NSC, we also utilize a hash table to accelerate the search speed. For example, given a NSC $ns =< a\neg bc\neg de\neg f >$, 1-$negMSSns = \{< a\neg bce >, < ac\neg de >, < ace\neg f >\}$, for each sequence 1-$negMS_i$ in 1-*negMSSns*, it is easy to obtain its corresponding positive partner P(1-$negMS_i$). Then we search all {*sid*} of P(1-$negMS_i$) and add each *sid* to the hash table if *sid* is not available in the hash table. Lastly, *ns*'s support can be easily calculated by $sup(MPS(ns))$ minus the size of the new *sid* set, according to Equation (2). Compared with the performance achieved using a common array, our tests show that the search speed with hash table is far more efficient.

(2) We assume that all data in the e-NSP data structure is stored in the main memory. We do not record the {*sid*} of 1-size PSP because the equations do not need to calculate the union set of those {*sid*} of 1-size PSP. Even in the worst situation in which the data is too big to fit completely into the main memory, by using the e-NSP candidate generation method for k-size positive sequential patterns, the size range of negative elements in the corresponding NSC is $1 \rightarrow \ulcorner k/2 \urcorner$. The size range of PSP used to calculate the support of these candidates is $(\llcorner k/2 \lrcorner + 1) \rightarrow (k-1)$, where $\llcorner k/2 \lrcorner$ is a maximum integer that is not larger than $k/2$. Therefore, it is necessary to put only {*sid*} of $(\llcorner k/2 \lrcorner + 1) \rightarrow (k-1)$ size PSP into the main memory.

UNCORRECTED PROOF

Table 3
Example: data set.

Sid	Data sequence
10	$< abc >$
20	$< a(ab) >$
30	$< (ae)(ab)c >$
40	$< aa >$
50	$< d >$

Example 13 *(Continuation of Example 12).* Given a PSP s, $size(s) = 5$, the neg-size range of NSC based on the PSP is $1 \rightarrow 3$, and the size range of PSP used is $3 \rightarrow 4$. When *neg-size(NSC)* $= 1$, the support of NSC can be calculated by Equation (2). When *neg-size(NSC)* $= 2$, we need to calculate the union set of 4-size PSP because $size(1\text{-}negMS_{nsc}) = 4$. When *neg-size(NSC)* $= 3$, we need to calculate the union set of 3-size PSP since $size(1\text{-}negMS_{nsc}) = 3$.

4.7. The e-NSP algorithm

In this section, we introduce an efficient NSP (e-NSP for short) mining algorithm to instantiate the proposed ST-NSP framework. The e-NSP, as described in Algorithm 2, is proposed for mining NSP based only on identifying PSP.

Algorithm 2 e-NSP Algorithm.

```
Input: Sequence dataset D and min_sup;
Output: NSP;
PSP = minePSP();
HashTable PSPHash = CreatePSPHashTable(PSP);
For (each psp in PSP){
  NSC = e-NSP_Candidate_Generation(psp);
  For (each nsc in NSC){
    if (nsc.size == 1 && nsc.neg_size == 1) {
      nsc.support = |D| − PP(nsc).support;
    } else if (nsc.size > 1 && nsc.neg_size == 1){
      nsc.support = MPS(nsc).support − PP(nsc).support;
    } else {
      1-negMSS_nsc = {1-negMS_i | 1 <= i <= nsc.neg_size};
      HashTable cHash = new HashTable();
      For (i = 1; i <= nsc.neg_size; i++) {
        For (each sid in PP(1-negMS_i).sidSet) {
          If (sid.hashcode NOT IN cHash)
            cHash.put(sid.hashcode(), sid);
        }
      }
      nsc.support = MPS(nsc).support − cHash.size();
    }
    If (nsc.support ≥ min_sup)
      NSP.add(nsc);
  }
}
return NSP;
```

e-NSP consists of three key steps: (1) All PSP are identified from the sequence database by using a PSP mining algorithm, such as GSP, PrefixSpan, and SPADE. All PSP and their *sid* sets are saved in a hash table *PSPHash*, in which the PSP hash codes are treated as ID codes. (2) For each PSP, NSC are generated by the approach presented in Section 4.5. (3) The support for each NSC *ns* is calculated by Equations (2)–(4).

4.8. An example

The above sections introduced the key concepts and components as well as the e-NSP algorithm for NSP mining. This section illustrates how these concepts and components are applied for mining NSP. The sequence database is shown in Table 3. Here we set $min\_sup = 2$.

The process is as follows.

(1) Positive sequential patterns are mining by using any existing algorithm, such as GSP, and filling in the e-NSP data structures, which are shown in Table 4.

(2) The e-NSP Candidate Generation method is applied to generate all NSC.

UNCORRECTED PROOF

Table 4
Example: results of positive patterns.

PSP	Support	Size	{*sid*}
$< a >$	4	1	–
$< b >$	3	1	–
$< c >$	2	1	–
$< aa >$	3	2	{20, 30, 40}
$< ab >$	3	2	{10, 20, 30}
$< ac >$	2	2	{10, 30}
$< bc >$	2	2	{10, 30}
$< (ab) >$	2	1	–
$< abc >$	2	3	{10, 30}
$< a(ab) >$	2	2	{20, 30}

Table 5
Example: results of NSC and supports ($min\_sup = 2$).

PSP	NSC	Related PSP	Sup
$< a >$	$< \neg a >$	$< a >$	1
$< b >$	$< \neg \mathbf{b} >$	$< b >$	**2**
$< c >$	$< \neg \mathbf{c} >$	$< c >$	**3**
$< aa >$	$< \neg aa >$	$< a >, < aa >$	1
	$< a \neg a >$	$< a >, < aa >$	1
$< ab >$	$< \neg ab >$	$< b >, < ab >$	0
	$< a \neg b >$	$< a >, < ab >$	1
$< ac >$	$< \neg ac >$	$< c >, < ac >$	0
	$< \mathbf{a} \neg \mathbf{c} >$	$< a >, < ac >$	**2**
$< bc >$	$< \neg bc >$	$< c >, < bc >$	0
	$< b \neg c >$	$< b >, < bc >$	1
$< (ab) >$	$< \neg (\mathbf{ab}) >$	$< (ab) >$	**3**
$< a(ab) >$	$< \neg a(ab) >$	$< (ab) >, < a(ab) >$	0
	$< \mathbf{a} \neg (\mathbf{ab}) >$	$< a >, < a(ab) >$	**2**
$< abc >$	$< \neg abc >$	$< bc >, < abc >$	0
	$< a \neg bc >$	$< ac >, < abc >$	0
	$< ab \neg c >$	$< ab >, < abc >$	1
	$< \neg ab \neg c >$	$< b >, < ab >, < bc >$	0

(3) Equations (2)–(4) are used to calculate the support of these NSC. The results are shown in Table 5, and the resulting NSP are marked in bold.

From this example, we can see that $< ac >$ and $< a \neg c >$, $< a(ab) >$ and $< a \neg (ab) >$ are frequent patterns. In practice, not all of them are useful for business purposes as some of the non-occurring patterns may be misleading. How to select meaningful and actionable patterns has been one of our ongoing tasks [10].

5. Theoretical analysis

In this section, we briefly analyze the performance of e-NSP and compare it with two baseline algorithms PNSP [11] and NegGSP [38]. We select PNSP and NegGSP because they are the only available algorithms comparable to e-NSP. As the three algorithms hold different definitions of negative containment, we adjust PNSP and NegGSP to follow the same definitions and constraints presented in Section 3.

By scrutinizing the three algorithms, we find that the runtime is mainly consumed by the calculation of the support of NSC, $sup(NSC)$. In PNSP and NegGSP, $sup(NSC)$ is obtained by comparing NSC with database sequences, requiring multiple instances of database re-scanning. In e-NSP, $sup(NSC)$ is obtained by calculating the union set of $\{p(1 - negMS_i)\}$, i.e., by comparing the *Sids* stored in hash tables. The number of comparison times determines the algorithm's runtime, and the arithmetic operation time in the algorithms is negligible. Therefore, the runtime analysis of these algorithms is converted to an analysis of their comparison times. As PNSP and NegGSP use a similar means to obtain $sup(NSC)$ and have similar time efficiency [38], we only compare e-NSP with PNSP here for simplicity, although we still compare the three algorithms in the experiments.

5.1. Runtime analysis of e-NSP

In e-NSP, $sup(NSC)$ is calculated by Equations (2)–(4), where Equations (3)–(4) calculate $sup(NSC)$ when NSC only contains a negative element, and Equation (2) calculates $sup(NSC)$ when NSC contains more than one negative element. As mentioned before, the runtime of obtaining $sup(NSC)$ by Equation (2) is mainly consumed by the calculation of the number of elements in the union set of $\{p(1 - negMS_i)\}$, i.e., by comparing the *sids* stored in hash tables. We do not consider the time consumed in processing hash table conflict (which will be discussed later). The comparison times for calculating a $k - size$ and $m -$

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

Table 6
$|NSC_{k,m}|$ for different k and m values.

k	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	…
1	1					
2	2					
3	3	1				
4	4	3				
5	5	6	1			
6	6	10	4			
7	7	15	10	1		
8	8	21	20	5		
9	9	28	35	15	1	
…						

$neg-size\ NSC$, denoted by $CT_{k,m}$, is no more than $\sum_{i=1}^{m}|\{p(1-negMS_i)\}|$, where $|\{p(1-negMS_i)\}|$ is the number of *sids* in $\{p(1-negMS_i)\}$, i.e., the support of $p(1-negMS_i)sup(p(1-negMS_i))$.

$$CT_{k,m}=\sum_{i=1}^{m}|\{p(1-negMSi)\}|=\sum_{i=1}^{m}sup(p(1-negMS_i)) \tag{5}$$

Now we discuss the generalized representation of Equation (5), because the discussions later are all based on the generalized representation and $p(1-negMS_i)$ in Equation (5) is the support of a very detailed set $\{p(1-negMS_i)\}$. We use the average support of $\{p(1-negMS_i)\}$, $\overline{sup}(p(1-negMS_i))$, instead of $sup(p(1-negMS_i))$. $\overline{sup}(p(1-negMS_i))$ indicates the degree of sparsity of the items in a database and it meets the need of generalization. For a $k-size$ and $m-neg-size$ NSC, the size of the PSP of its corresponding $p(1-negMS_i)$ is $k-m+1$. Hence Equation (5) can be rewritten as:

$$\begin{aligned}CT_{k,m}&=\sum_{i=1}^{m}\overline{sup}(p(1-negMS_i))\\&=m*\overline{sup}(p(1-negMS_i))\\&=m*\overline{sup}(PSP_{k-m+1})\end{aligned} \tag{6}$$

Equation (6) is used to calculate the comparison times of a NSC. If we know the total number of NSCs in e-NSP, then we can obtain the total comparison times of e-NSP. According to the definition of e-NSP candidate generation, NSC is generated from PSP. For a given database and a *min_sup*, the total number of its PSPs is a constant. If we know the number of NSCs generated from a PSP, then we can obtain the total number of NSCs easily by a summation operation. Below we calculate the number of NSC generated from a PSP.

5.1.1. The number of NSC generated from a PSP

Let $|NSC_{k,m}|$ denote the number of $m-neg-size$ NSCs generated from a $k-size$ PSP. According to the definition of e-NSP candidate generation and related properties of permutations and combinations, $|NSC_{k,m}|$ can be recognized as the combinations of taking m elements from $(k-m+1)$ elements and can be calculated by Equation (7).

$$|NSC_{k,m}|=C_{k-m+1}^{m}=\frac{(k-m+1)!}{m!*(k-2m+1)!}(1\le m\le\lceil k/2\rceil) \tag{7}$$

Let $|NSC_{k,\forall m}|$ denote the number of NSCs generated from a $k-size$ PSP, then $|NSC_{k,\forall m}|$ can be calculated by Equation (8).

$$|NSC_{k,\forall m}|=\sum_{m=1}^{\lceil k/2\rceil}|NSC_{k,m}| \tag{8}$$

Example 14. Get $|NSC_{k,m}|$ and NSCs of $<abc>$.

The size k of $<abc>$ is 3 and $m=1,2$. Therefore, $|NSC_{3,1}|=3$ and the corresponding NSCs are $<\neg abc>$, $<a\neg bc>$ and $<ab\neg c>$; $|NSC_{3,2}|=1$ and the corresponding NSC is $<\neg ab\neg c>$.

More values of $|NSC_{k,m}|$ at different k and m are shown in Table 6.

5.1.2. The total number of comparison times to calculate all union sets in e-NSP

Let $|PSP_k|$ denote the number of all $k-size$ PSPs in database D and the maximum size of PSPs be k. The number of NSCs generated from all $k-size$ PSPs is $|PSP_k|*|NSC_{k,\forall m}|$. The total number of NSCs generated from all PSPs by e-NSP, $|NSC|^{e-NSP}$, is

UNCORRECTED PROOF

$$|NSC|^{e-NSP} = \sum_{i=1}^{k} |PSP_i| * |NSC_{i,\forall m}| \tag{9}$$

According to the above discussions, the union set operation is only needed when $k \geq 3$ and $m \geq 2$. Hence the comparison times of all union set operations in e-NSP, CT^{e-NSP}, is

$$\begin{aligned} CT^{e-NSP} &= \sum_{i=3}^{k} |PSP_i| * (|NSC_{i,\forall m(m\geq 2)}| * CT_{k,m}) \\ &= \sum_{i=3}^{k} |PSP_i| * (\sum_{m=2}^{\lceil i/2 \rceil} |NSC_{i,m}| * CT_{k,m}) \\ &= \sum_{i=3}^{k} |PSP_i| * (\sum_{m=2}^{\lceil i/2 \rceil} \frac{(i-m+1)!}{m! * (i-2m+1)!} * m * \overline{sup}(PSP_{i-m+1})) \end{aligned} \tag{10}$$

Example 15. Let the maximum size of PSPs in database D be 8, the comparison times of all union set operation in e-NSP are as follows.

$$\begin{aligned} CT^{e-NSP} = {} & |PSP_3| * 1 * 2 * \overline{sup}(PSP_2) \\ & + |PSP_4| * 3 * 2 * \overline{sup}(PSP_3) \\ & + |PSP_5| * [6 * 2 * \overline{sup}(PSP_4) + 1 * 3 * \overline{sup}(PSP_3)] \\ & + |PSP_6| * [10 * 2 * \overline{sup}(PSP_5) + 4 * 3 * \overline{sup}(PSP_4)] \\ & + |PSP_7| * [15 * 2 * \overline{sup}(PSP_6) + 10 * 3 * \overline{sup}(PSP_5) + 1 * 4 * \overline{sup}(PSP_4)] \\ & + |PSP_8| * [21 * 2 * \overline{sup}(PSP_7) + 20 * 3 * \overline{sup}(PSP_5) + 5 * 4 * \overline{sup}(PSP_4)] \end{aligned}$$

5.1.3. Time for processing hash table conflict

The discussion above does not consider the time consumed in processing the hash table conflict. As the time consumed is considerable, it is necessary to consider resolutions for the hash table conflict. Although methods of resolving the hash table conflict in different operating systems and software setting may vary, our analysis shows that this does not affect the analysis. Here we use the secondary re-hashing detection to resolve the hash table conflict. In a hash table, the average search times are $\frac{1}{\alpha}\ln(1-\alpha)$ when a search is successful, and are $\frac{1}{1-\alpha}$ when the search is not successful, where α is the load factor and

$$\alpha = \frac{used \quad space \quad of \quad the \quad hash \quad table}{total \quad space \quad of \quad hash \quad table}. \tag{11}$$

Because $\frac{1}{1-\alpha} > \frac{1}{\alpha}\ln(1-\alpha)$, we use $\frac{1}{1-\alpha}$ as the average search time. Let t^{e-NSP} denote the time of a search in the hash table, then the total time for calculating all the union sets in e-NSP, T^{e-NSP} is calculated as

$$T^{e-NSP} = \frac{1}{1-\alpha} * t^{e-NSP} * CT^{e-NSP} \tag{12}$$

5.2. Runtime analysis of PNSP

Here we illustrate how to analyze the runtime of a NSP algorithm in terms of PNSP, which shares some similarity with ours. PNSP works in three phases. 1) PSPs are mined by GSP algorithms and from these PSPs, all positive itemsets are derived as $1-size$ PSP. 2) $1-size$ NSP are derived from these $1-size$ PSP, where the support of $1-size$ NSP is more than or equal to *min_sup* but less than or equal to a user specified missing frequency threshold *miss_feq*. 3) Both positive and negative itemsets are joined to generate NSC. Generally speaking, $k-size$ NSC are generated by appending a $(k-1)-size$ NSP and PSP with a $1-size$ PSP or a $1-size$ NSP. If the last element of a $(k-1)-size$ NSP is a negative itemset, a $1-size$ PSP is appended to form a NSC; otherwise, $1-size$ PSP or $1-size$ NSP is appended to form two NSCs. The supports of these candidates are then counted by scanning the database. The support of a negative sequence $< \neg e >$ is calculated by Equation (4), as in e-NSP.

Now we analyze the comparison times in PNSP. PNSP calculates the support of a NSC, $sup(NSC)$, by comparing a NSC with every data sequence in database D. The maximum comparison times of a NSC and a data sequence ds is the length of ds. Suppose the average length of data sequence in database D is denoted by $\overline{length_{ds}}$ and the number of data sequences in D is denoted by $|D|$, then the comparison times for obtaining a NSC, denoted by CT, is $\overline{length_{ds}} * |D|$. Suppose the total number of NSCs by PNSP is $|NSC|^{PNSP}$, then the total comparison time for all NSCs, CT^{PNSP}, is

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

$$CT^{PNSP} = |NSC|^{PNSP} * \overline{length_{ds}} * |D| \tag{13}$$

Next we analyze CT^{PNSP}. Because the generation method of NSC in PNSP is different from that in e-NSP, $|NSC|^{PNSP}$ is also different from $|NSC|^{e-NSP}$. Let $|NSC_k^{PNSP}|$ denote the number of all $k-size$ NSCs, $|PNSP_k|$ denote the number of all $k-size$ PSP and $k-size$ NSP in D. According to the generation method of NSC in PNSP,

$$|NSC_k^{PNSP}| = \begin{cases} |PNSP_{k-1}| * |PNSP_1|, \text{ when the last element in } NSP_{k-1} \text{ is positive} \\ |PNSP_{k-1}| * |PSP_1|, \text{ when the last element in } NSP_{k-1} \text{ is negative} \end{cases} \tag{14}$$

Suppose the maximum size of PSP in D is k, then the number of NSCs in D by PNSP is

$$|NSC|^{PNSP} = \sum_{i=1}^{k} |NSC_k^{PNSP}| \tag{15}$$

Accordingly, the total comparison times by PNSP, i.e., CT^{PNSP}, is

$$\begin{aligned} CT^{PNSP} &= |NSC|^{PNSP} * \overline{length_{ds}} * |D| \\ &= \sum_{i=1}^{k} |NSC_k^{PNSP}| * \overline{length_{ds}} * |D| \end{aligned} \tag{16}$$

Clearly, $|NSC|^{PNSP}$ is significantly larger than $|NSC|^{e-NSP}$. It is rather unfair if we directly use Equation (15) to derive all the comparison times of PNSP to compare with Equation (9). This is because the definition of negative containment in PNSP is different from that in e-NSP, as discussed earlier in this section. PNSP cannot generate NSC in the same way as e-NSP because of the constraints of its original definition. Since we change PSNP to follow the same definitions and constraints as in e-NSP, and adopt the same method of generating NSC as in e-NSP, we use Equation (9) instead of Equation (15) as the whole number of NSC in PNSP. Equation (16) is thus rewritten as Equation (17) below.

$$\begin{aligned} CT^{PNSP} &= |NSC|^{e-NSP} * \overline{length_{ds}} * |D| \\ &= \sum_{i=3}^{k} |PSP_i| * (\sum_{m=2}^{\lceil i/2 \rceil} \frac{(i-m+1)!}{m! * (i-2m+1)!} * \overline{length_{ds}} * |D|) \end{aligned} \tag{17}$$

Example 16. Let the maximum size of PSPs in database D be 8, we calculate CT^{PNSP} as:

$$\begin{aligned} CT^{PNSP} = (&|PSP_3| * 1 + |PSP_4| * 3 + |PSP_5| * (6+1) + |PSP_6| * (10+4) + |PSP_7| * (10+5+1) \\ &+ |PSP_8| * (21+20+5)) * \overline{length_{ds}} * |D| \end{aligned}$$

Let t^{PNSP} denote the time an item in NSC is compared with an item in a data sequence, then the total time for calculating the supports of all NSC, i.e., T^{PNSP}, is

$$T^{PNSP} = t^{PNSP} * CT^{PNSP} \tag{18}$$

5.3. Runtime comparison between e-NSP and PNSP against data factors

The runtime ratio between e-NSP and PNSP is the ratio of Equation (12) to Equation (18):

$$\begin{aligned} \frac{T^{e-NSP}}{T^{PNSP}} &= \frac{\frac{1}{1-\alpha} * t^{e-NSP} * CT^{e-NSP}}{t^{PNSP} * CT^{PNSP}} \\ &= \frac{1}{1-\alpha} * \frac{t^{e-NSP}}{t^{PNSP}} * \frac{CT^{e-NSP}}{CT^{PNSP}} \end{aligned} \tag{19}$$

Although t^{e-NSP}, t^{PNSP} and α are different for different experimental environments (hardware and software), $\frac{1}{1-\alpha} * \frac{t^{e-NSP}}{t^{PNSP}}$ is a constant for the same experimental environment. Accordingly, the runtime comparison between e-NSP and PNSP becomes a comparison of the time taken by each to generate NSC.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

5.3.1. Runtime in terms of data factors

We further assess the runtime of e-NSP T^{e-NSP} and PNSP T^{PNSP} in terms of the data factors describing the characteristics of a dataset. First, we define the concept "data factor" below.

Definition 11 *(Data Factor).* A *data factor* describes the characteristic of underlying data from a particular perspective. We specify the following data factors: C, T, S, I, DB and N to describe the characteristics of sequential data [2].

- C: Average number of elements per sequence;
- T: Average number of items per element;
- S: Average length of potentially maximal sequences;
- I: Average size of items per element in potentially maximal large sequences;
- DB: Number of sequences in a database; and
- N: Number of items.

We derive T^{e-NSP} in Equation (10) and T^{PNSP} in Equation (17) in terms of the above data factors. $\overline{length_{ds}}$ can be represented by $C * T$ and the size of sequence k can be represented by S/I. Accordingly, T^{e-NSP} in Equation (10) and T^{PNSP} in Equation (17) are converted to:

$$CT^{e-NSP} = \sum_{i=3}^{S/I} |PSP_i| * (\sum_{m=2}^{\lceil i/2 \rceil} \frac{(i-m+1)!}{m! * (i-2m+1)!} * m * \overline{sup}(PSP_{i-m+1})) \tag{20}$$

$$CT^{PNSP} = \sum_{i=3}^{S/I} |PSP_i| * (\sum_{m=2}^{\lceil i/2 \rceil} \frac{(i-m+1)!}{m! * (i-2m+1)!} * (C * T) * |DB|) \tag{21}$$

We observe the complexity by tuning one data factor while the others are fixed.

5.3.2. Effect of C on runtime

Here we analyze the impact of tuning data factor C on the runtime of e-NSP and PNSP while fixing other factors T, S, I, DB and N. The increase of C directly causes the increase of CT_{PNSP}. For e-NSP, although C does not directly appear in Equation (20), increasing C will result in the increase of m and $\overline{sup}(PSP_{i-m+1})$ to some degree. Hence, CT^{e-NSP} will also increase. Since the increasing speed of $m * \overline{sup}(PSP_{i-m+1})$ is slower than that of C, $CT^{PNSP} - CT^{e-NSP}$ will become greater when C increases.

5.3.3. Effect of T on runtime

Here we analyze the impact of tuning data factor T on the runtime of e-NSP and PNSP while fixing other factors C, S, I, DB and N. The effect of changing T is similar to that of adjusting C. The increase of T directly causes the increase of CT_{PNSP}. For e-NSP, although T does not directly appear in Equation (20), increasing T will result in the increase of m and $\overline{sup}(PSP_{i-m+1})$ to some degree. Hence, CT^{e-NSP} will also increase. Since the increasing speed of $\overline{sup}(PSP_{i-m+1})$ is slower than that of T, $CT^{PNSP} - CT^{e-NSP}$ will become greater when T increases.

5.3.4. Effect of S on runtime

This is to adjust data factor S while fixing others to observe its impact on the runtime. As shown in Equations (20) and (21), it will affect both e-NSP and PNSP in a complicated way since it will affect m. The trend of CT^{PNSP} decrease will be proportionally less than that of CT^{e-NSP}, and thus the overall gap may increase.

5.3.5. Effect of I on runtime

This is to tune the factor I to observe its impact on the runtime. As shown in Equations (20) and (21), it will affect both e-NSP and PNSP. Both will increase, while PNSP will increase proportionally faster than e-NSP when I increases, and the gap will thus also increase.

5.3.6. Effect of DB on runtime

Here we adjust DB to see its impact on the runtime while other factors are fixed. Increasing DB will directly increase CT_{PNSP}, but CT^{e-NSP} has nothing to do with DB. Therefore, the gap $CT^{PNSP} - CT^{e-NSP}$ will become greater when DB increases. We will further analyze scalability in Section 6.4.

5.3.7. Effect of N on runtime

Similarly, we adjust N while fixing all other data factors. Increasing N will decrease $\overline{sup}(PSP_{i-m+1})$ and thus decrease CT^{e-NSP}. However, CT_{PNSP} is not sensitive to N. Therefore, $CT^{PNSP} - CT^{e-NSP}$ will become smaller when N increases.

In summary, e-NSP performs generally more efficiently than PNSP from the various data factors perspective. e-NSP is particularly suitable for datasets with a smaller number of elements in a sequence (indicating that m is small) but a larger

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Table 7
Summary of datasets.

Dataset	Description	Number of sequences	Number of distinct items	Data file size
DS1	Synthetic data, generated by IBM data generator	100K	100	62.3M
DS2	Synthetic data, generated by IBM data generator	10K	200	10.2M
DS3	UCI data, web page visit sequences collected from MSNBC.com	989,818	17	12.3M
DS4	A real application dataset consisting of health insurance claim sequences	5269	around 4K	5.1M
DS5	Chain-Store real-life dataset, from [41]	1,112,949	46,086	45.6M
DS6	A KDD-CUP 2000 dataset containing sequences of click-stream data from e-commerce	59,601	497	0.8M
DS7	Another KDD-CUP 2000 dataset containing sequences of click-stream data	77,512	3340	2.3M
DS8	Sequential data of click streams from the website of FIFA World Cup 98	20,450	17	2.6M
DS9	Synthetic data, generated by IBM data generator which is further restructured into 16 subsets labeled as DS9.1.X, DS9.2.X, DS9.3.X, DS9.4.X ($X = 1, 2, 3$) and DS9.5.Y ($Y = 1, 2, 3, 4$)	10K	100	6.2M

number of itemsets ($|DB|$ is large). In Sections 6.3 and 6.4, the above empirical theoretical analysis from the data factor perspective is further verified by the corresponding experiments.

6. Experiments and evaluation

We conduct substantial experiments on three synthetic datasets and six real-life datasets to compare the efficiency and scalability of e-NSP with two comparable baseline approaches PNSP [11] and NegGSP [38]. Existing positive sequential pattern mining algorithm GSP is used to discover positive patterns first, then e-NSP, PNSP and NegGSP are applied to separately mine for NSP.

All algorithms are implemented in Java. e-NSP is specifically developed on the SPMF framework [42]. All the experiments are run on a virtual machine with 12 CPUs and 128 GB memory on the UTS High Performance Computing Linux Cluster. We compare their computational costs on different data sizes, data characteristics, and scalability. e-NSP is also used to detect fraudulent claims in health insurance data. 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.

6.1. Datasets

In total, nine source datasets are used for the experiments. They comprise six real datasets and three synthetic datasets generated by IBM data generator [2]. To describe and observe the impact of data characteristics on algorithm performance, we use the concept of data factors defined in the above section to summarize the data characteristics.

- *Dataset 1 (DS1)*, C8_T4_S6_I6_DB100k_N100.
- *Dataset 2 (DS2)*, C10_T8_S20_I10_DB10k_N0.2k.
- *Dataset 3 (DS3)* is from UCI and consists of MSNBC.com anonymous web data about web page visits. Visits were recorded at the page category and in a temporal order.
- *Dataset 4 (DS4)* is a real application dataset of health insurance claim sequences. The dataset contains 5269 customers/sequences. The average number of elements in a sequence is 21. The minimum number of elements in a sequence is 1, and the maximum number is 144. The file size is around 5M.
- *Dataset 5 (DS5)* is a Chain-Store real-life dataset containing 46,086 distinct items and 1,112,949 transactions [41], and is frequently used for testing utility-based sequential pattern mining.
- *Dataset 6 (DS6)* is a KDD-CUP 2000 dataset which contains 59,601 sequences of e-commerce clickstreams. It contains 497 distinct items. The average length of each sequence is 2.42 items with a standard deviation of 3.22. The dataset contains a number of long sequences. For example, 318 sequences contain more than 20 items.
- *Dataset 7 (DS7)* is another dataset used in the KDD-CUP 2000 competition. It contains 77,512 sequences of click-stream data with 3340 distinct items. The average length of a sequence is 4.62 items with a standard deviation of 6.07 items.
- *Dataset 8 (DS8)* is a dataset of 20,450 sequences of click stream data from the website of FIFA World Cup 98. It has 2990 distinct items (webpages). The average sequence length is 34.74 items with a standard deviation of 24.08 items. This dataset was created by processing a section of the World Cup web log.
- *Dataset 9 (DS9)* (C8_T4_S6_I6_DB10k_N100), was generated to test how different factors affect the performance of e-NSP, PNSP and NegGSP. The DS9 dataset was extended to 16 additional sub-datasets, labeled as $DS9.1.X$, $DS9.2.X$, $DS9.3.X$, $DS9.4.X$ ($X = 1, 2, 3$) and $DS9.5.Y$ ($Y = 1, \ldots, 4$), to embed different data distributions in terms of data factors.

Table 7 summarizes the characteristics of all the above datasets.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

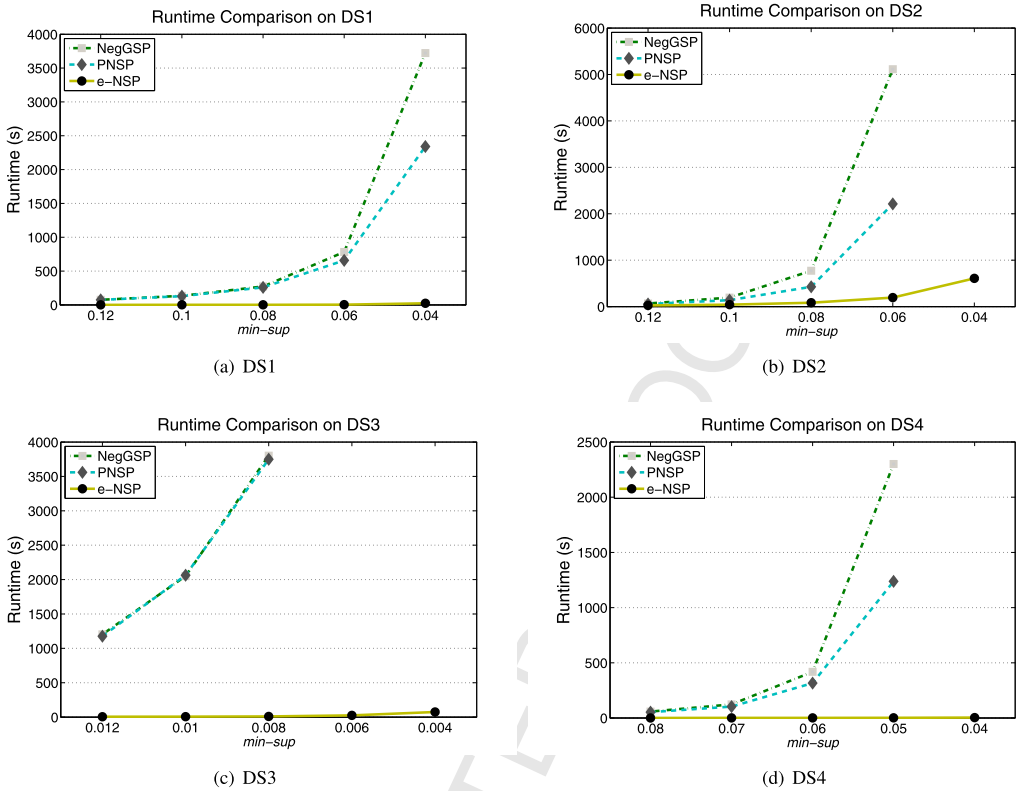


Fig. 4. Runtime on datasets DS1–DS4.

6.2. Computational cost

The execution time of mining NSP by the three algorithms is shown in Figs. 4 and 5. e-NSP always takes much less time than PNSP and NegGSP on all datasets. When the minimum support is set very low, e-NSP only uses a small percentage of the execution time of the other two algorithms. For example, e-NSP spends 0.5% to 0.3% of PNSP runtime on *DS*3 when *min_sup* decreases from 0.012 to 0.008. When *min_sup* reduces to 0.008, PNSP and NegGSP take around one hour, but e-NSP takes only 10 seconds. This is because e-NSP only needs to "calculate" the NSP supports based on the *sid* sets of corresponding positive patterns, while PNSP and NegGSP have to re-scan the whole dataset. In general, e-NSP only takes 0.1–2% of the runtime of PNSP or NegGSP on all datasets DS1 to DS8.

While all three algorithms suffer from data complexities, as shown in Fig. 4(b), (c) and (d) and in Fig. 5(a) and (b), PNSP and NegGSP are sometimes unable to generate outcomes when the minimum support is very low, whereas e-NSP works well. The trend observation shows that irrespective of whether or not the data is complex, e-NSP works in a near linear way, while both PNSP and NegGSP suffer substantially from the exponential increase of computational time.

When we scrutinize the performance dynamics of e-NSP and its baselines on eight datasets, it is challenging to draw a clear conclusion as to which data factors most affect performance. For this reason, we analyze the performance in terms of data characteristics in the section that follows.

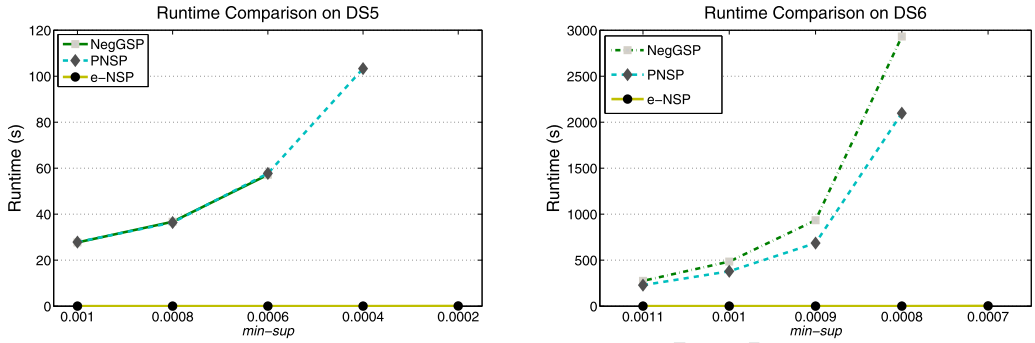
We also show the maximum length and number of negative patterns on the eight datasets (see the outcomes in Figs. 6 and 7). When the minimum support is substantially reduced, more patterns are exponentially generated. e-NSP is robust to the maximum length of negative patterns, since the runtime for e-NSP increases more slowly than the runtime for the other two algorithms when the maximum length increases.

6.3. Analysis of data characteristics

In this section, we explore the impact of data characteristics in terms of the data factors specified in Section 5.3 on the performance of e-NSP, compared to PNSP and NegGSP, as well as the sensitivity of e-NSP on particular data factors.

We generate various types of synthetic datasets with different distributions to evaluate the impact of data factors in dataset *DS9* on algorithm performance. *DS9* is extended to 16 different subsets by tuning each factor, as shown in Tables 8 and 9. For example, dataset *DS9.1.1*(C4T4S6I6.DB10k.N100) is different from *DS9*(C8T4S6I6.DB10k.N100) on *C* factor, which means each dataset has a different average number of elements in a sequence. We mark the differentiators by underlining the respective distinct factor for each dataset in Tables 8 and 9.

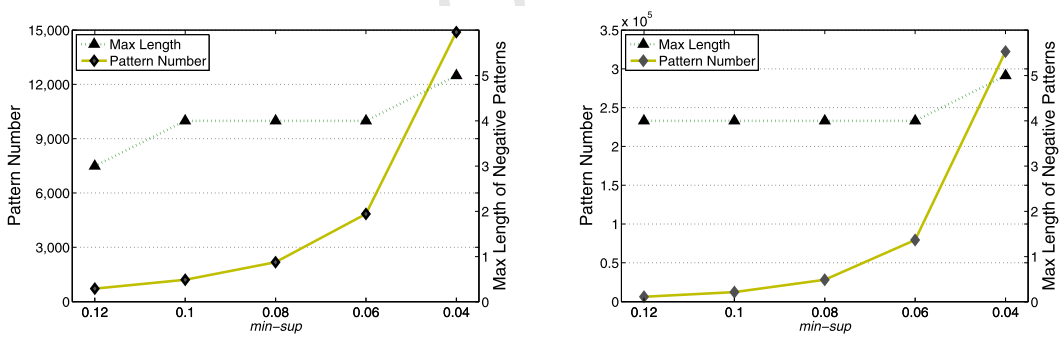
Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001



(a) DS5 (b) DS6

(c) DS7 (d) DS8

Fig. 5. Runtime on datasets DS5–DS8.



(a) DS1 (b) DS2

(c) DS3 (d) DS4

Fig. 6. Maximum length and number of negative patterns on the datasets DS1–DS4.

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

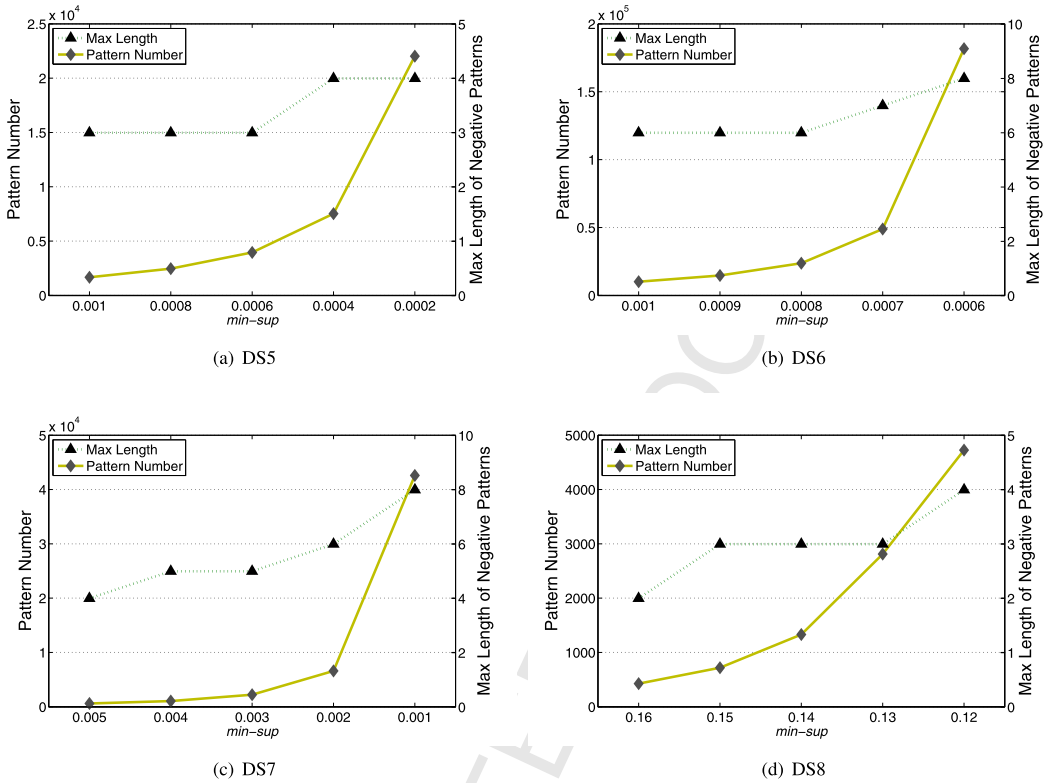


Fig. 7. Maximum length and number of negative patterns on the datasets DS5–DS8.

In Tables 8 and 9, $t_{NegGSP}(s)$, $t_{PNSP}(s)$ and $t_{e-NSP}(s)$ represent the runtime of NegGSP, PNSP and e-NSP in terms of seconds (s) respectively. Overall, e-NSP is much more efficient than NegGSP and PNSP under different combinations of data factors and with various minimum supports.

We use t_{e-NSP}/t_{PNSP} to show the proportion of e-NSP's performance compared to that of PNSP. It shows that e-NSP only spends about 0.5% to 3.6% of t_{PNSP} on all combinations of different data factors, in terms of factors C, T and S and I and N. In addition, with various minimum supports, e-NSP always shows outstanding performance compared to the other two algorithms. This observation is consistent with our theoretical analysis in Section 5.3. In most cases, e-NSP demonstrates better performance when *min_sup* gets lower, and the value of t_{e-NSP}/t_{PNSP} decreases slightly.

According to the results shown in Tables 8 and 9 and Figs. 8, 9 and 10, factors C, T, I and N seriously affect the performance of the three algorithms. There is not much change in runtime when S increases from 2 to 8, which is slightly different from our theoretical analysis. Note that in the synthetic data generator, factor S is constrained and affected by factors C, I and T, so purely changing S would not affect the real data distribution. It is reasonable to suppose that factor S does not greatly affect algorithm performance.

As discussed in Section 5.3, the impact of data factors on computational performance of NSP mining is very complicated. Figs. 8, 9 and 10 compare the different data combinations of e-NSP, PNSP and NegGSP in terms of various minimum supports. The figures show that e-NSP substantially beats PNSP on all datasets for any minimum support by spending only 0.5% to 3.6% of PNSP runtime. In particular, when minimum support decreases, the advantage of e-NSP remains relatively consistent. This shows the strong potential of e-NSP against different data characteristics.

6.4. Scalability test

e-NSP calculates supports based on the *sid* sets of corresponding positive patterns, thus its performance is sensitive to the size of *sid* sets. If a dataset is huge, it produces large *sid* sets. The scalability test is conducted to evaluate the e-NSP performance on large datasets. Fig. 11 shows the results of e-NSP on datasets DS6 and DS8, in terms of different data sizes: from 10 (i.e., 8M) to 50 (40M and 2,980,050 sequences) times of DS6, and from 5 (13M) to 25 (65M and 511,250 sequences) times of DS8, with various low minimum supports *min_sup* 0.0007, 0.0008, 0.0009 and 0.001 on DS6, and 0.12, 0.14, 0.16 and 0.18 on DS8, respectively.

On DS6, for example, when the sampled data size increases to 50 times its original size (see the results corresponding to label 'X50') and $min\_sup = 0.001$, e-NSP takes 140 seconds to obtain the results. This is around seven times of the runtime

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Table 8
Runtime against various data characteristics C, T and S.

Data factor	Dataset ID	*min_sup*	$t_{e-NSP}(s)$	$t_{NegGSP}(s)$	$t_{PNSP}(s)$	t_{e-NSP}/t_{PNSP}
$C = 4$	**DS9.1.1**=C4T4S6I6.DB10k.N100	0.06	0.04	1.23	1.16	3.6%
		0.04	0.07	3.52	2.99	2.2%
		0.02	0.14	24.2	16.1	0.9%
$C = 6$	**DS9.1.2**=C6T4S6I6.DB10k.N100	0.06	0.17	13.1	10.5	1.6%
		0.04	0.33	54.2	33.8	1.0%
		0.02	1.5	1226.0	304.0	0.5%
$C = 8$	**DS9**=C8T4S6I6.DB10k.N100	0.06	0.91	143.0	81.3	1.1%
		0.04	2.3	1042.0	309.0	0.8%
		0.02	13.8	N/A	3044.0	0.5%
$C = 10$	**DS9.1.3**=C10T4S6I6.DB10k.N100	0.06	4.7	1718.0	494.9	0.95%
		0.04	14.8	N/A	2122.0	0.7%
		0.02	101.0	N/A	N/A	0.0%
$T = 4$	**DS9**=C8T4S6I6.DB10k.N100	0.20	0.07	2.8	2.1	3.5%
		0.18	0.08	3.9	3.4	2.5%
		0.16	0.1	5.1	4.9	2.0%
$T = 6$	**DS9.2.1**=C8T6S6I6.DB10k.N100	0.20	1.3	71.9	50.9	2.5%
		0.18	1.4	218.0	116.0	1.2%
		0.16	1.5	228.0	225.0	0.7%
$T = 8$	**DS9.2.2**=C8T8S6I6.DB10k.N100	0.20	1.5	141.0	101.0	1.5%
		0.18	2.5	256.0	162.0	1.5%
		0.16	3.9	524.0	283.0	1.4%
$T = 10$	**DS9.2.3**=C8T10S6I6.DB10k.N100	0.20	11.2	2107.0	783.0	1.4%
		0.18	18.6	5358.0	1402.0	1.3%
		0.16	54.3	N/A	2630.0	2.1%
$S = 2$	**DS9.3.1**=C8T4S2I6.DB10k.N100	0.06	0.9	120.0	80.0	1.1%
		0.04	2.4	770.0	310.0	0.8%
		0.02	13.2	N/A	3005.0	0.4%
$S = 4$	**DS9.3.2**=C8T4S4I6.DB10k.N100	0.06	0.8	117.0	71.0	1.1%
		0.04	2.1	755.0	279.0	0.8%
		0.02	12.1	N/A	2676.0	0.5%
$S = 6$	**DS9**=C8T4S6I6.DB10k.N100	0.06	0.9	143.0	81.0	1.1%
		0.04	2.4	1042.0	309.0	0.8%
		0.02	13.9	N/A	3044.0	0.5%
$S = 8$	**DS9.3.3**=C8T4S8I6.DB10k.N100	0.06	0.9	158.0	82.0	1.1%
		0.04	2.5	1275.0	327.0	0.8%
		0.02	15.3	N/A	3628.0	0.4%

on the 10 times (X10) data size. This indicates that the increase of five times the data size leads to about seven times runtime growth.

Both results on DS6 and DS8 in Fig. 11 show that the growth of runtime of e-NSP on large scale data follows a roughly linear relationship with the data size increase on different minimum supports. The results in this scalability test show that e-NSP works particularly well on very large datasets.

6.5. Experimental summary

In summary, the above substantial experiments result in the following observations on e-NSP:

- e-NSP is highly efficient, performing tens to hundreds of times faster than the two baseline algorithms, and is applicable for mining NSP from very large scale data.
- e-NSP works extremely well for such scenarios as having a small number of elements in a sequence, a small number of items in an element, and a large number of itemsets. The length of patterns and the average number of items in an element of patterns are not sensitive to e-NSP.
- The scalability test shows that the runtime of e-NSP has a near linear relationship with the number of data sequences. It shows that e-NSP performs well on large scale datasets.
- The advantage of e-NSP remains well on both very low and high minimum supports, while the baselines work better on high minimum support and sometimes they cannot produce outcomes on low minimum supports.

UNCORRECTED PROOF

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Table 9
Runtime against various data characteristics I and N.

Data factor	Dataset ID	*min_sup*	$t_{e-NSP}(s)$	$t_{NegGSP}(s)$	$t_{PNSP}(s)$	t_{e-NSP}/t_{PNSP}
$I = 4$	**DS9.4.1**=C8T4S6I4.DB10k.N100	0.08	0.4	35.3	29.5	1.4%
		0.06	0.8	114.0	71.4	1.1%
		0.04	2.2	740.0	259.0	0.9%
$I = 6$	**DS9**=C8T4S6I6.DB10k.N100	0.08	0.4	40.8	30.3	1.4%
		0.06	0.9	143.0	81.4	1.1%
		0.04	2.4	1042	309.0	0.8%
$I = 8$	**DS9.4.2**=C8T4S6I8.DB10k.N100	0.08	0.5	62.9	44.1	1.2%
		0.06	1.2	260.0	121.0	1.0%
		0.04	3.4	2089.0	492.8	0.7%
$I = 10$	**DS9.4.3**=C8T4S6I10.DB10k.N100	0.08	0.7	116.0	68.9	1.1%
		0.06	1.6	516.0	199.0	0.8%
		0.04	4.6	N/A	934.0	0.5%
$N = 200$	**DS9.5.1**=C8T4S6I6.DB10k.N200	0.020	0.7	834.0	140.0	0.5%
		0.018	0.9	1197.0	225.0	0.4%
		0.016	1.1	N/A	257.0	0.4%
$N = 300$	**DS9.5.2**=C8T4S6I6.DB10k.N300	0.020	0.2	204.0	35.5	0.6%
		0.018	0.3	316.0	41.5	0.7%
		0.016	0.3	583.0	56.0	0.5%
$N = 400$	**DS9.5.3**=C8T4S6I6.DB10k.N400	0.020	0.1	60.6	15.5	0.8%
		0.018	0.2	101.0	20.9	0.7%
		0.016	0.2	181.0	29.6	0.6%
$N = 500$	**DS9.5.4**=C8T4S6I6.DB10k.N500	0.020	0.08	53.1	6.9	1.1%
		0.018	0.09	65.3	9.3	1.0%
		0.016	0.12	86.4	14.2	0.9%

The above experiment observations are supported by the theoretical design of e-NSP, and the results are consistent with the theoretical analysis.

- The proposed set theory-based NSP mining framework, namely ST-NSP framework, lays a solid theoretical foundation for the outstanding performance of e-NSP. The ST-NSP framework presents an innovative and efficient learning framework for handling not only NSP mining but also the general non-occurring behavior analytics problems [8] in large scale data and applications.
- The proposed constraints on frequency, format and negative elements make the ST-NSP framework and its instantiated algorithm e-NSP workable. The e-NSP data structure and optimization strategy also contribute to the performance of e-NSP.
- The theoretical analysis of the computational complexity of e-NSP and other NSP algorithms in terms of data factors provides solid argument for the soundness of e-NSP. The data factors-based analysis provides a valuable path for understanding why e-NSP works better and when e-NSP works better in terms of data characteristics and their matching to the theoretical design.

6.6. Case study: fraudulent claim detection

In the health insurance industry, medical treatments often follow certain rules. The non-occurrence of some medical service codes in claim transactions may indicate problems (such as fraud) in service procedures, protheses or specific diseases. For certain patients, for example, the medical service code a should always occur after another code b in their claim history. If a customer claims a but has not made a prior claim for b, which is represented as a NSP $< \neg ba >$, the claim might be treated as potentially suspicious. NSP mining can be applied here to identify non-occurring services in health insurance claims.

We applied e-NSP for more than one non-occurring medical service code (that is, more than one negative item) in insurance claims to detect claim fraud. We assume a fraud may exist if a customer's medical claim sequence s satisfies the following conditions:

- $s =< i_1 i_2 \dots i_n >$, i_x $(1 \leqslant x \leqslant n)$ is a positive item and represents one medical service code;
- s' is m-neg-size negative sequence, and the positive parter of s' is $p(s') = s$;
  For example, when $m = 1$, $s' =< i_1 \dots \neg i_x \dots i_n >$ $(1 \leqslant x \leqslant n)$; when $m = 2$, $s' =< i_1 \dots \neg i_x i_{x+1} \dots i_{y-1} \neg i_y \dots i_n >$ $(1 \leqslant x < y \leqslant n)$;
- $sup(s')/sup(s) < min\_ratio$ or $sup(s)/sup(s') < min\_ratio$ (i.e. $min\_ratio = 0.02$)

since medical codes that should occur together do not appear together in the claims.

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

UNCORRECTED PROOF

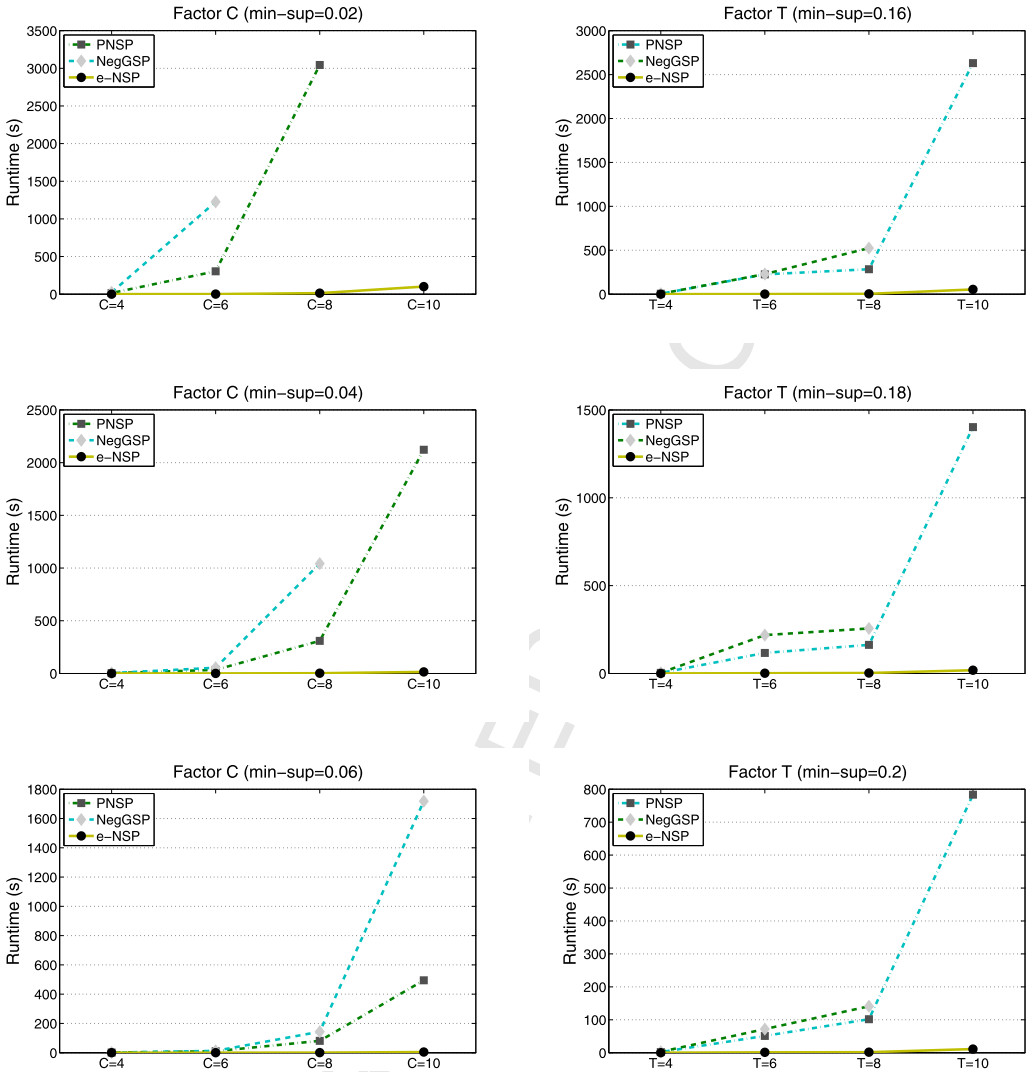


Fig. 8. Runtime comparison on data factors C and T.

We identified the above negative claim patterns, which have been converted into business rules for health insurance fraud detection. For example, if a patient claimed item CMBS [1] code 45530 for a rectus abdominis flap, item 45569 should also be claimed for the closure of the abdomen and reconstruction of the umbilicus. CMBS code 45530 represents breast reconstruction using a latissimus dorsi or other large muscle or myocutaneous flap, including repair of secondary skin defects; 45569 represents closure of the abdomen with reconstruction of the umbilicus, with or without lipectomy. That is to say, 45569 should be claimed with 45530; if the patient claim history shows $< 45530 \neg 45569 >$ then it is suspicious and should be reviewed. Similar examples can be identified in healthcare data by e-NSP to detect abnormal services that are unlikely to occur together in medical treatments.

7. Discussions

As the incorporation of various constraints into NSP mining shows, the study of NSP is at a very early stage. There are many problems that cannot be addressed by existing NSP methods. In fact, non-occurring behavior analytics (NBA) is a critical new area as discussed in [8]. We especially highlight the following important aspects in NBA, particularly in relation to NSP mining.

- ST-NSP framework: e-NSP successfully demonstrates the effectiveness of the ST-NSP framework, which requires more theoretical analysis to make it more effective for NSP mining with relaxed constraints, and more efficient for data with different data characteristics and complexities.

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

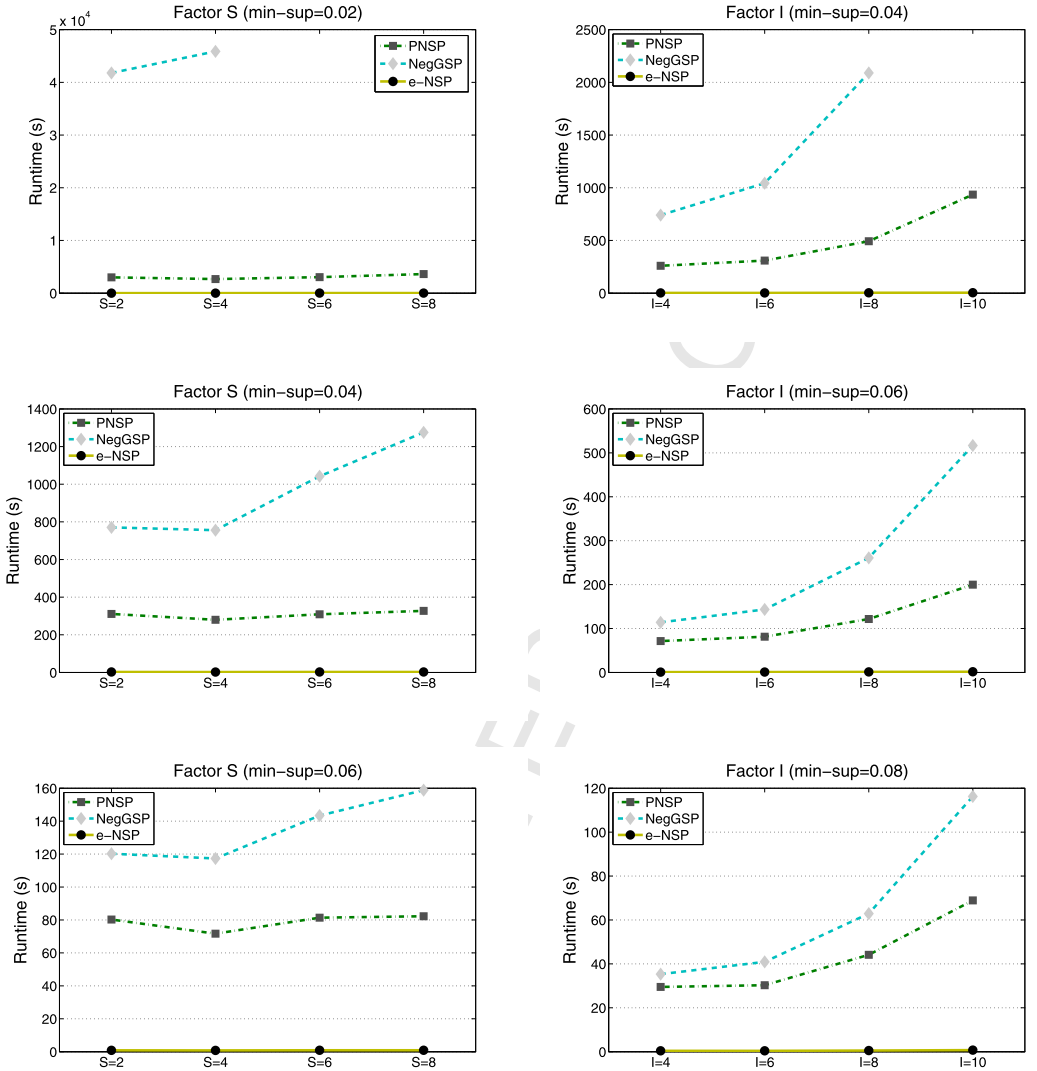


Fig. 9. Runtime comparison on data factors S and I.

- Constraints: The format constraint restricts those NSP with continuous negative elements, e.g., $< \neg a \neg bc >$, to be identified by existing NSP algorithms. The negative element constraint filters those patterns mixing non-occurring and occurring items in one element. Innovative and efficient frameworks and algorithms need to be developed in the future.
- Scalability: While e-NSP demonstrates great potential in discovering NSP in large scale data, the number of NSC increases exponentially when the length and size of a positive pattern increase. New design is needed to reduce the number of NSC while releasing the constraint on negative elements.
- Data characteristics: Limited research outcomes can be found in the literature concerning the theoretical analysis of the impact of data characteristics on learning performance and the incorporation into technical design in terms of data factors. As shown in e-NSP, this serves as the driving force for designing an effective and efficient learning framework and algorithms.

8. Conclusions and future work

As a critical tool for understanding complex non-occurring behaviors, NSP presents interesting information about NOB dynamics and patterns. mining NSP is very challenging due to the complexities surrounding non-occurring items, high computational cost, and the large search space of negative candidates. However, NSP discovery is becoming increasingly important for many intelligent systems and applications, as traditional PSP and association rule mining approaches cannot effectively detect such patterns and exceptions that are associated with non-occurring sequences. In the very limited research outcomes reported on NSP mining, existing techniques rely heavily on re-scanning databases after identifying pos-

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

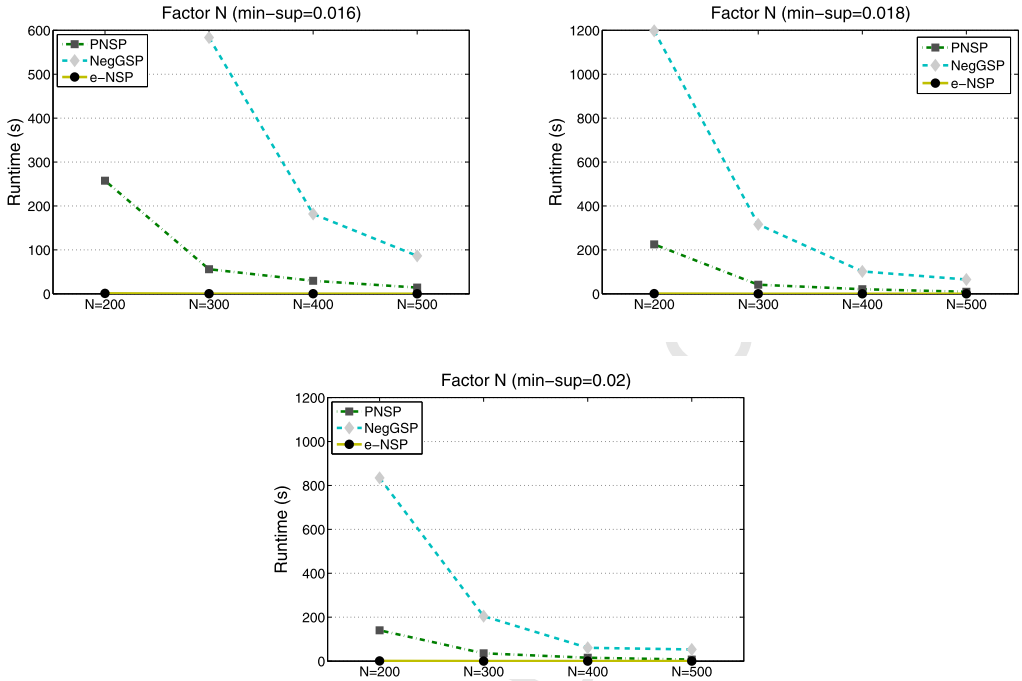
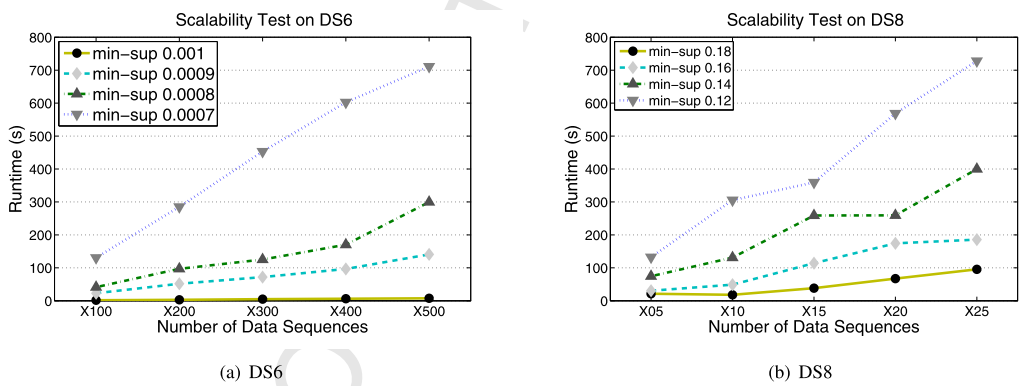


Fig. 10. Runtime comparison on data factors *N*.



(a) DS6 (b) DS8

Fig. 11. Scalability test on data factor *DB* on datasets DS6 and DS8.

itive patterns and incorporate specific designs and definitions for respective purposes. This has been shown to be highly inefficient and inapplicable for real-life large data.

In this paper, we have proposed an innovative framework, set theory-based NSP mining, and an instantiated algorithm e-NSP, which are built on a solid theoretical foundation including a comprehensive design of constraints, negative containment, negative conversion, NSC generation, and NSC support calculation. Both theoretical and experimental analyses have been provided for e-NSP on computational complexities, data characteristics in terms of data factors, and scalability on nine distinct datasets, compared with two typical benchmark NSP mining algorithms. Experimental results are consistent with the theoretical analyses, concluding that e-NSP is much more efficient than existing approaches. e-NSP has been shown to perform particularly well on datasets with a small number of elements in a sequence, a large number of itemsets and under low minimum supports. e-NSP offers a new strategy for efficiently mining large scale NSP.

We are currently working on effective approaches to select the most meaningful patterns from PSP and NSP, and on more effective data structures that can store less data but can support the easy calculation of union sets for efficient NSP mining on large scale data. Further efforts will be made in the exploration of the critical challenges discussed in Section 7, including data characteristics, constraints and scalability.

Please cite this article in press as: L. Cao et al., e-NSP: Efficient negative sequential pattern mining, Artif. Intell. (2016), http://dx.doi.org/10.1016/j.artint.2016.03.001

Acknowledgement

This work has been jointly supported by Australian Research Council Grants (DP130102691), National Natural Science Foundation of China (No. 71271125), and Shandong Provincial Natural Science Foundation (No. ZR2011FM028). We particularly thank Dr. Zhenjiang Lin and Mr. Yongshun Gong for their help in the experiments and discussions.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.artint.2016.03.001.

References

[1] http://www9.health.gov.au/mbs.
[2] R. Agrawal, R. Srikant, Mining sequential patterns, in: ICDE'95, 1995, pp. 3–14.
[3] A. Algarni, N. Zhong, Mining positive and negative patterns for relevance feature discovery, in: KDD'2010, 2010, pp. 753–762.
[4] J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, in: KDD'02, 2002, pp. 429–435.
[5] A. Bilal, A. Erhan, An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules, Soft Comput. 10 (3) (2005) 230–237.
[6] L. Cao, In-depth behavior understanding and use, Inf. Sci. 180 (17) (2010) 3067–3085.
[7] L. Cao, P. Yu (Eds.), Behavior Computing: Modeling, Analysis, Mining and Decision, Springer, 2012.
[8] L. Cao, P.S. Yu, V. Kumar, Nonoccurring behavior analytics: a new area, IEEE Intell. Syst. 30 (6) (2015) 4–11.
[9] L. Cao, Y. Ou, P.S. Yu, Coupled behavior analysis with applications, IEEE Trans. Knowl. Data Eng. 24 (8) (2012) 1378–1392.
[10] L. Cao, P. Yu, Y. Zhao, C. Zhang, Domain Driven Data Mining, Springer, 2010.
[11] S.C. Hsueh, M.-Y. Lin, C.-L. Chen, Mining negative sequential patterns for e-commerce recommendations, in: APSCC'08, IEEE, 2008, pp. 1213–1218.
[12] X. Dong, Z. Zheng, L. Cao, Y. Zhao, C. Zhang, J. Li, W. Wei, Y. Ou, e-NSP: efficient negative sequential pattern mining based on identified positive patterns without database rescanning, in: CIKM'2011, ACM, 2011, pp. 825–830.
[13] X. Dong, F. Sun, X. Han, R. Hou, Study of positive and negative association rules based on multi-confidence and chi-squared test, in: ADMA06, in: Springer LNCS, vol. 4093, Springer-Verlag, Berlin-Heidelberg, 2006, pp. 100–109.
[14] X. Dong, Y. Gong, L. Zhao, Comparisons of typical algorithms in negative sequential pattern mining, in: IEEE Workshop on Electronics, Computer and Applications 2014, 2014, pp. 387–390.
[15] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.C. Hsu, Freespan: frequent pattern-projected sequential pattern mining, in: KDD'00, 2000, pp. 355–359.
[16] Y. Gong, C. Liu, X. Dong, Research on typical algorithms in negative sequential pattern mining, Open Automat. Control Syst. J. 7 (2015) 934–941.
[17] S. Kamepalli, R. Kurra, Frequent negative sequential patterns: a survey, Int. J. Comput. Eng. Technol. 5 (3) (2014) 15–121.
[18] P. Kazienko, Mining sequential patterns with negative conclusions, in: DaWaK'2008, 2008, pp. 423–432.
[19] P. Kazienko, Filtering of web recommendation lists using positive and negative usage patterns, in: KES'2007, Springer, 2007, pp. 1016–1023.
[20] V.K. Khare, V. Rastogi, Mining positive and negative sequential pattern in incremental transaction databases, Int. J. Comput. Appl. 71 (1) (2013) 18–22.
[21] N.P. Lin, H.J. Chen, W.H. Hao, Mining negative sequential patterns, in: WSEAS'2007, 2007, pp. 654–658.
[22] N.P. Lin, H.J. Chen, W.H. Hao, Mining negative fuzzy sequential patterns, in: Proceedings of the 7th WSEAS International Conference on Simulation, Modeling and Optimization, 2007, pp. 52–57.
[23] N.P. Lin, H.J. Chen, W.H. Hao, Mining strong positive and negative sequential patterns, WSEAS Trans. Comput. 3 (7) (2008) 119–124.
[24] N.P. Lin, W.H. Hao, H.J. Chen, C.I. Chang, H.E. Chueh, An algorithm for mining strong negative fuzzy sequential patterns, Int. J. Comput. 3 (1) (2007) 167–172.
[25] S. Mesbah, F. Taghiyareh, A new sequential classification to assist Ad auction agent in making decisions, in: 5th International Symposium on Telecommunications 2010, 2010, pp. 1006–1012.
[26] W.M. Ouyang, Q.H. Huang, Mining negative sequential patterns in transaction databases, in: ICMLC'2007, 2007, pp. 830–834.
[27] W.M. Ouyang, Q.H. Huang, S. Luo, Mining positive and negative fuzzy sequential patterns in large transaction databases, in: FSKD'2008, 2008, pp. 18–23.
[28] W.M. Ouyang, Q.H. Huang, Mining positive and negative fuzzy multiple level sequential patterns in large transaction databases, in: GCIS'2009, 2009, pp. 500–504.
[29] W.M. Ouyang, Q.H. Huang, Mining positive and negative sequential patterns with multiple minimum supports in large transaction databases, in: GCIS'2010, 2010, pp. 190–193.
[30] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M.C. Hsu, Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth, in: ICDE'2001, 2001, pp. 215–224.
[31] V. Rastogi, V.K. Khare, Apriori based mining positive and negative frequent sequential patterns, Int. J. Latest Trends Eng. Technol. 1 (3) (2012) 24–33.
[32] R. Srikant, R. Agrawal, Mining sequential patterns: generalizations and performance improvements, in: EDBT'1996, vol. 1057, 1996, pp. 1–17.
[33] X. Wu, C. Zhang, S. Zhang, Efficient mining of both positive and negative association rules, ACM Trans. Inf. Syst. 22 (2004) 381–405.
[34] M.J. Zaki, Spade: an efficient algorithm for mining frequent sequences, Mach. Learn. 42 (2001) 31–60.
[35] Y. Zhao, H. Zhang, L. Cao, C. Zhang, H. Bohlscheid, Efficient mining of event-oriented negative sequential rules, in: WI-IAT'2008, vol. 1, IEEE/WIC/ACM, 2008, pp. 336–342.
[36] Y. Zhao, H. Zhang, L. Cao, C. Zhang, H. Bohlscheid, Mining both positive and negative impact-oriented sequential rules from transactional data, in: PAKDD'09, in: LNCS, vol. 5476, 2009, pp. 656–663.
[37] Y. Zhao, H.F. Zhang, S.S. Wu, J. Pei, L.B. Cao, C.Q. Zhang, H. Bohlscheid, Debt detection in social security by sequence classification using both positive and negative patterns, in: ECML/PKDD'2009, in: LNAI, vol. 5782, Part 2, 2009, pp. 648–663.
[38] Z. Zheng, Y. Zhao, Z. Zuo, L. Cao, Negative-GSP: an efficient method for mining negative sequential patterns, in: Data Mining and Analytics (AusDM'09), vol. 101, 2009, pp. 63–67.
[39] Z. Zheng, Y. Zhao, Z. Zuo, L. Cao, An efficient GA-based algorithm for mining negative sequential patterns, in: PAKDD'10, in: LNCS, vol. 6118, 2010, pp. 262–273.
[40] C. Liu, X. Dong, C. Li, Y. Li, SAPNSP select actionable positive and negative sequential patterns based on a contribution metric, in: International Conference on Fuzzy Systems and Knowledge Discovery (FSKD2015), 2015, pp. 847–851.
[41] J. Pisharath, Y. Liu, B. Ozisikyilmaz, R. Narayanan, W.K. Liao, A. Choudhary, G. Memik, NU-MineBench Version 2.0 Data Set and Technical Report, http://cucis.ece.northwestern.edu/projects/DMS/MineBenchDownload.html.
[42] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu, V.S. Tseng, SPMF: a Java open-source pattern mining library, J. Mach. Learn. Res. 15 (2014) 3389–3393.
[43] Z. Zheng, Negative sequential pattern mining, PhD Thesis, 2011.

UNCORRECTED PROOF

XMLVIEW: extended

Appendix A. Supplementary material

The following is the Supplementary material related to this article.

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 1
caption: This is file of Fig. 4.a.
link: **APPLICATION : mmc1**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 2
caption: This is file of Fig. 4.b.
link: **APPLICATION : mmc2**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 3
caption: This is file of Fig. 4.c.
link: **APPLICATION : mmc3**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 4
caption: This is file of Fig. 4.d.
link: **APPLICATION : mmc4**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 5
caption: This is file of Fig. 5.a.
link: **APPLICATION : mmc5**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 6
caption: This is file of Fig. 5.b.
link: **APPLICATION : mmc6**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 7
caption: This is file of Fig. 5.c.
link: **APPLICATION : mmc7**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 8
caption: This is file of Fig. 5.d.
link: **APPLICATION : mmc8**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 9
caption: This is file of Fig. 6.a.
link: **APPLICATION : mmc9**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 10
caption: This is file of Fig. 6.b.
link: **APPLICATION : mmc10**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 11
caption: This is file of Fig. 6.c.
link: **APPLICATION : mmc11**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

UNCORRECTED PROOF

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 12
caption: This is file of Fig. 6.d.
link: **APPLICATION : mmc12**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 13
caption: This is file of Fig. 7.a.
link: **APPLICATION : mmc13**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 14
caption: This is file of Fig. 7.b.
link: **APPLICATION : mmc14**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 15
caption: This is file of Fig. 7.c.
link: **APPLICATION : mmc15**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 16
caption: This is file of Fig. 7.d.
link: **APPLICATION : mmc16**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 17
caption: This is file of Fig. 8.factor c-1.
link: **APPLICATION : mmc17**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 18
caption: This is file of Fig. 8.factor c-2.
link: **APPLICATION : mmc18**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 19
caption: This is file of Fig. 8.factor c-3.
link: **APPLICATION : mmc19**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 20
caption: This is file of Fig. 8.factor t-1.
link: **APPLICATION : mmc20**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 21
caption: This is file of Fig. 8.factor t-2.
link: **APPLICATION : mmc21**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 22
caption: This is file of Fig. 8.factor t-3.
link: **APPLICATION : mmc22**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 23
caption: This is file of Fig. 9.factor i-1.
link: **APPLICATION : mmc23**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

UNCORRECTED PROOF

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 24
caption: This is file of Fig. 9.factor i-2.
link: **APPLICATION : mmc24**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 25
caption: This is file of Fig. 9.factor i-3.
link: **APPLICATION : mmc25**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 26
caption: This is file of Fig. 9.factor s-1.
link: **APPLICATION : mmc26**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 27
caption: This is file of Fig. 9.factor s-2.
link: **APPLICATION : mmc27**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 28
caption: This is file of Fig. 9.factor s-3.
link: **APPLICATION : mmc28**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 29
caption: This is file of Fig. 10.factor n-1.
link: **APPLICATION : mmc29**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 30
caption: This is file of Fig. 10.factor n-2.
link: **APPLICATION : mmc30**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 31
caption: This is file of Fig. 10.factor n-3.
link: **APPLICATION : mmc31**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 32
caption: This is file of Fig. 11.a.
link: **APPLICATION : mmc32**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent begin ecomponent

Label: Application 33
caption: This is file of Fig. 11.b.
link: **APPLICATION : mmc33**

end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent end ecomponent

UNCORRECTED PROOF

Sponsor names

Do not correct this page. Please mark corrections to sponsor names and grant numbers in the main text.

Australian Research Council, *country*=Australia, *grants*=DP130102691

National Natural Science Foundation of China, *country*=China, *grants*=71271125

UNCORRECTED PROOF