NegPSpan: efficient extraction of negative sequential patterns with embedding constraints

Thomas Guyet, René Quiniou

Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

NegPSpan: efficient extraction of negative sequential patterns with embedding constraints

Thomas Guyet · René Quiniou

Received: date / Accepted: date

Abstract Mining frequent sequential patterns consists in extracting recurrent behaviors, modeled as subsequences, in a big sequence dataset. Such patterns inform about which events are frequently observed in sequences, *i.e.* events that really happen. Sometimes, knowing that some specific event does not happen is more informative than extracting observed events. Negative sequential patterns (NSPs) capture recurrent behaviors by patterns having the form of sequences mentioning both observed events and the absence of events. Few approaches have been proposed to mine such NSPs. In addition, the syntax and semantics of NSPs differ in the different methods which makes it difficult to compare them. This article provides a unified framework for the formulation of the syntax and the semantics of NSPs. Then, we introduce a new algorithm, NegPSpan, that extracts NSPs using a PrefixSpan depth-first scheme, enabling *maxgap* constraints that other approaches do not take into account. The formal framework highlights the differences between the proposed approach and the methods from the literature, especially with the state of the art approach eNSP. Intensive experiments on synthetic and real datasets show that NegPSpan can extract meaningful NSPs and that it can process bigger datasets than eNSP thanks to significantly lower memory requirements and better computation times.

Keywords Sequential patterns mining · pattern semantics · absence modeling · negative containment

T. Guyet
Agrocampus Ouest/IRISA-UMR6074
65 rue de Saint Brieuc, 35042 Rennes, France
E-mail: thomas.guyet@irisa.fr

R. Quiniou
Inria, Univ Rennes, CNRS, IRISA

1 Introduction

In many application domains such as diagnosis or marketing, decision makers show a strong interest for rules that associate specific events (a context) to undesirable events to which they are correlated or that are frequently triggered in such a context. In the pattern mining field, such rules are extracted from dataset and formalized by patterns. Patterns are substructures that appear in a structured dataset. The patterns can refer to different structural forms: itemsets, subsequences, subgraphs. For example, a sequential pattern is a subsequence, such as buying first *milk*, then *bread*, and then *chocolate*. This subsequence of purchases is interesting when it occurs in a shopping history database (the structure dataset). Sequential pattern mining algorithms can extract such hidden rules from execution traces or transactions. In the classical setting, sequential patterns contain only positive events, *i.e.* really observed events. However, the absence of a specific action or event can often better explain the occurrence of an undesirable situation (Cao et al., 2015). For example in diagnosis, if some maintenance operations have not been performed, *e.g.* damaged parts have not been replaced, then a fault will likely occur in a short delay while if these operations would have been performed in time the fault would not occur. In marketing, if some market-place customer has not received special offers or coupons for a long time then she/he has a high probability of churning while if she/he would have been provided such special offers she/he should remain loyal to her/his market-place. In these two cases, mining specific events, some present and some absent, to discover under which context some undesirable situation occurs or not, may provide interesting so-called *actionable* information for determining which action should be performed to avoid the undesirable situation, *i.e.* fault in diagnosis, churn in marketing.

We aim at discovering sequential patterns that take into account the absence of some events called *negative events* (Cao et al., 2015). Moreover, we want to take into account some aspects of the temporal dimension as well, maximal span of pattern occurrences or maximal gap between the occurrences of pattern events. For example, suppose that from a sequence dataset, we want to mine a sequential pattern $\langle a\ b\rangle$ with the additional *negative* constraint telling that the event c should not appear between events a and b (the scope of the constraint). The corresponding negative pattern is represented as $\langle a\ \neg c\ b\rangle$, where the logical sign $\neg$ indicates the absence of the event or the set of events in its scope. Once the general idea of introducing negative events in a pattern has been stated, the syntax and semantics of such negative patterns should be clearly formulated since they have a strong impact both on algorithms outcome and their computational efficiency. As we will see, the few algorithms from literature do not use the same syntactical constraints and rely on very different semantical principles (see Section 7). More precisely, the efficiency of eNSP (Cao et al., 2016), the state-of-the-art algorithm for NSP mining, comes from a semantic for negation that enables efficient operations on the sets of supported sequences. The two major computational limits of eNSP are memory requirements and the impossibility for eNSP to handle embedding constraints such as the classical *maxgap* and *maxspan* constraints. *Maxgap* and *maxspan* constraints enforce the events of a pattern occurrence to not be too distant in time. Intuitively, if events are far from each others, they are not related/ Thus, they

might not match the pattern. In addition, such constraints can prune efficiently the exploration of occurrences during the sequence dataset scan.

The two main contributions of this article are as follows:

- a clarification of the syntactic definition of negative sequential patterns and different negation semantics with their associated properties.
- a complete and correct algorithm called NegPSpan, inspired by algorithm PrefixSpan, to extract negative sequential patterns with *maxgap* and *maxspan* constraints.

Intensive experiments compare, on synthetic and real datasets, the performance of NegPSpan and eNSP as well as the pattern sets extracted by each of them. As results, they show that algorithm NegPSpan is more time-efficient than eNSP for mining long sequences thanks to the maxgap constraint and that its memory requirement is several orders of magnitude lower, enabling to process much larger datasets. In addition, they highlight that eNSP misses interesting patterns on real datasets due to semantical restrictions.

2 Frequent sequential pattern mining

This section introduces frequent sequential pattern mining, its basic concepts and main results. This data mining task was introduced at the early ages of the pattern mining field (Srikant and Agrawal, 1996).

Let $(\mathcal{I}, <)$ be the finite set of items (alphabet) associated with a total order (*e.g.* lexicographic order). In the sequel, $[n] = \{1, \cdots, n\}$ denotes the set of the first n strictly positive integers.

Definition 1 (Sequence) An *itemset* $A = \{a_1\ a_2\ \cdots\ a_m\} \subseteq \mathcal{I}$ is a set of items. $|A|$ denotes the size of the itemset A. A *sequence* $\boldsymbol{s}$ is a finite set of sequentially ordered itemsets $\boldsymbol{s} = \langle s_1\ s_2\ \cdots\ s_n\rangle$. This means that s_i appears before s_j in sequence $\boldsymbol{s}$ for all $i, j \in [n],\ i < j$. This sequence starts by s_1 and finishes by s_n. n is the *length* of the sequence, denotes $|\boldsymbol{s}|$. The *size* of the sequence is the total number of items it contains.

Definition 2 (Subsequence and embedding) Let $\boldsymbol{s} = \langle s_1\ s_2\ \cdots\ s_n\rangle$ and $\boldsymbol{s'} = \langle s'_1\ s'_2\ \cdots s'_m\rangle$ be two sequences, $\boldsymbol{s'}$ is a *subsequence* of $\boldsymbol{s}$, denoted $\boldsymbol{s'} \preceq \boldsymbol{s}$, iff there exists an increasing sequence of m indexes $e_i \in [n]$ such that $e_i < e_{i+1}$ for all $i \in [m-1]$ and $s'_i \subseteq s_{e_i}$ for all $i \in [m]$. $(e_i)_{i\in[m]} \in [n]^m$ is called an *embedding* of $\boldsymbol{s'}$ in $\boldsymbol{s}$.

Definition 3 (Sequential pattern) A *sequential pattern* $\boldsymbol{p} = \langle p_1\ p_2\ \cdots\ p_m\rangle$ is a sequence over the set of items $\mathcal{I}$. Let $\boldsymbol{s}$ be a sequence and $\boldsymbol{p}$ be a sequential pattern. Sequence $\boldsymbol{s}$ *supports* pattern $\boldsymbol{p}$ (or $\boldsymbol{p}$ *occurs* in sequence $\boldsymbol{s}$) iff $\boldsymbol{p}$ is a subsequence of $\boldsymbol{s}$, *i.e.* $\boldsymbol{p} \preceq \boldsymbol{s}$.

Example 1 (Sequence and sequential patterns) In this example, we illustrate the different notions of frequent sequential pattern mining. We assume that $\mathcal{I} = \{a, b, c, d,$

$e, f\}$ and we consider the following dataset of six sequences:

$$\mathcal{D} = \left\{ \begin{array}{l} \boldsymbol{s_1} = \langle a\ (ef)\ b\ b\ (ce)\rangle \\ \boldsymbol{s_2} = \langle e\ a\ (cd)\ b\ d\ b\ (ce)\ (de)\rangle \\ \boldsymbol{s_3} = \langle a\ d\ b\ (ce)\ f\rangle \\ \boldsymbol{s_4} = \langle b\ (ce)\ a\ d\ f\ b\rangle \\ \boldsymbol{s_5} = \langle (acd)\ c\ b\ e\ b\ (ce)\ d\rangle \\ \boldsymbol{s_6} = \langle c\ e\ b\ d\rangle \end{array} \right\}.$$

The length of sequence $\boldsymbol{s_1}$ is 5. (ef) and (ce) denote itemsets containing 2 items occurring at the same time in the sequence.

Let $\boldsymbol{p} = \langle a\ b\ b\ (ce)\rangle$ be a sequential pattern, $\boldsymbol{p}$ occurs in sequences $\boldsymbol{s_1}$, $\boldsymbol{s_2}$, $\boldsymbol{s_5}$. The embedding of $\boldsymbol{p}$ in $\boldsymbol{s_5}$ is $(1, 3, 5, 6)$. Definition 2 requires that each itemset of the pattern is a subset of some sequence itemset and respects the pattern and the sequence orderings. In addition, according to the definition of a subsequence (Definition 2), successive itemsets of a sequential pattern do not need to occur in a row in the sequence. Some items or itemsets can appear between the sequence occurrences of two successive itemsets of the sequential pattern. Indeed, a, the first item of $\boldsymbol{p}$, is a subset of (acd) in $\boldsymbol{s_5}$. Also, item c appears between the occurrences of a and b in $\boldsymbol{s_5}$, and item e appears between the occurrences of the second and third itemsets of $\boldsymbol{p}$. But, $\boldsymbol{p}$ does not occur in $\boldsymbol{s_6}$ because $\boldsymbol{s_6}$ does not contain any item a. $\boldsymbol{p}$ does not occur in $\boldsymbol{s_4}$ because the items of the pattern $\boldsymbol{p}$ are not in the correct order. $\boldsymbol{p}$ does not occur in $\boldsymbol{s_3}$ because item b must occur twice (the embedding must strictly increase).

Let us now consider the problem of mining frequent sequential patterns from a dataset of sequences, denoted $\mathcal{D}$. The main idea behind mining frequent patterns assumes that the more frequent a pattern the more interesting it is. In practice, a pattern is said to be frequent when it occurs more frequently than a user-defined threshold σ. The following definitions formalize these intuitive definitions.

Definition 4 (Sequential pattern support) Let $\boldsymbol{p}$ be a sequential pattern and $\mathcal{D} = \{\boldsymbol{s_i}\}$ a dataset of sequences where $\boldsymbol{s_i}$ is the i-th sequence of $\mathcal{D}$.

The *support* of $\boldsymbol{p}$ in $\mathcal{D}$, denoted $supp(\boldsymbol{p})$, is the number of sequences that support $\boldsymbol{p}$:

$$supp(\boldsymbol{p}) = |\{\boldsymbol{s_i} \in \mathcal{D} \mid \boldsymbol{p} \preceq \boldsymbol{s_i}\}|\,.$$

The support of some sequential pattern $\boldsymbol{p}$ is the number of sequences from the dataset in which $\boldsymbol{p}$ occurs. It is worth noting that the support is the number of sequences that contains a pattern and not the number of embeddings of the pattern in the sequences. If the pattern occurs several times in a sequence, this sequence counts only for 1 in the support.

Definition 5 (Frequent sequential pattern mining) Let $\mathcal{D}$ be a dataset of sequences and σ be a threshold, *mining frequent sequential patterns* consists in extracting the complete list of patterns having a support greater than a given threshold σ:

$$\{\boldsymbol{p} \mid supp(\boldsymbol{p}) \geq \sigma\}.$$

A naive approach to solve the problem of mining frequent sequential pattern mining consists in enumerating all potential sequential patterns and to evaluate their support. But, this approach is practically intractable on large datasets due to the large size of the set of sequential patterns. The algorithmic strategies that have been developed in the field of sequential pattern mining are based on the anti-monotonicity property of the support measure.

Proposition 1 (Support anti-monotonicity (Srikant and Agrawal, 1996)) *Let $\mathcal{D}$ a dataset of sequences and $\boldsymbol{p}$, $\boldsymbol{p'}$ two sequential patterns, the anti-monotonicity states that:*

$$\boldsymbol{p} \preceq \boldsymbol{p'} \implies supp(\boldsymbol{p}) \geq supp(\boldsymbol{p'})$$

The anti-monotonicity property of the support with respect to pattern inclusion states that if a pattern $\boldsymbol{p'}$ is a super-pattern of $\boldsymbol{p}$ (in other words, that $\boldsymbol{p}$ is a subsequence of $\boldsymbol{p'}$), then the support of $\boldsymbol{p'}$ is lower than the support of $\boldsymbol{p'}$. The intuition behind this property, is that each time the super-pattern $\boldsymbol{p'}$ occurs in a sequence of $\mathcal{D}$, then any of its sub-pattern $\boldsymbol{p}$ also occurs in the same sequence, thus the support of $\boldsymbol{p}$ is at least equal to or greater than the support of $\boldsymbol{p'}$.

Example 2 (Anti-monotonicity of the support) Continuing Example 1, let $\boldsymbol{p'} = \langle a\ b\ b\ (ce)\ d \rangle$ be a super-pattern of $\boldsymbol{p}$, $\boldsymbol{p} \preceq \boldsymbol{p'}$. Indeed, $\boldsymbol{p'}$ occurs in sequence $\boldsymbol{s_2}$ and $\boldsymbol{s_5}$, but not in $\boldsymbol{s_1}$. $supp(\boldsymbol{p}) = 3$ whereas $supp(\boldsymbol{p'}) = 2$ and so $supp(\boldsymbol{p'}) \leq supp(\boldsymbol{p})$. It is useless to look at other sequences, if $\boldsymbol{p}$ does not occur in some sequence, $\boldsymbol{p'}$ does not as well.

It follows from Proposition 1 that if $\boldsymbol{p}$ and $\boldsymbol{p'}$ are two sequential patterns such that $\boldsymbol{p} \preceq \boldsymbol{p'}$ then if $\boldsymbol{p'}$ is frequent ($supp(\boldsymbol{p'}) \geq \sigma$), then $\boldsymbol{p}$ is also frequent ($supp(\boldsymbol{p}) \geq \sigma$). Conversely, if $\boldsymbol{p}$ is not frequent, then $\boldsymbol{p'}$ is not frequent.

The anti-monotonicity property of the support is used by all mining algorithms to make the exploration of the pattern space efficient. Intuitively, each time the algorithm encounters a *not frequent* pattern $\boldsymbol{p}$, it avoids the exploration of the super-patterns of $\boldsymbol{p}$. Decades of research have proposed many algorithmic strategies to exploit this anti-monotonicity property and numerous algorithms have been proposed. They can be classified in three main approaches:

- bread-first search: all potential frequent patterns of size $n + 1$ are generated from frequent patterns of size n and then evaluated. GSP (Srikant and Agrawal, 1996) uses this strategy.
- depth-first search: a frequent pattern is extended until being not frequent, and then alternative extensions are explored. FreeSpan (Han et al., 2000) and PrefixSpan (Pei et al., 2004) uses this strategy.
- vertical dataset: this strategy uses a vertical representation of the dataset of sequences to explore the search space using equivalent classes of patterns. This strategy has been introduced by SPADE (Zaki, 2001).

Several variations of the problem of sequential pattern mining have been proposed. We refer the reader wishing to have a broad and deep view of this specific field to a survey of the literature, such as Mooney and Roddick (2013).

Sequential pattern mining has many applications but sometimes a simple sequence of items is not precise enough to capture the behaviors that are looked for. In addition, under the general setting, sequential pattern mining may yield too many patterns for low support thresholds or trivial patterns for high thresholds. To address these limitations, the user would like to enrich the pattern syntax to obtain the information she/he is looking for.

Some of these variations consist in integrating constraints in sequential pattern mining (Pei et al., 2007), e.g. constraints on the size of pattern embeddings (*maxspan*) or on the number of sequence itemsets that may be skipped between the occurrences of two successive pattern itemsets (*maxgap*). The constraints that preserve the anti-monotonicity property of the support reduce the number of patterns and thus decrease the computation times. Some other variations consist in using alternative domains of sequential patterns. A pattern domain specifies new pattern shapes. For instance, temporally-annotated sequences (Giannotti et al., 2006) are sequential patterns with inter-event durations. There are many such extensions of sequential patterns devoted to answering specific questions.

3 Negative sequential patterns

This section introduces negative sequential pattern mining which aims at capturing the notion of the absence of some items in sequences. First, we introduce a syntax for the domain of negative patterns. Then, we present several possible semantics for patterns based on this syntax and show that some enjoys the anti-monotonicity property. Finally, we extend negative sequential patterns to constrain negative sequential patterns.

3.1 Syntax of negative sequential patterns

The negative sequential pattern (NSP) model extends classical sequential patterns by enabling the specification of the absence of some itemsets. For example, $\boldsymbol{p} = \langle a\ b\ \neg c\ e\ f\rangle$ is a negative pattern. The symbol $\neg$ before c denotes that item c must be absent. $\neg c$ is called a negative item. Semantically, $\boldsymbol{p}$ specifies that items a, b, e and f occur in a row, but no item c occurs between the occurrence of b and the occurrence of e.

In the field of string matching, negation is classically defined for regular expression. In this case, a pattern is an expression that can hold any kind of negated *pattern*. The same principle gives the following most generic definition of negative sequential patterns: Let $\mathcal{N}$ be the set of negative patterns. A negative pattern $\boldsymbol{p} = \langle p_1\ \cdots\ p_n\rangle \in \mathcal{N}$ is a sequence where, for all $i \in [n]$, p_i is a positive itemset ($p_i \subseteq \mathcal{I}$) or a negated pattern ($p_i = \neg\{q_i\},\ q_i \in \mathcal{N}$).

Due to its infinite recursive definition, $\mathcal{N}$ appears to be too huge to be an interesting and tractable search space for pattern mining. For instance, with $\mathcal{I} = \{a, b, c\}$, it is possible to express simple patterns like $\langle a\ \neg b\ c\rangle$ but also complex patterns like $\langle a\ \neg\langle b\ c\rangle\rangle$. The combinatorics for such patterns is infinite.

Now we provide our definition of negative sequential patterns (NSP) which introduces some syntactic restrictions compared with the most generic case. These simple

restrictions are broadly pointed in the literature (Kamepalli et al., 2014) and enable us to propose an efficient algorithm.

Definition 6 (Negative sequential patterns (NSP)) A negative pattern $\boldsymbol{p}$ is a sequence $\langle p_1 \cdots p_n \rangle$ where, for all $i \in [n]$, p_i is a positive itemset ($p_i = \{p_i^j\}_{j \in [m_i]},\ p_i^j \in \mathcal{I}$) or a negated itemset ($p_i = \neg\{q_i^j\}_{j \in [m'_i]},\ q_i^j \in \mathcal{I}$) under the two following constraints: consecutive negative itemsets are forbidden as well as negative itemsets at the pattern boundaries. m_i (resp. m'_i) is the size of the i-th positive (resp. negative) itemset. $\boldsymbol{p}^+$, the *positive part*[1] of pattern $\boldsymbol{p}$ denotes the subsequence of $\boldsymbol{p}$ restricted to its positive itemsets.

According to the non consecutive negative itemsets constraint, a negative pattern $\boldsymbol{p}$ has the general form $\boldsymbol{p} = \langle p_1\ \neg q_1\ p_2\ \neg q_2\ \cdots p_{n-1}\ \neg q_{n-1}\ p_n \rangle$ where $p_i \in 2^I \setminus \{\emptyset\}$ and $q_i \in 2^I$ for all $i \in [n]$ (q_i may be empty). Under this notation, $\boldsymbol{p}^+ = \langle p_1\ p_2\ \cdots p_{n-1}\ p_n \rangle$.

Let us illustrate the syntactic restrictions by some pattern counterexamples that our approach does not extract:

- first of all, a pattern is a sequence of positives and negative itemsets. It is not possible to have patterns such as $\langle a\ \neg\langle b\ c\rangle\rangle$
- then, successive negated itemsets such as $\langle a\ \neg b\ \neg c\ d\rangle$ are not allowed.
- finally, a pattern starting or finishing by a negated itemsets such as $\langle \neg b\ d\rangle$ is not allowed as well.

Finally, it is worth noticing that the syntax of NSP introduced in Definition 6 can not handle patterns with positive and negative events in the same itemset.

3.2 Semantics of negative sequential patterns

The semantics of negative sequential patterns relies on *negative containment*: a sequence $\boldsymbol{s}$ supports pattern $\boldsymbol{p}$ if $\boldsymbol{s}$ contains a sub-sequence $\boldsymbol{s'}$ such that every positive itemset of $\boldsymbol{p}$ is included in some itemset of $\boldsymbol{s'}$ in the same order and for any negative itemset $\neg q_i$ of $\boldsymbol{p}$, q_i is *not included* in any itemset occurring in the sub-sequence of $\boldsymbol{s'}$ located between the occurrence of the positive itemset preceding $\neg q_i$ in $\boldsymbol{p}$ and the occurrence of the positive itemset following $\neg q_i$ in $\boldsymbol{p}$.

So far in the literature, the absence or non-inclusion of itemsets (represented here as a negative itemset) has been specified by loose formulations. The authors of PNSP (Hsueh et al., 2008) have proposed the set symbol $\not\subseteq$ to specify non-inclusion. This symbol is misleading since it does not correspond to the associated semantics given in PNSP: an itemset I is absent from an itemset I' if the entire set I is absent from I' (as opposed to at least some item from I is absent from I') which corresponds to $I \cap I' = \emptyset$ in standard set notation, *i.e.* disjoint sets, and not $I \not\subseteq I'$. We will call PNSP interpretation *total non inclusion*, *i.e.* disjointness. It should be distinguished from *partial non inclusion* which corresponds (correctly) to the set symbol $\not\subseteq$. The

[1] Called the *maximal positive subsequence* in PNSP (Hsueh et al., 2008) and NegGSP (Zheng et al., 2009) or the *positive element id-set* in eNSP.

Table 1 Lists of supported sequences in $\mathcal{D}$ by negative patterns $\boldsymbol{p_i}$, $i = 1..4$ under the total and partial non inclusion semantics. Every pattern has the shape $\langle a\ \neg q_i\ b\rangle$ where q_i are itemsets such that $q_i \subset q_{i+1}$.

	partial non inclusion $\not\subseteq_G$	total non inclusion $\not\subseteq_D$
$\boldsymbol{p_1} = \langle b\neg ca\rangle$	$\{\boldsymbol{s_1}, \boldsymbol{s_3}, \boldsymbol{s_4}\}$	$\{\boldsymbol{s_1}, \boldsymbol{s_3}, \boldsymbol{s_4}\}$
$\boldsymbol{p_2} = \langle b\neg(cd)a\rangle$	$\{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}, \boldsymbol{s_4}\}$	$\{\boldsymbol{s_1}, \boldsymbol{s_4}\}$
$\boldsymbol{p_3} = \langle b\neg(cde)a\rangle$	$\{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}, \boldsymbol{s_4}\}$	$\{\boldsymbol{s_1}\}$
$\boldsymbol{p_4} = \langle b\neg(cdeg)a\rangle$	$\{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}, \boldsymbol{s_4}, \boldsymbol{s_5}\}$	$\{\boldsymbol{s_1}\}$
	monotonic	anti monotonic

symbol $\not\subseteq$ was further used by the authors of NegGSP (Zheng et al., 2009) and eNSP (Cao et al., 2016). The semantics of non inclusion is not detailed in NegGSP and one cannot determine whether it means total or partial non inclusion.[2] eNSP does not define explicitly the semantics of non inclusion but, from the procedure used to compute the support of patterns, one can deduce that it uses total non inclusion.

Definition 7 (Non inclusion) We introduce two relations between itemsets P and I:

- partial non inclusion: $P \not\subseteq_G I \Leftrightarrow \exists e \in P, e \notin I$
- total non inclusion: $P \not\subseteq_D I \Leftrightarrow \forall e \in P, e \notin I$

Choosing one non inclusion interpretation or the other has consequences on extracted patterns as well as on pattern search. Let us illustrate this on related pattern support in sequence dataset $\mathcal{D}$

$$\mathcal{D} = \left\{ \begin{array}{l} \boldsymbol{s_1} = \langle (bc)\ f\ a\rangle \\ \boldsymbol{s_2} = \langle (bc)\ (cf)\ a\rangle \\ \boldsymbol{s_3} = \langle (bc)\ (df)\ a\rangle \\ \boldsymbol{s_4} = \langle (bc)\ (ef)\ a\rangle \\ \boldsymbol{s_5} = \langle (bc)\ (cdef)\ a\rangle \end{array} \right\}.$$

Table 1 compares the support of progressively extended patterns under the two semantics to show whether anti-monotonicity is respected or not. Let us consider the inclusion of pattern $\boldsymbol{p_2} = \langle b\ \neg(cd)\ a\rangle$ in sequence $\boldsymbol{s_2}$. Since the positive part of $\boldsymbol{p_2}$ is in $\boldsymbol{s_2}$, $\boldsymbol{p_2}$ occurs in sequence $\boldsymbol{s_2}$ iff $(cd) \not\subseteq_* (cf)$ ((cd) is not included in (cf)). In case of total non inclusion, it is false that $(cd) \not\subseteq_D (cf)$ because c occurs in (cf), and thus $\boldsymbol{p_2}$ does not occur in $\boldsymbol{s_2}$. But in case of partial non inclusion, it is true that $(cd) \not\subseteq_G (cf)$, because d does not occur in (cf), and thus $\boldsymbol{p_2}$ occurs in $\boldsymbol{s_2}$.

Obviously, partial non inclusion satisfies anti-monotonicity while total non inclusion satisfies monotonicity. In the sequel we will denote the general form of itemset non inclusion by the symbol $\not\subseteq_*$, meaning either $\not\subseteq_G$ or $\not\subseteq_D$.

[2] Actually, though not clearly stated, it seems that the negative elements of NegGSP patterns consist of items rather than itemsets. In this case, total and partial inclusion are equivalent (see Proposition 6).

Now, we formulate the notions of sub-sequence, non inclusion and absence by means of the concept of embedding borrowed from (Negrevergne and Guns, 2015).

Definition 8 (Positive pattern embedding) Let $\boldsymbol{s} = \langle s_1 \cdots s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1 \cdots p_m \rangle$ be a (positive) sequential pattern. $\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ is an *embedding* of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff $p_i \subseteq s_{e_i}$ for all $i \in [m]$ and $e_i < e_{i+1}$ for all $i \in [m-1]$

Definition 9 (Strict and soft embeddings of negative patterns) Let $\boldsymbol{s} = \langle s_1 \cdots s_n \rangle$ be a sequence, $\boldsymbol{p} = \langle p_1 \cdots p_m \rangle$ be a negative sequential pattern and $\not\subseteq_* \in \{\not\subseteq_G, \not\subseteq_D\}$.

$\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ is a **soft-embedding** of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff for all $i \in [m]$:

- $p_i \subseteq s_{e_i}$ if p_i is positive
- $p_i \not\subseteq_* s_j,\ \forall j \in [e_{i-1}+1, e_{i+1}-1]$ if p_i is negative

$\boldsymbol{e} = (e_i)_{i \in [m]} \in [n]^m$ is a **strict-embedding** of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ iff for all $i \in [m]$:

- $p_i \subseteq s_{e_i}$ if p_i is positive
- $p_i \not\subseteq_* \bigcup_{j \in [e_{i-1}+1, e_{i+1}-1]} s_j$ if p_i is negative

Proposition 2 soft- *and* strict-*embeddings are equivalent when* $\not\subseteq_* \overset{\text{def}}{=} \not\subseteq_D$.

Proof The proofs of all propositions are in Appendix A.

Let $\boldsymbol{p}^+ = \langle p_{k_1} \cdots p_{k_l} \rangle$ be the positive part of some pattern $\boldsymbol{p}$, where l denotes the number of positive itemsets in $\boldsymbol{p}$. If $\boldsymbol{e}$ is an embedding of pattern $\boldsymbol{p}$ in some sequence $\boldsymbol{s}$, then $\boldsymbol{e}^+ = \langle e_{k_1} \cdots e_{k_l} \rangle$ is an embedding of the positive sequential pattern $\boldsymbol{p}^+$ in $\boldsymbol{s}$.

The following examples illustrate the impact of the itemset non-inclusion relation and of the embedding type on pattern occurrence.

Example 3 (Itemset absence semantics) Let $\boldsymbol{p} = \langle a\ \neg(bc)\ d \rangle$ be a pattern and four sequences:

Sequence	$\not\subseteq_D$	$\not\subseteq_G$ / strict-embedding	$\not\subseteq_G$ / soft-embedding
$\boldsymbol{s_1} = \langle a\ c\ b\ e\ d \rangle$			✓
$\boldsymbol{s_2} = \langle a\ (bc)\ e\ d \rangle$			
$\boldsymbol{s_3} = \langle a\ b\ e\ d \rangle$		✓	✓
$\boldsymbol{s_4} = \langle a\ e\ d \rangle$	✓	✓	✓

One can notice that each sequence contains a unique occurrence of $\langle a\ d \rangle$, the positive part of pattern $\boldsymbol{p}$. Using the soft-embedding and partial non-inclusion semantics ($\not\subseteq_* \overset{\text{def}}{=} \not\subseteq_G$), $\boldsymbol{p}$ occurs in sequences $\boldsymbol{s_1}$, $\boldsymbol{s_3}$ and $\boldsymbol{s_4}$ but not in $\boldsymbol{s_2}$. Using the strict-embedding and partial non-inclusion semantics, $\boldsymbol{p}$ occurs in sequences $\boldsymbol{s_3}$ and $\boldsymbol{s_4}$ considering that items b and c occur between occurrences of a and d in sequences $\boldsymbol{s_1}$ and $\boldsymbol{s_2}$. With total non inclusion ($\not\subseteq_* \overset{\text{def}}{=} \not\subseteq_D$) and either type of embeddings, the absence of an itemset is satisfied if any of its item is absent. As a consequence, $\boldsymbol{p}$ occurs only in sequence $\boldsymbol{s_4}$.

Another point that determines the semantics of negative containment concerns the multiple occurrences of some pattern in a sequence: should every or only one occurrence of the pattern positive part in the sequence satisfy the non inclusion constraints? This point is not discussed in previous propositions for negative sequential pattern mining. Actually, PNSP (Hsueh et al., 2008) and NegGSP (Zheng et al., 2009) require a weak absence (at least one occurrence should satisfy the non inclusion constraints) while eNSP requires a strong absence (every occurrence should satisfy non inclusion constraints).

Definition 10 (Negative pattern occurrence) Let $\boldsymbol{s}$ be a sequence, $\boldsymbol{p}$ be a negative sequential pattern. $\boldsymbol{p}^+$ is the positive part of $\boldsymbol{p}$.

- Pattern $\boldsymbol{p}$ *weakly-occurs* in sequence $\boldsymbol{s}$, denoted $\boldsymbol{p} \preceq \boldsymbol{s}$, iff there exists at least one (strict/soft)-embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.
- Pattern $\boldsymbol{p}$ *strongly-occurs* in sequence $\boldsymbol{s}$, denoted $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$, iff for any embedding $\boldsymbol{e'}$ of $\boldsymbol{p}^+$ in $\boldsymbol{s}$ there exists an embedding $\boldsymbol{e}$ of $\boldsymbol{p}$ in $\boldsymbol{s}$ such that $\boldsymbol{e'} = \boldsymbol{e}$.

Definition 10 allows for formulating two notions of absence semantics for negative sequential patterns depending on the occurrence of the positive part:

- *strong occurrence*: a negative pattern $\boldsymbol{p}$ occurs in a sequence $\boldsymbol{s}$ iff there exists at least one occurrence of the positive part of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ and **every** such occurrence satisfies the negative constraints,
- *weak occurrence*: a negative pattern $\boldsymbol{p}$ occurs in a sequence $\boldsymbol{s}$ iff there exists at least one occurrence of the positive part of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$ and **one** of these occurrences satisfies the negative constraints.

Example 4 (Strong vs weak occurrence semantics) Let $\boldsymbol{p} = \langle a\ b\ \neg c\ d\ \rangle$ be a pattern and $\boldsymbol{s_1} = \langle a\ b\ e\ d\rangle$ and $\boldsymbol{s_2} = \langle a\ b\ c\ a\ d\ e\ b\ d\rangle$ be two sequences. The positive part of $\boldsymbol{p}$ is $\langle a\ b\ d\rangle$. It occurs once in $\boldsymbol{s_1}$ so there is no difference for occurrences under the two semantics. But, it occurs three times in $\boldsymbol{s_2}$ with embeddings $(1, 2, 5)$, $(1, 2, 8)$ and $(4, 7, 8)$. The two first occurrences do not satisfy the negative constraint $(\neg c)$ while the third occurrence does. Under the weak occurrence semantics, pattern $\boldsymbol{p}$ occurs in sequence $\boldsymbol{s_2}$ whereas under the strong occurrence semantics it does not.

The definitions of pattern support, frequent pattern and pattern mining task derive naturally from the notion of negative sequential pattern occurrence, no matter the choices for embedding (soft or strict), non inclusion (partial or total) and occurrence (weak or strong). However, these choices concerning the semantics of NSPs impact directly the number of frequent patterns (under the same minimal threshold) and further the computation time. The stronger the negative constraints, the fewer the number of sequences that hold some pattern, and the fewer the number of frequent patterns.

Finally, we introduce a partial order on NSPs that is the foundation of our efficient NSP mining algorithm.

Definition 11 (NSP partial order) Let $\boldsymbol{p} = \langle p_1\ \neg q_1\ p_2\ \neg q_2\ \cdots p_{k-1}\ \neg q_{k-1}\ p_k\rangle$ and $\boldsymbol{p'} = \langle p'_1\ \neg q'_1\ p'_2\ \neg q'_2\ \cdots\ p'_{k'-1}\ \neg q'_{k'-1}\ p'_{k'}\rangle$ be two NSPs s.t. $p_i \neq \emptyset$ for all $i \in [k]$ and $p'_i \neq \emptyset$ for all $i \in [k']$. We write $\boldsymbol{p} \lhd \boldsymbol{p'}$ iff $k \leq k'$ and:

1. $\forall i \in [k-1],\ p_i \subseteq p'_i$ and $q_i \subseteq q'_i$
2. $p_k \subseteq p'_k$
3. $k' = k \implies p_k \neq p'_k$ (irreflexive)

Intuitively, $\boldsymbol{p} \lhd \boldsymbol{p'}$ if $\boldsymbol{p}$ is shorter than $\boldsymbol{p'}$ and the positive and negative itemsets of $\boldsymbol{p}$ are pairwise included into the itemsets of $\boldsymbol{p'}$. The classical pattern inclusion fails to be anti-monotonic (Zheng et al., 2009) because pattern extension may change the scope of negative itemsets. We illustrate what is happening on two examples. Let us first consider the case of a pattern ending with a negated itemset illustrated by Zheng et al. (2009). Let $\boldsymbol{p'} = \langle b\ \neg c\ a \rangle$ and $\boldsymbol{p} = \langle b\ \neg c \rangle$ be two NSPs. Removing a from $\boldsymbol{p'}$ makes the constraint of the pattern positive part weaker, and so the positive part is more frequent. But, removing a extends also the scope of the negative constraint and makes it stronger and so, the negative pattern may be less frequent which violates the anti-monotonicity property. This specific case does not impact our framework. In fact, our definition of NSP (see Definition 6) does not allow ending an NSP with negated itemsets. Let us now consider patterns $\boldsymbol{p'} = \langle b\ \neg c\ d\ a \rangle$ and $\boldsymbol{p} = \langle b\ \neg c\ a \rangle$, and the sequence $\boldsymbol{s} = \langle b\ e\ d\ c\ a \rangle$. $\boldsymbol{p'}$ occurs in $\boldsymbol{s}$ but not $\boldsymbol{p}$ since the scope of the negated itemset $\neg c$ is wider: it is restricted to the interval between the occurrences of b and d for $\boldsymbol{p'}$, but to the interval between the occurrences of b and a for $\boldsymbol{p}$.

What is important to note for the partial order $\lhd$ of Definition 11, is that the embedding of the pattern positive part yields an embedding for $\boldsymbol{p}$ that imposes the negative constraints on the exact same scope than the negative constraints of $\boldsymbol{p'}$. Thanks to the anti-monotonicity of $\not\subseteq_D$, additional itemsets in negative patterns lead to stronger constrain the containment relation. These remarks give some intuition behind the following anti-monotonicity property (Proposition 3).

Proposition 3 (Anti-monotonicity of NSP support) *The support of NSP is anti-monotonic with respect to $\lhd$ when $\not\subseteq_* \overset{\text{def}}{=} \not\subseteq_D$ and weak-occurrences ($\preceq$) are considered.*

Proof see Appendix A.

We can note that when the strong occurrence semantic ($\sqsubseteq$) is used, $\lhd$ violates the anti-monotonicity. Considering $\boldsymbol{p'} = \langle a\ (bc)\ \neg c\ d \rangle$, $\boldsymbol{p} = \langle a\ b\ \neg c\ d \rangle$ and $\boldsymbol{s} = \langle a\ \mathbf{b}\ (bc)\ e\ d \rangle$, then it is true that $\boldsymbol{p'} \sqsubseteq \boldsymbol{s}$, but not that $\boldsymbol{p} \sqsubseteq \boldsymbol{s}$. There are two possible embeddings for $\boldsymbol{p'}$ in $\boldsymbol{p'}$: $(1, 2, 5)$ and $(1, 3, 5)$. The first one ($(1, 2, 5)$, which does not derive from the embedding of $\boldsymbol{p'}$) does not satisfy the negative constraint.

A second example illustrates another case that is encountered when the postfix of a sequence restricts the set of valid embeddings: $\boldsymbol{p'} = \langle a\ \neg b\ d\ \mathbf{c} \rangle$, $\boldsymbol{p} = \langle a\ \neg b\ d \rangle$ and $\boldsymbol{s} = \langle a\ e\ d\ c\ b\ d \rangle$. Again, $\boldsymbol{p'}$ occurs only once while $\boldsymbol{p}$ occurs twice and one of its embeddings does not satisfy the negated itemset $\neg b$. This example shows that a simple postfix extension of an NSP leads to violate the anti-monotonicity property under the strong occurrence semantics.

3.3 Gap constraints over negative sequential patterns

Numerical constraints have been introduced early in sequential constraints. Such constraints can also be introduced in NSPs yielding *constrained negative sequential patterns*. We consider the two most common constraints on sequential patterns:

maxgap ($\theta \in \mathbb{N}$) and *maxspan* ($\tau \in \mathbb{N}$) constraints. These constraints impact NSP embeddings whatever their semantic. An embedding $\boldsymbol{e} = \{e_i, \cdots, e_n\}$ of a pattern $\boldsymbol{p}$ in some sequence $\boldsymbol{s}$ satisfies the *maxgap* (resp. *maxspan*) constraint iff $\boldsymbol{e}$ satisfies the following constraint: $e_{i+1} - e_i \leq \theta$ (resp. $e_n - e_1 \leq \tau$) for all $i \in [n-1]$.

Example 5 (Embedding of a constrained NSP) Let $\boldsymbol{p} = \langle a\ \neg b\ c\ d \rangle$ be a NSP and we consider the following constraints: not more than one itemset in between positive itemsets occurrences ($\theta = 2$) and not more than three itemsets in between the first and the last itemset occurrences ($\tau = 4$). Table below illustrates on five sequences whether the sequence supports pattern $\boldsymbol{p}$ or not. These simple patterns and sequences lead to the same results whatever the containment relation between NSP and sequences.

Sequence	$\tau = \infty$ $\theta = \infty$	$\tau = 4$ $\theta = \infty$	$\tau = \infty$ $\theta = 2$	$\tau = 4$ $\theta = 2$
$\langle a\ b\ c\ d \rangle$				
$\langle a\ e\ c\ d \rangle$	✓	✓		
$\langle a\ e\ e\ c\ d \rangle$	✓		✓	
$\langle a\ e\ c\ e\ d \rangle$	✓	✓	✓	✓
$\langle a\ e\ c\ b\ d \rangle$	✓	✓	✓	✓

It is worth noticing that the anti-monotonicity of the support is preserved with a *maxspan* constraint but not with a *maxgap* constraint for frequent sequential patterns. Nonetheless, the support becomes anti-monotonic for both constraints with the partial order that takes only into consideration extensions at the end of sequential patterns and not elsewhere. The partial order specified in Definition 11 for negative sequential patterns is based on the same idea of having extensions only at the end of a pattern. Thus, the support of constrained NSP also enjoys the anti-monotonicity property.

Proposition 4 (Anti-monotonicity of constrained NSP support) *The support of NSP with maxgap or maxspan constraints is anti-monotonic with respect to $\lhd$ when $\not\subseteq_* \stackrel{\text{def}}{=} \not\subseteq_D$ and weak-occurrences ($\preceq$) are considered.*

Proof see Appendix A.

4 Algorithm NegPSpan

In this section, we introduce algorithm NegPSpan for mining NSPs from a dataset of sequences under *maxgap* and *maxspan* constraints and under a soft absence semantics with $\not\subseteq_* \stackrel{\text{def}}{=} \not\subseteq_D$ for itemset inclusion. As stated in Proposition 2, no matter the embedding strategy, they are equivalent under total itemset inclusion. Considering occurrences, NegPSpan uses the weak-occurrence semantics: at least one occurrence of the negative pattern is sufficient to consider that it is supported by the sequence.

For computational reasons, we make an additional assumption for admissible itemsets as negative itemsets: the negative itemsets are restricted to one element

belonging to some language $\mathcal{L}^- \subseteq \mathcal{I}^{\mathbb{N}}$. Considering all possible itemsets (*i.e.* $\mathcal{I}^{\mathbb{N}}$) for negatives is computationally costly. The definition of a subset $\mathcal{L}^-$ allows to control partially the combinatorics introduced by negative itemsets. In algorithm NegPSpan presented below, $\mathcal{L}^- = \{I = \{i_1, \cdots, i_n\} | \forall k,\ supp(i_k) \geq \sigma\}$ denotes the set of itemsets that can be built from frequent items. But this set could be arbitrarily defined when the user is interested in the absence of some specific events. For instance, the definition of $\mathcal{L}^-$ could be changed into the set of frequent itemsets, which would be more restrictive than the set of itemsets made of frequent items.

4.1 Main algorithm

NegPSpan is based on algorithm PrefixSpan (Pei et al., 2004) which implements a depth first search and uses the principle of database projection to reduce the number of sequence scans. NegPSpan adapts the pseudo-projection principle of PrefixSpan which uses a projection pointer to avoid copying the data. For NegPSpan, a projection pointer of some pattern $\boldsymbol{p}$ is a triple $\langle sid, ppred, pos \rangle$ where sid is a sequence identifier in the dataset, pos is the position in sequence sid that matches the last itemset of the pattern (necessarily positive) and $ppred$ is the position of the previous positive pattern. In the PrefixSpan algorithm, the projection pointer is the couple $\langle sid, pos \rangle$. NegPSpan adds the position of the previous positive pattern to locate the region of the sequence in which evaluate the presence or absence of a candidate extensions of the last negative itemset.

Algorithm 1 details the main recursive function of NegPSpan for extending a current pattern $\boldsymbol{p}$. The principle of this function is similar to PrefixSpan. Every pattern $\boldsymbol{p}$ is associated with a pseudo-projected database represented by both the original set of sequences $\mathcal{S}$ and a set of projection pointers $occs$. First, the function evaluates the size of $occs$ to determine whether pattern $\boldsymbol{p}$ is frequent or not. If so, it is

Algorithm 1: NegPSpan: recursive function for negative sequential pattern extraction

in: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern of length k, σ: minimum support threshold, $occs$: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
1  Function NegPSpan (S, σ, p, occs, I^f, θ, τ):
       //Support evaluation of pattern p
2      if |occs| ≥ σ then
3          OutputPattern(p, occs);
4      else
5          return;
       //Positive itemset composition
6      PositiveComposition(S, σ, p, occs, I^f, θ, τ);
       //Positive sequential extension
7      PositiveSequence(S, σ, p, occs, I^f, θ, τ);
8      if |p| ≥ 2 and |p_k| = 1 then
           //Negative sequential extension
9          NegativeExtension(S, σ, p, occs, I^f, θ, τ);
```

outputted, otherwise, the recursion is stopped because no larger patterns are possible (anti-monotonicity property).

Then, the function explores three types of pattern extensions of pattern $\boldsymbol{p}$ into a pattern $\boldsymbol{p}'$:

- the positive sequence composition ($\rightsquigarrow_c$) consists in adding one item to the last itemset of $\boldsymbol{p}$ (following the notations of Definition 11, this extension corresponds to the case in which $\boldsymbol{p}'$ is the extension of $\boldsymbol{p}$ where $k' = k, q_i = q'_i$ for all $i \in [k-1]$ and $|p'_k| = |p_k| + 1$),
- the positive sequence extension ($\rightsquigarrow_s$) consists in adding a new positive singleton itemset at the end of $\boldsymbol{p}$ ($k' = k+1$, $q_i = q'_i$ for all $i \in [k-1]$ and $|p'_{k'}| = 1$),
- the negative sequence extension ($\rightsquigarrow_n$) consists in appending a negative itemset between the last two positive itemsets of $\boldsymbol{p}$ ($k' = k$, $q_i = q'_i$ for all $i \in [k-2]$, $|q'_{k-1}| = |q_{k-1}| + 1$ and $p'_k = p_k$). In addition, NSP are negatively extended iff $|p_k| = 1$. It prevents redundant exploration of same patterns (see Section 4.3).

The negative pattern extension is specific to our algorithm and is detailed in next section. The first two extensions are identical to PrefixSpan pattern extensions, including their gap constraints management, *i.e. maxgap* and *maxspan* constraints on positive patterns.

Proposition 5 *The proposed algorithm is correct and complete.*

Proof see Appendix A.

Intuitively, the algorithm is complete considering that the three extensions enable to generate any NSP. For instance, pattern $\langle a\ \neg e\ b\ (ce)\ \neg(bd)\ a\rangle$ would be generated after evaluating the following patterns: $\langle a\rangle \rightsquigarrow_s \langle a\ b\rangle \rightsquigarrow_n \langle a\ \neg e\ b\rangle \rightsquigarrow_s \langle a\ \neg e\ b\ c\rangle \rightsquigarrow_c \langle a\ \neg e\ b\ (ce)\rangle \rightsquigarrow_s \langle a\ \neg e\ b\ (ce)\ a\rangle \rightsquigarrow_n \langle a\ \neg e\ b\ (ce)\ \neg b\ a\rangle \rightsquigarrow_n \langle a\ \neg e\ b\ (ce)\ \neg(bd)\ a\rangle$. Secondly, according to Proposition 3, the pruning strategy is correct.

4.2 Extension of patterns with negated itemsets ($\rightsquigarrow_n$)

Algorithm 2 extends the current pattern $\boldsymbol{p}$ with negative items. It generates new candidates by inserting an item $it \in \mathcal{I}^f$, the set of frequent items. Let p_{k-1} and p_k denote respectively the penultimate itemset and the last itemset of $\boldsymbol{p}$. If p_{k-1} is positive, then a new negated itemset is inserted between p_{k-1} and p_k. Otherwise, if p_{k-1} is negative, item it is appended to p_{k-1}. To prevent the redundant enumeration of negative itemsets, only items it (lexicographically) greater than the last item of p_{k-1} can be added.

Once the pattern has been extended, lines 10 to 20 evaluate the candidate by computing the pseudo-projection of the current database. According to the selected semantics associated with $\not\subseteq_D$, *i.e.* total non inclusion (see Definition 9), it is sufficient to check the absence of it in the subsequence included between the occurrences of the two positive itemsets surounding it. To achieve this, the algorithm checks the sequence positions in the interval $[occ.ppred + 1, occ.pos - 1]$. If it does not occur in itemsets from this interval, then the extended pattern occurs in the sequence $occ.sid$.

Algorithm 2: NegPSpan: negative extensions

in: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern of length k, σ: minimum support threshold, $occs$: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
1  Function NegativeExtension(S, σ, p, occs, I^f, θ, τ):
2      for it ∈ I^f do
3          if p_{k-1} is pos then
               //Insert the negative item at the penultimate position
4              p.insert(¬it);
5          else
6              if it > p_{k-1}.back() then
                   //Append an item to the penultimate (negative) itemset
7                  p_{k-1}.append(¬it);
8              else
9                  continue;
10         newoccs ← ∅;
11         for occ ∈ occs do
12             found ← false;
13             for sp = [occ.pred + 1, occ.pos − 1] do
14                 if it ∈ s_{occ.sid}[sp] then
15                     found ← true;
16                     break;
17             if not found then
18                 newoccs ← newoccs ∪ {occ};
19             else
                   //Look for an alternative occurrence
20                 newoccs ← newoccs ∪ Match(s_{occ.sid}, p, θ, τ);
21         NegPSpan (D, σ, p, newoccs, I^f);
22         p_{k-1}.pop();
```

Otherwise, to ensure the completeness of the algorithm, another occurrence of the pattern has to be searched in the sequence (*cf.* `Match` function that takes into account gap constraints).

For example, the first occurrence of pattern $\boldsymbol{p} = \langle a\ b\ c \rangle$ in sequence $\langle a\ b\ e\ c\ a\ b\ c \rangle$ is $occ_{\boldsymbol{p}} = \langle sid, 2, 4 \rangle$. Let us now consider $\boldsymbol{p'} = \langle a\ b\ \neg e\ c \rangle$, a negative extension of $\boldsymbol{p}$. The extension of the projection-pointer $occ_{\boldsymbol{p}}$ does not satisfy the absence of e. So a new occurrence of $\boldsymbol{p}$ has to be searched for. $\langle sid, 6, 7 \rangle$, the next occurrence of $\boldsymbol{p}$, satisfies the negative constraint. Then, NegPSpan is called recursively for extending the new current pattern $\langle a\ b\ \neg e\ c \rangle$.

We can note that the gap constraints τ and θ are not mentioned explicitly in this algorithm (except when a complete matching is required), but they impact indirectly the algorithm by narrowing the interval of admissible sequence positions (line 13).

4.2.1 Extracting NSP without surrounding negations

An option may be used to restrict negated items to not be surrounded by itemsets containing this item. This option is motivated by the objective to ease pattern under-

standing. A pattern $\langle a\ \neg b\ b\ c\rangle$ may be interpreted as "there is exactly one occurrence of b between a and c". But, this may also lead to redundant patterns. For instance, $\langle a\ b\ \neg b\ c\rangle$ matches exactly the same sequences as $\langle a\ \neg b\ b\ c\rangle$ (see section 4.3). This second restriction can be disabled in our algorithm implementation. If so and for the sake of simplicity, we choose to yield only pattern $\langle a\ b\ \neg b\ c\rangle$.

The set of such restricted NSPs can be extracted using the same algorithm, simply by changing the candidate generation in Algorithm 2, line 2 by $it \in \mathcal{I}^f \setminus (p_k \cup p_{k-1})$. The items to be added to a negative itemset are among frequent items where items of surrounding itemsets are filtered out.

4.2.2 Extracting NSP with partial non inclusion ($\not\subseteq_* \stackrel{\text{def}}{=} \not\subseteq_G$)

Algorithm 3 presents the variant of the negative extension algorithm for the partial non-inclusion ($\not\subseteq_* \stackrel{\text{def}}{=} \not\subseteq_G$). The backbone of the algorithm is similar: a candidate pattern with a negated itemset at the penultimate position is generated and it assesses whether this candidate is frequent or not. The absence of the itemset is is checked in the itemsets of the sequence at positions between the occurrence of the preceding itemset and the occurrence of last itemset. The test in line 8 assesses that it is false that $is \not\subseteq_G \boldsymbol{s_{occ.sid}}[sp]$: is is not partially non-included in one sequence itemset iff is is a subset of this itemset.

Contrary to Algorithm 2, it is not possible to use negative sequence extension ($\rightsquigarrow_n$) to extend negative itemsets. With partial non inclusion the support is monotonic (and not anti-monotonic). Thus, candidate patterns extended with negative itemsets are generated based on a user defined $\mathcal{L}^- \subseteq \mathcal{I}^{\mathbb{N}}$. We remind that the default $\mathcal{L}^-$ is the set of itemsets that can be built from frequent items, *i.e.* $\mathcal{L}^- = \{I = \{i_1, \cdots, i_n\} | \forall k,\ supp(i_k) \geq \sigma\}$. According to the monotony property if a pattern $\langle a\ \neg(bc)\ d\rangle$ is not frequent, the pattern $\langle a\ \neg b\ d\rangle$ is not frequent either. But, in practice, both candidates will be evaluate by Algorithm 3. Thus, the combinatorics of this variant is significantly higher in practice because each element of $\mathcal{L}^-$ is evaluated.

4.3 Redundancy avoidance

Algorithm NegPSpan is syntactically non-redundant but it can in practice generate patterns that are semantically redundant.

For instance, pairs of patterns like $\langle a\ \neg b\ b\ c\rangle$ and $\langle a\ b\ \neg b\ c\rangle$ are syntactically different but match exactly the same sequences. Semantically, such pattern could be interpreted as "*there is not more than one occurrence of b between a and c*". It is possible to avoid generating both type of patterns efficiently. Our solution is to avoid the generation of candidate patterns with negative items that are in the last itemset. Thus, only $\langle a\ b\ \neg b\ c\rangle$ would be generated. In Algorithm 2 line 2, the list of frequent items $\mathcal{I}^f$ is replaced by $\mathcal{I}^f \setminus p_k$. But, this modification leads to loose the completeness of the algorithm. In fact, the pattern $\langle a\ \neg b\ b\rangle$ is not generated neither its semantically equivalent pattern $\langle a\ b\ \neg b\rangle$ because of the syntactic constraint on NSP that can not end with a negative itemset. In practice, we do not manage this kind of redundancy.

Algorithm 3: NegPSpan: negative extensions with partial non-inclusion (alternative to Algorithm 2)

in: $\mathcal{S}$: set of sequences, $\boldsymbol{p}$: current pattern, σ: minimum support threshold, $occs$: list of occurrences, $\mathcal{I}^f$: set of frequent items, θ: maxgap, τ: maxspan

```
1  Function NegativeExtension(S, σ, p, occs, I^f, θ, τ):
2      for is ∈ L^- do
           //Insert the negative itemset at the penultimate position
3          p.insert(¬is);
4          newoccs ← ∅;
5          for occ ∈ occs do
6              found ← false;
7              for sp = [occ.pred + 1, occ.pos − 1] do
8                  if is ⊆ s_{occ.sid}[sp] then
9                      found ← true;
10                     break;
11             if !found then
12                 newoccs ← newoccs ∪ {occ};
13             else
                   //Look for an alternative occurrence
14                 newoccs ← newoccs ∪ Match(s_{occ.sid}, p, θ, τ);
15         NegPSpan (D, σ, p, newoccs, I^f);
16         p_{k−1} = ∅;
```

We prefer the sound and correct option of not surrounding negative itemsets (see section 4.2.1).

A syntactic redundancy is introduced by extending patterns with negative items. For instance, the pattern $\langle a\ \neg b\ (cd)\rangle$ may be reached by two distinct paths p_1 : $\langle a\ c\rangle \rightsquigarrow_c \langle a\ (cd)\rangle \rightsquigarrow_n \langle a\ \neg b\ (cd)\rangle$ or $p_2 : \langle a\ c\rangle \rightsquigarrow_n \langle a\ \neg b\ c\rangle \rightsquigarrow_c \langle a\ \neg b\ (cd)\rangle$. To solve this problem, the algorithm first specifies the negative itemsets as a composition of negative items and then it composes the last itemset with new items. This discards the path p_1. In Algorithm 1, line 8 enables negative extension only if the last (positive) itemset is of size 1.

4.4 Execution example

This section illustrates the execution of the algorithm on a small example. Let us consider the dataset of sequences illustrated in Table 2 and the minimal support threshold $\sigma = 2$. In this example, we consider the following semantics for negative patterns: total non inclusion and strict absence. No gap constraints are considered ($\theta = \infty$ and $\tau = \infty$) and $\mathcal{I} = \{a, b, c, d, e, f\}$. In this example, candidate negative itemsets are simply frequent items: $\mathcal{L}^- = \{a, b, c, d, e\}$. Event f is not in $\mathcal{L}^-$ because it occurs only once and thus is not frequent under the minimal support σ.

Figure 1 illustrates the execution of NegPSpan algorithm on the dataset of Table 2 starting from pattern $\langle a\rangle$. The search tree illustrates the successive patterns explored by the depth-first search strategy. Each node details both the pattern and the corresponding

Table 2 Dataset of sequences used in the execution example.

SID	Sequence
s_1	$\langle a\ c\ b\ e\ d\rangle$
s_2	$\langle a\ (bc)\ e\rangle$
s_3	$\langle a\ b\ e\ d\rangle$
s_4	$\langle a\ e\ d\ f\rangle$

projected database. For the sake of space, the tree is simplified and some nodes are missing.

For patterns longer than two, projected sequences are represented in two gray levels: the bold part of the sequence can be used to make positive extensions and the gray part of the sequence is used to assess the absence of items for negative extension. Two markers locate the projection pointer positions. The second pointer is the same as the one computed by PrefixSpan.

Let us consider projected sequences of pattern $\langle a\ e\rangle$. In the first sequence, d is green as it is the part of the sequence ending the sequence after the position of e. c and b are in red because these events are in between occurrences of a and e. Pattern $\langle a\ e\rangle$ can be extended in two ways:

- with negative items among $\mathcal{L} \setminus \{a, e\}$ (a and e are removed if the restriction on second restriction is activated),
- with positive items among items that are frequent in the bold parts.

Considering extension of pattern $\langle a\ e\rangle$ with a negative item, *e.g.* $\neg c$, each sequence whose gray part contains the item is discarded, the others remain identical. The extension by $\neg c$ leads to pattern $\langle a\ \neg c\ e\rangle$ which is supported by sequences s_3 and s_4 only.

The extension of pattern $\langle a\ e\rangle$ by a positive item follows the same strategy as PrefixSpan. In this case, the algorithm explores only the extension by item d and the projected pointers are updated to reduce further scanning.

Adding a new negative item while the penultimate item is negative consists in appending it to the negative itemset. In case of pattern $\langle a\ \neg c\ e\rangle$, d is the only candidate because e is one of the surrounding events and b is before c in the lexicographic order. Under total non inclusion, again, we simply have to discard sequences that contain the item d within their gray part.
For extending $\langle a\ \neg d\ e\ d\rangle$, we can see that every combination of itemsets may quickly satisfy all the negation constraints. This suggests first to carefully select the appropriate $\mathcal{L}^-$ and second to use as big as possible negative itemsets to avoid pattern explosion.

Finally, we also notice that extensions with negative items are not terminal recursive steps. Once negative items have been inserted, new positive items can be append to the pattern. We encounter this case with pattern $\langle a\ \neg d\ e\rangle$ which is extended by item d.

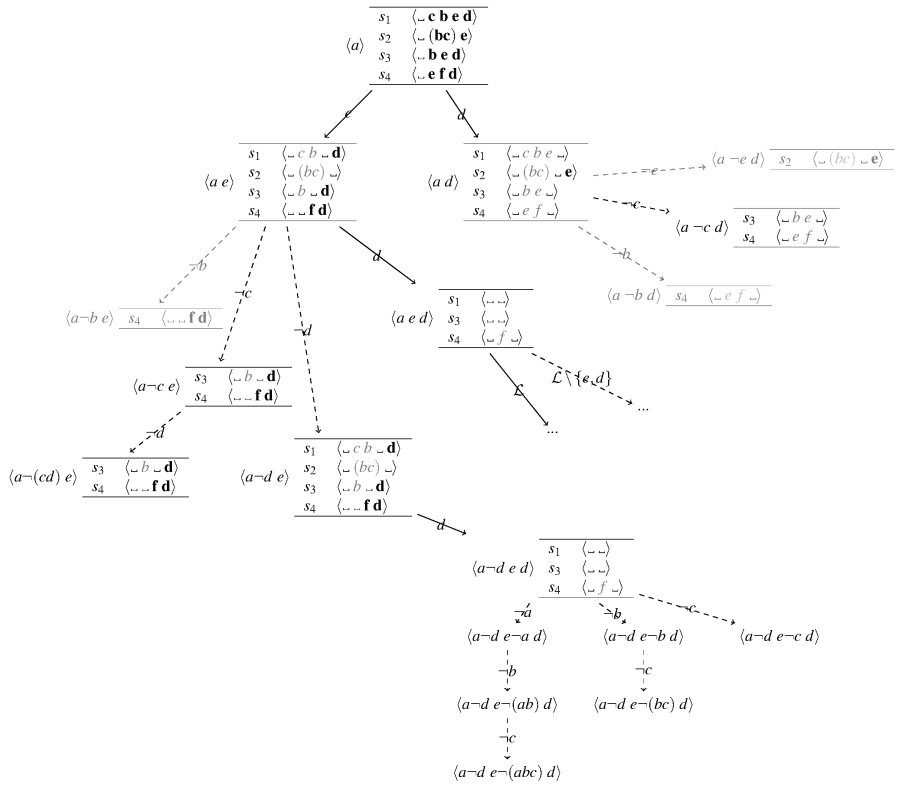

Fig. 1 Example of algorithm NegPSpan search tree on the dataset of Table 2. Each node tree represents the pattern on the left and the projected database on the right. The gray part of sequences is used to assess future negations while the bold part of sequences is used for sequential ($\rightsquigarrow_s$). Dashed arrows represent negative extensions ($\rightsquigarrow_n$) while plain arrows are sequential ($\rightsquigarrow_s$) or compositional extensions ($\rightsquigarrow_c$). Arrow label holds the item that is used in the extension.

5 Experiments

This section presents experiments on synthetic and real data. Experiments on synthetic data aims at exploring and comparing NegPSpan and eNSP for negative sequential pattern mining. The other experiments were conducted on real dataset to illustrate results for negative patterns. NegPSpan and eNSP have been implemented in C++.[3]

5.1 Benchmark on synthetic datasets

This section presents experiments on synthetically generated data. We first provide some details about the random sequence generator. This sequence generator enables to study the behavior of eNSP and NegPSpan for some specific dataset features and

[3] Code, data generator and synthetic benchmark datasets can be tested online and downloaded here: `http://people.irisa.fr/Thomas.Guyet/negativepatterns/`.

algorithm parameters. First, we study the time-efficiency and the number of extracted patterns with respect to the minimal frequency threshold. Then, we complement the analysis of the number of patterns with a study of the capacity of eNSP or NegPSpan to extract accurately some NSPs known to be frequent in the synthetic dataset. Note that additional experiments on the behavior of the algorithms with respect to the vocabulary size and the average sequence length can be found in Appendix C.

5.1.1 Random sequence generator

The principle of our sequence generator is the following: generate random negative patterns and randomly plant or not some of their occurrences into randomly generated sequences. The main parameters are the total number of sequences (n, default value is $n = 500$), the mean sequences length ($l = 20$), the vocabulary size *i.e.* the number of items ($d = 20$), the total number of patterns to plant (3), their mean length (4) and the pattern minimal occurrence frequency in the dataset ($f = 10\%$).

In addition, the generator ensures that the positive partners of a NSP occurs in a controlled number of the generated sequences. The number of occurrences of the positive partners is in $[\lambda_{min} \times f \times n, \lambda_{max} \times f \times n]$.

For sake of fairness of our evaluation, the generated sequences ensures that eNSP extracts less patterns than NegPSpan on our synthetic datasets. It is the case under two specific conditions. First, the generated sequences are sequences of items (not itemsets). In this case, Proposition 6 (see Appendix B) shows that any frequent NSP with the semantic of eNSP is also frequent with the semantic of NegPSpan. Second, we restrict $\mathcal{L}^-$ to the set of frequent items also for fairness of evaluation. Indeed, for sequences of items, patterns extracted by eNSP are only made of items and not itemsets because positive partners must be a frequent sequential pattern in sequences of items. This means that for sequence of items, eNSP does not manage any conjunction of negative items. The restriction of $\mathcal{L}^-$ for NegPSpan ensures that NegPSpan will not extract patterns with negative itemsets of size strictly larger than 1.

The occurrences of negative pattern are randomly generated without restriction on the gaps between item occurrences of a pattern. This means that the number of occurrences of planted NSP may be higher than its evaluated support while taking gap constraints into consideration.

5.1.2 Computation time and number of patterns

Figure 2 displays the computation time and the number of patterns extracted by eNSP and NegPSpan on sequences of length 20 and 30, under three minimal thresholds ($\sigma = 10\%$, 15% and 20%) and with different values for the maxgap constraint ($\tau = 4$, 7, 10 and ∞). For eNSP, the minimal support of positive partners, denoted ς, is set to 70% of the minimal threshold σ. For both approaches, we bound the pattern length to 5 items. Each boxplot has been obtained with a 20 different sequence datasets. Each run has a timeout of 5 minutes (300 s).

The main conclusion from Figure 2 is that NegPSpan is more efficient than eNSP when maxgap constraints are used. As expected, eNSP is more efficient than NegPSpan without any maxgap constraint. This is mainly due to the number of

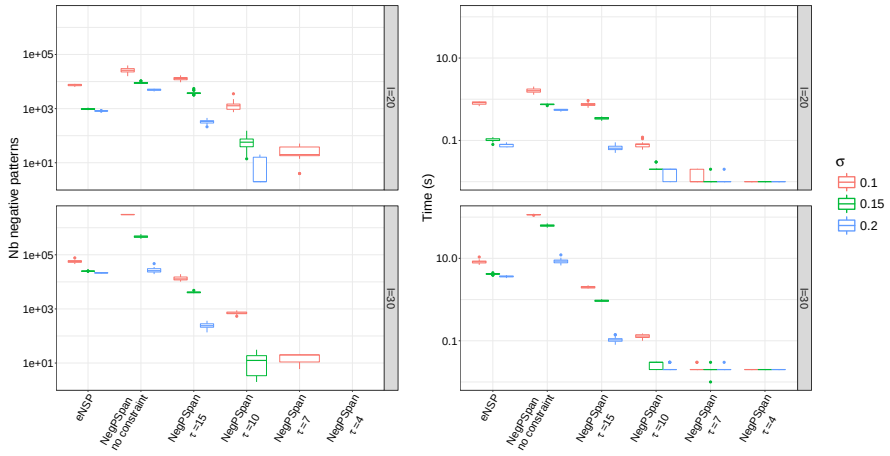

Fig. 2 Comparison of eNSP and NegPSpan on number of extracted patterns (left) and computation time (right) for different values of maxgap (τ). Top (resp. bottom) figures correspond to datasets with average sequence length equal to 20 (resp. 30). Boxplot colors correspond to different values of σ (10%, 15% and 20%).

extracted patterns. NegPSpan extracts significantly more patterns than eNSP because of different choices for the semantics of NSPs. First, eNSP uses a stronger negation semantics. Without maxgap constraints, the set of patterns extracted by NegPSpan is a superset of those extracted by eNSP (see proof in Appendix B).

An interesting result is that, for reasonably long sequences (20 or 30), even a weak maxgap constraint ($\tau = 10$) significantly reduces the number of patterns and makes NegPSpan more efficient. $\tau = 10$ is said to be a *weak* constraint because it does not cut early the search of a next occurring item compared to the length of the sequence (20 or 30). This is of particular interest because maxgap is a quite natural constraint when mining long sequences. It prevents from taking into account long distance correlations that are likely irrelevant. Note that section 5.1.4 discusses the fairness of this setting as NegPSpan extracts more planted patterns that eNSP does. Another interesting question raised by these results is the real meaning of extracted patterns by eNSP. In fact, under low frequency thresholds, it extracts numerous patterns that are not frequent when weak maxgap constraints are considered. As a consequence, the significance of most of the patterns extracted by eNSP seems low when processing datasets containing "long" sequences. Extensive experiments on the influence of sequence length can be found in Appendix C.2.

5.1.3 Influence of minimum threshold

Figure 3 illustrates computation time and memory consumption with respect to minimum threshold for different settings: eNSP is run with different values for ς, the minimal frequency of the positive partner of negative patterns (100%, 80% and 20%

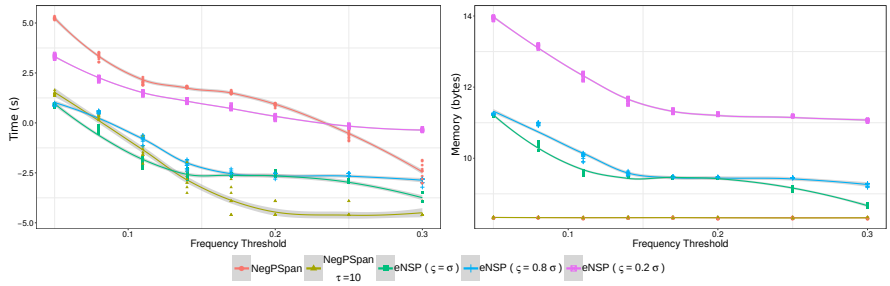

Fig. 3 Comparison of eNSP and NegPSpan computation time (left) and memory consumption (right) wrt minimal support.

of the minimal frequency threshold) and NegPSpan is run with *maxgap* = 10 or *maxgap* = 0. Computation times show similar behaviors as in previous experiments: NegPSpan becomes as efficient as eNSP with a (weak) maxgap constraint. We can also notice that the minimal frequency of the positive partners does not impact eNSP computation times neither memory requirements.

The main result illustrated by Figure 3 is that NegPSpan consumes significantly less memory than eNSP. This comes from the depth-first search strategy used by NegPSpan which prevents from storing in memory many patterns. On the opposite, eNSP requires to keep in memory all frequent positive patterns and their occurrence lists. The lower the threshold, the more memory is required. When ς is low, then the number of positive patterns is huge. We can see that the memory consumption for $\varsigma = .2\sigma$ is 2 orders of magnitude larger than for $\varsigma = .8\sigma$. The strategy of eNSP appears to be practically intractable for large or dense datasets for which the number of ς-frequent positive patterns is very high and requires a huge amount of memory. This conclusion is consolidated by the experiment on the influence of vocabulary size in Appendix C.1.

5.1.4 Study of the pattern recall

In this section, we study and compare the recall of eNSP and NegPSpan on simulated data. The recall is the proportion of planted patterns in the dataset that are actually extracted by the mining algorithm. The higher the recall, the better the mining algorithm. NegPSpan is complete (*i.e.* a recall 1.0), but in case of the use of maxgap constraints, it may miss some of the planted patterns. Conversely, eNSP is incomplete due to the constraint on positive partners (that must be frequent). A negative pattern can be frequent without satisfying this constraint and could be missed by algorithm eNSP.

The recall is computed as the number of negative patterns extracted from 10 datasets of $n = 1000$ random sequences each in which at least 10 negative patterns were planted in at least $25,5\%$ of the sequences ($\lambda_{min} = 0.15$ and $\lambda_{max} = 0.2$) and with positive partners in 30% of the data.

The incompleteness of eNSP comes from the minimal support of positive partner which is $\varsigma \times d$. The incompleteness of NegPSpan comes from the maxgap constraint

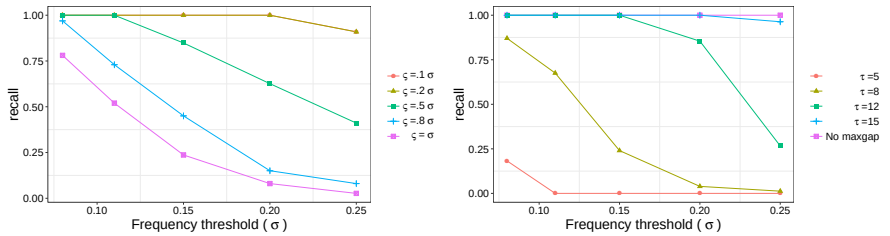

Fig. 4 Recall of algorithms eNSP (on the left) and NegPSpan (on the right) wrt rthe minimal threshold (σ) and parameters ς (minimal support of positive partners) or τ (maxgap constraint).

τ and also the minimal support threshold. In fact, the maxgap constraint reduces the number of occurrences of an NSP. If the minimal support threshold is sufficiently low, all the patterns are effectively extracted.

Figure 4 illustrates the recall wrt the minimal support threshold (σ). Each curve corresponds to one value of some algorithm parameter – maxgap (τ) for NegPSpan and the positive pattern threshold (ς) for eNSP – that impacts the recall.

We notice that:

- For eNSP and NegPSpan: the lower the support, the higher the recall but the longer the mining process. In fact, if the minimal support is low enough, it is always possible to find the planted pattern (recall is 1 with very low σ). But decreasing the support increases computational costs (see previous experiments).
- For eNSP: curves with lower ς are below, so the lower ς, the higher the recall. In fact, low ς values (*e.g.* $.1\sigma$) means that the constraint on the positive partner is very weak. But, this choice may lead to generate many positive partners that eNSP must analyse. Low ς values make the computation time increase and make the memory requirement explode.
- In addition, we can note that eNSP does not extract the complete set of patterns with a minimal threshold set to 25% where NegPSpan does. This means that eNSP requires to set the minimal support threshold lower than actually required by the dataset to reach the perfect recall of 1.
- For NegPSpan, curves with small maxgap value, τ, are below. The smaller τ, the lower the support of a pattern and the fewer patterns are frequent.
- Finally, curves with $\varsigma = .1\sigma$ or $\tau = 5$ exhibit a perfect recall whatever the minimal support between 1% and 25%. This means that NegPSpan extracts all the planted patterns. We can note that the recall could decrease for higher support. In fact, the generated datasets ensure that planted patterns occur in at least 25.5% of the sequences, if the minimal frequency threshold σ is above this value, some planted patterns may not be frequent at all.

Figure 5 compares the recall of eNSP with $\varsigma = 0.7\sigma$ and NegPSpan with $\tau = 10$. We can see on Figure 5 left that the recall curve of NegPSpan is above the recall curve of eNSP and, at the same time, on Figure 5, on the right side, that the computation times are lower. This demonstrates that NegPSpan extracts efficiently and accurately the planted negative patterns.

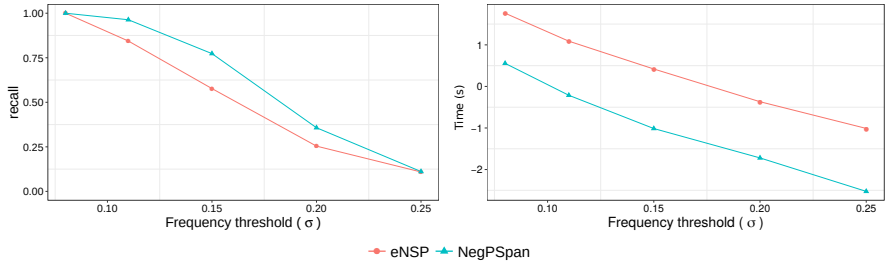

Fig. 5 Recall of algorithms eNSP (on the left) and NegPSpan (on the right) wrt the minimal threshold (σ) and parameters ς (ratio of positive patterns) or τ (maxgap constraint).

Let us now sum up the results on simulated datasets:

- NegPSpan extracts a correct and complete set of NSP while eNSP is correct but incomplete
- NegPSpan with $\tau = 10$ is more time-efficient than eNSP with $\varsigma = .7\sigma$ and has a better recall
- NegPSpan has a memory consumption several orders of magnitude lower than eNSP, and its memory consumption increases several orders of magnitude slower than eNSP with the average sequence length
- NegPSpan is more time-efficient than eNSP on dense datasets (small vocabulary size)

NegPSpan is then the best compromise for extracting efficiently a set of negative sequential patterns as much complete as possible.

These conclusions were drawn from synthetic datasets. Now, we supplement them with an analysis of results obtained from real datasets.

5.2 Experiments on real datasets

In this section, we present experiments on real datasets coming from the SPMF repository.[4] These datasets consist of click-streams or texts represented as sequences of items. For every dataset, we have computed the negative sequential patterns with a maximum length $l = 5$ and a minimal frequency threshold set to $\sigma = 5\%$. we set maxgap to $\tau = 10$ for NegPSpan and ς is set to $.7\sigma$ for eNSP. Table 3 provides the computation time, the memory consumption and the numbers of positive and negative extracted patterns. Note that the numbers of positive patterns for eNSP are given for ς threshold, *i.e.* the support threshold for positive partners used to generate negative patterns.

For the *sign* dataset, the execution has been stopped after $10mins$ to avoid running out of memory. The number of positive patterns extracted by eNSP considering the σ threshold is not equal to NegPSpan simply because of the maxgap constraint.

The results presented in Table 3 confirm the results from experiments on synthetic datasets. First, it highlights that NegPSpan requires significant less memory for mining

[4] `http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`

Table 3 Results on real datasets with settings $\sigma = 5\%$, $l = 5$, $\tau = 10$, $\varsigma = .7\sigma$. Bold faces highlight the lowest computation times or memory consumptions.

	NegPSpan				eNSP			
	time (s)	mem (kb)	#pos	#neg	time (s)	mem (kb)	#pos	#neg
Sign	**15.51**	**6,220**	348	1,357,278	349.84	13,901,600	1,190,642	1,257,177
Leviathan	**6.07**	**19,932**	110	39797	28.43	428,916	7,691	17,220
Bible	38.82	**68,944**	102	43,701	**27.38**	552,288	1,364	2,621
BMS1	**0.16**	**22,676**	5	0	0.18	34,272	8	7
BMS2	0.37	**39,704**	1	0	**0.35**	53,608	3	2
kosarak25k	0.92	**24,424**	23	409	**0.53**	43,124	50	51
MSNBC	**40.97**	**41,560**	613	56,418	41.44	808,744	2,441	5,439

every dataset. Second, NegPSpan outperforms eNSP for datasets having a long mean sequence length (*Sign*, *Leviathan*, and *MSNBC*). In case of the *Bible* dataset, the number of extracted patterns by eNSP is very low compared to NegPSpan due to the constraint on minimal frequency of positive partners. NegPSpan fails to identify negative patterns from *BMS* datasets, but eNSP extracts few negative patterns. It can be explained by the maxgap constraint for NegPSpan that lowers the support of the few eNSP patterns below the frequency threshold.

6 Case studies

In this section, we presents the use of NSPs on three case studies that show both the wide range of applications of NSPs and some benefits of NegPSpan for extracting NSPs. It presents the output of the algorithms and analyze them. The computational performances of the algorithms on these case studies datasets are presented in Appendix C.3.

First, we present briefly some outputs on the classical market basket dataset, then we present a complete study of care pathway data to compare patterns extracted by eNSP and NegPSpan, and finally we apply NSP mining for customer relationship management purpose and we present how to take decisions with extracted NSPs.

6.1 Instacart data

Instacart is an online grocery service. It has published a dataset containing information on 3 million grocery orders from more than 200,000 users from 2017.[5] The dataset contains information on what products users purchased, the sequence they bought them in and the amount of time between Instacart orders. The number of items is above $50,000$ with a lot of items that are rarely bought. The objective of Instacart was to foster research on algorithms that can predict what items customers will buy or may be interested in.

[5] The Instacart dataset is available on Kaggle: `https://www.kaggle.com/c/instacart-market-basket-analysis`

The motivation for extracting negative sequential patterns from purchase data is to identify which products are surprisingly absent from sequences of orders. A product is surprisingly absent when it is absent but the analyst assumes it should not. It is the case when retailers assume that some products are very correlated. For instance, customers that bought breakfast cereal without buying milk at any time could be surprising. This information can help retailers to better understand customer habits and to suggest commercial actions to increase sales. For instance, it could be profitable to advertise some milk products to the above customers.

In these experiments, NSPs were extracted with two minimal frequency thresholds: 0.5% and 1%. The vocabulary size requires to set such very low thresholds to identify non-trivial patterns, but such low thresholds prevent eNSP from processing the whole dataset. For this reason, we analyzed only the first $50,000$ customers. The main parameter values of each algorithm were set to $\tau \in \{2,3,4,6,8\}$ and $\varsigma \in \{0.4,0.5,0.6,0.7\}$.

Let us now analyze the set of patterns extracted with a minimal support $\sigma = 1\%$ and the settings $\varsigma = 0.4$ for eNSP and $\tau = 4$ for NegPSpan. eNSP extracted $6,499$ patterns, while NegPSpan extracted $8,096$ patterns. With higher values for τ, the number of NSPs makes the comparison difficult.

Despite the low threshold, a large part of the patterns extracted by NegPSpan falls into one of the following shapes of patterns (X and Y represent itemsets):

- the pattern is positive (contains no negation),
- $\boldsymbol{p_1^I} = \langle 13176\ \neg X\ 13176\rangle$ where 13176 is `bag` (86 patterns)
- $\boldsymbol{p_2^I} = \langle 21903\ \neg X\ 21903\rangle$ where 21903 is `organic baby spinach ravioli` (86 patterns)
- $\boldsymbol{p_3^I} = \langle 24852\ \neg X\ 24852\ \neg Y\ 24852\rangle$ where 24852 is `banana` ($7,830$ patterns)

The patterns of type $\boldsymbol{p_3^I}$ represent more than 90% of the extracted patterns. Among these NSPs, some seems interesting because they involve different fruits. It is interesting to identify customers purchasing lots of bananas and who do not purchase any other fruit during the same period. Nevertheless, most of these patterns involves products that are not fruits, and are not really meaningful. In this case, the positive partner constraint of eNSP would discard the patterns whose positive pattern, $\langle 24852\ \neg X\ 24852\ \neg Y\ 24852\rangle$, is not ς-frequent. And we can expect that eNSP would extract a small set of meaningful patterns, but eNSP extracts a similar amount of patterns ($6,499$).

eNSP extracts similar patterns but also a collection of more diverse patterns. The diversity comes from the absence of gap constraints. The embedding of a sequential pattern without gap constraint may involve purchases that are very distant in the sequence of purchases. This case study questions the significance of such patterns. Retailers are in reality not interested in relationships involving client purchases that are very distant in time. In some way, we can say that the absence of gap constraints makes eNSP over-estimates the support of patterns.

The conclusion of this case study is that eNSP has better computation time results on this dataset, but it did not succeed in processing the whole dataset. NegPSpan is time-consuming but fit in memory even on a large and sparse dataset. The computation

costs of NegPSpan comes from the enumeration of patterns with the same positive shape: it combines positive patterns with all possible negative items that could make them frequent. At the end, the patterns extracted by each algorithm are difficult to interpret. The patterns extracted by eNSP may be somehow more significant thanks to the positive partner constraint. But without gap constraint, most of the extracted NSPs are irrelevant for retailers.

To process such large but sparse dataset, it would be interesting to benefit from the best of the two approaches: a relevant NSP semantics including gap constraints and a reasonable number of diverse NSPs. Thanks to the anti-monotonicity property of its semantics for negative patterns, NegPSpan could be adapted to fit in some recent frameworks for mining diverse patterns efficiently (Bosc et al., 2018). Alternatively, the frequency constraint on positive partners could be applied as a post-processing step to prevent NegPSpan from extracting useless patterns. It would not be efficient, but could provide more interpretable results for this case study.

6.2 Care pathway analysis

This section presents the use of NSPs for analyzing epileptic patient care pathways. Recent studies suggest that medication changes may be associated with epileptic seizures for patients with long term treatment with anti-epileptic medication (Polard et al., 2015). NSP mining algorithms are used to extract patterns of drugs deliveries that may inform the suppression of a drug from a patient treatment. In (Dauxais et al., 2017), we studied discriminant temporal patterns but it does not explicitly extract the information about medication absence as a possible explanation of epiletic seizures.

Our dataset was obtained from the french insurance database (Moulis et al., 2015). 8,379 epileptic patients were identified by their hospitalization related to an epileptic seizure. For each patient, we built a sequence of drugs deliveries within the 90 days before the first epileptic seizure. For each drug delivery, an event is a tuple (m, g) where m is the ATC[6] code of the active molecule, $g \in \{0, 1\}$ is the brand-name (0) vs generic (1) status of the drug. For the sake of readability, an identifier is assigned to each event tuple. Table 4 gives the mapping between event identifiers and event tuples. For example, the following sequence representing a sequence of drug deliveries with a switch from generic to brand-name *valproic acid*:

$$\langle\ (\mathit{valproic\ acid}, \mathit{generic})\ \ ((\mathit{valproic\ acid}, \mathit{generic}), (\mathit{paracetamol}, \mathit{generic}))\ \ \ldots$$
$$(\mathit{valproic\ acid}, \mathit{generic})\ (\mathit{valproic\ acid}, \mathit{brand})\ (\mathit{valproic\ acid}, \mathit{brand})\ \rangle$$

is translated into the sequence:

$$\langle 383\ (383, 86)\ 383\ 114\ 114 \rangle$$

The dataset contains 251,872 events over 7,180 different drugs. The mean length of a sequence is 7.89±8.44 itemsets. The sequence length variance is high. This is due to the heterogeneous nature of care pathways. Some of them represent complex therapies

[6] ATC: Anatomical Therapeutic Chemical Classification System is a drug classification system that classifies the active ingredients of drugs.

Table 4 Selection of events identifiers that are used in the discussed patterns.

Event tuple	Event id
(*levetiracetam*, generic)	7
(*paracetamol*, generic)	86
(*zolpidem*, generic)	112
(*valproic acid*, brand)	114
(*clobazam*, generic)	115
(*zopiclone*, generic)	151
(*phenobarbital*, generic)	158
(*valproic acid*, generic)	383

Table 5 Patterns involving *valproic acid* switches with their supports computed by eNSP and NegPSpan. Empty supports indicate that the pattern has not been extracted. Drug names associated with identifiers in the figures are given within the text.

Pattern	support eNSP	support NegPSpan
$\boldsymbol{p}_1^{\boldsymbol{CP}} = \langle 383\ \neg(86, 383)\ 383 \rangle$	1,579	
$\boldsymbol{p}_2^{\boldsymbol{CP}} = \langle 383\ \neg 86\ 383 \rangle$	1,251	1,243
$\boldsymbol{p}_3^{\boldsymbol{CP}} = \langle 383\ \neg 112\ 383 \rangle$	1,610	
$\boldsymbol{p}_4^{\boldsymbol{CP}} = \langle 383\ \neg 114\ 383 \rangle$	1,543	1,232
$\boldsymbol{p}_5^{\boldsymbol{CP}} = \langle 383\ \neg 115\ 383 \rangle$	1,568	1,236
$\boldsymbol{p}_6^{\boldsymbol{CP}} = \langle 383\ \neg 151\ 383 \rangle$	1,611	
$\boldsymbol{p}_7^{\boldsymbol{CP}} = \langle 383\ \neg 158\ 383 \rangle$	1,605	
$\boldsymbol{p}_8^{\boldsymbol{CP}} = \langle 383\ \neg 7\ 383 \rangle$		1,243

involving the consumption of many different drugs while others are simple cases consisting of few deliveries of anti-epileptic drugs. In the french health system, drug deliveries are renewed every month. For epileptic patients that require a continuous medication, we expect to have mostly sequences with three or four deliveries of anti-epileptic drugs. Other items are additional medical treatments that are not necessarily related to epilepsy.

Let us now compare the pattern sets extracted by eNSP and NegPSpan. In fact, expressible pattern constraints are different (*maxgap* constraints for NegPSpan and minimal support of positive partners for eNSP) and the extracted pattern sets are different. In this qualitative experiment, the parameters is set to $\sigma = 14.3\%$ $(1,200$ sequences), $l = 3$ (maximal pattern length), $\tau = 3$ for NegPSpan and $\varsigma = .1 \times \sigma$ the minimal support for positive partners for eNSP. eNSP extracted 1,120 patterns and NegPSpan only 10 patterns (including positive and negative patterns). Due to a low ς threshold, many positive patterns were extracted by eNSP leading to generate a lot of patterns having a single negated item. In this analysis, we pay attention to the specialty of *valproic acid* which exists in generic form (event 383) or brand-named form (event 114). Table 5 presents all patterns starting and finishing with event 383. Other events correspond to alternative anti-epileptic drugs (*levetiracetam*, *phenobarbital*) or psycholeptic drugs (*zolpidem*, *clobazam*, *zopiclone*) except *paracetamol*.

It is interesting to note that only 3 patterns ($\boldsymbol{p_2^{CP}}$, $\boldsymbol{p_4^{CP}}$ and $\boldsymbol{p_5^{CP}}$) are extracted by both algorithms. Their supports are lower with NegPSpan because of the *maxgap* constraint. This constraint also explains that patterns $\boldsymbol{p_3^{CP}}$ and $\boldsymbol{p_6^{CP}}$ are not extracted by NegPSpan. These patterns illustrate that in some cases, the patterns extracted by eNSP may not be really interesting because they involve distant events in the sequence. Pattern $\boldsymbol{p_1^{CP}}$ is not extracted by NegPSpan due to the syntactic constraints. On the opposite, NegPSpan extracts patterns that eNSP misses. For instance, pattern $\boldsymbol{p_8^{CP}}$ is not extracted by eNSP because its positive partner, $\langle 383\ 7\ 383\rangle$, is not frequent. In this case, it leads eNSP to miss a potentially interesting pattern involving two anti-epileptic drugs.

Next, we examine a particular case to illustrate that NSPs are essential to have true insights about the meaning of frequent positive patterns. Remind that our objective is to conclude about the potential link between switches in epileptic drug deliveries and epileptic seizures. Our point is that positive patterns can be misinterpreted, because they only show what actually happens. Negative patterns may avoid some misinterpretation.

For this experiment, we changed the NegPSpan settings to focus on patterns involving a switch from the generic form to the brand-named form of *valproic acid*. The settings were $\sigma = 1.2\%$, $l = 3$ and $\tau = 5$. The only two frequent positive patterns are $\langle 114\ 383\ 114\rangle$ and $\langle 114\ 114\rangle$. In our experiment, it means that these patterns occur frequently in patient having an epileptic seizure. Because $\langle 114\ 114\rangle$ is a positive pattern, it does not means that there is no switches in sequences that hold this pattern. Thus, positive patterns are not conclusive about the impact of a switch from 114 to 383 on epileptic seizures.

NegPSpan also extracts the pattern $\langle 114\ \neg 383\ 114\rangle$, which specifies that the absence of a switch is frequent for patients having epileptic seizure. This time, it sheds light on the fact that switches and non-switches from generic form to brand-named form of *valproic acid* are both frequent behaviors in care pathways of patients having epileptic seizure. We can conclude that pattern mining techniques does not establish a statistical link between epileptic drug switches and epileptic seizure. This result is consistent with (Polard et al., 2015).

6.3 Customer Relationship Management

Customer Relationship Management (CRM) refers to a set of tools that is used for managing the interactions between a company and its customers. As its main objective is to maintain regular customers, by developing long-term relationships with them, and to acquire new ones, CRM is being used by any organization to identify the problems of customers and to improve the consistency with them. To fullfill the objectives above, it is required from the company to perform customer relationship analysis in order to meet customer needs and improve customer services. However, due to the increased number of business data generated by companies during the last years, the most useful and valuable information required for customer analysis is often hidden in the large CRM databases and is not easily accessible.

Table 6 Frequent contact reasons in the CRM database.

Id	Name	Frequency (%)
1CTR	Contract subscription	30.73%
2VIE	Contract modification	20.67%
1RES	Contract termination	12.42%
6DIS	Relationship with supplier	6.62%
8DOS	File handling	6.47%
3SER	Advices and services	4.56%
4FAC	Billing	4.08%
5REC	Recovery	3.04%
Unknown	Unknown	2.4%
3RCTR	Contract	1.9%

According to Ngai et al. (2009), a common set of supporting tools (statistical analysis, cluster analysis, probability theory, artificial neural networks, etc) widely used from companies for making relevant CRM decisions is data mining, which are good at extracting and identifying useful knowledge from large customer databases. Specific data mining techniques like association rules mining or sequential pattern mining are especially useful for analyzing customer data. Frequent sequential pattern mining is particularly useful in commercial applications, as it can be used to discover customers' behavioral and purchasing patterns over time (Mallick et al., 2013).

The negative events are meaningful in CRM to inform stakeholders about which actions was absent in the customer interactions. An absent interaction may be an action missed by customer services. This way, NSPs enable decision makers to identify potential improvements in the customer relationship management.

In this case study, we illustrate how some extracted NSPs can be interpreted to make decisions for improving customer relationship management. We analyze sequences corresponding to the interactions of customers with the services of an electricity supplier in order to prevent contract cancellations.

The dataset has $375,142$ customers (one sequence per customer) with a vocabulary of 93 different items. Items are tags that label the contact reasons (for instance invoicing procedures, technical issues or commercial purposes). Table 6 presents the 16 items that occurs more than 1% in the dataset. The dataset does not specify whether a contact was initiated by the company or by the customer. A contact between the company and the customer can be labeled with several tags. The average number of tags per contact is 1.21±0.34. This means that most of the itemsets hold only 1 item (a contact has only one contact reason), but it raises up to 10 tags for a unique contact. The average length of the sequences is 2.52±7.26: most of the sequences are short (2 or 3 contacts within a period of one year) but some customers have much more interactions with the company.

The frequent NSPs is run with a minimal support threshold $\sigma = 0.5\%$ (at least 1,875 occurrences of an NSP), $\varsigma = 0.4\sigma$ for eNSP and $\tau = 4$ NegPSpan. NegPSpan (resp. eNSP) extracts $18,652$ (resp. $6,746$) patterns. The two sets of extracted NSPs have $1,463$ patterns in common. So, NegPSpan extracts more NSPs than eNSP does

Table 7 Selection of patterns involving *3SER* (on the left) or *2VIE* (on the right) negative items with their supports computed by NegPSpan but not by eNSP.

Pattern	support NegPSpan
$\boldsymbol{p_1^{CRM}} = \langle 6DIS\ \neg 3SER\ 1RES \rangle$	5,352
$\boldsymbol{p_2^{CRM}} = \langle 1RES\ \neg 3SER\ 1RES \rangle$	5,265
$\boldsymbol{p_3^{CRM}} = \langle (1CTR, 8DOS)\ \neg 3SER\ 1RES \rangle$	6,597
$\boldsymbol{p_4^{CRM}} = \langle (1CTR, 2VIE, 8DOS)\ \neg 3SER\ 1RES \rangle$	1,946
$\boldsymbol{p_5^{CRM}} = \langle (3SER, 8DOS)\ \neg 2VIE\ 1RES \rangle$	3,032
$\boldsymbol{p_6^{CRM}} = \langle (1CTR, 3SER), \neg 2VIE\ 1RES \rangle$	2,258
$\boldsymbol{p_7^{CRM}} = \langle 1CTR\ 2VIE\ 8DOS\ \neg 2VIE\ 1RES \rangle$	1,912

in previous case studies but it does it faster. NegPSpan requires $113s$ computation time while eNSP requires $786s$. In the following, we do not compare the set of extracted patterns, but we focus our attention on the interpretation of patterns extracted by NegPSpan.

Among the $18,652$ patterns extracted by NegPSpan, we selected only the patterns ending by the contract termination item ($1RES$) and which have only one negative event. There are 281 patterns satisfying these conditions. This amount of patterns is still too large to be analyzed. Then, the practical objective of this case study suggests to focus our interest on actionable patterns. The information extracted aims at suggesting actions to improve customer relation management. The customer services can impact customer decisions mainly by providing services (*3SER* item) or by modifying the customer contract (*2VIE* item). For this reason, we look at the NSPs that involve the *3SER* item or the *2VIE* item in the negative part.

Table 7 illustrates some interesting patterns extracted only by NegPSpan. The complete lists of selected patterns is in Appendix D: 20 NSP are extracted by both eNSP and NegPSpan, 18 are extracted only by eNSP and 20 are extracted only by NegPSpan.

Patterns $\boldsymbol{p_1^{CRM}}$, $\boldsymbol{p_2^{CRM}}$ and $\boldsymbol{p_3^{CRM}}$ are interesting because they are almost frequent but are not extracted by eNSP. It seems that the support of their positive partner does not exceed $\varsigma = 0.4\sigma$ (750 sequences). Nonetheless, the following comments show that they are insightful and can provide interesting information about the behavior of customers:

- $\boldsymbol{p_1^{CRM}}$ are customers that contacted the company with a concern about distribution of the electricity ($6DIS$) and that did not get any advice or service ($3SER$). For the $5,368$ customers, it leads to a contract termination. It can be notice that customers that experience this pattern amounts to about 2% but contacts for the $6DIS$ purpose amounts to 6.62% of the contact reasons. Thus, it is a relatively frequent pathway for customers having this contact reason. This pattern highlights that customer services have to provide advice or services to prevent customer churn in this case.
- $\boldsymbol{p_2^{CRM}}$ are customers who contacted the company about contract cancellation but were not contacted for advice or services proposal afterwards, and then contacted again the customer services about contract cancellation (maybe to finalize their

contract cancellation procedure). This pattern identifies customers undergoing a potential failure in the customer relationship management: if a customer contacts the customer services about contract cancellation, it seems quite normal to investigate this customer's situation with a $3SER$ contact. This pattern shows that $5,265$ customers were not contacted back after asking for cancellation information.

- $\boldsymbol{p}_3^{CRM}$ (and its similar pattern $\boldsymbol{p}_4^{CRM}$) represents customers who subscribed to a contract and finally canceled their contract without having been contacted for advice. The maximum gap constraint that have been imposed by NegPSpan, $\tau = 4$, indicates that the customer relationship was faulty: for these customers there are at most three interactions between their contract subscription and their contract cancellation.

Patterns $\boldsymbol{p_5^{CRM}}$ and $\boldsymbol{p_6^{CRM}}$ involve absence of contract modifications ($2VIE$) because advice or service offers do not necessarily meet customers' needs. In these cases, they do not modify their contract. A possible explanation of these two patterns is that customers received some advice ($3SER$) but did not modify their contracts ($2VIE$) and finally churned. For the customer services, these patterns can be used to identify which advice or services have been offer to these particular customers and to assess whether an alternative advice would not lead to a contract termination.

Pattern $\boldsymbol{p_7^{CRM}}$ is relatively frequent regarding its length. We interpret this pattern as a failure in the file handling procedure. The customer subscribed to a contract, then she/he had a contact to modify it. This modification may not have been correctly taken into consideration and the customer had another contact about her/his file handling. Without effective contract modification after the last contact, the customer churned.

Let us now conclude about the three case studies. The first case study is a dataset on which both algorithms have difficulties to extract meaningful NSPs. The case study on medical care pathways highlights two important conclusions about the interpretation of sequential patterns. First, NSPs provide meaningful information about sequential patterns and may prevent from their misinterpretation. Second, the different semantics adopted by the two algorithms lead to different sets of extracted pattern. Without *maxgap* constraints, eNSP outputs some patterns whose frequency is over-estimated compared to the frequency computed by NegPSpan. But, eNSP misses some possibly important patterns at the boundary of the frequent NSPs. Finally, the third case study illustrates some complex interpretations of NSPs that would be useful for further analysis of customer relationship data and management procedures.

7 Related work

Kamepalli et al. (2014) and more recently Wang and Cao (2019) provide surveys of the approaches proposed for mining negative patterns. The three most significant algorithms appear to be PNSP (Hsueh et al., 2008), NegGSP (Zheng et al., 2009) and eNSP (Cao et al., 2016). We briefly review each of them in the following paragraphs.

PNSP (Positive and Negative Sequential Patterns mining) (Hsueh et al., 2008) is the first algorithm proposed for mining full negative sequential patterns where negative itemsets are not only located at the end of the pattern. PNSP extends algorithm

GSP (Srikant and Agrawal, 1996) to cope with mining negative sequential patterns. PNSP consists of three steps: i) use algorithm GSP for mining frequent positive sequential patterns, ii) preselect negative sequential itemsets — for PNSP, negative itemsets must not be too infrequent (should have a support lower than a threshold *miss_freq*) — iii) generate candidate negative sequences levelwise and scan the sequence dataset again to compute the support of these candidates and prune the search when the candidate is infrequent. This algorithm is incomplete: the second parameter reduces the set of potential negative itemsets. Moreover, the pruning strategy of PNSP is not correct (Zheng et al., 2009) and PNSP misses potentially frequent negative patterns.

Zheng et al. (2009) also proposed a negative version of algorithm GSP, called NegGSP, to extract negative sequential patterns. They showed that traditional Apriori-based negative pattern mining algorithms relying on support anti-monotonicity have two main problems. The first one is that the Apriori principle does not apply to negative sequential patterns. They gave an example of sequence that is frequent even if one of its sub-sequence is not frequent. The second problem has to do with the efficiency and the effectiveness of finding frequent patterns due to a vast candidate space. Their solution was to prune the search space using the support anti-monotonicity over positive parts. This pruning strategy is correct but incomplete and it is not really efficient considering the huge number of remaining candidates whose support has to be evaluated. To improve the efficiency of their approach, the authors proposed an incomplete heuristic search based on Genetic Algorithm to find negative sequential patterns (Zheng et al., 2010).

eNSP (efficient NSP) has been recently proposed by Cao et al. (2016). It identifies NSPs by computing only frequent positive sequential patterns and deducing negative sequential patterns from positive patterns. Precisely, Cao et *al.* showed that the support of some negative pattern can be computed by arithmetic operations on the support of its positive sub-patterns, thus avoiding additional sequence dataset scans to compute the support of negative patterns. However, this necessitates to store all the (positive) sequential patterns with their set of covered sequences (tid-lists) which may be impossible in case of big dense datasets and low minimal support thresholds. This approach makes the algorithm more efficient but it hides some restrictive constraints on the extracted patterns. First, a frequent negative pattern whose so-called positive partner (the pattern where all negative events have been switched to positive) is not frequent will not be extracted. Second, every occurrence of a negative pattern in a sequence should satisfy absence constraints. We call this *strong absence semantics* (see Section 3.2). These features lead eNSP to extract fewer patterns than previous approaches. In some practical applications, eNSP may miss potentially interesting negative patterns from the dataset.

The first constraint has been partly tackled by Dong et *al.* with algorithm eNSPFI, an extension of eNSP which mines NSPs from frequent and some infrequent positive sequential patterns from the negative border (Gong et al., 2017). E-msNSP (Xu et al., 2017b) is another extension of eNSP which uses multiple minimum supports: an NSP is frequent if its support is greater than a local minimal support threshold computed from the content of the pattern and not a global threshold as in classical approaches. A threshold is associated with each item, and the minimal support of a pattern is

defined from the most constrained item it contains. Such kind of adaptive support prevents from extracting some useless patterns still keeping the pattern support anti-monotonic. The same authors also proposed high utility negative sequential patterns based on the same principles (Xu et al., 2017a) and applied the framework on smart city data (Xu et al., 2018). An alternative approach has been proposed by Lin et al. (2016) consisting in mining high-utility itemsets with negative unit profits but it was not applied on sequential patterns. It is worth noting that this algorithm relies basically on the same principle as eNSP and so, presents the same drawbacks, heavy memory requirements, strong absence semantics for negation. F-NSP+ (Dong et al., 2018b) extends the eNSP algorithm to use bitmap representations of itemsets. Using bitmap representations enables to speed up algorithm eNSP, thanks to very efficient set operations on bitmaps. Algorithm F-NSP has a poor memory usage, while F-NSP+, which adapts the bitmap size to the dataset, requires slightly less memory.

e-RNSP (Dong et al., 2018a) extracts repetition NSP (RNSP). A RNSP is a NSP for which the support takes into account the repetition of the pattern in a sequence of the sequence dataset. Thus, it is not really a new pattern domain but more a new definition of support measure. The e-RNSP uses the same strategy as eNSP with a hash map to store all occurrences of the positive patterns (all repetition must be stored). The experiments have been conducted on small datasets with a maximum number of $10k$ sequences (of at most 15 itemsets). Unsurprisingly, the computation times of e-RNSP are slightly above e-RNSP. The memory consumption of the approach is not discussed but it is at most as expensive as eNSP.

SAPNSP (Liu et al., 2015) tackles the problem of large amount of patterns by selecting frequent negative and positive patterns that are actionable. Patterns are actionable if they conform to *special rules*.

NegI-NSP (Qiu et al., 2017) proposes additional syntactic constraints on negative itemsets and uses the same strategy as e-NSP.

To conclude this state of the art section, we provide in Table 8 a comparison of several negative sequential pattern mining approaches wrt several features investigated in section 3. It is also important to emphasize that one semantics is "more correct" than another one. Its relevance depends on the information the data scientists want to capture in datasets, and the nature of the data at hand. In this work, one of our objectives is to provide a sound and insightful framework for negative patterns to enable users to choose the tool to use and to make this choice according to the semantics of the negation they want to use. Execution time is obviously an important choice criteria but it must be consistent with semantical choices to provide interesting, intuitive and sound results.

8 Conclusion and perspectives

This article has investigated negative sequential pattern mining (NSP). It highlights that state of the art algorithms do not extract the same patterns, not only depending on their syntax and algorithm specificities, but also depending on the semantical choices adopted for each of them. In this article, we have proposed definitions that clarify the

Table 8 Comparison of negative pattern mining proposals. Optional constraints are specified in Italic. Question marks indicates that the article does not clearly state their semantic.

	PNSP (Hsueh et al., 2008)	**NegGSP** (Zheng et al., 2009)	**eNSP** (Cao et al., 2016)	NegPSpan
Negative elements	itemsets	items?	itemsets	itemsets
Itemsets	$\not\subseteq_G$?	$\not\subseteq_G$	$\not\subseteq_D$	$\not\subseteq_D$
Embeddings	strict	strict?	strict	strict/*soft*
Occurrences	weak	weak	strong	weak
Constraints on negative itemsets	not too infrequent ($supp \leqslant less\_freq$)	frequent items	positive partner is frequent	frequent items, *bounded size*
Global constraints on patterns			positive part is frequent (second greater threshold)	*maxspan*, *maxgap*

negation semantics encountered in the literature. We have shown that the support of NSP depends on the semantics of itemset non-inclusion, two possible alternatives for considering negation of itemsets and two ways for considering multiple embeddings in a sequence. So, we could point out the limits of the state of the art algorithm eNSP that imposes a minimum support for positive partners and that is not able to deal with embedding constraints, and more especially *maxgap* constraints.

We have proposed NegPSpan, a new algorithm for mining negative sequential patterns that overcomes these limitations. The experiments show that NegPSpan is more efficient than eNSP on datasets with medium-long sequences (more than 20 itemsets) even when weak *maxgap* constraints are applied and that it prevents from missing possibly interesting patterns.

In addition, NegPSpan is based on well-founded theoretical principles that makes possible to extend it to the extraction of closed or maximal patterns to reduce the number of extracted patterns even more.

Acknowledgements The authors would like to thank REPERES Team from Rennes University Hospital for spending time to discuss our case study results. We also would like to thanks M. Boumghar, L. Pierre and D. Lagarde for raising interesting issues and providing the dataset about customer relationship management. Finally, we would also like to thanks the reviewers for their insightful comments.

References

Bosc G, Boulicaut JF, Raïssi C, Kaytoue M (2018) Anytime discovery of a diverse set of patterns with monte carlo tree search. Data Mining and Knowledge Discovery 32(3):604–650

Cao L, Yu PS, Kumar V (2015) Nonoccurring behavior analytics: A new area. Intelligent Systems 30(6):4–11

Cao L, Dong X, Zheng Z (2016) e-NSP: Efficient negative sequential pattern mining. Artificial Intelligence 235:156–182

Dauxais Y, Guyet T, Gross-Amblard D, Happe A (2017) Discriminant chronicles mining - application to care pathways analytics. In: Proceedings of 16th Conference on Artificial Intelligence in Medicine (AIME), pp 234–244

Dong X, Gong Y, Cao L (2018a) e-RNSP: An efficient method for mining repetition negative sequential patterns. IEEE Transactions on Cybernetics pp 1–13, DOI 10.1109/TCYB.2018.2869907

Dong X, Gong Y, Cao L (2018b) F-NSP+: A fast negative sequential patterns mining method with self-adaptive data storage. Pattern Recognition 84:13–27

Giannotti F, Nanni M, Pedreschi D (2006) Efficient mining of temporally annotated sequences. In: Proceedings of the SIAM International Conference on Data Mining, pp 348–359

Gong Y, Xu T, Dong X, Lv G (2017) e-NSPFI: Efficient mining negative sequential pattern from both frequent and infrequent positive sequential patterns. International Journal of Pattern Recognition and Artificial Intelligence 31(02):1750002

Han J, Pei J, Mortazavi-Asl B, Chen Q, Dayal U, Hsu MC (2000) FreeSpan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth international conference on Knowledge discovery and data mining (SIGKDD), pp 355–359

Hsueh SC, Lin MY, Chen CL (2008) Mining negative sequential patterns for e-commerce recommendations. In: Proceedings of Asia-Pacific Services Computing Conference, pp 1213–1218

Kamepalli S, Sekhara R, Kurra R (2014) Frequent negative sequential patterns – a survey. International Journal of Computer Engineering and Technology 5, 3:115–121

Lin JCW, Fournier-Viger P, Gan W (2016) FHN: An efficient algorithm for mining high-utility itemsets with negative unit profits. Knowledge-Based Systems 111:283–298

Liu C, Dong X, Li C, Li Y (2015) SAPNSP: Select actionable positive and negative sequential patterns based on a contribution metric. In: Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery, pp 811–815

Mallick B, Garg D, Grover PS (2013) CRM customer value based on constrained sequential pattern mining. International Journal of Computer Applications 64(9):21–29

Mooney CH, Roddick JF (2013) Sequential pattern mining – approaches and algorithms. ACM Computing Survey 45(2):1–39

Moulis G, Lapeyre-Mestre M, Palmaro A, Pugnet G, Montastruc JL, Sailler L (2015) French health insurance databases: What interest for medical research? La Revue de Médecine Interne 36:411–417

Negrevergne B, Guns T (2015) Constraint-based sequence mining using constraint programming. In: Michel L (ed) Proceedings of the conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR), Springer International Publishing, pp 288–305

Ngai E, Xiu L, Chau D (2009) Application of data mining techniques in customer relationship management: A literature review and classification. Expert Systems with Applications 36(2, Part 2):2592 – 2602

Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE Transactions on knowledge and data engineering 16(11):1424–1440

Pei J, Han J, Wang W (2007) Constraint-based Sequential Pattern Mining: The Pattern-growth Methods. Journal of Intelligent Information Systems 28(2):133–160

Polard E, Nowak E, Happe A, Biraben A, Oger E (2015) Brand name to generic substitution of antiepileptic drugs does not lead to seizure-related hospitalization: a population-based case-crossover study. Pharmacoepidemiology and drug safety 24:1161–1169

Qiu P, Zhao L, Dong X (2017) NegI-NSP: Negative sequential pattern mining based on loose constraints. In: Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON), pp 3419–3425

Srikant R, Agrawal R (1996) Mining sequential patterns: Generalizations and performance improvements. In: Proceedings of the International Conference on Extending Database Technology (EDBT), Springer, pp 1–17

Wang W, Cao L (2019) Negative sequences analysis: A review. ACM Computing Survey 52

Xu T, Dong X, Xu J, Dong X (2017a) Mining high utility sequential patterns with negative item values. International Journal of Pattern Recognition and Artificial Intelligence 31(10):1750035

Xu T, Dong X, Xu J, Gong Y (2017b) E-msNSP: Efficient negative sequential patterns mining based on multiple minimum supports. International Journal of Pattern Recognition and Artificial Intelligence 31(02):1750003

Xu T, Li T, Dong X (2018) Efficient high utility negative sequential patterns mining in smart campus. IEEE Access 6:23839–23847

Zaki MJ (2001) SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning 42(1/2):31–60

Zheng Z, Zhao Y, Zuo Z, Cao L (2009) Negative-GSP: An efficient method for mining negative sequential patterns. In: Proceedings of the Australasian Data Mining Conference, pp 63–67

Zheng Z, Zhao Y, Zuo Z, Cao L (2010) An efficient GA-based algorithm for mining negative sequential patterns. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Springer, pp 262–273

A Proofs

Proof (Proof of Proposition 2) Let $\boldsymbol{s} = \langle s_1 \ \cdots \ s_n \rangle$ be a sequence and $\boldsymbol{p} = \langle p_1 \ \cdots \ p_m \rangle$ be a negative sequential pattern. Let $\boldsymbol{e} = (e_i)_{i\in[m]} \in [n]^m$ be a soft-embedding of pattern $\boldsymbol{p}$ in sequence $\boldsymbol{s}$. Then, the definition matches the one for strict-embedding if p_i is positive. If p_i is negative then $\forall j \in [e_{i-1}+1, e_{i+1}-1],\ p_i \not\subseteq_D s_j$, *i.e.* $\forall j \in [e_{i-1}+1, e_{i+1}-1],\ \forall \alpha \in p_i,\ \alpha \notin s_j$ and then $\forall \alpha \in p_i,\ \forall j \in [e_{i-1}+1, e_{i+1}-1],\ \alpha \notin s_j$. Thus, it implies that $\forall \alpha \in p_i,\ \alpha \notin \bigcup_{j\in[e_{i-1}+1,e_{i+1}-1]} s_j$, *i.e.* by definition, $p_i \not\subseteq_D \bigcup_{j\in[e_{i-1}+1,e_{i+1}-1]} s_j$.

The exact same reasoning is done in the reverse way to prove the equivalence.

Proof (Proof of Proposition 3 (Anti-monotonicity of NSP)) Let $\boldsymbol{p} = \langle p_1 \ \neg q_1 \ p_2 \ \neg q_2 \ \cdots p_{k-1} \ \neg q_{k-1} \ p_k \rangle$ and $\boldsymbol{p'} = \langle p'_1 \ \neg q'_1 \ p'_2 \ \neg q'_2 \ \cdots p'_{k'-1} \ \neg q'_{k'-1} \ p'_{k'} \rangle$ be two NSP s.t. $\boldsymbol{p} \lhd \boldsymbol{p'}$. By definition we have that $k \leq k'$.

To prove the anti-monotonicity, we prove that any embedding $(e_i)_{i\in[k]}$ of $\boldsymbol{p'}$ in a sequence $\boldsymbol{s}$ generates an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.

Let $\boldsymbol{s} = \langle s_1 \ \cdots \ s_n \rangle$ be a sequence s.t. $\boldsymbol{p'} \preceq \boldsymbol{s}$, *i.e.* there exists an embedding $(e_i)_{i\in[k']}$:

- $\forall i,\ e_{i+1} > e_i$,
- $\forall i,\ p'_i \subseteq s_{e_i}$,
- $\forall j \in [e_i+1, e_{i+1}-1],\ q'_i \not\subseteq_D s_{e_j}$

By definitions of $\lhd$ and embedding,

(i) $\forall i \in [k],\ p_i \subseteq p'_i \subseteq s_{e_i}$, ($e_i$ exists because $k \leq k'$)
(ii) $\forall i \in [k-1],\ \forall j \in [e_i+1, e_{i+1}-1],\ q'_j \not\subseteq_D s_{e_i}$, and thus $q_j \not\subseteq_D s_{e_i}$ (because $\not\subseteq_D$ is anti-monotone and $q_i \subseteq q'i$)

This means that $(e_i)_{i\in[k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.

Proof (Proof of Proposition 4 (Anti-monotonicity of the support of Constrained NSP)) The proof of this property is similar to the proof of Proposition 3.

Let $\boldsymbol{p} = \langle p_1 \ \neg q_1 \ p_2 \ \neg q_2 \ \cdots p_{k-1} \ \neg q_{k-1} \ p_k \rangle$ and $\boldsymbol{p'} = \langle p'_1 \ \neg q'_1 \ p'_2 \ \neg q'_2 \ \cdots p'_{k'-1} \ \neg q'_{k'-1} \ p'_{k'} \rangle$ be two NSP s.t. $\boldsymbol{p} \lhd \boldsymbol{p'}$. And let $\boldsymbol{s} = \langle s_1 \ \cdots \ s_n \rangle$ be a sequence s.t. $\boldsymbol{p'} \preceq \boldsymbol{s}$, *i.e.* there exists an embedding $(e_i)_{i\in[k']}$:

- $\forall i,\ e_{i+1} > e_i$ (embedding), $e_{i+1} - e_i \leq \theta$ (*maxgap*) and $e_{k'} - e_1 \leq \tau$ (*maxspan*),
- $\forall i,\ p'_i \subseteq s_{e_i}$,
- $\forall j \in [e_i+1, e_{i+1}-1],\ q'_i \not\subseteq_D s_{e_j}$

To prove that $\boldsymbol{p} \preceq \boldsymbol{s}$, we prove that $(e_i)_{i\in[k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$.

Let us first consider that $k = k'$, then by definitions of $\lhd$ and the embedding,

(i) $\forall i \in [k],\ p_i \subseteq p'_i \subseteq s_{e_i}$,
(ii) $\forall i \in [k-1],\ \forall j \in [e_i+1, e_{e+1}-1],\ q'_j \not\subseteq_D s_{e_i}$, and thus $q_j \not\subseteq_D s_{e_i}$ (because of anti-monotonicity of $\not\subseteq_D$ and $q_i \subseteq q'i$)

In addition, we know that *maxgap* and *maxspan* constraints are satisfied by the embedding, *i.e.*

(iv) $\forall i \in [k],\ e_{i+1} - e_i \leq \theta$
(v) $e_k - e_1 = e_{k'} - e_1 \leq \tau$

Let us now consider that $k' > k$, (i), (ii) and (iii) still holds, and we know that if $e_k < e_{k'}$ (embedding property), then $e_k - e_i < \theta$.

This means that $(e_i)_{i\in[k]}$ is an embedding of $\boldsymbol{p}$ in $\boldsymbol{s}$ that satisfies gap constraints.

Proof (Proof of Proposition 5 (Complete and correct algorithm)) The correction of the algorithm is given by lines 2-3 of Algorithm 1. A pattern is outputted only if it is frequent (line 2).

We now prove the completeness of the algorithm. First of all, we have to prove that any pattern can be reached using a path of elementary transformations ($\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$). Let $\boldsymbol{p'} = \langle p'_1 \cdots p'_m \rangle$ be a pattern with a total amount of n items, $n > 0$, then it is possible to define $\boldsymbol{p}$ such that $\boldsymbol{p} \rightsquigarrow \boldsymbol{p'}$ where $\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$, and $\boldsymbol{p}$ will have exactly $n-1$ items:

- if the last itemset of $\boldsymbol{p'}$ is such that $|p'_m| > 1$ we define $\boldsymbol{p} = \langle p'_1 \cdots p'_{m-1} \ p_m \rangle$ as the pattern with the same prefix as $\boldsymbol{p'}$ and an additional itemset, p_m such that $|p_m| = |p'_m| - 1$ and $p_m \subset p'_m$: then $\boldsymbol{p} \rightsquigarrow_c \boldsymbol{p'}$

- if the last itemset of $\boldsymbol{p'}$ is such that $|p'_m| = 1$ and p'_{m-1} is positive then we define $\boldsymbol{p} = \langle p'_1 \cdots p'_{m-2}\ p'_{m-1}\rangle$: then $\boldsymbol{p} \rightsquigarrow_s \boldsymbol{p'}$
- if the last itemset of $\boldsymbol{p'}$ is such that $|p'_m| = 1$ and p'_{m-1} is negative (non-empty) then we define $\boldsymbol{p} = \langle p'_1 \cdots p_{m-1}\ p'_m\rangle$ where p_{m-1} is such that $|p_{m-1}| = |p'_{m-1}| - 1$ and $p_{m-1} \subset p'_{m-1}$: then $\boldsymbol{p} \rightsquigarrow_n \boldsymbol{p'}$

Applying this rules recursively, for any pattern $\boldsymbol{p}$ there is a path from the empty sequence to $\boldsymbol{p}$: $\emptyset \rightsquigarrow^* \boldsymbol{p}$. Also, exactly one of the three extensions can be used at any step, meaning that these path is unique. This prove that our algorithm is not redundant.

Second, the pruning strategy is correct, so, no frequent pattern will be missed. This comes from the anti-monotonicity property.

Let $\boldsymbol{p}$ and $\boldsymbol{p'}$ be two patterns such that $\boldsymbol{p} \rightsquigarrow \boldsymbol{p'}$ where $\rightsquigarrow \in \{\rightsquigarrow_n, \rightsquigarrow_s, \rightsquigarrow_c\}$, then is is quite obvious that $\boldsymbol{p} \lhd \boldsymbol{p'}$. Let us now consider that $\boldsymbol{p} \rightsquigarrow^* \boldsymbol{p'}$ from $\boldsymbol{p}$ to $\boldsymbol{p'}$ then, by transitivity of $\lhd$, we also have that $\boldsymbol{p} \lhd \boldsymbol{p'}$. And then by anti-monotonicity of the support, we have that $supp(\boldsymbol{p}) \geq supp(\boldsymbol{p'})$.

Let us now proceed by absurd and consider that $\boldsymbol{p'}$ is a pattern with support $supp(\boldsymbol{p'}) \geq \sigma$ that was not found by the algorithm. This means that for all paths[7] $\emptyset \rightsquigarrow^* \boldsymbol{p'}$ there exists some pattern $\boldsymbol{p}$ such that $\emptyset \rightsquigarrow^* \boldsymbol{p} \rightsquigarrow^* \boldsymbol{p'}$ with $supp(\boldsymbol{p}) < \sigma$. $\boldsymbol{p}$ is the pattern that has been used to prune the search for this path to $\boldsymbol{p'}$. This is not possible considering that $\boldsymbol{p} \rightsquigarrow^* \boldsymbol{p'}$ and thus $supp(\boldsymbol{p}) \geq supp(\boldsymbol{p'}) \geq \sigma$.

B NegPSpan extracts a superset of eNSP

Proposition 6 *Soft-embedding $\implies$ strict-embedding for patterns consisting of items.*

Proof Let $\boldsymbol{s} = \langle s_1 \cdots s_n\rangle$ be a sequence and $\boldsymbol{p} = \langle p_1 \cdots p_m\rangle$ be a NSP s.t. $|p_i| = 1$ for all $i \in [n]$ and $\boldsymbol{p}$ occurs in $\boldsymbol{s}$ according to the soft-embedding semantic.

There exists $\boldsymbol{\epsilon} = (e_i)_{i\in[m]} \in [n]^m$ s.t. for all $i \in [n]$, p_i is positive implies $p_i \in s_{e_i}$ and p_i is negative implies that for all $j \in [e_{i-1}+1, e_{i+1}-1]$, $p_i \notin s_j$ (items only) then $p_i \notin \bigcup_{j\in[e_{i-1}+1,e_{i+1}-1]} s_j$ *i.e.* $p_i \not\subseteq_* \bigcup_{j\in[e_{i-1}+1,e_{i+1}-1]} s_j$ (whatever $\not\subseteq_G$ or $\not\subseteq_D$). As a consequence $\boldsymbol{\epsilon}$ is a strict-embedding of p.

Proposition 7 *Let $\mathcal{D}$ be a dataset containing sequences made of items and $\boldsymbol{p} = \langle p_1 \cdots p_m\rangle$ be a sequential pattern extracted by eNSP. Then, without embedding constraints $\boldsymbol{p}$ is extracted by* NegPSpan *with the same minimum support.*

Proof If $\boldsymbol{p}$ is extracted by eNSP, then its positive partner is frequent in the dataset $\mathcal{D}$. As a consequence, each $p_i, i \in [m]$ is a singleton itemset.

According to the search space of NegPSpan defined by $\lhd$ if $\boldsymbol{p}$ is frequent then it will be reached by the depth-first search. Then it is sufficient to prove that for any sequence $\boldsymbol{s} = \langle s_1 \cdots s_n\rangle \in \mathcal{D}$ such that $\boldsymbol{p}$ occurs in $\boldsymbol{s}$ according to eNSP semantics (strict-embedding, strong absence), then $\boldsymbol{p}$ also occurs in $\boldsymbol{s}$ according to the NegPSpan semantics (soft-embedding, weak absence). Consequently, considering the same minimum support threshold, $\boldsymbol{p}$ is frequent according to NegPSpan. Proposition 6 gives this result.

Then we conclude that NegPSpan extracts more patterns than eNSP on sequences of items. In fact, NegPSpan can extract patterns with negative itemsets larger than 2.

eNSP extracts patterns that are not extracted by NegPSpan on sequences of itemsets. Practically, NegPSpan uses a size limit for negative itemsets $\nu \geq 1$. eNSP extracts patterns whose positive partners are frequent. The positive partner, extracted by PrefixSpan may hold itemsets larger than ν, and if the pattern with negated itemset is also frequent, then this pattern is extracted by eNSP, but not by NegPSpan.

C Additional experiments

C.1 Influence of vocabulary size

Figure 6 shows computation time and memory consumption with respect to vocabulary size: eNSP is run with different values of ς, the minimal frequency of the positive partner of negative patterns (100%, 80%

[7] Note that we proved that this path is actually unique.

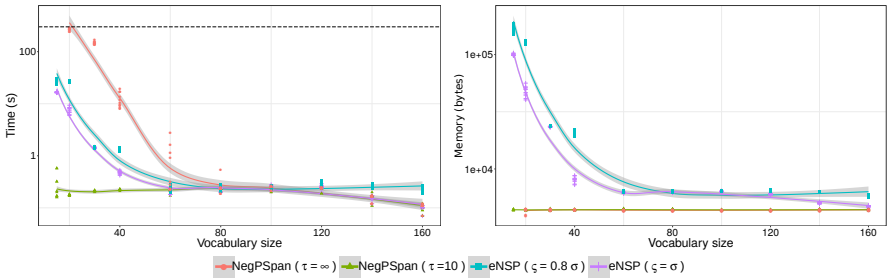

Fig. 6 Comparison of eNSP and NegPSpan computation time (left) and memory consumption (right) wrt vocabulary size. The dashed line shows the limit for timeout executions.

and 20% of the minimal frequency threshold) and NegPSpan is run with a *maxgap* of 10 or without. The timeout is set to 5 minutes.

Similarly to experiments of the previous section, with no constraint, NegPSpan is less time-efficient than eNSP but it becomes more time-efficient with a gap constraint whatever the vocabulary size. Memory consumption curves show that NegPSpan requires significantly less memory than eNSP.

Figure 6 clearly shows that the smaller the vocabulary is, the more frequent patterns there are, and thus the more memory is required and time is high. Indeed, the smaller vocabulary (with a given sequence length), the higher the probability to extract some sequential pattern. This is the case for positive patterns as well as for negative patterns considering that the positive part of the negative pattern may more likely occur in a raw (and may necessarily satisfy the negative constraints). Then, generated datasets have more positive and negative patterns to extract.

More especially, there are more positive patterns to extract and thus the memory required by eNSP increases because it requires to store all positive patterns (with a support above ς). We can see that when the vocabulary size decreases, the memory required by eNSP increases very quickly (faster than exponential growth), while NegPSpan requires almost the same amount of memory for any vocabulary size.

The use of a *maxgap* constraint ($\tau = 10$) makes NegPSpan several orders of magnitude more time-efficient for small vocabulary size. With very small vocabulary size (< 20), the number of negative patterns extracted by NegPSpan explodes and the execution time exceeds the timeout of 5 minutes ($300s$). For greater vocabulary size, the differences between algorithms disappear. NegPSpan ($\tau = \infty$) is more efficient than eNSP for big vocabulary size because only few frequent negative patterns should be extracted, but there are many positive patterns: in this case eNSP has to evaluate the support of potential negative sequential patterns on the basis of the positive patterns while NegPSpan stops the exploration as soon as an unfrequent negative pattern is found. Thus, eNSP with $\varsigma = 0.8\sigma$, which explores more positive patterns than eNSP with $\varsigma = \sigma$, is less time-efficient.

C.2 Influence of average sequence length

Figure 7 shows the computation time and memory consumption with respect to average length of sequences with a minimal support $\sigma = 20\%$. eNSP is run with different values for ς, the minimal frequency of the positive partner of negative patterns (100%, 80% and 20% of the minimal frequency threshold) and NegPSpan is run with a maxgap $\tau = 10$ or without *maxgap* constraint. The timeout is set to 5 min.

Computation times and memory consumptions are exponential with respect to the average sequence length. Curves differ by their factors of exponential growth. In the remainder of this section α represents the exponential growth.

Figure 7 on the left compares the computation times. The exponential growth of NegPSpan without maxgap ($\alpha \approx 10^{-11}$) is high and the timeout is reached for datasets with an average sequence length of about 30 itemsets. eNSP is one order of magnitude more time-efficient and can analyze dataset with an average sequence length about 45 itemsets. But, parameter ς does not changes the computation time significantly. Indeed, the exponential growths are close to each other ($\alpha_{\varsigma=\sigma} \approx 5.08\times10^{-12}$, $\alpha_{\varsigma=0.8\sigma} \approx 2.42\times10^{-12}$, $\alpha_{\varsigma=0.5\sigma} \approx 2.72\times10^{-12}$). In contrast, the use of the maxgap constraint ($\tau = 10$) changes significantly

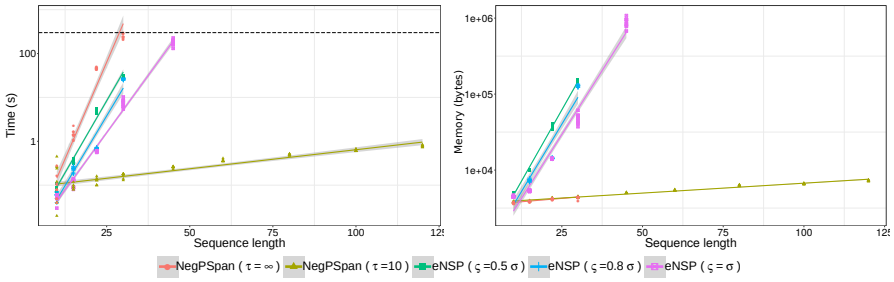

Fig. 7 Comparison of computing time (left) and memory consumption (right) between eNSP and NegPSpan wrt average length of sequences. The dashed line shows the limit for timeout executions.

the difficulty of the task: the exponential growth is significantly lower ($\alpha \approx 10^{-53}$) and the timeout is not reached even for sequence containing about 120 itemsets. This result was expected considering that the maxgap constraint avoids to explore the full sequence to evaluate the support of the pattern. Using NegPSpan with maxgap constraints is thus very time-efficient to mine negative patterns in long sequences.

Figure 7 on the right compares the memory consumption of the two algorithms. The curves look very similar to the computation time curves, but it is important to note that the memory consumption requirement increases significantly slower with NegPSpan than with eNSP, whatever the use of maxgap constraint. The explanation is similar to previous benchmark results: the depth-first search strategy does not store patterns while eNSP does to evaluate the support of negative patterns.

C.3 Computational performances on case study datasets

This appendix presents the computational performances of eNSP and NegPSpan on two datasets of the case studies (see Section 6): instacart and care pathway analysis.

C.3.1 Instacart data

Figure 8 shows the computation times, the memory requirements and the number of NSPs extracted by both algorithms on the Instacart dataset (see Section 6.1). On this dataset, we can first note that the number of patterns extracted by NegPSpan is about two orders of magnitude larger. As a consequence, the computation time is higher even with strong gap constraints: NegPSpan takes about $1,000s$ with $\tau = 2$ while eNSP takes always less than $500s$ to extract 1% NSPs (whatever ς). These results can be explained by the dataset features. With a large vocabulary the support of patterns decreases rapidly when the pattern length increases. This means that eNSP prunes a lot of patterns while extracting the positive partners. This explains that few patterns are extracted. In contrast with this fast pruning strategy, NegPSpan explores lots of potential negative extensions because most of them could be frequent due to the relative low frequency of each item. eNSP seems more efficient on this dataset, but we recall that it failed to explore large datasets, not for time reason, but for heavy memory requirements. Figure 8 middle, shows this limitation well: the memory requirement is several orders larger for eNSP than for NegPSpan and it increases exponentially when ς decreases.

C.3.2 Care pathway analysis

Figure 9 gives a comparison of the computation performances (time and memory usage) between eNSP and NegPSpan with respect to the minimal frequency threshold ($\sigma \in [0.08, 0.3]$) on the care pathway dataset (see Section 6.2). Each algorithm is run with different settings: the *maxgap* constraint $\tau \in \{3, 5, 8\}$ for NegPSpan and the minimal support of positive partners $\varsigma \in \{0.4\sigma, 0.8\sigma, \sigma\}$ for eNSP. The maximal pattern length is set to $l = 5$. The results obtained on real data confirm the results obtained in Section 5 on

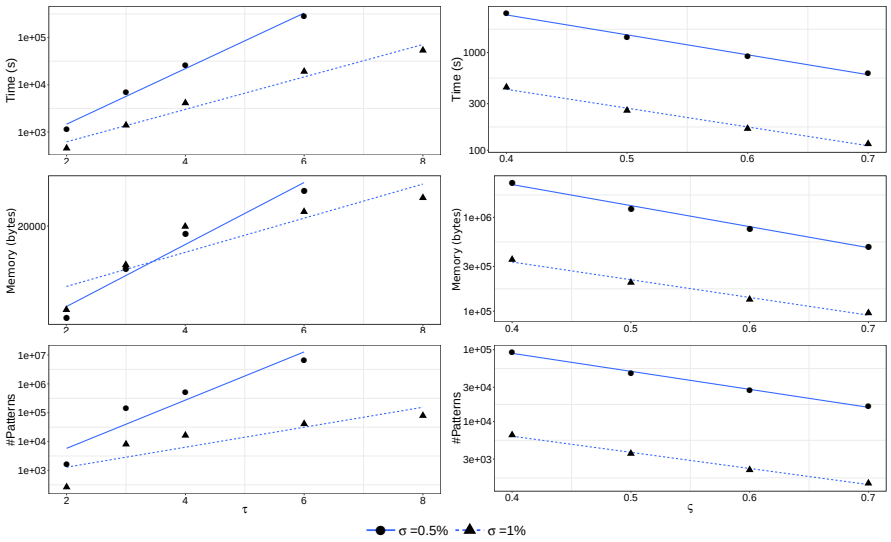

Fig. 8 Computation times (on top), memory requirements (in the middle) and numbers of NSP (at the bottom) extracted from the Instacart dataset with respect to the gap constraint τ for NegPSpan (on the left) and with respect to frequent positive ratio (ς) for eNSP (on the right).

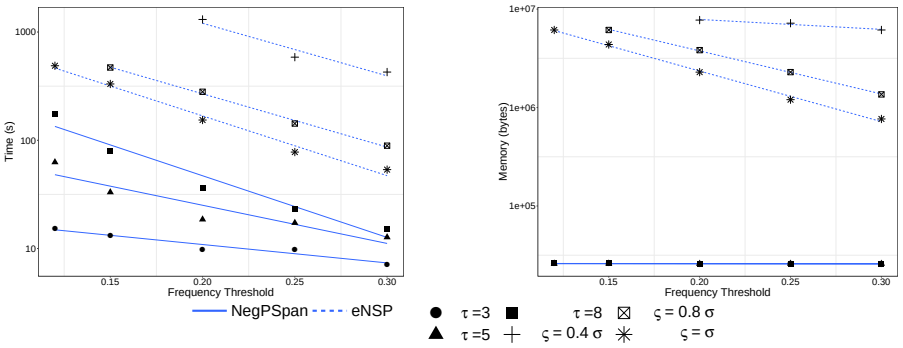

Fig. 9 Time, memory usage and number of extracted patterns for NSP and NegPSpan with respect to the minimal frequency support, and different algorithm settings. (care pathways dataset, see Section 6.2)

synthetic data. On the one hand, NegPSpan requires several orders of magnitude less memory than eNSP. eNSP does not terminate with lowest σ and ς values. Its memory requirement exceeds the computer capacity (8Go). On the other hand, this heavy memory requirement has consequences on computation times and NegPSpan is several orders of magnitude more time-efficient than eNSP whatever the setting. We observe that the computation time increases exponentially when the frequency threshold (σ) decreases, and, the lower *maxgap*, the lower the computation time. This is mainly due to the number of extracted patterns that grows also exponentially when the frequency threshold decreases.

Table 9 NSP extracted by NegPSpan but not by eNSP

Pattern	support NegPSpan
$\langle 6DIS\ \neg 3SER\ 1RES\rangle$	5,352
$\langle (1CTR, 3SER)\ \neg 3SER\ 1RES\rangle$	2,268
$\langle (1CTR, 2VIE, 8DOS)\ \neg 3SER\ 1RES\rangle$	1,946
$\langle 1RES\ \neg 3SER\ 1RES\rangle$	5,265
$\langle (1CTR, 3SER, 8DOS)\ \neg 3SER\ 1RES\rangle$	1,893
$\langle 7ACC\ \neg 3SER\ 1RES\rangle$	1,947
$\langle (3SER, 8DOS)\ \neg 3SER\ 1RES\rangle$	3,060
$\langle (2VIE, 3SER)\ \neg 3SER\ 1RES\rangle$	2,058
$\langle (2VIE, 8DOS)\ \neg 3SER\ 1RES\rangle$	3,106
$\langle (1CTR, 2VIE)\ \neg 3SER\ 1RES\rangle$	2,294
$\langle 4FAC\ 4FAC\ \neg 3SER\ 1RES\rangle$	2,123
$\langle (1CTR, 8DOS)\ \neg 3SER\ 1RES\rangle$	6,597
$\langle 2VIE\ 8DOS\ \neg 3SER\ 1RES\rangle$	2,048
$\langle (2VIE, 8DOS)\ \neg 2VIE\ 1RES\rangle$	3,029
$\langle (1CTR, 3SER, 8DOS)\ \neg 2VIE\ 1RES\rangle$	1,900
$\langle (1CTR, 3SER), \neg 2VIE\ 1RES\rangle$	2,258
$\langle 1CTR\ 2VIE\ 8DOS\ \neg 2VIE\ 1RES\rangle$	1,912
$\langle 7ACC\ \neg 2VIE\ 1RES\rangle$	1,901
$\langle (3SER, 8DOS)\ \neg 2VIE\ 1RES\rangle$	3,032
$\langle (1CTR, 2VIE)\ \neg 2VIE\ 1RES\rangle$	2,266

D List of extracted patterns for the CRM dataset

This Appendix provides the complete list of negative sequential patterns involving *3SER* or *2VIE* negative items extracted by eNSP or NegPSpan.

Table 10 NSP extracted by eNSP but not by NegPSpan

Pattern
$\langle 8DOS\ \neg(2VIE, 3SER)\ 1RES\rangle$
$\langle 5REC\ \neg 3SER\ 1RES\rangle$
$\langle 5REC\ 5REC\ \neg 2VIE\ 1RES\rangle$
$\langle 2VIE\ \neg 2VIE\ 1RES\rangle$
$\langle 2VIE''3SER\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ 5REC\ \neg 2VIE\ 1RES\rangle$
$\langle 4FAC\ \neg 2VIE\ 1RES\rangle$
$\langle 3SER\ \neg 3SER\ 1RES\rangle$
$\langle 2VIE\ 2VIE\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle 2VIE\ 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle 4FAC\ 4FAC\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ \neg 3SER\ 1RES\rangle$
$\langle 1CTR\ \neg 3SER\ 1RES\rangle$
$\langle 8DOS\ 2VIE\ \neg 2VIE\ 1RES\rangle$
$\langle 2VIE\ \neg 3SER\ 1RES\rangle$
$\langle 5REC\ 5REC\ \neg 3SER\ 1RES\rangle$
$\langle 4FAC\ \neg 3SER\ 1RES\rangle$
$\langle 5REC\ \neg 2VIE\ 1RES\rangle$
$\langle 6DIS\ \neg 2VIE\ 1RES\rangle$
$\langle 1RES\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ 8DOS\ \neg 3SER\ 1RES\rangle$
$\langle 1CTR\ \neg 2VIE\ 1RES\rangle$
$\langle 3SER\ \neg 2VIE\ 1RES\rangle$
$\langle (1CTR, 8DOS)\ \neg 2VIE\ 1RES\rangle$

Table 11 NSP extracted by both eNSP and NegPSpan

Pattern
$\langle 3SER\ \neg 3SER\ 1RES\rangle$
$\langle 2VIE\ \neg 3SER\ 1RES\rangle$
$\langle 1RES\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ 8DOS\ \neg 3SER\ 1RES\rangle$
$\langle 5REC\ \neg 3SER\ 1RES\rangle$
$\langle 2VIE\ 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle 4FAC\ \neg 3SER\ 1RES\rangle$
$\langle 2VIE\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ 8DOS\ \neg 2VIE\ 1RES\rangle$
$\langle (2VIE, 3SER)\ \neg 2VIE\ 1RES\rangle$
$\langle 4FAC\ 4FAC\ \neg 2VIE\ 1RES\rangle$
$\langle 4FAC\ \neg 2VIE\ 1RES\rangle$
$\langle 1CTR\ \neg 2VIE\ 1RES\rangle$
$\langle 5REC\ \neg 2VIE\ 1RES\rangle$
$\langle 6DIS\ \neg 2VIE\ 1RES\rangle$
$\langle 8DOS\ \neg 3SER\ 1RES\rangle$
$\langle 3SER\ \neg 2VIE\ 1RES\rangle$
$\langle (1CTR, 8DOS)\ \neg 2VIE\ 1RES\rangle$
$\langle 1CTR\ \neg 3SER\ 1RES\rangle$