Negative Sequence Analysis: A Review

WEI WANG, University of Technology Sydney
LONGBING CAO*, University of Technology Sydney

Negative sequential patterns (NSPs) produced by negative sequence analysis (NSA) capture more informative and actionable knowledge than classic positive sequential patterns (PSPs) due to involving both occurring and non-occurring items, which appear in many applications. However, the research on NSA is still at an early stage and NSP mining involves very high computational complexity and a very large search space, there is no widely accepted problem statement on NSP mining, and different settings on constraints and negative containment have been proposed in existing work. Among existing NSP mining algorithms, there are no general and systemic evaluation criteria available to assess them comprehensively. This paper conducts a comprehensive technical review of existing NSA research. We explore and formalize a generic problem statement of NSA, investigate, compare and consolidate the definitions of constraints and negative containment, and compare the working mechanisms and efficiency of existing NSP mining algorithms. The review is concluded by discussing new research opportunities in NSA.

CCS Concepts: • **General and reference → Surveys and overviews**;

Additional Key Words and Phrases: Negative Sequential Pattern Mining, Negative Sequence Analysis, Non-occurring Behavior Analysis, Behavior Analytics

ACM Reference Format:
Wei Wang and Longbing Cao. 2018. Negative Sequence Analysis: A Review. *ACM Comput. Surv.* 9, 4, Article 39 (February 2018), 37 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Sequences widely appear in ordered and behavioral data, and sequence analysis is increasingly recognized in many sequential applications including gene analysis, behavior analytics, and text analysis [4, 8, 55]. Dominating efforts have been made on positive sequence analysis (PSA), in particular, mining frequent positive sequential patterns (PSP) in occurring sequences [40], leading to many PSP mining algorithms, e.g., PrefixSpan [46], SPADE [67], SPAM [3], bitSPADE [2], LAPIN [62], IMSP [53], FAST [49] and LCMSeq [65], and closed PSP mining algorithms such as ClaSP [24], ClosedISP [34], CloFAST [22] and FCloSM [33]. Further work is on PSP pruning to improve the performance of PSP mining algorithms, such as CMAP [21] and DISC [18].

By contrast, negative sequence analysis (NSA), i.e., discovering important yet non-occurred [13] sequential patterns, also called negative sequential patterns (NSP), has only received limited attention. NSPs are patterns consisting of both occurring and non-occurring elements in sequential

*Corresponding author.

Authors' addresses: Wei Wang, University of Technology Sydney, Wei.Wang-8@student.uts.edu.au; Longbing Cao, University of Technology Sydney, longbing.cao@uts.edu.au.

0360-0300/2018/2-ART39 $15.00
https://doi.org/0000001.0000001

data. While rarely studied, NSA and NSP are shown useful for understanding and managing non-occurring behavior (NOB) [13] and complicated behavior relations [6, 7], as they can disclose additional yet usually hidden information that cannot be replaced or informed by PSPs [1, 8].

The non-occurring elements in a negative sequence stand for the absence of specific events or actions, corresponding to important but undeclared behaviors [13], often appearing in manipulated or undesirable behaviors [12], concerned situations [36], and business applications [14]. While non-occurring events are usually overlooked, they could be very useful for detecting missing health/medical treatments or health insurance claims [74, 75], undeclared fraudulent social welfare behaviors for debt detection [15, 68–71], undeclared behaviors of taxpayers [73], manipulated trading behaviors [9, 10, 51], pattern relation analysis [5], poor academic learning performance [30], terrorist activities, security, and risk management, etc.

For example, in social welfare debt detection, it was found those allowance recipients who did not claim activities: PYR, RPR, REA and STM, may likely receive overpayment paid by the government due to the misleading information unprovided (i.e., NSP: $\neg < PYR, RPR, REA, STM > \rightarrow DEB$) [71]. More generally, in detecting insurance claim fraud, suppose $S_{pos} =< a, b, c, d >$ is a claim sequence of a customer, if $S_{neg} =< a, b, \neg c, d >$ is an NSP and $sup(S_{pos})/sup(S_{neg}) < min\_ratio$, then S_{pos} is likely fraudulent, since code c should be claimed together with others but it does not appear together [72]. Here, each item a, b, c and d stands for a claim item code, $sup(S_{pos})$ and $sup(S_{neg})$ are the supports, and min_ratio is a predefined threshold. For the example of system diagnosis, if some maintenance operations should be but were not conducted following certain alarms, a system fault or even disaster may occur; while if these operations were performed in time, then alarms would stop and no fault would occur, i.e., $S_\alpha =< a, b, o, X >$ and $S_\beta =< a, b, \neg o, Y >$. Here a and b stand for two alarms, o stands for the conduction of maintenance operation, and X and Y stand for no-fault and fault. Given alarms a and b are received, S_α represents a pattern that no-fault X would occur if operation o was performed while S_β represents that otherwise fault Y would likely occur.

However, PSP methods cannot be directly applied or adjusted to analyze the above NOB scenarios. In addition to the hidden nature of NOB, the downward property, which forms the foundation of PSA, does not hold in NSA. It is also far more challenging to discover NSPs even in a medium-sized dataset with a not-too-low support threshold, because the search space of NSP mining is much larger and its computational complexity is significantly higher than in PSP mining. In addition, few pruning strategies are available due to the lack of downward property. As a result, only a few algorithms have been proposed in NSA, including NSPM [36], NFSPM [38], PNSPM [37], Neg-GSP [74] and e-NSP [8]. There is a significant gap between the wide NOB applications and very limited research on NSA.

As NSA is fundamental and challenging, significant inconsistencies exist between existing algorithms, with each only focusing on specific scenarios and settings. There are no unified problem statements, constraint settings, negative containment definitions, formal representations, or NSP formulas [8]. This has limited the theoretical development and applications of NSA compared to the widespread NOB scenarios. The only survey in [25] only covers very limited and specific aspects in terms of three NSP mining algorithms and three measures on limited datasets.

To address the demand of and significant gaps in existing NSA research, this paper provides a comprehensive and systematic technical overview and survey on NSA.

- By reviewing all NSA algorithms, we propose a systematic formalization of the NSA problem, in which NSP formulas are specified on top of formal concepts: *item*, *element* and *sequence*.
- By investigating different definitions of constraints, a comprehensive formalization of constraints is provided in terms of size constraint, frequency constraint, format constraint, and negative element constraint.

- Building on understanding existing definitions, negative containment is defined and formalized in terms of levels such as on element, sub-sequence and super-sequence and various containing relationships such as disjunction and conjunction.
- An empirical evaluation demonstrates the pros and cons of existing NSA algorithms and their settings in terms of algorithm efficiency.
- Lastly, open issues and prospects for NSA research are discussed.

2 RELATED WORK AND CHALLENGES

The following NSP algorithms have so far been reported in the literature: NSPM [36], two extended NSPM algorithms NFSPM [38] and PNSPM [37], MSIS [44], three MSIS extensions including MBFIFS [43], CPNFMLSP [41] and CPNFSP [42], Incremental CPNFSP [32], SpamNeg [68], PNSP [28], Negative-GSP [74], GA-NSP [75], e-NSP [8] and three extended e-NSP algorithms SAPNSP [39], e-msNSP [61] and e-NSPFI [26]. Below, we conduct a preliminary comparison between them and summarize the main problems associated with these methods and technical challenges in NSA.

2.1 Categorization of Existing Methods

According to analytical objectives and settings, existing NSP mining algorithms can be roughly categorized into four groups: format-specific NSP mining, complete NSP mining, stochastic NSP mining, and PSP-based NSP mining.

Format-specific NSPs refer to particular types of NSPs that impose specific constraints on NSP structures and formats. They thus generate negative sequential candidates (NSCs) in a small search space and reduce the number of NSCs to be verified. Typical algorithms include NSPM, NFSPM, PNSPM, MSIS, MBFIFS, CPNFMLSP and CPNFSP.

Among these algorithms, NSPM, NFSPM and PNSPM introduce a concept of *location format constraint* (LFC) to generate NSCs in the format of $< e_1, e_2, \dots, \neg e_s >$, where a negative element can only appear at the end of an NSC. MSIS, MBFIFS, CPNFMLSP and CPNFSP introduce another LFC constraint to define NSCs in the form of $< e_1, \neg e_2 >$, $<\neg e_1, e_2 >$ and $<\neg e_1, \neg e_2 >$ so that an NSC can have only two elements and there is at most one positive element that contains all positive items, which resembles the mining of negative association rules [20, 29, 59, 69, 71].

This category of NSP algorithms obtains NSCs by a simplified NSC generation strategy with low computational complexity, but the algorithms are only applicable for specific applications since the discovered NSPs are quite constrained.

The second category of NSP algorithms aims to discover the complete set of NSPs. They adjust the existing PSP algorithms to mine all the NSPs satisfying a given threshold. Typical algorithms include PNSP and Negative-GSP (NegGSP for short), which were both extended from GSP [52] algorithm. Both PNSP and NegGSP adopt the NSC generation-and-testing strategy. This strategy first adapts traditional PSP mining algorithms to generate long-length or long-size NSCs based on mined PSPs and NSCs, and then tests whether they are interesting NSPs by calculating their negative supports in a pass over the whole dataset.

PNSP adopts an appending-based NSC generation strategy, which generates a s-size NSC by appending a (s-1)-size NSC or PSP with a frequent positive or negative itemset. NegGSP adopts a joining-based strategy which generates a l-length NSC by joining a (l-1)-length seed or PSP with another (l-1)-length seed. This category of NSP mining algorithms can discover a larger number of NSPs, especially for long-size or long-length NSPs with wider item distribution. However, these algorithms always consume more runtime in generating and testing enormous NSCs, and they require more memory space to save the generated NSCs. In addition, many of the mined NSPs may not be actionable [11, 55] since their interestingness may be too low to attract business interest.

The third category targets highly frequent NSP mining by introducing a stochastic strategy to NSP mining and only generating NSCs with potentially high frequency by specific operations between selected optimal NSPs in a shrunken search space. GA-NSP is a typical stochastic NSP mining algorithm which is built on genetic algorithm. GA-NSP calculates the dynamic fitness for each NSC or NSP. It selects NSPs with high dynamic fitness and conducts crossover and mutation operations on these chosen NSPs to generate NSCs, thus consuming less runtime and memory usage during the mining process. Suffering from the characteristics of stochastic processing, these algorithms cannot guarantee the coverage of discovered NSPs, and sometimes only a small number of NSPs may be mined.

Lastly, PSP-based NSP mining invents a new NSA theory based on the PSP-to-NSC conversion. This method derives NSCs and calculates the negative supports by only using the corresponding information of discovered PSP and converts negative containment to positive containment. e-NSP is the only algorithm proposed, which is a set theory-based algorithm that applies sequence frequency constraint (SFreC) and strictly-negative containment. It generates NSCs by a negative conversion strategy and calculates the strictly-negative-support of each NSC by using the support of its maximum positive sub-sequence and 1-negative-size maximum sub-sequences. Benefiting from avoiding testing NSCs by rescanning the dataset, e-NSP is highly efficient and scalable with small runtime and memory usage. However, e-NSP may only discover a very small coverage of NSPs, and long-size or long-length NSPs may be lost. Built on e-NSP, SAPNSP mines patterns through e-NSP and proposes an interestingness measure to judge whether a mined pattern is actionable [16]. In addition, e-msNSP extends e-NSP to mine NSP with multiple minimum supports, where each item is associated with a minimum item support (MIS) and the minimum support threshold for a negative sequence is calculated by the MIS value of items within this sequence. In e-msNSP, a negative sequence is an NSP if its support is greater than its minimum support threshold, rather than a predefined global threshold as in other algorithms. Moreover, e-NSPFI is another extension of e-NSP to discover NSP form both frequent and some constrained infrequent PSP. Finally, HUSP-NIV [60] mines high utility sequential patterns (HUSP) from sequential utility-based databases, also built on e-NSP. Since HUSP-NIV introduces utility to each item and focuses on the discovery of patterns with negative item values (NIV), it is beyond the scope of this paper.

The above categorization of existing NSP mining algorithms is summarized in Table 1 according to the above analysis and their main research ideas, advantages and disadvantages.

In conclusion, even though research on NSP mining is attracting more and more attention and several algorithms have already been proposed, existing work cannot address the wide NOB applications, and further efforts in this field are required.

2.2 Technical Challenges

The above analysis of related work shows: (1) NSA is attracting increasing interest, with more advanced theories and algorithms progressively being reported; (2) NSP mining is much more difficult and complex than PSP mining; and (3) Existing work only discovers a partial coverage of interesting NSPs with no universal and systematic definitions and mechanisms accepted in the area. Despite the difficulty in obtaining non-occurring behavioral data, the main technical challenges facing NSP research include: inconsistencies in NSA problem formalization, violation of the downward closure property, large search space, high computational complexity, and the lack of systematic evaluation criteria.

First, there are serious inconsistencies in the NSA definition and formalization. In contrast to PSP mining, no widely accepted problem statement, constraint settings, and systematic definitions about negative containment exist in the current NSA research. This reflects the much more challenging nature of NSA. Accordingly, different algorithms aim to discover specific forms of NSPs in respective

Table 1. Categorization of Existing NSP Mining Algorithms

Research Category	**Main Idea**	**Typical Algorithm**	**Advantage**	**Disadvantage**
Format-specific NSP mining	Adopt specific constraints to define interesting NSPs of special formats	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CP-NFMLSP [41], CPNFSP [42], Incremental CP-NFSP [32], SpamNeg [68]	Smaller space is required to search, and fewer NSCs are generated	NSPs against predefined formula are missed and applications are limited
Complete NSP mining	Adopt the NSC generation-and-testing strategy, and adapt PSP mining algorithms to generate long NSCs based on mined PSPs and NSCs	PNSP [28], NegGSP [74]	Maintains the maximum coverage of NSPs	Resource consumption is large, and many discovered NSPs may be meaningless
Stochastic NSP mining	Adopt a stochastic strategy and only generate NSCs with potentially high frequency by taking stochastic operations on selected optimal NSPs	GA-NSP [75]	Search space is reduced, and the average resource consumption is small	Coverage of discovered NSPs cannot be guaranteed
PSP-based NSP mining	Convert negative containment to positive containment, and generate and test NSPs by only using the information of discovered PSPs	e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]	Effective and scalable in runtime and memory usage by avoiding rescanning dataset	Problem statement is much stricter, and only a very small coverage of NSPs is discovered

search spaces by adopting diverse constraints. One of the reasons why different algorithms result in divergent pattern coverage is that they adopt different definitions of negative containment. For example, PNSP adopts a stricter definition of negative containment than NegGSP. Hence, PNSP maintains a smaller pattern coverage than NegGSP, even though they mine PSPs in the same search space. In addition, NSPM defines the length of sequences as the number of elements in a sequence, while NegGSP, GA-NSP and e-NSP define the length as the total number of items in all elements of a sequence. The definition of sequence length in NSPM is actually the definition of sequence size in the latter algorithms. The inconsistencies and confusion require a more systematic design of NSA.

Second, NSPs do not satisfy the downward closure property as PSPs do. This means that a super-sequence of an infrequent negative sequence may be a frequent NSP, and a sub-sequence of a frequent NSP may also be an infrequent negative sequence. For example, $S_\alpha =< a, \neg b, c >$ is a super-sequence of $S_\beta =< \neg b, c >$, but S_α is contained in $S_\gamma =< b, a, c >$ while S_β is not contained in S_γ. The fact that NSP does not satisfy the downward closure property further enlarges the search space and computational complexity.

Third, NSP mining faces a large search space and high computational complexity due to the lack of downward closure property. Despite the existence of diverse definitions of negative containment, a data sequence can contain many more NSCs compared with positive candidates [72]. This means that the search space of NSP mining is much larger than that of PSP mining. A huge number of NSCs can be generated, and the computational complexity can be much higher if an NSP algorithm aims to maintain a large pattern coverage, such as PNSP and NegGSP. For example, for a set of items $I = \{a, b, c\}$, data sequence $S =< a, b, c >$ can contain at most $\sum_{k=1}^{3} C_3^k = 7$ positive candidates, but it can contain at most $\sum_{k=1}^{3}(C_3^k 3^{k+1}) = 2916$ NSCs. This requires strategies such as adopting reasonable constraints and effective pruning to reduce the research space and improve the performance of NSP mining algorithms. Pruning strategies can filter some invalid NSCs and reduce the computational complexity of NSP mining. However, few efficient pruning strategies are currently available. Applying constraints leads to a trade-off between algorithm performance and pattern coverage but also reduces the number of NSCs generated. In practice, a well-designed constraint may ensure that more informative NSPs can be identified in a smaller search space, while the corresponding theoretical analysis should be provided.

Lastly, there are no comprehensive and systematic evaluation criteria available so far. It results in the fact that different researchers apply different methods to evaluate their work, and the nature of behavior non-occurrences is not effectively captured and reflected in existing NSA research and evaluation. This also makes it difficult for existing NSP mining algorithms to be effectively validated in real-life applications.

In the following sections, we systematically review and formalize the NSA problem, with the aim of forming a comprehensive review of existing work as well as forming a systematic theoretical system for NSA research.

3 NSA PROBLEM STATEMENT

In this section, we introduce the basic entities required in NSA and their definitions, including *item*, *element*, and *sequence*. The problem of NSP mining is then defined. The main notations used in this paper are described in Table 2.

Table 2. Main Notations in NSA

Notation	Description	Notation	Description
I	A set of items, i.e., $I = \{i_1, i_2, \ldots, i_n\}$	$e_{ce} \subseteq_{ce} e_{pos}$	Conjunction element e_{ce} is a sub-element of positive element e_{pos}, and e_{pos} is a super-element of e_{ce}
i_k	The k-th item of I, $1 \leqslant k \leqslant I'ssize$	$e_{de} \subseteq_{de} e_{pos}$	Disjunction element e_{de} is a sub-element of positive element e_{pos}, and e_{pos} is a super-element of e_{de}
$i_k.state$	The state of item i_k	S	A sequence, which is an ordered list of elements
$RI(i_k)$	The reverse item of item i_k	$length(S)$	The length of sequence S, which is the total number of items in all elements in S
$PI(i_k)$	The positive item partner of item i_k	$size(S)$	The size of sequence S, which is the total number of elements in S
e	An element, which is a non-empty set of I	$neg - size(S_{neg})$	The negative size of negative sequence S_{neg}, which is the number of negative elements in S
E^+	The element containing all the positive items in I	$PS(S_{neg})$	The positive sequence partner of a negative sequence S_{neg}
E^-	The element containing all the negative items in I	$S_\alpha \subseteq_{pos} S_\beta$	Positive sequence S_α is a sub-sequence of positive sequence S_β, and S_β is a super-sequence of S_β
$size(e)$	The size of element e, which is the number of items in e	D	Sequence dataset, which is a set of tuples of data sequences and their identifiers
$neg - size(e)$	The negative size of element e, which is the number of negative items in e	$\|D\|$	The size of sequence dataset D, which is the number of tuples in D
$RE(e)$	The reverse element of element e	$SC(S)$	The count of sequence S contained in D.
$PE(e_{neg})$	The positive element partner of negative element e_{neg}	$sup(S)$	The support (in percentage) of sequence of S in D
$MPE(e_{neg})$	The maximum positive element of a negative element e_{neg}	min_sup	The minimum support threshold predefined by a user

3.1 Item and Its Properties

The concept *item* is the lowest level of unit and the smallest unit in NSA. Let I be a non-empty set of items, i.e., $\{i_1, i_2, \ldots, i_n\}$, where each item i_k ($1 \leqslant k \leqslant n$) is an atomic entity in a sequence and is associated with a set of attributes, such as the state of an item, etc. The value of item i on attribute A is denoted by $i.A$. The *state* of an item i, denoted as $i.state$, can be either *positive* or *negative*. A *negative item* is represented by the symbol $\neg$ in front of its corresponding positive item. An item in a positive state is called a *positive item*, which represents the *occurrence* (*appearance*) of a specific event (i.e., the item); while an item in a negative state is called a *negative item*, which represents the *non-occurrence* (*absence*) of its corresponding positive item. For example, the negative item $\neg a$ means that its positive item a does not appear or is absent.

The corresponding item with the opposite state of an item i_k is called its *reverse item* (RI), denoted as $RI(i_k)$. For example, $RI(a) = \neg a$ and $RI(\neg a) = a$. The *positive item partner* (PP) of a positive item is itself, while the *positive item partner* of a negative item is its reverse item, denoted as $PP(i_k)$. For example, $PP(a) = a$ and $PP(\neg a) = a$.

3.2 Element and Its Properties

The concept *element* is the second lowest level in the NSA conceptual system. An *element* is a non-empty subset of I, which is denoted as $e = (x_1, x_2, \ldots, x_s), x_k \in I, 1 \leqslant k \leqslant s$. An element is a compound entity of items and contains two attributes: *state* and *size*. The *state of an element* can be also positive or negative: if an element contains positive items only, it is called a *positive element*; if an element includes at least one negative item, it is called a *negative element*. If an element is a negative element and is composed of negative items only, it is called a *proper negative element*. A proper negative element $(\neg x_1, \neg x_2, \ldots, \neg x_s)$ can be also represented as $\neg(x_1, x_2, \ldots, x_s)$ for short, where $x_k (1 \leqslant k \leqslant s)$ is a positive item. For instance, a proper negative element $(\neg a, \neg b, \neg c)$ can be represented as $\neg(a, b, c)$. The set of all positive items is called the *complete positive element*, denoted as $E^+ = (i_1, i_2, \ldots, i_n)$, and the set of all negative items is called the *complete negative element*, denoted as $E^- = (\neg i_i, \neg i_2, \ldots, \neg i_n)$ or $E^- = \neg(i_1, i_2, \ldots, i_n)$. It is self-evident that $I = E^+ \cup E^-$.

If a negative element consists of both positive items and negative items, it can contain at most one of an item and its reverse item, but cannot contain both of them. For example, $(\neg a, b, \neg c)$ is allowed while $(\neg a, a, \neg c)$ is invalid in NSA. Items in an element are on the same level, and their orders are not differentiated. Without loss of generality, items in the same element are sorted in the ascending order of absolute item values, alphabetically, or per domain knowledge.

The *size* of an element e is the number of items in e, denoted as $size(e)$. An element e is called a *s-size element* if $size(e) = s$. The *negative size of element e* is the number of negative items in e, denoted as $neg-size(e)$. An element e is called a *s-neg-size element* if $neg-size(e) = s$. For example, if element $e = (\neg a, b, \neg c)$, then $size(e) = 3$ and $neg - size(e) = 2$.

Element $e_\alpha = (x_{\alpha_1}, x_{\alpha_2}, \ldots, x_{\alpha_m})$ is called a *reverse element* of element $e_\beta = (x_{\beta_1}, x_{\beta_2}, \ldots, x_{\beta_n})$ if $size(e_\alpha) = size(e_\beta)$ and $x_{\alpha_k} = RI(x_{\beta_k}), 1 \leqslant k \leqslant size(e_\alpha)$, denoted as $e_\alpha = RE(e_\beta)$. For example, $(\neg a, b, \neg c)$ is a reverse element of $(a, \neg b, c)$.

The *positive element partner of a negative element* $e_{neg} = (x_1, x_2, \ldots, x_s)$ is a positive element consisting of all positive items and reverse items of all negative items in e_{neg}, denoted as $PE(e_{neg})$, i.e., $PE(e_{neg}) = (x'_1, x'_2, \ldots, x'_s)$, where $x'_k = PI(x_k), 1 \leqslant k \leqslant s$. For example, the positive element partner of negative element $e_{neg} = (\neg a, b, \neg c)$ is $PE(e_{neg}) = (a, b, c)$.

The *maximum positive element of a negative element* $e_{neg} = (x_1, x_2, \ldots, x_s)$ is a positive element that consists of all positive items in e_{neg}, denoted as $MPE(e_{neg})$, i.e., $MPE(e_{neg}) = (x'_1, x'_2, \ldots, x'_s)$, where $x'_k \in E^+, 1 \leqslant k \leqslant s$. For example, the maximum positive element partner of negative element $e_{neg} = (\neg a, b, \neg c)$ is $MPE(e_{neg}) = (b)$.

The negative items in an element share certain logic relationships, which can be categorized in terms of two inferential coupling relationships between items: *conjunction coupling* and *disjunction coupling* [5, 9, 56, 57]. We define them below.

Coupling 1 (Conjunction Coupling). *If any one reverse item of the negative items in element e is not allowed to occur, these negative items are coupled with a conjunction relationship. e is called a* conjunction element, *represented as* $(x_1 \wedge x_2 \wedge \ldots \wedge x_s)$.

A conjunction element e_{ce} is called a *sub-element of positive element* e_{pos} and e_{pos} is a *super-element* of e_{ce}, denoted as $e_{ce} \subseteq_{ce} e_{pos}$, if all positive items of e_{ce} appear in e_{pos} and no reverse item of negative items in e_{ce} appears in e_{pos}.

Example 1. *A conjunction element* $(\neg a \wedge b \wedge \neg c)$ *is a sub-element of positive element* (b, d)*, but not a sub-element of positive element* (b, c, d)*, since item c occurs in* (b, c, d).

Coupling 2 (Disjunction Coupling). *If the reverse items of the negative items in element e are required not to co-occur, in other words, at least one reverse item of these negative items does not occur, these negative items are coupled by a disjunction relationship. e is called a* disjunction element, *represented as* $(x_1 \vee x_2 \vee \ldots \vee x_s)$.

A disjunction element e_{de} is called a *sub-element of positive element* e_{pos} and e_{pos} is a *super-element* of e_{de}, denoted as $e_{de} \subseteq_{de} e_{pos}$, if all positive items of e_{de} appear in e_{pos} and at least one reverse item of negative items in e_{de} does not appear in e_{pos}.

Example 2. *A disjunction element* $(\urcorner a \vee b \vee \urcorner c)$ *is a sub-element of positive element* (b, c, d)*, since items a and c do not co-occur in* (b, c, d)*, even though item a occurs alone.*

The conjunction couplings between negative items in an element specify the “AND” logical relationships, and all negative conditions are required to be satisfied simultaneously. For example, if a positive element e_{pos} is a super-element of a conjunction element $(\urcorner a \wedge b \wedge \urcorner c)$, it means that “$e_{pos}$ contains $\urcorner a$” AND “e_{pos} contains b” AND “e_{pos} contains $\urcorner c$”.By contrast, the disjunction couplings in an element specify the “OR” logical relationship between negative items, and at least one of these negative conditions is required to be satisfied. For example, if positive element e_{pos} is a super-element of a disjunction element $(\urcorner a \vee b \vee \urcorner c)$, it means that “($e_{pos}$ contains b)” AND “(e_{pos} contains $\urcorner a$” OR “e_{pos} contains $\urcorner c$)”.If a positive element e_{pos} is a super-element of negative element e_{neg}, e_{pos} is also called *containing* e_{neg}, denoted as $e_{neg} \subseteq e_{pos}$; otherwise denoted as $e_{neg} \nsubseteq e_{pos}$.

3.3 Sequence and Its Properties

A *sequence S* is an ordered list of elements, denoted as $S =< e_1, e_2, \ldots, e_s >$, where e_j $(1 \leqslant j \leqslant s)$ is an element. An element e in a sequence is also called an *element of a sequence*, denoted as $e = (x_1, x_2, \ldots, x_m) \in S, x_k \in I, 1 \leqslant k \leqslant m$. For simplicity, if only one item is contained in an element, the brackets around it can be omitted, i.e., element $e_k = (x)$ can be represented as $e_k = x$. In general, in NSA, an item is only allowed to appear at most one time in an element, but it can occur multiple times in several divergent elements of a sequence.

A sequence contains five attributes: *state*, *length*, *size*, *width* and *frequency*. The *state of a sequence* is either positive or negative: if a sequence $S =< e_1, e_2, \ldots, e_s >$ is composed of positive elements only, it is a *positive sequence*; if a sequence S contains at least one negative element, it is a *negative sequence*. The *length of sequence S* is the total number of items in all elements in S, denoted as $length(S) = \sum_{k=1}^{s} size(e_k)$. A sequence S is called a *l-length* or *l-item sequence* if $length(S) = l$. The *size of sequence S* is the total number of elements in S, denoted as $size(S) = s$. A sequence S is called a *s-size* or *s-element sequence* if $size(S) = s$. In addition, the *negative size* of sequence S is the number of negative elements in S, denoted as $neg - size(S) = |E|, \forall e' \in E, e' \in S \cup e' \cap E^- \neq \varnothing$. Moreover, the k-th element of sequence S is denoted as $S[k]$, and the *width of sequence S* is the maximum size of any element in S, denoted as $width(S) = \max\limits_{1 \leqslant k \leqslant size(S)} size(S[k])$. A sequence S is called a *w-width sequence* if $width(S) = w$. For example, a negative sequence $S_{neg} =< (\urcorner a, b, \urcorner c), (a, c), \urcorner(b, d), a, \urcorner c >$ consists of nine items, five elements, and three negative elements, and its maximum-size element is $(\urcorner a, b, \urcorner c)$, which is a 3-size element. Accordingly, S is a 9-length, 5-size, 3-neg-size and 3-width sequence, and $S[3] = \urcorner(b, d)$. The *positive sequence partner* of a negative sequence $S_{neg} =< e_1, e_2, \ldots, e_s >$ is a sequence that transforms all its elements to their positive element partners, denoted as $PS(S_{neg})$, i.e., $PS(S_{neg}) =< e'_1, e'_2, \ldots, e'_s >$ where $e'_k = PS(e_k), 1 \leqslant k \leqslant s$. For example, the positive sequence partner of negative sequence $S_{neg} =< (\urcorner a, b, \urcorner c), (a, c), \urcorner(b, d), a, \urcorner c >$ is $PS(S_{neg}) =< (a, b, c), (a, c), (b, d), a, c >$.

Definition 3.1 (Sub-sequence & Super-sequence of A Positive Sequence). A positive sequence $S_\alpha =< e_{\alpha_1}, e_{\alpha_2}, \ldots, e_{\alpha_m} >$ is called a *sub-sequence* of another positive sequence $S_\beta =< e_{\beta_1}, e_{\beta_2}, \ldots, e_{\beta_n} >$, and S_β is called a *super-sequence* of S_α, denoted as $S_\alpha \subseteq_{pos} S_\beta$. If $\forall e_{\alpha_k} \in S_\alpha, 1 \leqslant k \leqslant size(S_\alpha)$, there exists $j_k, 1 \leqslant j_k \leqslant size(S_\beta)$ such that $e_{\alpha_k} \subseteq e_{\beta_{j_k}}$ and $1 \leqslant j_1 < j_2 < \ldots < j_{size(S_\alpha)} \leqslant size(S_\beta)$, which also means that *S_β contains S_α*.

Example 3. *Given positive sequences $S_\alpha =< (a, c), c >$ and $S_\beta =< (a, b, c), (a, c), (b, d), a, c >$, then $S_\alpha \subseteq_{pos} S_\beta$.*

A *sequence dataset* is a set of binary tuples, denoted as $D = \{< Sid, S >\}$, where S is a positive sequence and Sid is the *sequence identifier* of S. A sequence in the sequence dataset is called a *data sequence*, and a positive element in a data sequence is called a *data element*. The *size* of sequence dataset D is the number of binary tuples in D, denoted as $|D|$. The set of binary tuples which contains sequence S is denoted as $\{< S >\}$.

The *support count* of S in D is the number of $\{< S >\}$, i.e., the number of data sequences in D that contain S, which is denoted as $SC(S) = |\{< S >\}| = |\{< Sid', S' >| < Sid', S' > \in D \wedge S \subseteq_{pos} S'\}|$. Furthermore, the *support* of S in D is the percentage of its support count with respect to the size of the sequence dataset, denoted as $sup(S) = \frac{SC(S)}{|D|}$.

Definition 3.2 (Positive Sequential Pattern & Negative Sequential Pattern). Given a predefined minimum support threshold min_sup, a sequence S is called a *frequent sequence* or a *sequential pattern* if the support of S is not less than min_sup, i.e., $sup(S) \geqslant min\_sup$. S is called an *infrequent sequence* if its support is less than min_sup, i.e., $sup(S) < min\_sup$. If a sequential pattern is a positive sequence, it is called a *positive sequential pattern* (PSP); if this pattern is a negative sequence, it is called a *negative sequential pattern* (NSP).

Definition 3.3 (NSP Mining). Given a sequence dataset D and a minimum support threshold min_sup predefined by users, *sequential pattern mining* is a process of discovering all the sequential patterns with supports not less than min_sup. If the mining process only focuses on discovering all PSPs, it is called *PSP mining*; if the mining process aims to discover all the sequential patterns including PSPs and NSPs, it is called *NSP mining*. PSP mining is a subtask of NSP mining.

Intrinsically, NSP has different semantics from PSP. A PSP means that its elements highly likely occur sequentially, while an NSP emphasizes that its negative elements would not occur in certain situations. For example, given a PSP $S_{psp} =< a, b, d >$ and a NSP $S_{nsp} =< a, \neg c, d >$, for a data sequence $S_{data} =< e_1, e_2, \ldots, e_m >$, we can see that $S_{psp} \subseteq S_{data}$ if $\exists 1 \leqslant i_1 \leqslant i_2 \leqslant i_3 \leqslant m$ such that $a \subseteq e_{i_1}, b \subseteq e_{i_2}, d \subseteq e_{i_3}$, while $S_{nsp} \subseteq S_{data}$ if $\exists 1 \leqslant i_1 \leqslant i_2 \leqslant m$ such that $a \subseteq e_{i_1}, d \subseteq e_{i_2}$ and $\forall i_n, (i_1 < i_n < i_2)$ such that $c \nsubseteq e_{i_n}$. For instance, suppose $S_{data} =< (a, c), (b, e), (c, e), (d, f) >$, we can see that $S_{psp} \subseteq S_{data}$ but $S_{nsp} \nsubseteq S_{data}$, because $a \subseteq (a, c)$, $d \subseteq (d, f)$ but $c \subseteq (c, e)$. The formal definitions of various negative containments are investigated in Section 5, but as can be seen from the above example, NSP mining cannot be simply considered as a sub-field of traditional PSP mining and existing PSP mining algorithms cannot be directly applied to mining NSPs, due to the intricate nature of negative entities.

3.4 NSP Formats

Different NSP formats and problem statements are defined in NSP mining compared to those for PSPs, due to the much richer and more challenging nature of NSPs as discussed in Section 2.2. There are no universal NSP formulas defined in the literature. For example, NSPM, NFSPM and PNSPM define NSP in the form of $< e_1, e_2, \ldots, \neg e_s >$, where a negative element can only be located at the end. They discover NSPs w.r.t. high support. MSIS, MBFIFS, CPNFMLSP and CPNFSP identify the NSP in the form of $< e_1, \neg e_2 >$, $< \neg e_1, e_2 >$ or $< \neg e_1, \neg e_2 >$, and filter invalid NSC with low support or low interestingness. PNSP, NegGSP, GA-NSP and e-NSP mine the complete set of NSPs with predefined constraints, whose supports are more than the given threshold. Among these algorithms, NFSPM and MBFIFS consider numerical attributes and introduce a fuzzy membership function to define fuzzy support and fuzzy interestingness for filtering. Unlike these algorithms for mining NSPs on static datasets, the Incremental CPNFSP discovers and updates NSPs on dynamic datasets.

4 CONSTRAINT SETTINGS

As discussed in Section 2.2, NSP mining involves much higher computational complexity and a much larger search space than PSP mining, and part of the discovered NSP may be meaningless, thus constraints have been set at different levels and aspects to make a trade-off between computational efficiency and pattern coverage and make NSA less costly and more feasible [72].

While existing PSP constraints can still be empowered on the positive components in NSA, whether they can be applied to the negative components is an open issue. However, there may be different forms of constraints, e.g., structural and logic constraint, and semantic constraints, to be introduced on specific negative entities or aspects of NSA, which are called negative constraints. A logic *constraint* C for sequential pattern mining can be considered as a boolean function $C(S)$ defined on a sequential pattern S. Instead, semantic constraints were used in traditional PSP mining and frequent itemset mining, which usually represent user interest and focus on and confine the discovered patterns to a particular subset satisfying some strong conditions [45, 66]. Existing constraints introduced in NSP mining mainly focus on restricting the form of NSP to be mined in order to lower the computational cost of NSP mining, which are specific to NSA. We will investigate the potential semantic constraints of NSP mining as open issues, which are discussed in Section 7.

In this work, only the constraints adopted by existing NSP mining algorithms are summarized. We categorize the constraint types into *size constraint*, *frequency constraint*, *format constraint* and *negation constraint*. We review and define them below.

4.1 Size Constraint (SC)

A *size constraint* (SC) sets a specific restriction on the size of entities (items, elements or sequences) and reduces the search space by avoiding testing whether some NSCs of excessive size are contained by certain data sequences. Below, we discuss the *item size constraint* and *element size constraint*.

Constraint 1 (Item Size Constraint (ISC)). *An* item size constraint *(ISC) is defined as follows: A data element e_{data} cannot contain a negative element e_{neg} with a size larger than e_{data}, i.e., $e_{neg} \nsubseteq e_{data}$ if $size(e_{data}) < size(e_{neg})$. It is in the form of $C_{ISC}(S_{neg}) \equiv (|\{< Sid_{data}, S_{data} >| (< Sid_{data}, S_{data} > \in D) \wedge (S_{neg} \subseteq S_{data}) \wedge (\forall e_{data} \in S_{data}, e_{neg} \in S_{neg} : (e_{neg} \cap E^{-} \neq \emptyset) \wedge (e_{neg} \subseteq e_{data})\ s.t.\ size(e_{data}) \geqslant size(e_{neg})\}| \geqslant min\_sup \times |D|)$, where D is the sequence dataset.*

Example 4. *The data element $e_{data} = (a, c)$ cannot contain negative element $e_{neg} = (a, \neg b, c)$ since $size(e_{data}) = 2$ while $size(e_{neg}) = 3$.*

ISC is a practical size constraint set on elements, and is adopted by NegGSP and GA-NSP [74, 75]. Since many applications only focus on the negative element whose positive element partner exists in the sequence dataset, and since excessive negative elements are often neglected, discovering NSPs that have excessively large elements is wasteful and futile, thus ISC can avoid testing these uninteresting NSCs. For example, for data element $e_{data} = (a, c)$, only the NSCs including 3 negative elements $(a, \neg c)$, $(\neg a, c)$ and $\neg(a, c)$ need to be tested to discover whether they are contained by e_{data} if ISC is applied. Otherwise, it is necessary to test NSCs that have more negative elements, such as $(a, \neg b, \neg c)$,$(a, \neg c, \neg d)$ and $(a, \neg b, \neg c, \neg d)$. The introduction of ISC leads to the loss of NSPs that have longer elements, especially when a sequence dataset is insufficient and sparse.

Constraint 2 (Element Size Constraint (ESC)). *An* element size constraint *(ESC) is defined as follows: A data sequence S_{data} cannot contain and support a negative sequence S_{neg} with a size larger than the data sequence, i.e., $S_{neg} \not\subseteq S_{data}$ if $size(S_{data}) < size(S_{neg})$. It is in the form of $C_{ESC}(S_{neg}) \equiv (|\{< Sid_{data}, S_{data} >| (< Sid_{data}, S_{data} >\in D) \wedge (S_{neg} \subseteq S_{data}) \wedge (size(S_{data}) \geqslant size(S_{neg})\}| \geqslant min\_sup \times |D|)$.*

Example 5. *The negative sequence $S_{neg} =< (a, c), \neg(b, d), a, \neg c >$ cannot be contained by data sequence $S_{data} =< (a, c), c >$ since $size(S_{data}) = 2$ while $size(S_{neg}) = 4$.*

ESC is set on the hierarchy of sequences and is adopted by PNSP to prune NSCs with excessive elements. It guarantees that the maximum size of the NSPs discovered from dataset D is $size(S)$, where $< Sid, S >\in D$ and $\forall < Sid\prime, S\prime >\in D, size(S\prime) \leqslant size(S)$, and maintains the upper boundary of the search space [28]. In some applications, big-size NSPs express worthless information, while it is very costly to search a large space to discover such NSPs. Therefore, ESC can avoid testing uninteresting and big-size NSCs and can shrink the search space. For example, if the data sequence $S_{data} =< (a, c), a >$, then NSC $S_{neg} =<\neg(a, b, c), (a, c), \neg(b, d), a, \neg c >$ means that an element does not appear between data elements, which may be meaningless in some applications. However, the introduction of ESC causes the loss of informative NSPs that are of large size and loses the knowledge extracted from them.

4.2 Frequency Constraint (FC)

In contrast to size constraints which aim to avoid the testing of some NSCs and are only adopted by a small number of algorithms, *frequency constraints* (FCs) reduce the search space by limiting the scale of possible entities that constitute NSCs or generating specific NSCs. FCs are widely used in NSP mining. Common FCs are listed below, and the reasons for their proposal are explained.

Constraint 3 (Item Frequency Constraint (IFC)). *The* item frequency constraint *(IFC) is defined as follows: A negative item i_{neg} cannot appear in an NSC unless its reverse item $RI(i_{neg})$ is a frequent positive item, i.e., $C_{IFC}(S_{NSC}) \equiv (\forall i_{neg} \in e_{neg}, e_{neg} \in S_{NSC}, sup(< RI(i_{neg}) >) \geqslant min\_sup)$.*

Example 6. *The negative sequence $< (a, c), \neg(b, d), a, \neg c >$ cannot be generated as an NSC unless all items a, b, c and d are frequent.*

IFC is set on the item level and specifies that NSC cannot contain the reverse items of infrequent positive items as negative items. Specific IFCs were adopted by NegGSP and GA-NSP [74, 75]. The IFC is introduced because, in some real-life applications, only the non-occurrence of focused items draws attention, and the NSP related to the non-occurrence of infrequent items may be regarded as less valuable. IFC can enable NSP mining algorithms to discover essential NSPs without searching an unnecessary space by pruning those NSCs containing negative items whose positive item partners are infrequent. Compared with other frequency constraints, however, the search space of NSP mining algorithms is still quite large and a large number of NSCs can be generated. This is because the number of possible combinations of frequent positive items and their reverse items is still enormous, of which the reverse elements may not be frequent positive elements.

Constraint 4 (Element Frequency Constraint (EFC)). *The* element frequency constraint *(EFC) is defined as follows: A negative element e_{neg} cannot appear in NSCs unless its positive element partner $PE(e_{neg})$ is a frequent positive element, i.e., $C_{EFC}(S_{NSC}) \equiv (\forall e_{neg} \in S_{NSC}, sup(< PE(e_{neg}) >) \geqslant min\_sup)$.*

Example 7. *The negative element $(\neg a, b, \neg c)$ cannot appear in any NSC unless (a, b, c) has a greater support than the user-given threshold, even if all three items a, b and c are frequent positive items.*

EFC is set on elements and is a stricter frequency constraint than IFC. EFC was adopted in PNSP, NSPM, NFSPM and PNSPM, and it is introduced because sometimes only the non-occurrence of valuable item combinations of sufficient frequency is of business interest, and EFC removes trivial situations that occur coincidentally such as in PNSP and NSPM. NSP algorithms adopting EFC can only discover a coverage of NSPs smaller than those adopting IFC, since they cannot generate those NSCs consisting of the negative elements whose positive element partners are infrequent.

Constraint 5 (Element Independence & Frequency Constraint (EIFC)). *The* element independence & frequency constraint *(EIFC) is defined as follows: Only NSCs in the form of* $< e_1, \neg e_2 >$, $< \neg e_1, e_2 >$ *and* $< \neg e_1, \neg e_2 >$ *can be generated as NSCs, and they can be NSPs if both elements* e_1 *and* e_2 *are frequent and* $e_1 \cap e_2 = \emptyset$*, i.e.,* $C_{EIFC}(S_{NSC}) \equiv (size(S_{NSC}) = 2, neg - size(S) \geqslant 1,\ and\ \forall e_1, e_2 \in S_{NSC}, sup(< PE(e_1) >) \geqslant min\_sup, sup(< PE(e_2) >) \geqslant min\_sup, PE(e_1) \cap PE(e_2) = \emptyset)$.

Example 8. *The negative sequence* $< \neg(a, b), (b, c) >$ *cannot be generated as NSC since* $(a, b) \cap (b, c) \neq \emptyset$ *regardless of whether* (a, b) *and* (b, c) *are frequent.*

EIFC is also set on elements and is an extension of EFC which considers the intersections between elements, and which was adopted by MSIS, MBFIFS, CPNFMLSP, CPNFSP and Incremental CPNFSP. EIFC-enabled algorithms mine NSPs in a far smaller search space and generate a much smaller number of NSCs. However, it is a quite strict frequency constraint, requiring that no NSP that denies the given formulas can be discovered. Accordingly, the pattern coverage of EIFC-based algorithms is even smaller.

Constraint 6 (Sequence Frequency Constraint (SFreC)). *The* sequence frequency constraint *(SFreC) is defined as follows: A negative sequence* S_{neg} *cannot be generated as an NSC unless its positive sequence partner is a PSP, i.e.,* $C_{SFreC}(S_{neg}) \equiv (sup(PS(S_{neg})) \geqslant min\_sup)$.

Example 9. *The negative sequence* $< (\neg a, b, \neg c), (a, \neg c), \neg(b, d), a, \neg c >$ *can be an NSC only if* $< (a, b, c), (a, c), (b, d), a, c >$ *is a PSP.*

SFreC is set on sequences. To the best of our knowledge, SFreC is the strictest frequency constraint proposed so far. It maintains a minimum number of generated NSCs and discover the least number of NSPs compared to other constraints, and was adopted by SpamNeg, e-NSP and SAPNSP. SFreC is introduced based on the hypothesis that users are only interested in the absence of certain frequent elements, whose positive element partners appearing in NSPs should have high frequency. Though the algorithms adopting SFreC can only discover a small coverage of NSPs because of the strictness of SFreC, they can avoid generating a large number of NSCs and enable the use of a set theory-based mining approach such as e-NSP, which leads to super high efficiency in NSP mining. The cost is that algorithms adopting SFreC maintain the smallest coverage of NSPs compared to other existing algorithms, and all the NSPs whose positive sequence partners are not PSPs are missed.

4.3 Format Constraint

In addition to frequency constraints, format constraints (ForC) are a category of constraints placed on elements to reduce the search space by limiting the formulas of NSPs to be discovered and avoiding the generation of NSCs that are inconsistent with the given formulas. Common format constraints include *continuity format constraint* (CFC), *location format constraint* (LFC), and *size format constraint* (SForC).

Constraint 7 (Continuity Format Constraint (CFC)). *The continuity format constraint (CFC) is defined as follows: Two or more continuous negative elements in an NSC are not allowed, i.e.,* $C_{CFC}(S_{neg}) \equiv (\forall k : 1 \leqslant k \leqslant size(S_{neg}) - 1,\ if\ S_{neg}[k] \cap E^{-} \neq \emptyset,\ then\ S_{neg}[k + 1] \cap E^{-} = \emptyset)$.

Example 10. *The negative sequence* $S_\alpha =< (a,b,c),(a,c),\neg(b,d),a,\neg c >$ *satisfies constraint* 7 *while* $S_\beta =< (a,b,c),\neg(a,c),\neg(b,d),a,\neg c >$ *denies.*

CFC specifies that adjacent negative elements are not permitted in an NSC and this is a practical format constraint introduced by most existing NSP mining algorithms, such as PNSP, Neg-GSP, GA-NSP and e-NSP. The reasons why CFC is widely adopted include the following. On one hand, if multiple adjacent negative elements are allowed in an NSP, a potentially infinite number of NSCs will be generated and tested even in a small dataset, which will lead to extremely high computational complexity. For example, a data sequence $< (a,b,c),(a) >$ can support NSCs in the form of $< (a,b,c),\neg(a,c),\neg(b,d),a,\neg c >$, $< (a,b,c),\neg(a,c),\neg(a,c),\neg(b,d),a,\neg c >$ and $< (a,b,c),\neg(a,c),\dots,\neg(a,c),\neg(b,d),a,\neg c >$. On the other hand, if adjacent negative elements exist in an NSC, their ordering is sophisticated for many applications, and it would be quite difficult to distinguish the correct order of those negative elements if no positive elements exist between them. For example, the negative sequence S_β specifies that neither (a,c) nor (b,d) appears between elements (a,b,c) and a. However, it is quite unclear whether element (a,c) does not occur before (b,d) or after (b,d). In fact, S_β may be also represented as $< (a,b,c),\neg(a,b,c,d),a,\neg c >$, which also means that none of a,b,c or d occurs between (a,b,c) or a. Therefore, S_β may be meaningless in some applications, and CFC can avoid generating invalid NSCs to reduce the search space.

However, for some applications, sliding windows may be defined to distinguish the periods in which elements may occur or may not occur [72]. NSPs containing consecutive negative elements may be quite informative but they cannot be discovered under CFC. For example, in a medical service, $S_\gamma =< (a,b,c),\neg(a,c),\neg(b,d),a,X >$ specifies that, if a series of treatments (a,b,c) are undertaken in the first week, the treatment (a,c) are not taken in the second week, and the treatments (b,d) are not taken in the third week, then the outcome X appears after another treatment a is taken. By contrast, the treatment sequence $S_\delta =< (a,b,c),\neg(b,d),\neg(a,c),a,Y >$ specifies that if the treatment combination (b,d) is not taken in the second week and the treatment combination (a,c) is not taken in the third week, then the outcome Y appears. If both S_γ and S_δ are NSPs, then the non-occurring order of $\neg(a,c)$ and $\neg(b,d)$ may be associated with the outcomes of medical treatments.

Constraint 8 (Location Format Constraint (LFC)). *The location format constraint (LFC) is defined as follows: Negative elements can only appear in certain locations of an NSC.*

An example of LFC is that negative elements can only be located at the end of a sequence, and NSCs can only be in the form of $< e_1,e_2,\dots,\neg e_s >$ where $e_k \subseteq E^+, 1 \leqslant k \leqslant s$. Such LFCs were adopted by NSPM, NFSPM and PNSPM, and emphasize the relationships between the preceding sub-sequence $< e_1,e_2,\dots,e_{s-1} >$ and the target sub-sequence $<\neg e_s >$. For example, the negative sequence $< (a,b,c),(a,c),(b,d),a,\neg c >$ is valid while $< (\neg a,b,\neg c),(a,c),\neg(b,d),a,\neg c >$ is invalid.

Another widely adopted LFC is that there can be only two elements in an NSC and all positive items can only be included in a positive element at the beginning or end of an NSC. This LFC is adopted by MSIS, MBFIFS, CPNFMLSP, CPNFSP and Incremental CPNFSP. For example, the negative sequences $<\neg(a,c),(b,d) >$ and $< (a,c),\neg(b,d) >$ are valid while $< (\neg a,c),(b,\neg d) >$ is invalid. LFC causes algorithms to generate fewer NSCs, which is only concerned by application requirements. However, any NSPs containing negative elements in undefined locations are lost and result in a significantly smaller coverage.

Constraint 9 (Size Format Constraint (SForC)). *The size format constraint (SForC) is defined as follows: Each negative element in NSCs is composed of only one negative item, i.e.,* $C_{SForC}(S_{neg}) \equiv (\forall k : 1 \leqslant k \leqslant size(S_{neg}),\ if\ S_{neg}[k] \cap E^- \neq \varnothing,\ then\ size(S_{neg}[k+1]) = 1)$.

Example 11. *The negative sequence* $<\neg a,(a,c),b,a,\neg c >$ *satisfies Constraint* 9 *while* $< (\neg a,b,\neg c),(a,c),\neg(b,d),a,\neg c >$ *is invalid.*

SForC is adopted by SpamNeg [68]. Because of the strictness of SForC, algorithms with SForC can avoid generating a large number of NSCs and maintain a small search space. However, they can only cover a small proportion of the full pattern set, and the NSPs containing negative elements with multiple negative items cannot be mined. Adopting a rational LFC or SForC can reduce the number of NSCs and can ensure that algorithms focus only on discovering interesting NSPs. However, this may also cause the loss of informative NSPs which reject the adopted format constraints.

4.4 Negative Element Constraint (NEC)

The *negative element constraint* (NEC) is set on elements and prunes invalid NSCs by formulating the composition of a negative element.

Constraint 10 (Negative Element Constraint (NEC)). *An NEC is defined as follows: The smallest negative unit in an NSC is required to be an element; if an element consists of more than one item, either all or none of these items are allowed to be negative, i.e.,* $C_{NEC}(S_{neg}) \equiv (\forall k : 1 \leqslant k \leqslant size(S_{neg}),\ if\ S_{neg}[k] \cap E^* \neq \emptyset,\ then\ S_{neg}[k] \cap (E - E^*) = \emptyset)$*, where* $E^* \in \{E^+, E^-\}$.

Example 12. *The negative sequence* $<(a,b), \urcorner(c,d)>$ *satisfies Constraint 10 while* $S_\alpha =< (\urcorner a, b), (c, \urcorner d) >$ *and* $S_\beta =< (a,b), (\urcorner c, d) >$ *does not.*

NEC is widely introduced into NSP mining algorithms such as NegGSP, GA-NSP, e-NSP and SAPNSP. NEC is introduced to reduce the search space and lower the computational complexity by avoiding handling improper NSCs, especially for mining NSP in a dense dataset. However, improper NSPs may be quite informative for some applications. In the above case, whether item *a* co-occurs with *b* if S_α and S_β are both NSPs results in absolutely different consequences. Algorithms with NEC cannot capture this knowledge. Therefore, NSP mining algorithms with a loose NEC can deliver more informative results, allowing the improper negative elements of NSPs to be discovered.

Among the constraints listed above, *size constraint* are *support-related*, i.e., they are applied to confine how a data sequence contains a pattern. The sequence dataset is required to be examined once in order to validate whether a pattern can satisfy *size constraint*. With regard to other constraints, whether the constraints are satisfied can be determined only be the patterns themselves, without considering the sequence dataset to which these patterns are applied.

Table 3 summarizes the constraint settings and their relevant constraint type, hierarchy, definitions, and pros and cons with respect to existing NSP algorithms.

Table 3. Constraint Settings in Existing NSP Algorithms

Type	Entity	Constraint	Definition	Advantage	Disadvantage	Algorithm
SC	Element	ISC	Data element e_{data} cannot contain negative element e_{neg} whose size is larger than the data element, i.e., $e_{neg} \not\subseteq e_{data}$ if $size(e_{data}) < size(e_{neg})$	Avoid testing NSCs with elements of excessive size	Missing the NSPs with longer elements, especially for insufficient and sparse sequence datasets	NegGSP [74], GA-NSP [75]
	Sequence	ESC	Data sequence S_{data} cannot contain and support negative sequence S_{neg} whose size is larger than the data sequence, i.e., $S_{neg} \not\subseteq S_{data}$ if $size(S_{data}) < size(S_{neg})$	Avoid testing large NSCs	Missing some knowledge induced from long-size NSPs	PNSP [28]

Table 3. Constraint Settings in Existing NSP Algorithms

Type	Entity	Constraint	Definition	Advantage	Disadvantage	Algorithm
FreC	Item	IFC	A negative item cannot appear in an NSC unless its reverse item is a frequent positive item	Avoid generating NSCs containing negative items whose positive item partners are infrequent	Search space is still quite large and a large number of NSCs can be generated	NegGSP [74], GA-NSP [75], e-NSPFI [26]
	Element	EFC	A negative element cannot appear in NSCs unless its positive element partner is frequent	Avoid generating NSCs that contain negative elements whose positive element partners are infrequent	Algorithms with EFC can only discover a smaller coverage of NSPs than IFC	PNSP [28], NSPM [36], NFSPM [38], PNSPM [37]
		EIFC	Only NSCs in the form of $< e_1, \neg e_2 >$, $< \neg e_1, e_2 >$ and $< \neg e_1, \neg e_2 >$ can be NSCs, and they can be NSPs if both e_1 and e_2 are frequent and $e_1 \cap e_2 = \emptyset$	Lead to a much smaller search space and generate far fewer NSCs	NSPs denying the given formulas cannot be discovered and the coverage with EIFC is also smaller	MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPNFSP [32]
	Sequence	SFreC	A negative sequence S_{neg} cannot be generated as an NSC unless its positive sequence partner is a PSP, i.e., $sup(PS(S_{neg})) \geqslant min\_sup$	Avoid generating NSCs whose positive sequence partners are infrequent and enable the set theory-based mining approach	Maintain a small coverage of NSPs and ensure that NSPs whose positive sequence partners are not PSPs are missed	SpamNeg [68], e-NSP [8], SAPNSP [39], e-msNSP [61]
ForC	Element	CFC	Two or more continuous negative elements in an NSC are not allowed	Avoid generating NSCs that contain continuous negative elements, which may be meaningless in some applications	Inapplicable for applications with sliding windows to distinguish between the order of elements and may miss knowledge deduced from NSPs that contain consecutive negative elements	PNSP [28], NegGSP [74], GA-NSP [75], e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]
		LFC	Negative elements can only appear in certain locations of an NSC	Generate a small number of NSCs, which is only concerned by application requirements	NSPs containing negative elements in undefined locations are lost and the coverage is small	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPNFSP [32]
		SForC	Each negative element is composed of only one negative item	Avoid generating a large number of NSCs and maintain a small search space	Have only a very small coverage and NSPs containing negative elements with multiple negative items cannot be discovered	SpamNeg [68]
NEC	Element	NEC	The smallest negative unit in an NSC is an element	Reduce the search space and lower the computational complexity by avoiding handling improper NSCs	Miss knowledge induced from improper NSPs	NegGSP [74], GA-NSP [75], e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]

5 NEGATIVE CONTAINMENT

Negative containment determines whether a negative sequence can be contained and supported by a given data sequence. With various constraints, a variety of definitions of negative containment have been given and adopted by existing NSP mining algorithms, leading to difference in pattern coverage. Here, we review the key concepts for, definitions of, pruning strategies for and coupling relationships in negative containment for NSA.

5.1 Key Concepts for Negative Containment

To clearly describe negative containment, the following relevant definitions are introduced.

Definition 5.1 (Element-id Set (EidS)). An *element identifier* is the order number of an element in a sequence. Given a sequence $S =< e_1, e_2, \ldots, e_s >$, $id(e_k) = k, 1 \leqslant k \leqslant s$ is the element identifier of element e_k. An *element-id set* of S, denoted as $EidS(S)$, is the set that includes all elements and their identifiers, i.e., $EidS(S) = \{(e_k, id(e_k)) \mid e_k \in S, 1 \leqslant k \leqslant s\} = \{(e_1, 1), (e_2, 2), \ldots, (e_s, s)\}$.

The element-id set including all positive elements and their identifiers of sequence S is called *positive element-id set* of S, denoted as $EidS^+(S)$. Similarly, the element-id set including all negative elements and their identifiers of sequence S is called the *negative element-id set* of S, denoted as $EidS^-(S)$. It is clear that $EidS(S) = EidS^+(S) \cup EidS^-(S)$.

Example 13. *Given a sequence* $S =< (\neg a, b, \neg c), (a, c), \neg(b, d), a, \neg c >$*, its* $EidS(S) = \{((\neg a, b, \neg c), 1), ((a, c), 2), (\neg(b, d), 3), (a, 4), (\neg c, 5)\}$*,* $EidS^+(S) = \{((a, c), 2), (a, 4)\}$ *and also* $EidS^-(S) = \{((\neg a, b, \neg c), 1), (\neg(b, d), 3), (\neg c, 5)\}$.

Definition 5.2 (Order-preserving Sequence (OPS)). For any subset $EidS'(S) = \{(e'_1, id(e'_1)), (e'_2, id(e'_2)), \ldots, (e'_m, id(e'_m))\}, 1 \leqslant m \leqslant s$ of $EidS(S)$, sequence $S_\alpha =< e'_1, e'_2, \ldots, e'_m >$ is called the *order-preserving sequence* of $EidS'(S)$, denoted as $S_\alpha = OPS(EidS'(S))$, if for $\forall e'_k, e'_{k+1} \in S_\alpha, 1 \leqslant k < m$, there exists $id(e'_k) < id(e'_{k+1})$.

Example 14. *In Example 13, given a subset of* $EidS(S)$ *as* $EidS'(S) = \{((\neg a, b, \neg c), 1), ((a, c), 2), (a, 4)\}$*, its* $OPS(EidS'(S)) =< (\neg a, b, \neg c), (a, c), a >$.

Definition 5.3 (Sub-sequence & Super-sequence of A Sequence). The sequence S_α is called a *sub-sequence* of another sequence S_β, and S_β is called a *super-sequence* of S_α, if $\exists EidS'(S_\beta) \subseteq EidS(S_\beta)$ such that $S_\alpha \subseteq OPS(EidS'(S_\beta))$, denoted as $S_\alpha \subseteq S_\beta$.

Example 15. *Given* $S_\alpha =< (\neg a, b, \neg c), c, a >$ *and* $S_\beta =< (\neg a, b, \neg c), (a, c), \neg(b, d), a, \neg c >$*,* $EidS(S_\beta) = \{((\neg a, b, \neg c), 1), ((a, c), 2), ((b, d), 3), (a, 4), (\neg c, 5)\}$*. Since* $EidS'(S_\beta) = \{((\neg a, b, \neg c), 1), (c, 2), (a, 3)\} \subseteq EidS(S_\beta)$ *and* $S_\alpha \subseteq OPS(EidS'(S_\beta))$*,* S_α *is a sub-sequence of* S_β*, i.e.,* $S_\alpha \subseteq S_\beta$.

Corollary 5.4 (Sub-sequence Corollary). *Given positive sequences* $S_\alpha =< e_{\alpha_1}, e_{\alpha_2}, \ldots, e_{\alpha_m} >$ *and* $S_\beta =< e_{\beta_1}, e_{\beta_2}, \ldots, e_{\beta_n} >$ *, if* $S_\alpha \subseteq_{pos} S_\beta$*, then* $S_\alpha \subseteq S_\beta$.

Definition 5.5 (Maximum Positive Sub-sequence (MPS)). Let S_{neg} be a negative sequence, $OPS(EidS^+(S_{neg}))$ is called the *maximum positive sub-sequence* of S_{neg}, denoted as $MPS(S_{neg})$.

Example 16. *Given a negative sequence* $S_{neg} =< (\neg a, b, \neg c), (a, c), \neg(b, d), a, \neg c >$*, its* $EidS^+(S_{neg}) = \{((a, c), 2), (a, 4)\}$*, therefore* $MPS(S_{neg}) =< (a, c), a >$

According to Definition 5.5, the maximum positive sub-sequence of a negative sequence includes all of its positive elements.

Definition 5.6 (1-neg-size Maximum Sub-sequence). Given a negative sequence S_{neg}, its sub-sequences $OPS(EidS^+(S_{neg}), e_{neg})$, where $(e_{neg}, id(e_{neg})) \in EidS^-(S_{neg})$, are called *1-neg-size maximum sub-sequences*, denoted as $1 - negMS(S_{neg})$. The sub-sequence set including all 1-neg-size maximum sub-sequences of S_{neg} is called *1-neg-size maximum sub-sequence set*, denoted as $1 - negMSS(S_{neg})$, i.e., $1 - negMSS(S_{neg}) = \{OPS(EidS^+(S_{neg}), e_{neg}) \mid \forall(e_{neg}, id(e_{neg})) \in EidS^-(S_{neg})\}$.

EXAMPLE 17. *In Example 16,* $1 - negMSS(S_{neg}) = \{< (\neg a, b, \neg c), (a, c), a >, < (a, c), \neg(b, d), a >, < (a, c), a, \neg c >\}$.

Definition 5.7 (1-neg-item Maximum Sub-sequence). Given a negative sequence S_{neg}, its sub-sequences $OPS(EidS^{+}(S_{neg}), e'_{neg})$ are called *1-neg-item maximum sub-sequences*, denoted as $1 - niMS(S_{neg})$, if there exists $(e_{neg}, id(e_{neg})) \in EidS^{-}(S_{neg})$ such that $MPE(e_{neg}) = MPE(e'_{neg})$ and $neg - size(e'_{neg}) = 1$. The sub-sequence set including all 1-neg-item maximum sub-sequences of S_{neg} is called a *1-neg-item maximum sub-sequence set*, denoted as $1 - niMSS(S_{neg})$, i.e., $1 - niMSS(S_{neg}) = \{OPS(EidS^{+}(S_{neg}), e'_{neg})\}, \forall e'_{neg}, \exists(e_{neg}, id(e_{neg})) \in EidS^{-}(S_{neg}), MPE(e_{neg}) = MPE(e'_{neg}), neg - size(e'_{neg}) = 1$.

EXAMPLE 18. *In Example 16,* $1 - negMSS(S_{neg}) = \{< (\neg a, b), (a, c), a >, < (b, \neg c), (a, c), a >, < (a, c), \neg b, a >, < (a, c), \neg d, a >, < (a, c), a, \neg c >\}$.

According to Definition 5.6, for any negative sequence S_{neg}, $neg - size(1 - negMS(S_{neg})) = 1$, $MPS(S_{neg}) \subseteq 1 - negMS(S_{neg})$ and $size(1 - negMSS(S_{neg})) = neg - size(S_{neg})$.

Definition 5.8 (First Sub-sequence Ending Position (FSE)). Given a data sequence $S_{data} =< de_1, de_2, \ldots, de_s >$ and a positive sequence S_{pos}, if $\exists k, 1 < k \leqslant s$ such that $S_{pos} \subseteq_{pos} < de_1, de_2, \ldots, de_k > \wedge S_{pos} \nsubseteq_{pos} < de_1, de_2, \ldots, de_{k-1} >$, then k is called *the first sub-sequence ending position* of S_{pos} in S_{data}, denoted as $FSE(S_{pos}, S_{data})$. In particular, if $S_{pos} \subseteq_{pos} < de_1 >$, then $FSE(S_{pos}, S_{data}) = 1$; if $S_{pos} \nsubseteq_{pos} S_{data}$, then $FSE(S_{pos}, S_{data}) = 0$.

Definition 5.9 (Last Sub-sequence Beginning Position (LSB)). Given a data sequence $S_{data} =< de_1, de_2, \ldots, de_s >$ and a positive sequence S_{pos}, if $\exists k, 1 \leqslant k < s$ such that $S_{pos} \subseteq_{pos} < de_k, de_{k+1}, \ldots, de_s > \wedge S_{pos} \nsubseteq_{pos} < de_{k+1}, de_{k+2}, \ldots, de_s >$, then k is called *the last sub-sequence beginning position* of S_{pos} in S_{data}, denoted as $LSB(S_{pos}, S_{data})$. In particular, if $S_{pos} \subseteq_{pos} < de_s >$, then $LSB(S_{pos}, S_{data}) = s$; if $S_{pos} \nsubseteq_{pos} S_{data}$, then $LSB(S_{pos}, S_{data}) = 0$.

EXAMPLE 19. *Given a data sequence* $S_{data} =< (a, b, c), (a, c), (b, d), a, c >$, $FSE(< (a, c) >, S_{data}) = 1$, $FSE(< (a, c), a >, S_{data}) = 2$, $FSE(< (a, d) >, S_{data}) = 0$, $LSB(< (a, c) >, S_{data}) = 2$,$LSB(< c >, S_{data}) = 5$, $LSB(< (a, d) >, S_{data}) = 0$.

As mentioned in Section 3, a sub-sequence may be contained by multiple sub-sequences of a positive sequence. FSE and LSB are introduced by e-NSP to determine the exact position of a data sequence that contains a given positive sequence for the first and last time respectively [8].

Definition 5.10 (Maximum Equivalent Element (MEE)). Given a set of items I and a data sequence S_{data}, for a positive element $e_{data} \in S_{data}$, its *maximum equivalent element*, denoted as $MEE(e_{data})$, is a negative element that contains all positive items in e_{data} and other negative items in $I - e_{data}$, i.e., $MEE(e_{data}) = (i_1, \ldots, i_s)$ where $\forall i_k \in MEE(e_{data}), (i_k \in e_{data}) \vee (i_k \in E^{-} \cap (I - e_{data}))$.

EXAMPLE 20. *Given* $S = \{a, b, c, d\}$ *and a data sequence* $S_{data} =< (a, b, c), (a, c), (b, d), a, c >$, *for element* $e = (a, c) \in S_{data}$, $MEE(e) = (a, \neg b, c, \neg d)$.

Definition 5.11 (Maximum Equivalent Sequence). Given a set of items I and a data sequence $S_{data} =< de_1, \ldots, de_{ds} >$, the *maximum equivalent sequence* of S_{data}, denoted as $MES(S_{data})$, is a negative sequence that transfers all elements into their maximum equivalent element, i.e., $MES(S_{data}) = (e_1, \ldots, e_{ds})$ where $\forall e_k \in MES(S_{data}), e_k = MEE(de_k)$.

EXAMPLE 21. *In Example 20,* $MES(S_{data}) =< (a, b, c, \neg d), (a, \neg b, c, \neg d), (\neg a, b, \neg c, d), (a, \neg b, \neg c, \neg d), (\neg a, \neg b, c, \neg d) >$.

It is worth noting that Definitions 5.10 and 5.11 are only applicable for data sequences.

5.2 Various Definitions of Negative Containment

Based on the above concepts defined for negative containment, we discuss various definitions of negative containment in existing NSP mining algorithms in this section. We define and formalize these definitions in alignment with the systematic settings adopted in this paper.

Containment 1 (Negative Cover). *A negative sequence $S_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$ is covered by a data sequence $S_{data} =< de_1, de_2, \ldots, de_{ds} >$, if the following two conditions are satisfied:*

(1) Its maximum positive sub-sequence of S_{neg} is contained by S_{data}, i.e., $MPS(S_{neg}) \subseteq_{pos} S_{data}$;

(2) $\forall ne_k \in S_{neg}$, where $(ne_k, id(ne_k)) \in EidS^{-}(S_{neg}), 1 \leqslant k \leqslant ns$, there exist integers p, q and r, $(p < q < r)$ such that $\exists ne_{k-1} \subseteq de_p$ and $\exists ne_{k+1} \subseteq de_r$, and $\exists de_q$ such that $PE(ne_k) \not\subseteq de_q$.

Example 22. *In Example 16, S_{neg} is covered by data sequences $S_\alpha =< b, (a, c), c, a >$ and $S_\beta =< b, (a, c), c, (b, d), a >$ but not covered by $S_\gamma =< b, (a, c), (b, d), a >$, since element c, which $PE(\neg(b, d)) \not\subseteq c$, exists between elements (a, c) and a in both S_α and S_β.*

The concept *negative cover* was first introduced in [74] and also adopted in [75] to reduce the number of generated NSCs. If a data sequence S_{data} covers a negative sequence S_{neg}, it is also called *base-support S_{neg}* [72], which is denoted as $S_{neg} \subseteq_{base} S_{data}$. The *base-support count* of a negative sequence S_{neg} in a sequence dataset D is the number of data sequences which *base-support S_{neg}* in D, denoted as $SC_{base}(S_{neg}) = |\{< Sid, S > | < Sid, S > \in D \wedge S_{neg} \subseteq_{base} S\}|$. The *base-support* of S_{neg} in D is the percentage of its base-support count with respect to the size of sequence dataset, denoted as $sup_{base}(S_{neg}) = \frac{SC_{base}(S_{neg})}{|D|}$.

According to Containment 1, it is easy to derive Corollary 5.12: MPS Corollary.

Corollary 5.12 (MPS Corollary). *Given a negative sequence S_{neg}, if $MSP(S_{neg})$ is not a PSP, then $sup_{base}(S_{neg}) < min\_sup$.*

Containment 2 (MPS-based Negative Containment). *A negative sequence $S_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$ is contained by a data sequence $S_{data} =< de_1, de_2, \ldots, de_{ds} >$, denoted as $S_{neg} \subseteq_{con} S_{data}$, if the following two conditions are satisfied:*

(1) The maximum positive sub-sequence of S_{neg} is contained by S_{data}, i.e., $MPS(S_{neg}) \subseteq_{pos} S_{data}$;

(2) $\forall ne_k \in S_{neg}$, where $(ne_k, id(ne_k)) \in EidS^{-}(S_{neg}), 1 \leqslant k \leqslant ns$, there exist integers p, q and r, $(p < q < r)$ such that $\exists ne_{k-1} \subseteq de_p$ and $\exists ne_{k+1} \subseteq de_r$, and $\forall de_q$ such that $PE(ne_k) \not\subseteq de_q$.

Example 23. *In Example 22, S_{neg} is only contained by S_α but not contained by S_β and S_γ, since element (b, d) can be found between elements (a, c) and a in both S_β and S_γ.*

The concept *MPS-based negative containment* was first introduced in [74] and also adopted in [75] to judge whether a data sequence supports a negative sequence. For an NSP mining algorithm adopting MPS-based negative containment, if a data sequence S_{data} contains a negative sequence S_{neg}, it is also called *negative-support S_{neg}*. The *negative-support count of a negative sequence S_{neg}* in a sequence dataset D is the number of data sequences which negative-support S_{neg} in D, denoted as $SC_{con}(S_{neg}) = |\{< Sid, S > | < Sid, S > \in D \wedge S_{neg} \subseteq_{con} S\}|$. The *negative-support* of S_{neg} in D is the percentage of its negative-support count with respect to the size of the sequence dataset, denoted as $sup_{con}(S_{neg}) = \frac{SC_{con}(S_{neg})}{|D|}$.

It is clear that if $S_{neg} \subseteq_{con} S_{data}$, then $S_{neg} \subseteq_{base} S_{data}$ and $sup_{con}(S_{neg}) \leqslant sup_{base}(S_{neg})$. In addition, Corollary 5.13 can be drawn.

Corollary 5.13 (Negative Cover Corollary). *Given a negative sequence $S_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$, where $(ne_1, id(ne_1)) \notin EidS^{-}(S_{neg})$ and $(ne_{ns}, id(ne_{ns})) \notin EidS^{-}(S_{neg})$, if $sup_{base}(S_{neg}) < min\_sup$, then for any super-sequence S'_{neg} of S_{neg}, i.e., $S_{neg} \subseteq S'_{neg}$, $sup_{con}(S'_{neg}) < min\_sup$.*

Compared with MPS-based negative containment, a number of stricter definitions of negative containment were proposed. *N-containment* was proposed in [28], and a stricter containment which we call *strictly-negative containment* was proposed in [8]. These are formalized as follows:

Containment 3 (N-containment). *A negative sequence* $S_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$ *is N-contained by a data sequence* $S_{data} =< de_1, de_2, \ldots, de_{ds} >$*, denoted as* $S_{neg} \subseteq_{nc} S_{data}$*, if the following two conditions are satisfied:*
(1) $S_{neg} \subseteq_{con} S_{data}$*;*
(2) $PS(S_{neg}) \nsubseteq_{pos} S_{data}$*.*

Example 24. *Given a negative sequence* $S_{neg} =< b, \neg(b, d), a >$ *and two data sequences* $S_\alpha =< (b, d), a, c >$ *and* $S_\beta =< (a, b, c), (b, d), a, c >$*, then* S_{neg} *is N-contained by* S_α *but not N-contained by* S_β*, since* $PS(S_{neg}) =< b, (b, d), a > \subseteq_{pos} S_\beta$

Similar to the definition of MPS-based negative containment, the *N-containment support count of a negative sequence* S_{neg} in a sequence dataset D is the number of data sequences which *N-contain* S_{neg} in D, denoted as $SC_{nc}(S_{neg}) = |\{< Sid, S > | < Sid, S > \in D \wedge S_{neg} \subseteq_{nc} S\}|$. The *N-containment support* of S_{neg} in D is the percentage of its *N-containment support count* with respect to the size of the sequence dataset, denoted as $sup_{nc}(S_{neg}) = \frac{SC_{nc}(S_{neg})}{|D|}$.

Comparing Containment 3 with Containment 2 for negative sequence S_{neg} and data sequence S_{data}, if $S_{neg} \subseteq_{nc} S_{data}$, then $S_{neg} \subseteq_{con} S_{data}$. However, if $S_{neg} \subseteq_{con} S_{data}$, S_{neg} may not be *N-contained* by S_{data}. For instance, in Example 24, $S_{neg} \subseteq_{con} S_\beta$ but $S_{neg} \nsubseteq_{nc} S_\beta$.

In addition to Containment 3, Lin *et al* proposed a similar representation, here called *N-end-containment*, and adopted it in [36–38] to fit their problem statement of NSP mining. Their work focuses on discovering NSPs which contain negative elements only at the end of a negative sequence, hence their adopted containment can be specified as follows.

Containment 4 (N-end-containment). *A negative sequence* $S_{neg} =< e_1, e_2, \ldots, e_s >$*, where* $(e_k, id(e_k)) \in EidS^+(S_{neg}), 1 \leqslant k < s$ *and* $(e_s, id(e_s)) \in EidS^-(S_{neg})$*, is N-end-contained by a data sequence* S_{data}*, denoted as* $S_{neg} \subseteq_{nec} S_{data}$*, if*
(1) $PS(S_{neg}) =< e_1, e_2, \ldots, PE(e_s) > \nsubseteq_{pos} S_{data}$*, and*
(2) its positive sub-sequence $< e_1, e_2, \ldots, e_{s-1} > \subseteq_{pos} S_{data}$*, i.e,* $MPS(S_{neg}) \subseteq_{pos} S_{data}$*.*

Example 25. *Given a negative sequence* $S_{neg} =< b, \neg d >$ *and two data sequences* $S_\alpha =< (a, c), (b, d), a, c >$ *and* $S_\beta =< (a, b, c), (a, c), (b, d), a, c >$*, then* S_{neg} *is N-end-contained by* S_α *but not N-end-contained by* S_β*, since* $PS(S_{neg}) =< b, d > \subseteq_{pos} S_\beta$

It is clear that Containment 3 satisfies the above Containment 4, but Containment 2 does not. For instance, in Example 25, $S_{neg} \subseteq_{con} S_\beta$ since element $b \subseteq (b, d)$ and element $(d \nsubseteq a) \wedge (d \nsubseteq c)$. However, $S_{neg} \nsubseteq_{sc} S_\beta$ since $FSE(MPS(S_{neg}), S_\beta) = 1$ and element $d \subseteq (b, d)$, and $S_{neg} \nsubseteq_{nec} S_\beta$ since $PS(S_{neg}) =< b, d > \subseteq_{pos} S_\beta$ and $d \subseteq (b, d)$.

Containment 5 (Strictly-negative Containment). *An m-neg-size negative sequence* $S_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$ *is strictly-negative-contained by a data sequence* $S_{data} =< de_1, de_2, \ldots, de_{ds} >$*, denoted as* $S_{neg} \subseteq_{sc} S_{data}$*, if (1)* $ns = 1$ *and* $m = 1$*,* $PS(S_{neg}) \nsubseteq_{pos} S_{data}$*; otherwise (2)* $\forall (ne_k, id(ne_k)) \in EidS^-(S_{neg}), 1 \leqslant k \leqslant m$*, one of the following three conditions is satisfied:*
(1) $(lsb = 1)$ *or* $(lsb > 1 \wedge PE(e_1) \nsubseteq < de_1, \ldots, de_{lsb-1} >)$*, when* $k = 1$*;*
(2) $(fse = ds)$ *or* $(0 < fse < ds \wedge PE(ne_{ns}) \nsubseteq < de_{fse+1}, \ldots, de_{ds} >)$*, when* $k = ns$*;*
(3) $(fse > 0 \wedge lsb = fse + 1)$ *or* $(fse > 0 \wedge lsb > fse + 1 \wedge PE(ne_k) \nsubseteq < de_{fse+1}, \ldots, de_{lsb-1} >)$*, when* $1 < k < ns$*, where* $fse = FSE(MPS(< ne_1, ne_2, \ldots, ne_{k-1} >), S_{data})$ *and* $lsb = LSB(MPS(< ne_{k+1}, \ldots, ne_{ns} >), S_{data})$*;*

Example 26. *Given a data sequence* $S_{data} =< (a,b,c),(a,c),(b,d),a,c >$, *and then (1) for* $S_{ns1} =< \neg a,b,d >$, *we have* $S_{ns1} \subseteq_{sc} S_{data}$ *since* $EidS^{-}(S_{ns1}) = (\neg a,1)$ *and* $lsb = 1$; *(2) for* $S_{ns2} =< d,c,\neg a >$, *we have* $S_{ns2} \subseteq_{sc} S_{data}$ *since* $EidS^{-}(S_{ns2}) = (\neg a,3)$ *and* $fse = 5$; *and (3) for* $S_{ns3} =< (a,c),\neg b,d,\neg c,a >$, *we have* $S_{ns3} \subseteq_{sc} S_{data}$. *Because* $EidS^{-}(S_{ns3}) = (\neg b,2),(\neg c,4)$, *for* $(\neg b,2)$, *we have* $fse = 1$, $fsb = 3$ *and* $PE(\neg b) \not\subseteq < (a,c) >$; *and for* $(\neg c,4)$, *we have* $fse = 3$, $fsb = 4$ *and* $(fse > 0 \wedge lsb = fse + 1)$.

For the NSP mining algorithm adopting the above *strictly-negative containment* concept, if a data sequence S_{data} *strictly-negative-contains* a negative sequence S_{neg}, it is also called *strictly-negative-support* S_{neg}, and the *strictly-negative-support count* of a negative sequence S_{neg} in a sequence dataset D is the number of data sequences which strictly-negative-support S_{neg} in D, denoted as $SC_{sc}(S_{neg}) = |\{< Sid,S >| < Sid,S > \in D \wedge S_{neg} \subseteq_{sc} S\}|$. The strictly-negative-support of S_{neg} in D is the percentage of its strictly-negative-support count with respect to the size of the sequence dataset, denoted as $sup_{sc}(S_{neg}) = \frac{SC_{sc}(S_{neg})}{|D|}$.

According to Containment 5, the following Corollary 5.14 *strictly-negative containment corollary* can be acquired.

Corollary 5.14 (Strictly-negative Containment Corollary). *Given a negative sequence* S_{neg} *and data sequence* S_{data}, *if* $S_{neg} \subseteq_{sc} S_{data}$, *then:*

(1) The maximum positive sub-sequence of S_{neg} *is contained by* S_{data}, *i.e.,* $MPS(S_{neg}) \subseteq_{pos} S_{data}$;

(2) $\forall ne_k \in S_{neg}, (ne_k, id(ne_k)) \in EidS^{-}(S_{neg}), 1 \leqslant k \leqslant ns, \exists ne_{k-1} \subseteq de_p$ *and* $ne_{k+1} \subseteq de_r$, *then* $\forall de_q, (p < q < r)$ *such that* $PE(ne_k) \not\subseteq de_q$.

Per Corollary 5.14, if $S_{neg} \subseteq_{sc} S_{data}$, then $S_{neg} \subseteq_{con} S_{data}$. However, if $S_{neg} \subseteq_{con} S_{data}$, S_{neg} may not be strictly contained by S_{data}. For example, given $S_{data} =< (a,c),(b,d)(a,c),a >$ and $S_{neg} =< (a,c),\neg(b,d),a,\neg c >$, $S_{neg} \subseteq_{con} S_{data}$ but $S_{neg} \not\subseteq_{sc} S_{data}$, since $FSE(MPS(< (a,c) >),S_{data}) = 1$, $LSB(MPS(< a,\neg c >),S_{data}) = 4$ and $PE(\neg(b,d)) \subseteq (b,d)$.

Comparing Containment 5 with Containment 3, a subsequent Corollary 5.15 *equivalence containment corollary* is proposed and proved as follows.

Corollary 5.15 (Equivalence Containment Corollary). *Given a negative sequence* S_{neg} *and a data sequence* S_{data}, *if* $S_{neg} \subseteq_{sc} S_{data}$, *then* $S_{neg} \subseteq_{nc} S_{data}$; *and if* $neg - size(S_{neg}) = 1$ *and* $S_{neg} \subseteq_{nc} S_{data}$, *then* $S_{neg} \subseteq_{sc} S_{data}$.

For Corollary 5.15, if $neg - size(S_{neg}) > 1$ and $S_{neg} \subseteq_{nc} S_{data}$, then S_{neg} may not be strictly-negative-contained by S_{data}, which can be seen in Example 27.

Example 27. *Given a negative sequence* $S_{neg} =< (\neg a,b,\neg c),(a,c),\neg(b,d),a,\neg c >$ *and a data sequence* $S_{data} =< b,(a,c),a,(b,d),a >$, *then* $S_{neg} \subseteq_{con} S_{data}$ *and* $S_{neg} \subseteq_{nc} S_{data}$ *but* $S_{neg} \not\subseteq_{sc} S_{data}$.

5.3 Pruning Strategies for Negative Containment

Since NSPs do not hold the property of downward closure, pruning strategies are required to lower the search space as well as computational complexity of NSP mining. Here we will discuss several pruning strategies which can be widely accepted.

As can be seen from Section 5.2, if negative sequence S_{neg} has a base-support smaller than threshold *min_sup*, none of its negative super-sequences can be NSPs. Therefore, the base-support of a negative sequence can be used to judge whether a negative sequence and its negative super-sequences need to be further generated and tested as NSCs. Pruning Strategy 1 *MPS pruning strategy* and Pruning Strategy 2 *Cover pruning strategy* are accordingly designed.

Pruning Strategy 1 (MPS Pruning Strategy). *Given a negative sequence* S_{neg}, *where* $(ne_1, id(ne_1)) \notin EidS^{-}(S_{neg})$ *and* $(ne_{ns}, id(ne_{ns})) \notin EidS^{-}(S_{neg})$, *if* $MPS(S_{neg})$ *is not a PSP, i.e.,* $sup(MPS(S_{neg})) < min\_sup$, S_{neg} *and its negative super-sequences cannot be NSPs and can be pruned.*

Pruning Strategy 2 (Cover Pruning Strategy). *Given a negative sequence S_{neg}, where $(ne_1, id(ne_1)) \notin EidS^-(S_{neg})$ and $(ne_{ns}, id(ne_{ns})) \notin EidS^-(S_{neg})$, if $sup_{base}(S_{neg}) < min\_sup$, then S_{neg} and its negative super-sequences cannot be NSPs and can be pruned.*

However, if Pruning Strategy 2 is adopted by an NSP mining algorithm which uses a joining-based NSC generation strategy, such as in NegGSP, it may lead to a loss of partial NSCs and a small coverage of discovered NSPs, since a l-length NSC generated by joining-based strategy may not be a super-sequence of its (l-1)-length seeds, especially when a disjunction coupling relationship exists in negative elements. For example, given a negative sequence $S_{neg} =< (a,c), \neg(b,d) >$ and a sequence dataset $D = \{< 1, < (a,c), b >>, < 2, < (a,c), d >>\}$, if the disjunction coupling relationship exists in negative elements, it can be seen that $SC_{con}(S_{neg}) = 2$, $sup_{base}(< (a,c), \neg b >) = 1$ and $sup_{base}(< (a,c), \neg d >) = 1$. Therefore, if $min\_sup = 2$, both $< (a,c), \neg b >$ and $< (a,c), \neg b >$ are pruned. According to the design of the joining operation adopted by NegGSP, NSC S_{neg} is generated by $<\neg(b,d) >$ and $< (a,c), \neg b >$ or $< (a,c), \neg d >$, and thus S_{neg} cannot be generated and discovered by NegGSP. If a sequence dataset is a dense dataset with a long data sequence, the NSP coverage of NegGSP will be much smaller. Another example is that, for a negative sequence $S_{neg} =< a, \neg a, b >$ and a sequence dataset $D = \{< 1, < a, b >>\}$, $SC_{con}(S_{neg}) = 1$, $sup_{base}(< a, \neg a >) = 1$ and $sup_{base}(<\neg a, b >) = 0$. If $min\_sup = 1$, then NSC $<\neg a, b >$ is pruned and NSP S_{neg} is overlooked. The third example is that, given a negative sequence $S_{neg} =< a, \neg(b,d) >$ and a sequence dataset $D = \{< 1, < (b,d), a >>\}$, $SC_{con}(S_{neg}) = 1$, $sup_{base}(< a, \neg b >) = 1$ and $sup_{base}(<\neg(b,d) >) = 0$. If $min\_sup = 1$, then NSC $<\neg(b,d) >$ is pruned and NSP S_{neg} is overlooked.

The above three examples demonstrate that if an NSP algorithm with a joining-based NSC generation strategy adopts Pruning Strategy 2, it may overlook some kinds of NSPs and thus maintain smaller coverage. A similar problem may also appear in those NSP algorithms which adopt an appending-based NSC generation strategy, such as PNSP. For example, given a negative sequence $S_{neg} =<\neg(b,d), a >$ and a sequence dataset $D = \{< 1, < a, (b,d) >>\}$, $SC_{con}(S_{neg}) = 1$ but $sup_{base}(<\neg(b,d) >) = 0$. Since PNSP generates NSC S_{neg} by appending $<\neg(b,d) >$ with a, if $min\_sup = 1$, then NSC $<\neg(b,d) >$ is pruned and NSP S_{neg} cannot be discovered by PNSP.

The above pruning strategies 1 and 2 can be adopted by MPS-based negative containment, N-containment and strictly-negative containment, but not by N-end-containment because of their definition conflict.

5.4 Coupling Relationships in Negative Containment

According to the above analysis, the MPS-based negative containment defined in Containment 2 judges whether there exists at least one sub-sequence of the given data sequence that can match all the negative elements in a negative sequence, while the strictly-negative containment concept defined in Containment 5 judges whether all corresponding sub-sequences of a data sequence match all negative elements.

According to Section 3, there exist two categories of inferential coupling relationships shared by negative items in negative elements of a sequence: *conjunction coupling relationships* and *disjunction coupling relationships*. The strictly-negative containment adopting different coupling relationships can exhibit different properties. If a disjunction coupling relationship is adopted, the following Corollary 5.16 can be proposed and proved.

Corollary 5.16 (Disjunctive Negative Conversion Corollary (DNC Corollary)). *Given a negative sequence $S^{dis}_{neg} =< ne_1, \dots, ne_{ns} >$ and a data sequence $S_{data} =< de_1, \dots, de_{ds} >$, $S^{dis}_{neg} \subseteq_{sc} S_{data}$ if and only if the following two conditions hold: (1) $MPS(S^{dis}_{neg}) \subseteq_{pos} S_{data}$; and (2) $\forall 1-negMS(S^{dis}_{neg}) \in 1-negMSS(S^{dis}_{neg})$, $PS(1-negMS(S_{neg})) \nsubseteq_{pos} S_{data}$.*

Corollary 5.16 judges whether a negative sequence is strictly-negative-contained by a given data sequence from the perspective of its 1-neg-size maximum sub-sequence set, and the subsequent Corollary 5.17 *strictly-negative-support of disjunctive negative sequence* can be derived from Corollary 5.16 and used to calculate the strictly-negative-support count of a disjunctive negative sequence.

Corollary 5.17 (Strictly-negative-support of Disjunctive Negative Sequence). *Given a s-size and n-neg-size disjunctive negative sequence S^{dis}_{neg} and a sequence dataset D, for $\forall 1-negMSS_k \in 1-negMSS(S^{dis}_{neg}), 1 \leqslant k \leqslant n$, the strictly-negative-support count of S^{dis}_{neg} in D, $SC_{sc}(S^{dis}_{neg})$, can be calculated as follows:*

$$SC_{sc}(S^{dis}_{neg}) = |MPS(S^{dis}_{neg}) - \cup^{n}_{k=1}\{PS(1-negMS_k)\}| \tag{1}$$

Since $\cup^{n}_{k=1}\{PS(1-negMS_k)\}\subseteq_{pos}\{MPS(S^{dis}_{neg})\}$, Equation (1) can also be rewritten as follows:

$$\begin{aligned} SC_{sc}(S^{dis}_{neg}) &= |MPS(S^{dis}_{neg})| - |\cup^{n}_{k=1}\{PS(1-negMS_k)\}| \\ &= SC(MPS(S^{dis}_{neg})) - |\cup^{n}_{k=1}\{PS(1-negMS_k)\}| \end{aligned} \tag{2}$$

Example 28. *For a negative sequence $S^{dis}_{neg} =< (\neg a, b, \neg c), (a, c), \neg(b, d), a, \neg c >$,$SC_{sc}(S^{dis}_{neg}) = SC(< b, (a, c), a >) - |\{< (a, b, c), (a, c), a >\} \cup \{< b, (a, c), (b, d), a >\} \cup \{< b, (a, c), a, c >\}|$.*

However, if a conjunction coupling relationship is adopted, the following Corollary 5.18 *conjunctive negative conversion corollary* (CNC Corollary) can be proposed. Its proof is similar to the Proof of Corollary 5.16 and we ignore it here.

Corollary 5.18 (Conjunctive Negative Conversion Corollary (CNC Corollary)). *Given a negative sequence $S^{con}_{neg} =< ne_1, ne_2, \ldots, ne_{ns} >$ and data sequence $S_{data} =< de_1, de_2, \ldots, de_{ds} >$, $S^{con}_{neg}\subseteq_{sc}S_{data}$ if and only if the following two conditions hold: (1) $MPS(S^{con}_{neg})\subseteq_{pos}S_{data}$; and (2) $\forall 1-niMS(S^{con}_{neg}) \in 1-niMSS(S^{con}_{neg})$, $PS(1-niMS(S^{con}_{neg}))\nsubseteq_{pos}S_{data}$.*

Corollary 5.18 judges whether a negative sequence is strictly-negative-contained by a given data sequence in terms of its 1-neg-item maximum sub-sequence set, and the subsequent Corollary 5.19 can be derived from Corollary 5.18 and used to calculate the strictly-negative-support count of a conjunctive negative sequence.

Corollary 5.19 (Strictly-negative-support of Conjunctive Negative Sequence). *Given an s-size and n-neg-size negative sequence S^{con}_{neg} and a sequence dataset D, for $\forall 1-niMSS_k \in 1-niMSS(S^{con}_{neg}), 1 \leqslant k \leqslant n$, the strictly-negative-support count of S^{con}_{neg} in D, $SC_{sc}(S^{con}_{neg})$, can be calculated as follows:*

$$SC_{sc}(S^{con}_{neg}) = |MPS(S^{con}_{neg}) - \cup^{n}_{k=1}\{PS(1-niMS_k)\}| \tag{3}$$

Similarly, Equation (3) can be also rewritten as follows:

$$\begin{aligned} SC_{sc}(S^{con}_{neg}) &= |MPS(S^{con}_{neg})| - |\cup^{n}_{k=1}\{PS(1-niMS_k)\}| \\ &= SC(MPS(S^{con}_{neg})) - |\cup^{n}_{k=1}\{PS(1-niMS_k)\}| \end{aligned} \tag{4}$$

Example 29. *For negative sequence $S^{con}_{neg} =< (\neg a, b, \neg c), (a, c), \neg(b, d), a, \neg c >$,$SC_{sc}(S^{con}_{neg}) = SC(< b, (a, c), a >) - |\{< (a, b), (a, c), a >\} \cup \{< (b, c), (a, c), a >\} \cup \{< b, (a, c), b, a >\} \cup \{< b, (a, c), d, a >\} \cup \{< b, (a, c), a, c >\}|$.*

According to the above deduction, we can analyze the difference between PSPs and data sequences in terms of set theory. Even though both PSPs and data sequences are composed of only positive elements, they have different meanings. Since data sequences in a dataset are collected from business, the items absent in a data sequence reflect that they do not occur in that sequence. For example, if $I = \{a, b, c, d\}$ and $S_{data} =< b, (a, c), a >$, it means that items a, c and d do not occur in the first element, b and d do not appear in the second element, and b, c and d do not appear in the third element in S_{data}. For a PSP with disjunction coupling S^{dis}_{pos}, Equation (5) is as follows according to Equation (2):

$$SC(S^{dis}_{pos}) = SC_{sc}(MES(S^{dis}_{pos})) + |\cup^{n}_{k=1}\{PS(1 - negMS_k)\}| \tag{5}$$

For a PSP with conjunction coupling S^{con}_{pos}, Equation (6) is as follows according to Equation (4):

$$SC(S^{con}_{pos}) = SC_{sc}(MES(S^{con}_{pos})) + |\cup^{n}_{k=1}\{PS(1 - niMS_k)\}| \tag{6}$$

Equation (5) and Equation (6) show that the support count of a PSP captures the number of data sequences strictly-negative-containing both its maximum equivalent sequence and all its 1-neg-size or 1-neg-item maximum sub-sequences. This means that a PSP covers all its super-sequences irrespective of whether any other items in I occur. In other words, an item which does not appear in a PSP is also covered.

In addition, it can be seen that the strictly-negative-support count of a negative sequence equals the number of data sequences which support its maximum positive sub-sequence but cannot strictly-negative-contain any one of its 1-neg-size or 1-neg-item maximum sub-sequences. This indicates the remaining applicability or support count difference of its maximum positive sub-sequence because of the introduction of negative items. For example, this means that, for a negative sequence $S_{neg} =< (a, c), \neg(b, d), a >$, if $SC_{sc}(S_{neg}) = 3$, three data sequences in $\{MPS(S_{neg})\}$ can also support the generated negative pattern if element (b, d) cannot take place between (a, c) and a. Therefore, after introducing negative items, the NSP mining task can discover the set of negative sequences with significant remaining applicability.

Furthermore, the formula $ratio(S_{neg}) = SC(PS(S_{neg}))/SC_{sc}(S_{neg})$ is an effective measure for evaluating the abnormality of the non-occurrences of negative items in S_{neg}, i.e., the higher $ratio(S_{neg})$ is, the more abnormal the non-occurring negative items are.

5.5 NSP Mining Algorithms Summary

Here we provide the theoretical analysis of the time and space complexity of five NSP mining algorithms which are the representative ones in the four categories: NSPM, MSIS, PNSP, NegGSP and e-NSP respectively. The basis for selecting these algorithms are explained in Section 6.1, and we will not analyze the complexity of GA-NSP because it is a stochastic NSP mining algorithm which is built on genetic algorithm. As discussed in [8], the performance of NSP mining algorithms can be calculated by the data factors of applied datasets. Because the derivation process of theoretical complexity analysis is quite space-consuming, only the final conclusion is provided here, and one worthy example of detailed derivation can be seen in [8].

The following data factors are specified to describe the characteristic of applied dataset: C is the average number of elements per sequence; T is the average number of items per element; S is the average length of potentially maximal frequent positive sequences; I is the average number of items per element in potentially maximal frequent positive sequences; and DB is the number of data sequences in a sequence dataset. In addition, we suppose that the runtime of extending an itemset to the end of a sequence (i.e., appending a sequence with an itemset) is t^{je}, denote the runtime of extending an item to the end of a sequence (i.e., appending a sequence with an item) is t^{ji}, the runtime of generating a new sequence based on given items is t^{g}, the runtime of comparison between two items is t^{c}, the memory usage consumed by storing an item is m_i, the number of s-size PSPs is $|PSP^{S}_{s}|$, the average size of all frequent positive itemset is $\overline{size}(PSP^{S}_{1})$. Let $|NSC^{S}_{s}|^{alg}$ and S^{alg}_{max} be the number of s-size NSCs and the maximum size of all NSCs generated by algorithm *alg* respectively, while $|NSC_l^L|^{alg}$ and L_{max}^{alg} be the number of l-length NSCs and the maximum length of all NSCs generated by *alg*. Accordingly, the total runtime of NSPM T^{NSPM} and its space usage S^{NSPM} can be calculated as follows:

$$T^{NSPM} = \sum_{i=1}^{S_{max}^{NSPM}} |NSC_i^S|^{NSPM} \times (C \times T \times DB \times t^c + (i-1) \times \overline{size}(PSP_1^S) \times t^c + t^{je}) \quad (7)$$

$$S^{NSPM} = \sum_{i=1}^{S_{max}^{NSPM}} |NSC_i^S|^{NSPM} \times i \times \overline{size}(PSP_1^S) \times m_i \quad (8)$$

In addition, the runtime of MSIS T^{MSIS} and its space usage S^{MSIS} can be calculated as follows:

$$T^{MSIS} = \sum_{i=1}^{\frac{S}{I}} \sum_{j=1}^{\left\lfloor \frac{i \times \overline{size}(PSP_1^S)}{2} \right\rfloor} \frac{(i \times \overline{size}(PSP_1^S))! \times (|PSP_{i-1}^S|^2 - |PSP_i^S|)}{j! \times (i \times \overline{size}(PSP_1^S) - j)!} \times (C \times T \times DB \times t^c + t^g) \quad (9)$$

$$S^{MSIS} = \sum_{i=1}^{\frac{S}{I}} \sum_{j=1}^{\left\lfloor \frac{i \times \overline{size}(PSP_1^S)}{2} \right\rfloor} \frac{2 \times (i \times \overline{size}(PSP_1^S))! \times (|PSP_{i-1}^S|^2 - |PSP_i^S|)}{j! \times (i \times \overline{size}(PSP_1^S) - j)!} \times \overline{length}(IPS_1) \times m_i \quad (10)$$

Here, $\overline{length}(IPS_1)$ is the average length of 1-size infrequent positive sequences. Furthermore, the runtime of PNSP T^{PNSP} and its space usage S^{PNSP} can be calculated as follows:

$$T^{PNSP} = \sum_{i=1}^{S_{max}^{PNSP}} |NSC_i^S|^{PNSP} \times (C \times T \times DB \times t^c + t^{je}) \quad (11)$$

$$S^{PNSP} = \sum_{i=1}^{S_{max}^{PNSP}} |NSC_i^S|^{PNSP} \times i \times \overline{length}(PNSP_1) \times m_i \quad (12)$$

Here, $\overline{length}(PNSP_1)$ is the average length of all 1-size PSPs and 1-size NSPs. Moreover, the runtime of NegGSP T^{NegGSP} and its space usage S^{NegGSP} can be calculated as follows:

$$T^{NegGSP} = \sum_{i=1}^{L_{max}^{NegGSP}} |NSC_i^L|^{NegGSP} \times (C \times T \times DB \times t^c + (i-1) \times t^c + t^{ji}) \quad (13)$$

$$S^{NegGSP} = \sum_{i=1}^{L_{max}^{NegGSP}} |NSC_i^L|^{NegGSP} \times i \times m_i \quad (14)$$

Finally, the runtime of e-NSP T^{e-NSP} and its space usage S^{e-NSP} can be calculated as follows:

$$T^{e-NSP} = \sum_{i=1}^{\frac{S}{I}} |PSP_i^S| \times (\sum_{k=1}^{\lceil \frac{i}{2} \rceil} \frac{(i-k+1)!}{k! \times (i - 2 \times k + 1)!} \times k \times (t^{tran} + \overline{sup}(PSP_{i-k+1}^S) \times t^s)) \quad (15)$$

$$S^{e-NSP} = \sum_{i=1}^{\frac{S}{I}} |PSP_i^S| \times (\sum_{k=1}^{\lceil \frac{i}{2} \rceil} \frac{(i-k+1)!}{k! \times (i - 2 \times k + 1)!} \times i \times \overline{size}(PSP_1^S) \times m_i) \quad (16)$$

Here, t^{tran} is the runtime of transforming a positive element to its reverse element, t^s is the runtime of searching a record in a hash table, and $\overline{sup}(PSP_s^S)$ is the average support count of all s-size PSPs.

Taking the systematic technical specifications discussed in the above sections, Table 4 provides a summary of existing NSP mining algorithms in terms of their main objectives, constraint settings, negative containments, advantages and disadvantages. The notations representing the advantages and disadvantages quoted in Table 4 are defined in Table 5 and Table 6 respectively.

Table 4. Technical Design Comparison of Existing NSP Mining Algorithms

Algorithm	**Main Objective**	**Constraint**	**Containment**	**Advantage**	**Disadvantage**
NSPM [36] (Lin *et al.* 2007)	Format-specific NSP mining	ISC,EFC, LFC, NEC	N-end-containment	Adv[1], Adv[2], Adv[3]	Dis[1], Dis[2]
NFSPM [38] (Lin *et al.* 2007)	Format-specific NSP mining	ISC,EFC, LFC, NEC	N-end-containment	Adv[1], Adv[3]	Dis[1], Dis[2]
PNSPM [37] (Lin *et al.* 2008)	Format-specific NSP mining	ISC,EFC, LFC, NEC	N-end-containment	Adv[1], Adv[3]	Dis[1], Dis[2]
MSIS [44] (Ouyang *et al.* 2007)	Format-specific NSP mining	ISC, EIFC, LFC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1], Adv[2], Adv[3]	Dis[1], Dis[2], Dis[3]
MBFIFS [43] (Ouyang *et al.* 2008)	Format-specific NSP mining	ISC, EIFC, LFC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1], Adv[3]	Dis[1], Dis[2], Dis[3]
CPNFMLSP [41] (Ouyang *et al.* 2009)	Format-specific NSP mining	ISC, EIFC, LFC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1], Adv[3]	Dis[1], Dis[2], Dis[3]
CPNFSP [42] (Ouyang *et al.* 2010)	Format-specific NSP mining	ISC, EIFC, LFC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1], Adv[3]	Dis[1], Dis[2], Dis[3]
Incremental CPNFSP [32] (Khare *et al.* 2013)	Format-specific NSP mining	ISC, EIFC, LFC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1], Adv[3]	Dis[1], Dis[2], Dis[3]
SpamNeg [68] (Zhao *et al.* 2008)	Format-specific NSP mining	ISC, SFreC, SForC, NEC	MPS-based Negative Containment, Strictly-Negative Containment	Adv[1]	Dis[1]
PNSP [28] (Hsueh *et al.* 2008)	Complete NSP mining	ISC, ESC, EFC, CFC, NEC	N-containment	Adv[4]	Dis[2], Dis[4]
NegGSP [74] (Zheng *et al.* 2009)	Complete NSP mining	ISC, IFC, CFC, NEC	Negative Cover, MPS-based Negative Containment	Adv[3], Adv[4]	Dis[2], Dis[5]
GA-NSP [75] (Zheng *et al.* 2010)	Stochastic NSP mining	ISC, IFC, CFC, NEC	Negative Cover, MPS-based Negative Containment	Adv[3],Adv[5]	Dis[2], Dis[6], Dis[7]
e-NSP [8] (Cao *et al.* 2016)	PSP-based NSP mining	ISC, SFreC, CFC, NEC	Strictly-negative Containment	Adv[5], Adv[6], Adv[7]	Dis[7]
SAPNSP [39] (Liu *et al.* 2015)	PSP-based NSP mining	ISC, SFreC, CFC, NEC	Strictly-negative Containment	Adv[1], Adv[5], Adv[6], Adv[7]	Dis[7]
e-msNSP [61] (Xu *et al.* 2017)	PSP-based NSP mining	ISC, SFreC, CFC, NEC	Strictly-negative Containment	Adv[5], Adv[6], Adv[7]	Dis[7]
e-NSPFI [26] (Gong *et al.* 2017)	PSP-based NSP mining	ISC, IFC, CFC, NEC	Strictly-negative Containment	Adv[5], Adv[6], Adv[7]	Dis[7]

Table 5. Advantages of NSP Mining Algorithms

Advantage	**Explanation**	**Algorithm**
Adv[1]	Define an interestingness measure to extract meaningful NSPs from a large number of frequent NSPs	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPNFSP [32], SpamNeg [68], SAPNSP [39]
Adv[2]	Can be extended to consider fuzzy elements to mine negative fuzzy sequential patterns	NSPM [36], MSIS [44]
Adv[3]	Avoid generating a number of redundant or invalid NSCs	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPNFSP [32], NegGSP [74], GA-NSP [75]
Adv[4]	Discover the complete set of NSPs	PNSP [28], NegGSP [74]
Adv[5]	Effective for NSC generation	GA-NSP [75], e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]
Adv[6]	Effective for calculating negative support of NSC and avoid rescanning dataset	e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]
Adv[7]	Scalable for mining NSPs on large dataset	e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]

Table 6. Disadvantages of NSP Mining Algorithms

Disadvantage	Explanation	Algorithm
Dis[1]	Cannot discover NSPs against predefined formats	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPN-FSP [32], SpamNeg [68]
Dis[2]	Need to scan dataset multiple times to calculate negative-support or N-containment support of NSCs	NSPM [36], NFSPM [38], PNSPM [37], MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPN-FSP [32], PNSP [28], NegGSP [74], GA-NSP [75]
Dis[3]	Consume a large amount of memory to save infrequent positive sequences	MSIS [44], MBFIFS [43], CPNFMLSP [41], CPNFSP [42], Incremental CPNFSP [32]
Dis[4]	Need to generate a very large number of invalid NSCs	PNSP [28]
Dis[5]	Ineffective in generating NSCs and needs a large number of iteration passes to generate long NSCs	NegGSP [74]
Dis[6]	Need to set several rates, and no related research outcomes for setting them properly are available	GA-NSP [75]
Dis[7]	Can only discover a very small coverage of NSPs	GA-NSP [75], e-NSP [8], SAPNSP [39], e-msNSP [61], e-NSPFI [26]

Table 7. NSP Evaluation Aspects and Description

Evaluation Aspect	Notation	Description
NSP count	Nct	The number of discovered NSPs
NSP runtime	Nrt	Runtime consumed to discover NSPs
Total length of NSCs	Tlnc	The total length of the generated NSCs

6 EXPERIMENTAL ANALYSIS

In this section, the main NSP mining algorithms PNSP, NegGSP, GA-NSP, e-NSP, NSPM and MSIS and their extended versions are selected for empirical analysis on two real-life datasets, based on the above analysis and the evaluation criteria listed in Table 7, which are proposed to evaluate NSP mining algorithms in terms of three evaluation aspects, *NSP count*, *NSP runtime* and *Total length of NSCs*. In addition, we evaluate the influence of adopting different constraint settings and negative containments on algorithm performance.

All the above algorithms are implemented in Java, and all experiments are conducted on a running node with Intel Xeon W3690 (6 Core) CPU of 3.47GHz, 12GB memory and Red Hat Enterprise Linux 6.7 (64bit) OS on the FEIT Linux Cluster at UTS. Since e-NSP can only be applied to NSP mining with disjunction coupling and space is limited, only the experimental results and discussion under disjunction coupling are shown in this paper.

6.1 Datasets and Algorithms to Be Compared

Two real-life sequence datasets are used for this evaluation, which are summarized as follow:

- Real-Life Dataset 1 (RL_1), is a real-life application dataset of health insurance claim sequences [8], which has averagely 21 elements per sequence, averagely 2 items per element, 5269 data sequences and 340 divergent items. The file size is around 5M.
- Real-Life Dataset 2 (RL_2), is a KDD-CUP 2000 dataset from SPMF, which contains 59,601 sequences of clickstream data from an e-commerce [8]. It contains 497 distinct items. The average length of sequences is 2.42 items with a standard deviation of 3.22. In this dataset, there are some long sequences.

Six representative NSP algorithms are evaluated in this paper because of the space limitation, which are NSPM, MSIS, PNSP, NegGSP, GA-NSP and e-NSP. These algorithms are selected because

Table 8. Comparison Algorithms and Settings

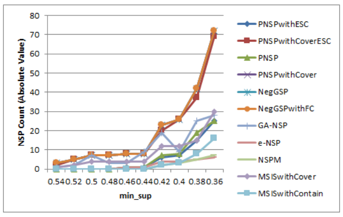
Algorithm	**Constraint**	**Negative Containment**
PNSPwithESC	ISC, ESC, EFC, CFC, NEC	N-containment
PNSPwithCoverESC	ISC, ESC, EFC, CFC, NEC	MPS-based Negative Containment
PNSP	ISC, EFC, CFC, NEC	N-containment
PNSPwithCover	ISC, EFC, CFC, NEC	MPS-based Negative Containment
NegGSP	ISC, IFC, CFC, NEC	MPS-based Negative Containment
NegGSPwithFC	ISC, EFC, CFC, NEC	MPS-based Negative Containment
MSISwithCover	ISC, EIFC, LFC, NEC	MPS-based Negative Containment
MSISwithContain	ISC, EIFC, LFC, NEC	Strictly-Negative Containment

they represent typical and different methodologies in NSA. NSPM [36] is selected as the representative of NSPM-based algorithms, which include NFSPM [38] and PNSPM [37], because NSPM has a much higher number of citations and NFSPM and PNSPM focus on the discovery of fuzzy sequential patterns, which lack comparability to others. Due to the similar consideration, MSIS [44] is chosen as the representative of MSIS-based algorithms, which cover MBFIFS [43], CPNFMLSP [41] and CPNFSP [42], because MBFIFS and CPNFMLSP focus on fuzzy sequential pattern mining while CPNFSP focus on NSP mining with multiple minimum supports. Incremental CPNFSP [32] is not evaluated in this paper, because it has a low number of citation and it is not more than an extension of CPNFSP which is applied to dynamic datasets, which lacks representation. In addition, SpamNeg [68] is also not evaluated, because it actually aims to discover the event-oriented negative sequential rules and it holds a restriction that each element is composed of only one item, which has a high limitation in real application and thus lacks actionablity and representation. Finally, e-NSP [8] is chosen as the representative of the algorithms based on e-NSP, which include SAPNSP [39], e-msNSP [61] and e-NSPFI [26], because e-NSP is an original algorithm for set theory-based NSP mining. Furthermore, SAPNSP and e-msNSP focuses on the mining of NSP based on interestingness measure and multiple minimum supports respectively, while e-NSPFI focuses on the discovery of NSP from both frequent and infrequent PSP, which lack comparability to other algorithms.

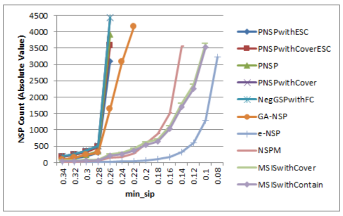
We extend the representative NSP algorithms NSPM, MSIS, PNSP, NegGSP, GA-NSP and e-NSP by adjusting the constraint settings and negative containments. First, AprioriAll is adopted by MSIS to discover PSPs, and GSP is adopted by other algorithms. In addition, PNSP adopts ESC, EFC and N-containment (Containment 3) while NegGSP adopts IFC and MPS-based negative containment (Containment 2), which results in six adjusted versions of these two algorithms: PNSPwithESC, PNSPwithCoverESC, PNSP, PNSPwithCover, NegGSP and NegGSPwithFC. Furthermore, two similar negative containments, negative-support and strictly-negative-support, are incorporated into MSIS, which results in two MSIS-related algorithms: MSISwithCover and MSISwithContain. The constraint settings and negative containments adopted by eight algorithms are listed in Table 8. In addition, GA-NSP is implemented with the crossover rate at 100%, mutation rate at 10% and decay rate at 10%. In the following figures, the X-axis stands for the value of minimum support and the Y-axis stands for the value of the evaluation criteria, the unit of NSP runtime (*Nrt*) is the millisecond, and the unit of NSP count (*Nct*) and total length of NSCs (*Tlnc*) is one.

6.2 Algorithm Efficiency Analysis

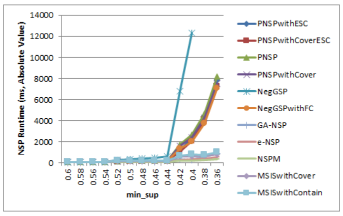
Here, we analyse the efficiency of these representative algorithms on two real-life datasets, which is shown in Figures 1 and 2.



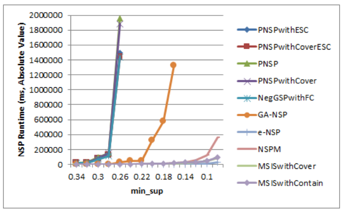
(a) NSP Count (Absolute Value, High Threshold)



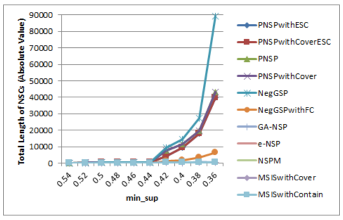
(b) NSP Count (Absolute Value, Low Threshold)



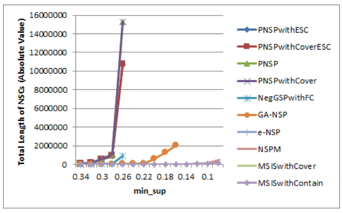
(c) NSP Runtime (Absolute Value, High Threshold)



(d) NSP Runtime (Absolute Value, Low Threshold)



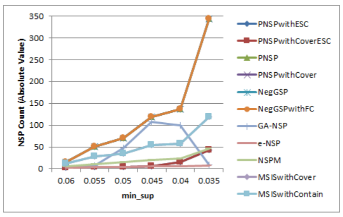
(e) Total Length of NSC (Absolute Value, High Threshold)



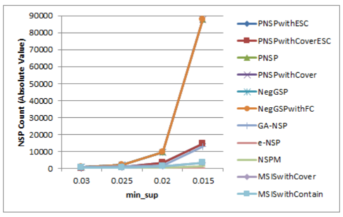
(f) Total Length of NSC (Absolute Value, Low Threshold)

Fig. 1. Algorithm Efficiency Analysis on RL_1 (X-axis stands for threshold, and Y-axis stands for algorithm efficiency)

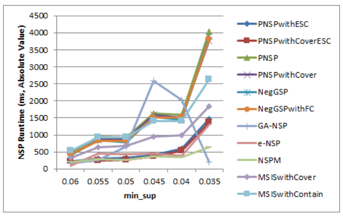
The efficiency of algorithms on RL_1 is illustrated in Figure 1. As seen from Figure 1a, when mining on RL_1 under threshold greater than 0.36, the *Nct* of PNSPwithCoverESC and PNSPwith-Cover is quite similar to that of two NegGSP-based variants, which is twice higher than that of PNSPwithESC and PNSP. It shows that the adoption of negative containment leaves a higher impact on *Nct* than constraint setting on high threshold on RL_1, and a similar phenomenon is also seen when comparing the *Nct* of MSISwithCover and MSISwithContain. Another interesting observation is that GA-NSP can achieve a slightly higher *Nct* than two PNSP-based variants with N-containment, and e-NSP and NSPM maintain the lowest *Nct*. However, from Figure 1b we can see, when threshold declines to 0.26, PNSP gets a similar *Nct* with PNSPwithCover and NegGSPwithFC, which is higher than that of two PNSP-based variants with ESC. It demonstrates that when threshold is low, negative constraint leaves a smaller impact on *Nct*, which is also verified by MSISwithCover and MSISwithContain. Different from the observation of Figure 1a, GA-NSP only discovers half number of NSPs than PNSPwithESC and PNSP, which shows its GA limits its competition for NSP mining under low threshold. It is noted that the results of GA-NSP show a fluctuation in these figures, because GA-NSP is based on genetic algorithm, which makes the efficiency of GA-NSP less stable and the coverage of the discovered NSPs cannot be guaranteed. Similar phenomena can be also seen in Figure 2. In addition, the *Nct* of e-NSP is only one-eighth of NSPM when threshold is less than 0.2. It is shown from Figure 1c that when threshold is high, the *Nrt* of NegGSP is significantly higher than that of NegGSPwithFC and PNSP-based variants, especially when $min\_sup \leqslant 0.44$. Moreover, four PNSP-based variants consume similar runtime, which means different constraint and negative containment have little effect on *Nrt*. NSPM achieves the lowest *Nrt*, followed by e-NSP with a mildly higher *Nrt*. But Figure 1d indicates, when threshold is lower than 0.3, PNSP and PNSPwithCover consume a little higher *Nrt* than other PNSP-based variants with ESC, which means ESC can help reduce *Nrt* and has more influence than negative containment. The *Nrt* of



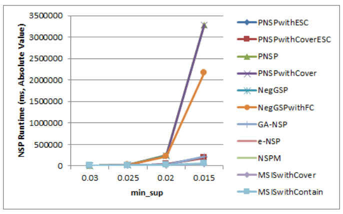
(a) NSP Count (Absolute Value, High Threshold)



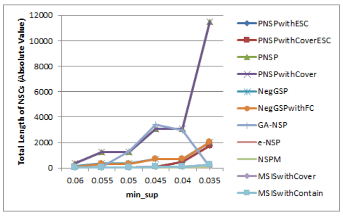
(b) NSP Count (Absolute Value, Low Threshold)



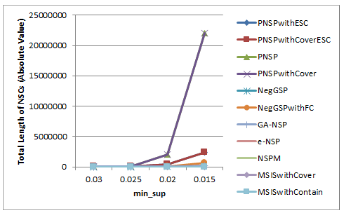
(c) NSP Runtime (Absolute Value, High Threshold)



(d) NSP Runtime (Absolute Value, Low Threshold)



(e) Total Length of NSC (Absolute Value, High Threshold)



(f) Total Length of NSC (Absolute Value, Low Threshold)

Fig. 2. Algorithm Efficiency Analysis on RL_2 (X-axis stands for threshold, and Y-axis stands for algorithm efficiency)

NSPM gets increased clearly when *min_sup* $\leqslant 0.22$, and so does NSPM when *min_sup* $\leqslant 0.14$. In comparison, e-NSP maintains a gentle *Nrt*. Figure 1e shows, when threshold greater than 0.44, almost identical *Tlnc* is generated by PNSP-based and NegGSP-based variants, but thereafter the *Tlnc* of PNSP-based variants with ESC is slightly less than that of PNSP and PNSPwithCover, which is over four times of that of NegGSPwithFC. Obviously, NegGSP generates the highest *Tlnc*, indicating IFC imposes a huge pressure on memory capacity even though NegGSPwithFC generates the lowest *Tlnc* among the complete NSP mining algorithms. Noted from Figure 1f, PNSP-based variants generate significantly higher *Tlnc* than NegGSPwithFC, especially when *min_sup* $\leqslant 0.3$, showing the advantage of NegGSPwithFC on the quality of generated NSCs. Finally, *Tlnc* of NSPM rises rapidly and exceeds that of MSIS-based variants when *min_sup* $\leqslant 0.14$, and e-NSP retains the smallest *Tlnc* on all thresholds.

The efficiency of algorithms on RL_2 is illustrated in Figure 2. Figure 2a shows that NegGSP-based variants, PNSP and PNSPwithCover produce the same *Nct* under high threshold, which is dramatically higher than that of PNSPwithESC and PNSPwithCoverESC. It reveals that on RL_2, *Nct* is less impacted by negative containment than by constraints. This phenomenon is quite different from that of Figure 1a, and a reason is that the average length of sequences of RL_2 is far smaller than that of RL_1 but RL_2 contains more distinct items (N), showing that the impact of negative containment on the *Nct* of algorithms is highly influenced by the data factors of datasets. This statement is also verified by the *Nct* of two MSIS-based algorithms. Another observation is that GA-NSP has a fluctuating *Nct*, and it shows its instability. From Figure 2b we can see, the *Nct* of NegGSP-based variants, PNSP and PNSPwithCover is identical and always more than three times higher than that of the PNSP-based variants with ESC, and it reveals the adoption of different negative containment and IFC leave little impact on *Nct* when mining on sparse datasets but ESC can reduce the number of discovered patterns. In addition, *Nct* of GA-NSP and NSPM is raised quickly under low threshold but that of e-NSP changes smoothly. Different from Figure 1c, 2c reveals when mining on RL_2 under high threshold, the *Nrt* of NegGSP-based variants is similar to that of PNSP and PNSPwithCover, which is more than two times higher than that of two PNSP-based variants with ESC. Moreover, e-NSP consumes a slightly higher *Nrt* than NSPM, which is also seen in Figure 2d. Figure 2d further shows that PNSP and PNSPwithCover consume a significantly higher *Nrt* than PNSPwithESC and PNSPwithCoverESC especially when $min\_sup \leqslant 0.20$, and NegGSP-based variants consume a relatively lower runtime than PNSP and PNSPwithCover. It illustrates, when applied on sparse dataset, ESC can accelerate the mining process obviously, and with the same constraints and negative containment NegGSP outperforms PNSP under low threshold in terms of runtime. Furthermore, GA-NSP consumes a higher *Nrt* than PNSPwithCoverESC, and the *Nrt* of e-NSP also exceeds that of NSPM, showing the runtime advantage of GA-NSP and e-NSP is less obvious for sparse dataset. As shown in Figure 2e, PNSP and PNSPwithCover achieve the highest *Tlnc*, followed by two NegGSP-based variants, of which the *Tlnc* is a little higher than that of PNSPwithESC and PNSPwithCoverESC. In addition, in Figure 2f, NegGSP-based variants consume less than half of the *Tlnc* of PNSPwithESC and PNSPwithCoverESC, demonstrating that NegGSP is a quite competitive complete NSP mining algorithm on sparse dataset under low threshold. Furthermore, the *Tlnc* of GA-NSP is higher than that of PNSPwithESC and PNSPwithCoverESC on most high thresholds, but relatively lower when $min\_sup \leqslant 0.025$. Under all thresholds, e-NSP maintains the lowest *Tlnc*, which is clearly lower than NSPM and MSIS-based variants.

Based on the above analysis, we can find that the efficiency of NSP mining algorithms is highly influenced by the data factors of a dataset. Hence, it is quite reasonable to select the optimal NSP mining algorithm based on the data characteristics of dataset, which can be also seen in Section 5.5.

7 PROSPECTS

Our comprehensive technical review discloses the significant challenges facing NSA research and the enormous opportunities NSA presents. NSA-related research is still in an early stage and there is huge potential for NSA for various non-occurring behaviors and applications [13, 19, 31, 35, 39]. In this section, we highlight several open issues in NSA research, inspired by the above technical and experimental analysis.

Semantic Constraint-based NSP Mining aims to push the semantic constraints into the mining process of NSP and discover the complete set of NSPs satisfying both a given threshold and a semantic constraint S. Different from the existing negative constraints presented in Section 4, semantic constraints confine the mined patterns to a particular subset of conditions based on user interest and focus, which is widely used in PSP mining and frequent itemset mining [27, 48, 66]. To the best of our knowledge, no research has been done to apply the semantic constraints which are widely used in PSP mining into NSA problems, as PSP mining follows the downward closure property while NSA does not. As NSA also covers PSP, hence it is understandable that existing PSP constraints can still be introduced into the positive sequences in NSA, however, the negative constraints on the negation of items, elements, sequences and patterns are very different. Nonetheless, semantic constraint-based NSP mining is highly worthwhile because it can significantly improve the effectiveness and efficiency of pattern mining [47]. Potential NSP semantic constraints can include but are not limited to *item constraint*, *length/size/width constraint*, *super-pattern constraint*, *aggregate constraint*, *regular expression constraint* and *time-stamp constraint*. Below, we present these semantic constraints first from the perspective of application based on the forms of these constraints, and then characterize them from the perspective of constraint properties.

Constraint 11 (Item Constraint (IC)). *An* item constraint *(ISC) specifies the particular set of items which should or should not be presented in the elements of a pattern S, i.e.,* $C_{IC}(S) \equiv (\varphi\ k :$

$1 \leqslant k \leqslant size(S), (S[k]\ \theta\ V_\alpha)\ \delta\ (S[k] \cap V_\beta\ \mu\ \varnothing))$, *where* V_α *is the set of items that should occur in elements while* V_β *is the one that should not,* $\varphi \in \{\forall, \exists\}$, $\delta \in \{\vee, \wedge\}$, $\theta \in \{\subseteq, \nsubseteq, \supseteq, \nsupseteq, \subset, \not\subset, \supset, \not\supset, \in, \notin\}$ *and* $\mu \in \{=, \neq\}$.

Example 30. *Given constraint* $C_{IC}(S) \equiv (\forall k : 1 \leqslant k \leqslant size(S), (S[k] \subseteq (a, \neg a, b, \neg c, d)) \wedge (S[k] \cap (b, d) \neq \varnothing))$, *then negative sequence* $S_\alpha = <(\neg c, d), (a, b), b, (\neg a, d)>$ *satisfies* $C_{IC}(S)$ *while* $S_\beta = <\neg(a, b, d), (a, c)>$ *does not, because* $S_\beta[1] \nsubseteq (a, \neg a, b, \neg c, d)$ *and* $S_\beta[2] \cap (b, d) = \varnothing$.

Constraint 12 (Length/Size/Width Constraint (LSWC)). *An* length/Size/width constraint *(LSWC) specifies the restriction on the length, size, neg-size or width of a pattern S, or the requirement on the size or neg-size of the elements in S i.e.,* $C_{LSWC}(S) \equiv ((length(S)\ \sigma\ t_l)\ \delta\ (size(S)\ \sigma\ t_s)\ \delta\ (neg-size(S)\ \sigma\ t_{ns})\ \delta\ (width(S)\ \sigma\ t_w)\ \delta\ (\varphi\, k : 1 \leqslant k \leqslant size(S), (size(S[k])\ \sigma\ t_{es})\ \delta\ (neg-size(S[k])\ \sigma\ t_{ens}))$, *where* $\sigma \in \{>, \leqslant, <, \geqslant, =, \neq\}$, t_l, t_s, t_{ns} *and* t_w *are the thresholds on the length, size, neg-size and width of S while* t_{es} *and* t_{ens} *are the thresholds on the size and neg-size of the elements in S.*

Example 31. *Given constraint* $C_{LSWC}(S) \equiv ((length(S) > 12) \wedge (neg - size(S) < 2) \wedge (width(S) < 4) \wedge \forall k : 1 \leqslant k \leqslant size(S), size(S[k]) > 2)$, *then* $S_{neg} = <(\neg c, d), (a, b), \neg c, b, (\neg a, b, \neg c, d)>$ *does not satisfy* $C_{LSWC}(S)$ *because* $length(S) = 10$, $neg - size(S) = 3$, $width(S) = 4$, *and* $size(\neg c) = 1$.

Constraint 13 (Super-pattern Constraint (SPC)). *An* super-pattern constraint *(SPC) specifies a particular set of pattern that should or should not be contained by a pattern S as sub-patterns, i.e.,* $C_{SPC}(S) \equiv (\varphi\ \gamma \in P\ s.t.\ \gamma\ \omega\ S)$, *where P is the set of pattern that should or should not be contained, and* $\omega \in \{\subseteq, \supseteq\}$.

Example 32. *In Example 30, given constraint* $C_{SPC}(S) \equiv (<\neg c, b, \neg a> \subseteq S)$, *then* S_α *satisfies* $C_{SPC}(S)$ *while* S_β *does not, because* $<\neg c, b, \neg a> \nsubseteq S_\beta$.

Constraint 14 (Aggregate Constraint (AC)). *Suppose the items in a pattern are associated with a numeric attribute A, an* aggregate constraint *(SPC) specifies on an aggregate of items in a pattern S, i.e.,* $C_{AC}(S) \equiv (\eta_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A\ \sigma\ t_{ac})$. *Here* t_{ac} *is a given threshold and* η *is an aggregate function on items, which can be sum, avg, max, min, standard deviation, etc.*

Example 33. *In marketing analysis, suppose each item i is associated with a* profit *attribute, i.e.,* $i.profit$, *a constraint* $C_{AC} \equiv (\sum_{i \in S[k], 1 \leqslant k \leqslant size(S)} i.profit \geqslant 80)$ *specifies the patterns where the total profit is no less than 80, where the profit of a negative item is the cost of this commodity.*

Constraint 15 (Regular Expression Constraint (REC)). *A* regular expression constraint *(SPC) is specified as a regular expression over the sequence elements using the established set of regular operations, such as disjunction (|) and Kleene closure (*) [23]. A pattern S satisfies a regular expression constraint* $S_{REC}(S)$ *if and only if S is accepted by the equivalent deterministic finite automata [48], thus* $S_{REC}(S)$ *specifies a regular family of sequential patterns which is of interest to users.*

Example 34. *In Example 11, given constraint* $S_{REC}(S) = (b^*(\neg a|d))$, *then* S_α *satisfies* $S_{REC}(S)$ *while* S_β *does not.*

Constraint 16 (Time-stamp Constraint (TSC)). *Suppose each element e of the data sequences in sequence dataset is associated with a time-stamp attribute, denoted as* $e.ts$, *a* time-stamp constraint *(TSC) specifies that the time-stamp difference between particular elements in pattern S should satisfy a given period requirement, i.e.,* $C_{TSC}(S) \equiv (|\{<Sid_{data}, S_{data}>| (<Sid_{data}, S_{data}> \in D) \wedge (S_{neg} \subseteq S_{data}) \wedge (for 1 \leqslant i < j \leqslant size(S), \exists 1 \leqslant k_i < k_j \leqslant size(S_{data})\ s.t.\ (S[i] \subseteq S_{data}[k_i]) \wedge (S[j] \subseteq S_{data}[k_j]) \wedge (S_{data}[k_j].ts - S_{data}[k_i].ts)\ \sigma\ t_{ts})\}| \geqslant min\_sup \times |D|)$. *To avoid confusion,* $C_{TSC}(S)$ *specifies that S does not contain proper negative elements, since it is impractical to determine the time-stamp when no item occurs.*

Table 9. Monotonicity, Anti-monotonicity and Succinctness of Commonly Used Semantic Constraints (NN stands for Not Necessary)

Constraint	**Definition**	**Mono**	**Anti-Mono**	**Succ**
IC	$C_{IC}(S) \equiv (\varphi\ k : 1 \leqslant k \leqslant size(S), S[k] \subseteq V_\alpha), \varphi \in \{\forall, \exists\}$	No	Yes	Yes
	$C_{IC}(S) \equiv (\varphi\ k : 1 \leqslant k \leqslant size(S), S[k] \supseteq V_\alpha), \varphi \in \{\forall, \exists\}$	Yes	No	Yes
	$C_{IC}(S) \equiv (\varphi\ k : 1 \leqslant k \leqslant size(S), S[k] \cap V_\beta = \emptyset), \varphi \in \{\forall, \exists\}$	No	Yes	Yes
	$C_{IC}(S) \equiv (\varphi\ k : 1 \leqslant k \leqslant size(S), S[k] \cap V_\beta \neq \emptyset), \varphi \in \{\forall, \exists\}$	Yes	No	Yes
LSWC	$C_{LSWC}(S) \equiv (fun(S) \leqslant t), fun(S) \in \{length(S), size(S), neg-size(S), width(S)\}$	Yes	No	Yes
	$C_{LSWC}(S) \equiv (fun(S) \geqslant t), fun(S) \in \{length(S), size(S), neg-size(S), width(S)\}$	No	Yes	Yes
SPC	$C_{SPC}(S) \equiv (\varphi\ \gamma \in P\ s.t.\ \gamma \subseteq S), \varphi \in \{\forall, \exists\}$	Yes	No	Yes
	$C_{SPC}(S) \equiv (\varphi\ \gamma \in P\ s.t.\ \gamma \supseteq S), \varphi \in \{\forall, \exists\}$	No	Yes	Yes
AC	$C_{AC}(S) \equiv (\max_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A \leqslant t)$	No	Yes	Yes
	$C_{AC}(S) \equiv (\max_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A \geqslant t)$	Yes	No	Yes
	$C_{AC}(S) \equiv (\min_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A \leqslant t)$	Yes	No	Yes
	$C_{AC}(S) \equiv (\min_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A \geqslant t)$	No	Yes	Yes
	$C_{AC}(S) \equiv (\eta'_{i \in S[k], 1 \leqslant k \leqslant size(S)}\ i.A\ \sigma'\ t), \eta' \in \{sum, avg\}$ and $\sigma' \in \{>, \leqslant, <, \geqslant\}$	No	No	No
REC	Regular expression	NN	NN	NN
LTSC	$C_{DC}(S) \equiv (dur(S)\ \sigma\ t_{ts}), \sigma \in \{>, \leqslant, <, \geqslant, =, \neq\}$	No	No	No
	$C_{DC}(S) \equiv (gap(S)\ \sigma\ t_{ts}), \sigma \in \{>, \leqslant, <, \geqslant, =, \neq\}$	No	Yes	No

Example 35. *Two commonly derived TSC are* duration constraint *and* gap constraint. Duration constraint *(DC) specifies that the time-stamp difference between the first and last elements in a pattern S satisfy a period requirement, i.e.,* $C_{DC}(S) \equiv (|\{< Sid_{data}, S_{data} >| (< Sid_{data}, S_{data} > \in D) \wedge (S_{neg} \not\subseteq S_{data}) \wedge (\exists 1 \leqslant k_1 < k_{size(S)} \leqslant size(S_{data})\ s.t.\ (S[1] \subseteq S_{data}[k_1]) \wedge (S[size(S)] \subseteq S_{data}[k_{size(S)}]) \wedge (S_{data}[k_{size(S)}].ts - S_{data}[k_1].ts)\ \sigma\ t_{ts})\}| \geqslant min\_sup \times |D|)$*, which is also denoted as* $C_{DC}(S) \equiv (dur(S)\ \sigma\ t_{ts})$*. And* gap constraint *(GC) specifies that the time-stamp difference between two adjacent elements in S satisfy a period requirement, i.e.,* $C_{GC}(S) \equiv (|\{< Sid_{data}, S_{data} >| (< Sid_{data}, S_{data} > \in D) \wedge (S_{neg} \not\subseteq S_{data}) \wedge (\forall 1 \leqslant i \leqslant size(S) - 1, \exists 1 \leqslant k_i \leqslant size(S_{data})\ s.t.\ (S[i] \subseteq S_{data}[k_i]) \wedge (S_{data}[k_{i+1}].ts - S_{data}[k_i].ts)\ \sigma\ t_{ts})\}| \geqslant min\_sup \times |D|)$*, which is also denoted as* $C_{DC}(S) \equiv (gap(S)\ \sigma\ t_{ts})$.

Among the above constraints, we can see that TSC is support-related, while others can be judged whether they are satisfied by the patterns themselves. Below, we characterize these semantic constraints from the perspective of three constraint properties, including monotonicity, anti-monotonicity and succinctness, commonly-used for traditional constrained PSP mining [48] and constrained frequent itemset mining [45]. Here a semantic constraint C_m is *monotonic* if a sequence S satisfies C_m implies that each super-sequence of S also satisfies C_m. And a constraint C_a is *anti-monotonic* if a sequence S satisfies C_a implies that each subsequence of S also satisfies C_m. In addition, a constraint C_s is *succinct* if a sequence S satisfies C_s implies that all the elements of S can be expressed in terms of the strict powersets of a fixed number of succinct sets using the set union and/or set difference operators, where a succinct set is an itemset in which items are selected from I using a selection operator $\tau_p(I)$ for some selection predicate p. In other words, all the patterns satisfying a succinct constraint C_s can be explicitly and preciously generated without an iterative generation-and-testing approach [47]. As per the definition of the above semantic constraints, the monotonicity, anti-monotonicity and succinctness property of some commonly used constraints are shown in Table 9. It is noted that a regular expression constraint $S_{REC}(S)$ is not necessarily a monotonic, anti-monotonic or succinct constraint, despite some particular regular expression constraints have these properties. Because of the space limitation, the relevant proof are omitted.

Some efforts have already been made to the related research on the PSP mining with semantic constraints, including some Apriori-based vertical formatting algorithms [66] and projection-based pattern-growth algorithms [47, 48]. However, none of them can be applied to semantic constraint-based NSP mining directly. On one hand, suffering from the violation of downward closure property, Apriori-based algorithms cannot work in NSP mining, such as cSPADE [66]. On the other hand, since negative items do not appear in a sequence dataset, NSC cannot be enumerated and generated by projection-based pattern-growth algorithms, such as [48]. In addition, the property of semantic constraints in NSP is a little different from that in PSP, thus some new theoretical breakthroughs and relevant algorithms are required for semantic constraint-based NSP mining.

NSP Mining with Loose Constraint Settings aims to achieve a reasonably large search space which generates sufficient NSCs and avoid the missing of informative patterns. With an increased number of generated NSCs, the comparison time and number of items to be saved rise accordingly, which substantially increases the runtime and space complexity. This requires the development of efficient data structures for storage and calculations. Moreover, effective pruning strategies need to be studied to avoid the generation of invalid NSCs and reduce the search space.

Incremental NSP Mining over Sequence Stream is much more challenging than classic NSP mining on static sequences but applicable to real-life applications involving data streams in which new data sequences are continuously inserted [17]. New data sequences may be generated at high speed, and NSP algorithms can only process the incoming new sequences once and do not rescan the dataset, as is usually done by existing algorithms [50]. By incorporating the new streamed sequences, the amount of space and memory resource required to process sequences may be quickly exhausted. This involves many issues including how to store, represent, and process evolving NSPs, and how to define related constraint settings and negative containments.

Quantitative NSP Mining considers the different significance of each negative containment and evaluates a negative sequence in terms of its utility, instead of frequency only, which can discover more informative NSPs and achieve a higher actionability of the resultant patterns [5, 11, 63]. In some real-life applications, items with high utility [63] may appear infrequently. For example, in a PC shop, failing to sell a laptop would lead to a greater reduction in profit than failing to sell a mouse. Proper strategies may be informed by analyzing how to promote laptop sales to achieve higher profitability. For this, we need to quantify the utility of a negative item, element and sequence, and build a new theoretical framework for quantitative NSP mining, for which existing NSP algorithms do not work. This involves many issues such as new evaluation measures, data structures, negative containment, NSC selection, and pruning strategies.

NSP Mining with Complex Hierarchical Structure considers hierarchical structures in sequences and breaks the assumption that items in the same element have no order and are on the same level, taken by existing NSP algorithms. For example, in marketing, let $<(coke, tissue, sprite), (bread, milk)>$ be a purchasing transaction. It is clear that both *coke* and *sprite* belong to a subclass of food and that they can be considered separately from the item *tissue*. The existing NSP problem statement is not applicable to this scenario and the introduction of a complex hierarchical structure can simplify the solution to such issues. When hierarchical and coupling relationships [5, 7] are considered, many interesting research issues arise that are aligned with real-life behavior interactions and applications. It requires theoretical breakthroughs in representing, modeling, reasoning about, storing, and managing complex structures, hierarchies and relationships in NSA.

Top-K NSP Mining allows users to determine the number of patterns to be discovered and target mining only patterns with high frequency within a limited search space and lower computational complexity, instead of setting an ideal threshold and mining all NSPs. Compared with top-K PSP mining, top-K NSP mining is more challenging and sophisticated, because the downward closure property does not hold in NSP mining. Accordingly, classic top-K PSP mining algorithms, such as

TSP [54], TUS [64] and TKU [58], cannot be used or adjusted directly. Therefore, new theories and algorithms are required to discover top-K NSPs.

8 CONCLUSIONS

Non-occurring behaviors are widely apparent in many applications, while effective theories and algorithms for discovering their patterns are unavailable. A specific direction is to identify non-occurring sequences in behavioral data, a process which is also called negative sequence analysis. Compared to positive sequential pattern discovery, negative sequential pattern mining is a relatively new area with diversified problem statements and technical designs, as well as limited research outcomes. This is because of the complex intrinsic characteristics and significant challenges in NSA, which are often managed by the proposal of specific constraint settings and negative containments.

This paper reviews all available NSP algorithms. Since existing work has been built on specific problem definitions and settings, this work consolidates and forms a comprehensive and systematic representation, formalization, and theoretical system for defining and representing NSP problems, constraints, negative containment, and evaluation. Existing algorithms are evaluated in terms of the proposed theoretical systems. In addition to the comprehensive technical review, this work also provides empirical analysis of representative NSP mining algorithms. Several new opportunities are discussed which represent a proportion of the ongoing and future work we are working on.

REFERENCES

[1] Fahad Anwar, Ilias Petrounias, Tim Morris, and Vassilis Kodogiannis. 2010. Discovery of events with negative behavior against given sequential patterns. In *Proceedings of the 5th IEEE International Conference on Intelligent Systems*. IEEE, 373–378.

[2] Sujeevan Aseervatham, Aomar Osmani, and Emmanuel Viennet. 2006. bitSPADE: A lattice-based sequential pattern mining algorithm using bitmap representation. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 792–797.

[3] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 429–435.

[4] Longbing Cao. 2010. In-depth behavior understanding and use: The behavior informatics approach. *Inf. Sci.* 180, 17 (2010), 3067–3085.

[5] Longbing Cao. 2012. Combined Mining: Analyzing Object and Pattern Relations for Discovering Actionable Complex Patterns. *sponsored by Australian Research Council Discovery Grants (DP1096218 and DP130102691) and an ARC Linkage Grant (LP100200774)* (2012).

[6] Longbing Cao. 2014. Non-IIDness Learning in Behavioral and Social Data. *Comput. J.* 57, 9 (2014), 1358–1370.

[7] Longbing Cao. 2015. Coupling Learning of Complex Interactions. *Journal of Information Processing and Management* 51, 2 (2015), 167–186.

[8] Longbing Cao, Xiangjun Dong, and Zhigang Zheng. 2016. eNSP: Efficient negative sequential pattern mining. *Artificial Intelligence* 235 (2016), 156–182.

[9] Longbing Cao, Yuming Ou, and Philip S Yu. 2012. Coupled behavior analysis with applications. *Knowledge and Data Engineering, IEEE Transactions on* 24, 8 (2012), 1378–1392.

[10] Longbing Cao, Yuming Ou, Philip S. Yu, and Gang Wei. 2010. Detecting abnormal coupled sequences and sequence changes in group-based manipulative trading behaviors. In *KDD'2010*. 85–94.

[11] Longbing Cao, Philip Yu, Yanchang Zhao, and Chengqi Zhang. 2010. *Domain Driven Data Mining*. Springer.

[12] Longbing Cao and Philip S. Yu. 2012. *Behavior Computing - Modeling, Analysis, Mining and Decision*. Springer.

[13] Longbing Cao, Philip S Yu, and Vipin Kumar. 2015. Nonoccurring Behavior Analytics: A New Area. *Intelligent Systems, IEEE* 30, 6 (2015), 4–11.

[14] Longbing Cao, Philip S. Yu, Chengqi Zhang, and Huaifeng Zhang. 2008. *Data Mining for Business Applications* (1 ed.). Springer.

[15] Longbing Cao, Yanchang Zhao, and Chengqi Zhang. 2008. Mining Impact-Targeted Activity Patterns in Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* 20, 8 (2008), 1053–1066.

[16] Longbing Cao, Yanchang Zhao, Huaifeng Zhang, Dan Luo, Chengqi Zhang, and E. K. Park. 2010. Flexible Frameworks for Actionable Knowledge Discovery. *IEEE Trans. Knowl. Data Eng.* 22, 9 (2010), 1299–1312.

[17] Hong Cheng, Xifeng Yan, and Jiawei Han. 2004. IncSpan: Incremental mining of sequential patterns in large database. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 527–532.

[18] Ding-Ying Chiu, Yi-Hung Wu, and Arbee LP Chen. 2004. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *Proceedings of the 20th International Conference on Data Engineering*. IEEE, 375–386.

[19] Xiangjun Dong, Yongshun Gong, and Lulin Zhao. 2014. Comparisons of typical algorithms in negative sequential pattern mining. In *2014 IEEE Workshop on Electronics, Computer and Applications*. IEEE, 387–390.

[20] Xiangjun Dong, Fengrong Sun, Xiqing Han, and Ruilian Hou. 2006. Study of positive and negative association rules based on multi-confidence and chi-squared test. In *International Conference on Advanced Data Mining and Applications*. Springer, 100–109.

[21] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas. 2014. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 40–52.

[22] Fabio Fumarola, Pasqua Fabiana Lanotte, Michelangelo Ceci, and Donato Malerba. 2016. CloFAST: closed sequential pattern mining using sparse and vertical id-lists. *Knowledge and Information Systems* 48, 2 (2016), 429–463.

[23] Minos N Garofalakis, Rajeev Rastogi, and Kyuseok Shim. 1999. SPIRIT: Sequential pattern mining with regular expression constraints. In *VLDB*, Vol. 99. 7–10.

[24] Antonio Gomariz, Manuel Campos, Roque Marin, and Bart Goethals. 2013. ClaSP: An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 50–61.

[25] Yongshun Gong, Chuanlu Liu, and Xiangjun Dong. 2015. Research on Typical Algorithms in Negative Sequential Pattern Mining. *Open Automation and Control Systems Journal* 7 (2015), 934–941.

[26] Yongshun Gong, Tiantian Xu, Xiangjun Dong, and Guohua Lv. 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 (2017), 1750002.

[27] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. 2007. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery* 15, 1 (2007), 55–86.

[28] Sue-Chen Hsueh, Ming-Yen Lin, and Chien-Liang Chen. 2008. Mining negative sequential patterns for e-commerce recommendations. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*. IEEE, 1213–1218.

[29] He Jiang, Xiumei Luan, and Xiangjun Dong. 2012. Mining weighted negative association rules from infrequent itemsets based on multiple supports. In *2012 International Conference on Industrial Control and Electronics Engineering (ICICEE)*. IEEE, 89–92.

[30] Xiaoqi Jiang, Qian Gao, Tiantian Xu, and Xiangjun Dong. 2018. Campus Data Analysis based on Positive and Negative Sequential Patterns. *International Journal of Pattern Recognition and Artificial Intelligence* (2018).

[31] S Kamepalli and R Kurra. 2014. Frequent negative sequential patterns: A survey. *International Journal of Computer Engineering and Technology* 5, 3 (2014), 15–121.

[32] Vinay Kumar Khare and Vedant Rastogi. 2013. Mining Positive and Negative Sequential Pattern in Incremental Transaction Databases. *International Journal of Computer Applications* 71, 1 (2013).

[33] Bac Le, Hai Duong, Tin Truong, and Philippe Fournier-Viger. 2017. FCloSM, FGenSM: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. *Knowledge and Information Systems* 53, 1 (2017), 71–107.

[34] Bac Le, Minh-Thai Tran, and Bay Vo. 2015. Mining frequent closed inter-sequence patterns efficiently using dynamic bit vectors. *Applied Intelligence* 43, 1 (2015), 74–84.

[35] Yuefeng Li, Abdulmohsen Algarni, and Ning Zhong. 2010. Mining positive and negative patterns for relevance feature discovery. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 753–762.

[36] Nancy P Lin, Hung-Jen Chen, and Wei-Hua Hao. 2007. Mining negative sequential patterns. In *Proceedings of the 6th WSEAS International Conference on Applied Computer Science, Hangzhou, China*. 654–658.

[37] Nancy P Lin, Hung-Jen Chen, Wei-Hua Hao, Hao-En Chueh, and Chung-I Chang. 2008. Mining strong positive and negative sequential patterns. *WSEAS Transactions on Computers* 7, 3 (2008), 119–124.

[38] Nancy P Lin, Wei-Hua Hao, Hung-Jen Chen, Chung-I Chang, and Hao-En Chueh. 2007. An Algorithm for Mining Strong Negative Fuzzy Sequential Patterns. *International Journal of Computers* 1, 3 (2007).

[39] Chuanlu Liu, Xiangjun Dong, Caoyuan Li, and Yan Li. 2015. SAPNSP: Select actionable positive and negative sequential patterns based on a contribution metric. In *12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. IEEE, 811–815.

[40] Carl H Mooney and John F Roddick. 2013. Sequential pattern mining–approaches and algorithms. *ACM Computing Surveys (CSUR)* 45, 2 (2013), 19.

[41] Weimin Ouyang and Qinhua Huang. 2009. Mining Positive and Negative Fuzzy Multiple Level Sequential Patterns in Large Transaction Databases. In *2009 WRI Global Congress on Intelligent Systems*, Vol. 1. IEEE, 500–504.

[42] Weimin Ouyang and Qinhua Huang. 2010. Mining positive and negative sequential patterns with multiple minimum supports in large transaction databases. In *2010 Second WRI Global Congress on Intelligent Systems*, Vol. 2. IEEE, 190–193.

[43] Weimin Ouyang, Qinhua Huang, and Shuanghu Luo. 2008. Mining Positive and Negative Fuzzy Sequential Patterns in Large Transaction Databases. In *Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Vol. 5. IEEE, 18–23.

[44] Wei-min Ouyang and Qin-hua Huang. 2007. Mining negative sequential patterns in transaction databases. In *2007 International Conference on Machine Learning and Cybernetics*, Vol. 2. IEEE, 830–834.

[45] Jian Pei and Jiawei Han. 2002. Constrained frequent pattern mining: a pattern-growth view. *ACM SIGKDD Explorations Newsletter* 4, 1 (2002), 31–39.

[46] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. 2004. Mining sequential patterns by pattern-growth: The PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering* 16, 11 (2004), 1424–1440.

[47] Jian Pei, Jiawei Han, and Wei Wang. 2002. Mining sequential patterns with constraints in large databases. In *Proceedings of the eleventh international conference on Information and knowledge management*. ACM, 18–25.

[48] Jian Pei, Jiawei Han, and Wei Wang. 2007. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems* 28, 2 (2007), 133–160.

[49] Eliana Salvemini, Fabio Fumarola, Donato Malerba, and Jiawei Han. 2011. Fast sequence mining based on sparse id-lists. In *International Symposium on Methodologies for Intelligent Systems*. Springer, 316–325.

[50] Amany F Soliman, Gamal A Ebrahim, and Hoda K Mohammed. 2011. SPEDS: A framework for mining sequential patterns in evolving data streams. In *2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PacRim)*. IEEE, 464–469.

[51] Yin Song, Longbing Cao, Xindong Wu, Gang Wei, Wu Ye, and Wei Ding. 2012. Coupled behavior analysis for capturing coupling relationships in group-based market manipulations. In *KDD'12*. 976–984.

[52] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology*. Springer, 1–17.

[53] Zaiping Tao. 2012. An Incremental Update Algorithm for Sequential Patterns Mining. In *Information Engineering and Applications*. Springer, 134–140.

[54] Petre Tzvetkov, Xifeng Yan, and Jiawei Han. 2005. TSP: Mining top-k closed sequential patterns. *Knowledge and Information Systems* 7, 4 (2005), 438–457.

[55] Can Wang and Longbing Cao. 2012. Modeling and analysis of social activity process. In *Behavior computing*. Springer, 21–35.

[56] Can Wang, Longbing Cao, and Chi-Hung Chi. 2015. Formalization and verification of group behavior interactions. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 8 (2015), 1109–1124.

[57] Can Wang, Zhong She, and Longbing Cao. 2013. Coupled clustering ensemble: Incorporating coupling relationships both between base clusterings and objects. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*. IEEE, 374–385.

[58] Cheng Wei Wu, Bai-En Shie, Vincent S Tseng, and Philip S Yu. 2012. Mining top-K high utility itemsets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 78–86.

[59] Xindong Wu, Chengqi Zhang, and Shichao Zhang. 2004. Efficient mining of both positive and negative association rules. *ACM Transactions on Information Systems (TOIS)* 22, 3 (2004), 381–405.

[60] Tiantian Xu, Xiangjun Dong, Jianliang Xu, and Xue Dong. 2017. Mining High Utility Sequential Patterns with Negative Item Values. *International Journal of Pattern Recognition and Artificial Intelligence* 31, 10 (2017), 1750035.

[61] Tiantian Xu, Xiangjun Dong, Jianliang Xu, and Yongshun Gong. 2017. E-msNSP: Efficient Negative Sequential Patterns Mining Based on Multiple Minimum Supports. *International Journal of Pattern Recognition and Artificial Intelligence* 31, 02 (2017), 1750003.

[62] Zhenglu Yang, Yitong Wang, and Masaru Kitsuregawa. 2007. LAPIN: Effective sequential pattern mining algorithms by last position induction for dense databases. In *International Conference on Database Systems for Advanced Applications*. Springer, 1020–1023.

[63] Junfu Yin, Zhigang Zheng, and Longbing Cao. 2012. USpan: An efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 660–668.

[64] Junfu Yin, Zhigang Zheng, Longbing Cao, Yin Song, and Wei Wei. 2013. Efficiently mining top-k high utility sequential patterns. In *13th International Conference on Data Mining*. IEEE, 1259–1264.

[65] N Yusof, R Zurita-Milla, and MJ Kraak. 2015. Mining sequential pattern of multi-dimensional wind profiles. In *Proceedings of the 13th international conference on GeoComputation geospatial information sciences, 20-23 May 2015,*

Richardson TX, United States of America.

[66] Mohammed J Zaki. 2000. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*. ACM, 422–429.

[67] Mohammed J Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning* 42, 1-2 (2001), 31–60.

[68] Yanchang Zhao, Huaifeng Zhang, Longbing Cao, Chengqi Zhang, and Hans Bohlscheid. 2008. Efficient mining of event-oriented negative sequential rules. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 1. IEEE, 336–342.

[69] Yanchang Zhao, Huaifeng Zhang, Longbing Cao, Chengqi Zhang, and Hans Bohlscheid. 2009. Mining both positive and negative impact-oriented sequential rules from transactional data. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 656–663.

[70] Yanchang Zhao, Huaifeng Zhang, Fernando Figueiredo, Longbing Cao, and Chengqi Zhang. 2007. Mining for Combined Association Rules on Multiple Datasets. In *DDDM'07*. 18–23.

[71] Yanchang Zhao, Huaifeng Zhang, Shanshan Wu, Jian Pei, Longbing Cao, Chengqi Zhang, and Hans Bohlscheid. 2009. Debt detection in social security by sequence classification using both positive and negative patterns. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 648–663.

[72] Zhigang Zheng. 2012. *Negative Sequential Pattern Mining*. Ph.D. Dissertation. University of Technology, Sydney.

[73] Zhigang Zheng, Wei Wei, Chunming Liu, Wei Cao, Longbing Cao, and Maninder Bhatia. 2016. An effective contrast sequential pattern mining approach to taxpayer behavior analysis. *World Wide Web* 19, 4 (2016), 633–651.

[74] Zhigang Zheng, Yanchang Zhao, Ziye Zuo, and Longbing Cao. 2009. Negative-GSP: An efficient method for mining negative sequential patterns. In *Proceedings of the Eighth Australasian Data Mining Conference-Volume 101*. Australian Computer Society, 63–67.

[75] Zhigang Zheng, Yanchang Zhao, Ziye Zuo, and Longbing Cao. 2010. An efficient GA-Based algorithm for mining negative sequential patterns. In *Advances in Knowledge Discovery and Data Mining*. Springer, 262–273.

Received February 2007; revised March 2009; accepted June 2009