Mining Conjunctive Sequential Patterns

Chedy Raïssi[1], Toon Calders[2], and Pascal Poncelet[3]

[1] LIRMM, University of Montpellier, France
raissi@lirmm.fr
[2] Eindhoven University of Technology, Netherlands
t.calders@tue.nl
[3] LGI2P, Ecole des Mines d'Alès, France
pascal.poncelet@ema.fr

Abstract. In this paper we aim at extending the non-derivable condensed representation in frequent itemset mining to sequential pattern mining. We start by showing a negative example: in the context of frequent sequences, the notion of non-derivability is meaningless. Therefore, we extend our focus to the mining of conjunctions of sequences. Besides of being of practical importance, this class of patterns has some nice theoretical properties. Based on a new unexploited theoretical definition of equivalence classes for sequential patterns, we are able to extend the notion of a non-derivable itemset to the sequence domain. We present a new depth-first approach to mine non-derivable conjunctive sequential patterns and show its use in mining association rules for sequences. This approach is based on a well known combinatorial theorem: the Möbius inversion. A performance study using both synthetic and real datasets illustrates the efficiency of our mining algorithm. These new introduced patterns have a high-potential for real-life applications, especially for network monitoring and biomedical fields with the ability to get sequential association rules with all the classical statistical metrics such as confidence, conviction, lift etc.

1 Introduction

In this paper we study the discovery of frequent sequences. Many algorithms have been proposed for mining all frequent sequences, such as SPAM [1], SPADE [15]. Even more than in the frequent itemset domain, however, frequent sequence mining is suffering from the huge amount of patterns a mining operation produces. It is not uncommon that mining a rather modest database results in a gigantic number of frequent patterns. Given that the original aim of data mining is to discover those precious little nuggets of knowledge hidden in a huge pile of data, this situation is a *contradictio in terminis*. In this respect, for the frequent itemset domain [7], many studies have been conducted which aim at reducing the redundancy in the output set. A logical approach is to see to what extent condensed representations and deduction in the context of frequent itemsets can be extended to the sequential pattern domain. Up to now, only closed sequential

patterns have already been studied as a condensed representation for the sequential pattern domain [13]. In this paper, we look at non-derivable itemsets as a candidate to extend. Loosely speaking, a derivable itemset is one of which the support is perfectly determined by the support of its subsets. A derivable itemset can thus be considered as redundant and be removed from the output set. This seemingly straightforward exercise, however, turns out to be slightly harder than expected. We start our paper with a negative result for non-derivable sequences: unlike in the frequent itemset domain, the notion of non-derivability is meaningless in the sequential pattern domain; except for some extremely degenerated cases, every sequence is non-derivable.

This negative result motivated us to look at a slightly different problem: the mining of *conjunctions* of sequential patterns. This extended class of patterns turns out to have much nicer mathematical properties. For example, for this class of patterns we are able to extend the notion of non-derivable itemsets in a non-trivial way, based on a new unexploited theoretical definition of equivalence classes for sequential patterns. As a side-effect of considering conjunctions of sequences as the pattern type, we can easily form association rules between sequences. Compared to the unordered structure of an itemset pattern, only few works [2] focus on the association rule mining problem for sequential pattern. This is largely due to the difficult formalization needed for these patterns compared to the set-theory based formalization used in itemset mining. Furthermore, and as highlighted by the authors in [14], sequences are helpful in real-world critical applications like medical diagnoses and disaster prediction by proposing sequence rules associated with statistical metrics in order to build classifiers that efficiently takes into account temporal order. We believe that building a theoretical framework and an efficient approach for sequence association rules extraction problem is the first step toward the generalization of association rules to all complex and ordered patterns.

Our contributions in this article are twofold:

1. We present a new equivalence relation and extend it to equivalence classes for sequential patterns. We discuss the role of these classes in sequential patterns concise representations and we exhibit a strong result for sequential pattern concise representations that are based on frequency bounds.
2. We introduce a new mining task with a new type of pattern: the *Conjunctive Sequence Pattern* based on the equivalence classes and investigate the algorithmic aspects along with the possibility of computing a set of frequent non-redundant conjunctive sequence patterns by using the combinatorial theorem of Möbius inversion [10].

The rest of the paper is presented as follows. In Section 2, the basic concepts of sequential pattern mining are introduced. Section 3 presents the equivalences classes for sequential patterns along with their related properties. Section 4 introduces the Conjunctive Sequence Pattern mining problem and discusses the computation of a non-redundant set of frequent conjunctive sequence patterns. Section 5 introduces CSPminer, our depth-first algorithm for mining conjunc-

tive sequence patterns. An experimental study is reported in section 6 and we conclude our work in Section 7.

2 Preliminary concepts and definitions

2.1 Frequent Sequence Mining

In this section we define the sequential pattern mining problem in large databases and give an illustration. This description of sequence datasets was first introduced in [12] and extended in [11]. Let $\mathcal{I} = \{i_1, i_2 \dots i_m\}$ be the finite set of items. An itemset is a non-empty set of items. A sequence S over $\mathcal{I}$ is an ordered list $\langle it_1, \dots, it_k \rangle$, with it_j an itemsets over $\mathcal{I}$, $j = 1 \dots k$.

$\mathbb{T}(\mathcal{I})$ will denote the (infinite) set of all possible sequences over $\mathcal{I}$. A sequence database $\mathcal{D}$ over $\mathcal{I}$ is a finite set of pairs (SID, T), called transactions, with $SID \in \{1, 2, \dots\}$ an identifier and $T \in \mathbb{T}(\mathcal{I})$ a sequence over $\mathcal{I}$. For any two transaction $(SID_1, T_1) \neq (SID_2, T_2) \in \mathcal{D}$, it must be that $SID_1 \neq SID_2$.

Definition 1 (Inclusion). *A sequence $S' = \langle is'_1\ is'_2\ \dots\ is'_n \rangle$ is a subsequence of another sequence $S = \langle is_1\ is_2\ \dots\ is_m \rangle$, denoted $S' \preceq S$, if there exist $i_1 < i_2 < \dots\ i_j\ \dots < i_n$ such that $is'_1 \subseteq is_{i_1}$, $is'_2 \subseteq is_{i_2} \dots is'_n \subseteq is_{i_n}$.*

Example 1. Sequence $\langle (a)(c)(d) \rangle$ is included in $\langle (ab)(c)(ab)(de) \rangle$. We say that sequence $\langle (ab)(c)(ab)(de) \rangle$ supports $\langle (a)(c)(d) \rangle$. However, $\langle (a)(c) \rangle$ is not included in $\langle (c)(a) \rangle$.

Definition 2 (Support). *The support of a sequence S in a transaction database $\mathcal{D}$, denoted $Support(S, \mathcal{D})$, is defined as: $Support(S, \mathcal{D}) = |\{(SID, T) \in \mathcal{D} | S \preceq T\}|$. The frequency of S in $\mathcal{D}$, denoted $f_S^{\mathcal{D}}$, is $f_S^{\mathcal{D}} = \frac{Support(S,\mathcal{D})}{|\mathcal{D}|}$.*

Given a user-defined minimal frequency threshold σ, the problem of sequential pattern mining is the extraction of all the sequences S in $\mathcal{D}$ such that $f_S \geq \sigma$. The set of all frequent sequences for a threshold σ in a database $\mathcal{D}$ is denoted $FSeqs(\mathcal{D}, \sigma)$.

$$FSeqs(\mathcal{D}, \sigma) := \{S \mid f_S^{\mathcal{D}} \geq \sigma\} \ .$$

Example 2. Consider the following database over the items $\mathcal{I} = \{a, b, c, d\}$. There are 3 transactions, with identifiers 1, 2, and 3. Let the minimal frequency threshold be $\sigma = \frac{2}{3}$, the frequent sequences in $\mathcal{D}$ are: $\langle (a) \rangle$, $\langle (b) \rangle$, $\langle (c) \rangle$, $\langle (d) \rangle$, $\langle (ab) \rangle$, $\langle (ad) \rangle$, $\langle (a)(c) \rangle$ and $\langle (d)(c) \rangle$. Notice that we use brackets to separate the different itemsets in the sequences from each other.

$\mathcal{D} =$

S_1	$(a, b, c, d)(a, c)$
S_2	(a, b)
S_3	$(a, d)(c)$

2.2 Problem Statement

The problem studied in this paper now is as follows: for many datasets $\mathcal{D}$ and thresholds σ, the size of the set of frequent sequences $FSeqs(\mathcal{D}, \sigma)$ is extremely large and contains a lot of redundancies. It is the goal of this paper to study how we can reduce this enormous set of frequent sequences using techniques of pattern condensation from the frequent itemset domain. The closed itemsets have already been adopted successfully in this domain, leading to the closed sequential patterns [13]. A sequence S in a database $\mathcal{D}$ is called closed if there does not exist a sequence $S' \neq S$ such that $S \preceq S'$ and $f_S = f_{S'}$. Only mining the closed sequences results in a reduced set of patterns. In this paper we want to extend another class of condensed representations, the k-free sets [5], to the frequent sequence domain. This class includes the free sets, disjunction-free sets, and the non-derivable itemsets. For an overview of condensed representations in the field of frequent itemsets, see [7]. Central in the construction of these representation is the deduction of frequencies. That is, for a given set of frequencies, we ask ourselves the question: what can be derived for the frequency of other patterns? This problem is formalized as follows: Let $\mathcal{C} = \{f_{S_1}, \ldots, f_{S_n} \in [0,1]\}$ be the respective frequencies of $S_1, \ldots, S_n$. A database is said to be *consistent* with $\mathcal{C}$ if and only if, for $i = 1 \ldots n$, $f_{S_i,\mathcal{D}} = f_{S_i}$. Typically, for given frequencies there are many consistent databases. Let now S be another sequence. The *best bounds* for f_S given these n frequencies is then defined as

$$[LB_{\mathcal{C}}(S), UB_{\mathcal{C}}(S)] := [\min\{f_{S,\mathcal{D}} \mid \mathcal{D} \text{ consistent with } f_{S_1}, \ldots, f_{S_n}\}, \max\{f_{S,\mathcal{D}} \mid \mathcal{D} \text{ consistent with } f_{S_1}, \ldots, f_{S_n}\}]$$

In the frequent itemset domain deduction rules that allow to compute these bounds under some assumptions of the set $\mathcal{C}$ have been studied. For example, among others, there are the following reasoning rules for itemsets: $f_{abc} \leq f_{ab}$ and $f_{abc} \geq f_{ab} + f_{ac} - f_a$. If now it happens that the lower bound f_{ab} equals the upper bound $f_{ab} + f_{ac} - f_a$, abc is redundant w.r.t. the itemsets a, ab, and ac. Hence, in this situation we can remove abc from the output set of frequent itemset mining. In general, this problem of deciding on the best bounds is **NP**-hard [4], but for the special case where we consider only a downwardly closed set $\mathcal{C}$ with a single top element, the problem becomes tractable and forms the basis of the frequent non-derivable itemset mining representation. This representation turns out to be quite successful in reducing the output set in frequent itemset mining [8, 6]. In the next section we start with extending these results to the sequential domain. An important notion is that of equivalence classes for sequences.

3 Equivalence Classes For Sequential Patterns

Previous works in sequential pattern mining [15] introduced sequential patterns equivalence classes based on a prefix equivalence relation in order to decompose the mining task in smaller easily-solvable problems. In this section, we introduce

the concept of sequential patterns equivalence classes based on a support equivalence relation. We discuss complexity issues associated with these equivalence classes and show that these equivalence classes definition can be used to prove an important theorem on the lower bound of sequential patterns frequency.

3.1 Definitions

Definition 3. *Let $\mathcal{S}$ be a set of sequences. Two sequences T_1 and T_2 are said to be* $\mathcal{S}$-equivalent*, denoted $T_1 \equiv_{\mathcal{S}} T_2$, if, for all $S \in \mathcal{S}$ it holds that $S \preceq T_1$ if and only if $S \preceq T_2$. The set of all sequences equivalent to T under $\equiv_{\mathcal{S}}$ is denoted $[T]_{\mathcal{S}}$.*

Let $\mathcal{IN}$ and $\mathcal{OUT}$ be sets of sequences. $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ denotes the following set of sequences:

$$\mathcal{E}(\mathcal{IN}, \mathcal{OUT}) \ := \ \{T \in \mathbb{T} \mid \forall S \in \mathcal{IN} : S \preceq T, \forall S \in \mathcal{OUT} : S \not\preceq T\} \ .$$

Hence, $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ denotes the set of all sequences that support all sequences in $\mathcal{IN}$, and none of the sequences in $\mathcal{OUT}$.

Example 3. Let $\mathcal{I} = \{a, b, c\}$, and let

$$\mathcal{S} = \left\{ \begin{array}{l} \langle (a)(b)(c) \rangle \\ \langle (b)(c) \rangle \\ \langle (a)(b) \rangle \end{array} \right\}$$

We then have: $\langle (abc)(abc)(abc) \rangle \equiv_{\mathcal{S}} \langle (a)(bc)(abc) \rangle$ and $\langle (b)(c) \rangle \not\equiv_{\mathcal{S}} \langle (a)(bc) \rangle$.

Consider now $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ for:

- $\mathcal{IN} = \{\langle (a)(b)(c) \rangle, \langle (ac) \rangle\}$, $\mathcal{OUT} = \{\langle (bc) \rangle\}$. $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ contains, among others, the sequences $\langle (ac)(b)(c) \rangle$, $\langle (ac)(c)(ab)(c) \rangle$, but not $\langle (abc) \rangle$.
- $\mathcal{IN} = \{\langle (ab) \rangle, \langle (ac) \rangle\}$, $\mathcal{OUT} = \{\langle (c) \rangle\}$. $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is empty, as every sequence that contains $\langle (ac) \rangle$ must also contain $\langle (c) \rangle$.
- $\mathcal{IN} = \{\langle (a)(c) \rangle, \langle (b) \rangle\}$, $\mathcal{OUT} = \{\langle (a)(b) \rangle, \langle (b)(c) \rangle\}$. $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is also empty, as every sequence T that contains $\langle (a)(c) \rangle$ and $\langle (b) \rangle$ must also contain either $\langle (a)(b) \rangle$ or $\langle (b)(c) \rangle$; since T contains $\langle (a)(c) \rangle$, the first occurrence of a in T must come before the last occurrence of c. As such, the first occurrence of (b) in T either comes after the first occurrence of a or before the last occurrence of c.

The following lemma is immediate given the definition of equivalence of sequences.

Lemma 1. *Let $\mathcal{S}$ be a set of sequences. $\equiv_{\mathcal{S}}$ is an equivalence relation on the set of all sequences $\mathbb{T}$. The number of equivalence classes $|\mathbb{T}/\equiv_{\mathcal{S}}|$ is at most $2^{|\mathcal{S}|}$.*

Proof. This follows easily from the fact that for every transaction $T \in \mathbb{T}$,

$$[T]_{\mathcal{S}} = \mathcal{E}(\{S \in \mathcal{S} \mid S \preceq T\}, \{S \in \mathcal{S} \mid S \not\preceq T\}) \ .$$

Furthermore, equivalence classes can be written, without loss of generality, by only using the top elements present in the sets $\mathcal{IN}$ and the bottom elements in $\mathcal{OUT}$. Semantically, $\mathcal{E}(\mathcal{IN}, \mathcal{OUT}) = \emptyset$ means that no sequences can be built supporting all sequences from set $\mathcal{IN}$ and not supporting any sequence from the set $\mathcal{OUT}$. Here a first divergence with the itemset domain emerges; whereas the structure of the equivalence classes is extremely simple in the itemset case, for sequences this is not at all true. For itemsets, $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is non-empty if and only if every set in $\mathcal{OUT}$ has at least one item that is not in any of the sets in $\mathcal{IN}$; e.g., $\mathcal{E}(\{ab, ac\}, \{bc\})$ is empty as every transaction that contains the itemsets ab and ac, also contains bc. As such, deciding non-emptyness of an equivalence class is trivial for itemsets as this test can be performed in linear time. For sequences, this problem turns out to be much harder. Consider, e.g., the example $\mathcal{E}(\{\langle (a)(b)\rangle, \langle (a)(c)\rangle\}, \{\langle (b)(c)\rangle\})$. This class is non-empty, as it contains, among others, the sequence $\langle (a)(c)(b)\rangle$. The following lemma states exactly how much more complex this problem becomes. The importance of this lemma will become apparent later on in the paper, where we introduce deduction rules for the frequency of (conjunctions of) sequences.

Lemma 2. *Let $\mathcal{IN}$ and $\mathcal{OUT}$ be sets of sequences. Deciding if $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is nonempty is an* **NP***-complete problem.*

Proof. The inclusion in **NP** is straightforward; if $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is nonempty, then it contains at least 1 sequence T with $size(T)$ at most $\sum_{S \in \mathcal{IN}} size(S)$, where $size(\langle is_1, \ldots, is_k\rangle)$ denotes $\sum_{j=1}^{k} |is_j|$. Indeed, let T be a sequence in $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$. Then, for every $S \in \mathcal{IN}$, there exists a set of indices $i_1 < i_2 < \ldots < i_{|S|}$, such that the jth set in S is a subset of the i_jth subset of T. Fix for every $S \in \mathcal{IN}$ one such set of indices $i[S]$. Let now T' be the following subsequence of T: let $1 \leq t_1 < \ldots < t_m \leq |T|$ be the set of indices $\cup_{S \in \mathcal{S}'} i[S]$.

$$T' = \left\langle \bigcup_{\substack{S \in \mathcal{IN} \\ t_j = i_k \in i[S]}} S[k] \right\rangle_{j=1}^{m}$$

Since $T' \preceq T$, for all $S \in \mathcal{OUT}$, $S \not\preceq T'$. Furthermore T' is constructed in such a way that for all $S \in \mathcal{IN}$, $S \preceq T'$. $size(T') \leq \sum_{S \in \mathcal{IN}} size(S)$. Such a sequence T' is a succinct certificate for the non-emptiness of $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$.

For the completeness, we reduce the following variant EQUAL-3COL of the 3COL problem to the problem of deciding on the non-emptyness of $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$: *Given a graph G with $3k$ vertices, does there exist a coloring of the vertices that uses only 3 colors, and every color exactly k times?* This problem is equivalent to the 3COL problem. On the one hand we can reduce 3COL to it as follows: a graph G is three-colorable if and only if the graph consisting of 3 separate copies of G is in EQUAL-3COL. On the other hand, if a graph is in EQUAL-3COL the coloring itself is clearly a succinct certificate and thus EQUAL-3COL is in **NP**.

So, let G be a graph with $3k$ edges. We show how we can reduce the 3COL problem for a graph G with $3k$ vertices to a non-emptiness problem $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$.

Let $V = \{v_1, \ldots, v_{3k}\}$ be the set of vertices, and E be the set of edges of G. The set of items is $\{i_1, i_2, \ldots, i_n, i_{3k+1}, R, G, B\}$. The sets $\mathcal{IN}$ and $\mathcal{OUT}$ will be constructed in such a way that the only sequences in $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ are of the form $\langle i_1 C_1 \ldots i_{3k} C_{3k} i_{3k+1} \rangle$ with C_i either R, G, or B, and such that $C(v_i) = C_i$, for all $i = 1 \ldots 3k$ is a valid coloring of G. We first describe the sequences in the set $\mathcal{IN}$:

1. $\langle (i_1)(i_2) \ldots (i_{3k})(i_{3k+1}) \rangle$, *all markers must be present in the right order.*
2. $\left\langle \overbrace{(C)(C)\ldots(C)}^{k\times} \right\rangle$, for $C = R, G, B$ *every color occurs at least* k *times.*

We now describe the sequences in $\mathcal{OUT}$:

1. $\langle (i, j) \rangle$, $\forall i \neq j \in \mathcal{I}$, *all sets in the sequence are singletons.*
2. $\langle (i_j)(i_j) \rangle$, $j = 1 \ldots 3k + 1$, *no marker occurs twice.*
3. $\left\langle \overbrace{(C)(C)\ldots(C)}^{k+1\times} \right\rangle$, for $\forall C \in \{R, G, B\}$, *no color occurs* $k + 1$ *times.*
4. $\langle (i_j)(C_1)(C_2)(i_{j+1}) \rangle$, $j = 1 \ldots 3k$, $\forall C_1, C_2 \in \{R, G, B\}$. *there are no two colors between two subsequent markers.*
5. $\langle (i_j)(C)(i_{j+1})(i_k)(C)(i_{k+1}) \rangle$, $\forall (v_j, v_k) \in E$, $\forall C \in \{R, G, B\}$. *no adjacent vertices can have the same color* C with $C = R, G, B$.

Every sequence in $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ encodes a 3-coloring of G as described above.

Notice that the high complexity of this seemingly simple problem also indicates that deducing bounds $[LB_{\mathcal{C}}(S), UB_{\mathcal{C}}(S)]$ on the frequency of a sequence S, given the frequencies of other sequences $\mathcal{C}$, is at least as hard as **NP**. This can be seen as follows: the equivalence class $\mathcal{E}(\mathcal{IN}, \mathcal{OUT})$ is empty if and only if for any sequence S in $\mathcal{IN}$, given the frequencies $f_{S'} = 1$ for all $S' \in \mathcal{IN} \setminus \{S\}$, and $f_{S'} = 0$ for all $S \in \mathcal{OUT}$, the best bound is $[0, 0]$. Nevertheless, low complexity for the equivalence class problem does not guarantee efficient algorithms for computing bounds.

The **NP**-completeness of this problem motivates the study of special cases that could be interesting from a practical point of view. The special case we consider is the following: suppose that the set of given constraints $\mathcal{C}$ contains the frequency $f_{S'}$ of every strict subsequence $S' \prec S$. This is a useful subcase as it reflects exactly the information we have in Apriori-like algorithms. However, as shown in the next subsection, in this case, the lower bound will always be trivial.

3.2 Lower bound on the frequency of sequential patterns

Based on the notion of equivalence classes, we can give a negative result effectively eliminating all hope for an easy extension of the notion of non-derivable itemsets to the sequential pattern domain.

Lemma 3. *For any non-empty sequence S, the equivalence class $\mathcal{E}(\{S' \mid S' \prec S\}, \{S\})$ is nonempty.*

Theorem 1. *Set S be a sequence, and let $\mathcal{C}$ be a set of given frequencies that has the frequency for every $S' \prec S$. If there exists a database that is consistent with $\mathcal{C}$, the lower bound $LB_{\mathcal{C}}(S)$ is 0.*

Proof. Let $\mathcal{D}$ be a database consistent with C. As $\mathcal{E}(\{S' \mid S' \prec S\}, \{S\})$ is non-empty (Lemma 3), we can replace every transaction in $\mathcal{D}$ that supports S by a transaction of $\mathcal{E}(\{S' \mid S' \prec S\}, \{S\})$. This transformation does not affect the frequency of any of the strict subsequences of S, hence the transformed database still satisfies $\mathcal{C}$. The frequency of S in the transformed database, however, is 0, and hence $LB_{\mathcal{C}}(S)$ is 0.

Example 4. Let $\mathcal{D}$ be a sequence database containing the sequence $S = \langle (a)(b)(c) \rangle$. We can always exhibit a database $\mathcal{D}'$ that gives the same frequency to every strict subsequence of S but reduces the frequency of S itself to 0.

$\mathcal{D} =$

S_1	$(a)(b)(c)$
S_2	$(a)(b)(c)$
S_3	$(c)(a)$
S_4	$(b)(c)$

$\mathcal{D}' =$

S_1	$(b)(c)(a)(c)(a)(b)$
S_2	$(b)(c)(a)(c)(a)(b)$
S_3	$(c)(a)$
S_4	$(b)(c)$

This theorem is very important from the concise representations point of view as it clearly states that condensed representations based on support computations like non-derivable representation [8] and all the other *k-free* representations like 0-free-sets or disjunct-free-sets [3, 5] are *meaningless* for sequential patterns; only sequences with a frequency of 0 are derivable and thus requiring that a sequence is non-derivable will not reduce the set of frequent sequences at all.

4 Conjunctive sequence patterns

In the previous section we introduced equivalence classes for sequential patterns and discussed their theoretical usefulness. In this section we introduce a more practical approach in order to mine a more specific subset of equivalence classes: the downward closed equivalence classes regardless of their emptiness or not.

Definition 4. *A conjunctive sequence pattern (CSP) is a subset C of $\mathbb{T}$ such that all $S \neq S' \in C$, S and S' are incomparable; i.e., neither $S \preceq S'$, nor neither $S' \preceq S$. The set of all CSPs is denoted $\mathcal{C}$.*

A sequence $T \in \mathbb{T}$ is said to support a CSP C if for all $S \in C$, $S \preceq T$. The support *of a CSP C in a database $\mathcal{D}$, denoted $Support_{\mathcal{D}}(C)$, is defined as the number of sequence transactions in $\mathcal{D}$ that satisfies C.*

Let $C_1, C_2 \in \mathcal{C}$. C_1 is said to be a subpattern of C_2, denoted $C_1 \preceq C_2$ if and only if for all $S_1 \in C_1$ there exists a $S_2 \in C_2$ such that $S_1 \preceq S_2$.

Example 5. Let $C_1 = \{\langle (a)(b) \rangle, \langle (a, c) \rangle\}$ and $C_2 = \{\langle (b) \rangle, \langle (c) \rangle\}$ conjunctive sequence patterns. And $\mathcal{D}$ defined as:

$$\mathcal{D} = \begin{array}{|c|c|}\hline S_1 & (a)(b)(a,c) \\ \hline S_2 & (a,b)(c) \\ \hline S_3 & (c)(a) \\ \hline \end{array}$$

- C_2 is a subpattern of C_1 ($C_2 \preceq C_1$) as $\langle(b)\rangle \preceq \langle(a)(b)\rangle$ and $\langle(c)\rangle \preceq \langle(a,c)\rangle$
- $Support_{\mathcal{D}}(C_1) = 1$ as only the sequence $S_1 = \langle(a)(b)(a,c)\rangle$ supports C_1.
- $Support_{\mathcal{D}}(C_2) = 2$ as S_1 and S_2 support C_2.

The following lemma is immediate:

Lemma 4 (Anti-Monotonicity). *Let C_1, C_2 be CSPs. If $C_1 \preceq C_2$, then, for all databases $\mathcal{D}$, $Support(C_1) \geq Support(C_2)$.*

Hence, for mining frequent CSPs we can exploit the anti-monotonicity for pruning the search space. Next we show how we can generate all direct specializations of a pattern.

Let $P \subseteq \mathbb{T}$. $\lceil P \rceil$ denotes the CSP $\{S \in P \mid \nexists S' \in P : S' \prec S\}$. On the other hand, $\downarrow P$ denotes the downward closure of P; i.e., the set $\{S \in \mathbb{T} \mid \exists S' \in P : S \preceq S'\}$.

A CSP C_2 is said to cover a CSP C_1, denoted $C_1 \rightarrow C_2$, if $C_1 \prec C_2$, and there does not exist a CSP C_3 such that $C_1 \prec C_3 \prec C_2$; i.e., C_2 is a direct specialization of C_1, and C_1 a direct generalization of C_2. For the sequences case, the computation of all direct specializations is straightforward: given a sequence $S = \langle I_1, \ldots, I_n\rangle$. The direct generalizations of S, denoted $dg(S)$, are the sequences:

$$\bigcup_{\substack{j=1 \\ |I_j|>1}}^{n} \{\langle I_1, \ldots, I_{j-1}, I_j \setminus \{i\}, I_{j+1}, \ldots, I_n\rangle \mid i \in I_j\} \cup \bigcup_{\substack{j=1 \\ |I_j|=1}}^{n} \{\langle I_1, \ldots, I_{j-1}, I_{j+1}, \ldots, I_n\rangle \mid i \in I_j\} \ .$$

Lemma 5 (Generalization and Specialization). *Let C be a CSP. The set of direct generalizations of C is:*

$$\{\lceil (C \setminus S) \cup dg(S) \rceil \mid S \in C\}$$

and the set of direct specializations of C is:

$$\{C \cup \{S\} \mid S \notin C, dg(S) \subseteq C\}$$

Proof. The proof is based on the simple fact that if C_1 is a generalization of C_2 if and only if $\downarrow C_1 \subseteq \downarrow C_2$ and that $\downarrow C_1 = \downarrow C_2$ implies that $C_1 = C_2$. E.g., for the set of direct generalizations, it can easily be checked that for all $S \in C$, $\downarrow (C \setminus S) \cup dg(S) \subseteq \downarrow C$, and that $\downarrow C \setminus \downarrow (C \setminus S) \cup dg(S) = \{S\}$.

The generalization and specialization lemma allows for generating the set of all patterns from general to specific, thus exploiting the anti-monotonicity of support as much as possible.

4.1 Support Bounding and Möbius Inversion

The next theorem shows that the set of all CSPs equipped with the partial order $\preceq$ forms a lattice. This in contrast to the set of sequences without conjunctions, on which the structure is a partial order. E.g., the sequences $\langle (a,b)(a) \rangle$ and $\langle (a)(a,b) \rangle$ do not have a unique meet; both $\langle (a),(a) \rangle$ and $\langle (a,b) \rangle$ are meets. In the set of all CSPs, the meet is unique: $\{\langle (a),(a) \rangle, \langle (a,b) \rangle\}$.

Theorem 2. *The partial order $(\mathcal{C}, \preceq)$ forms a lattice.*

Proof. Let $C_1, C_2 \in \mathcal{C}$. It is easy to see that the following two sets are respectively the unique meet and the join of C_1 and C_2:

$$\bigwedge \lceil \downarrow C_1 \cap \downarrow C_2 \rceil \quad \text{and} \quad \bigwedge \lceil C_1 \cup C_2 \rceil$$

The fact that $(\mathcal{C}, \preceq)$ is a lattice opens up a whole mathematical toolbox of useful properties that can be applied. Most importantly, we can use the technique of Möbius inversion to get rules bounding the support of sequences in much the similar way as was done for the Non-Derivable Itemsets, as we will explain next.

Let C be a CSP. With every element $C' \in \downarrow C$ in the lattice, we can associate two numbers: $s(C') = \mathit{Support}(C', \mathcal{D})$, and $a(C') = A(C')$, with

$$A(C') := |\{T \in \mathcal{D} \mid \forall C'' \in \downarrow C : T \models C'' \Leftrightarrow C'' \preceq C'\}|$$

Hence, $s(C')$ is the normal support in the database $\mathcal{D}$, and $a(C')$ denotes the number of transactions that support *exactly* C', and nothing more. For all transactions that are counted in $a(C')$, they support a pattern only if that pattern is more general than C'.

Then, the following relation between a and s holds:

Lemma 6. *For all $C' \preceq C$,*

$$s(C') \;=\; \sum_{C' \preceq C''} a(C'')$$

From this lemma the following theorem follows quite easily:

Theorem 3. *Let, for all $C' \preceq C$, an integer s_C be given. There exists a database $\mathcal{D}$ with for all $C' \preceq C$, $\mathit{Support}(C', \mathcal{D}) = s_C$ if and only if the following system of inequalities and equalities in the variables $a(C')$, $C' \preceq C$ has a solution:*

$$\begin{cases} a(C') = 0 & \forall C' : \mathcal{E}(C', \downarrow C \setminus C') = \emptyset \\ a(C') \geq 0 & \forall C' : \mathcal{E}(C', \downarrow C \setminus C') \neq \emptyset \\ \sum_{C' \preceq C''} a(C'') = s'_C & \forall C' \preceq C \end{cases}$$

Moreover, for every database that satisfies for all $C' \preceq C$, $\mathit{Support}(C', \mathcal{D}) = s_C$, $a(C') = a(C', \mathcal{D})$ is a solution to the system.

The theory on Möbius inversion, which can be thought of as a generalization of the inclusion-exclusion principle, now learns us that under this condition there always exists a function μ, the so-called Möbius inverse, that allows us to express the $a(C')$ in function of the $s(C')$; i.e.:

Lemma 7. *There exists a function $\mu(\cdot,\cdot)$ that maps a pair of CSPs $C', C'' \preceq C$ to a integer $\mu(C_1, C_2)$, such that for all databases $\mathcal{D}$, the following holds:*

$$a(C') \;=\; \sum_{C' \preceq C''} \mu(C', C'') s(C'')$$

Hence, $a(C')$ can be expressed as a simple linear combination of the supports of the CSPs.

Combining the lemma with the theorem give us:

Theorem 4. *Let, for all $C' \preceq C$, an integer s_C be given. There exists a database $\mathcal{D}$ with for all $C' \preceq C$, Support$(C', \mathcal{D}) = s_C$ if and only if:*

$$\begin{cases} \sum_{C' \preceq C''} \mu(C', C'') s(C'') = 0 & \forall C' : \mathcal{E}(C', \downarrow C \setminus C') = \emptyset \\ \sum_{C' \preceq C''} \mu(C', C'') s(C'') \geq 0 & \forall C' : \mathcal{E}(C', \downarrow C \setminus C') \neq \emptyset \end{cases}$$

Example 6. Let $\epsilon = 1$ and let $\mathcal{D}$ be defined as:

$$\mathcal{D} = \boxed{S_1 \boxed{(a)(b)(c)}}$$

Suppose that we are trying to compute support bounds for the conjunctive sequence pattern $csp = \{\langle (a) \rangle ; \langle (b) \rangle\}$ and suppose that all frequencies are already known for the set of sub-conjunctions $P = \{\ \{\langle\rangle\}, \{\langle a \rangle\}, \{\langle b \rangle\}\ \}$. The lattice $(\mathcal{L}, \preceq)$ based on the set $P \cup \{csp\}$ can be represented as a matrix and the inverse matrix μ contains the values needed to compute the bounds :

$$\zeta = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \;,\; \mu = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

From μ, we get the 4 possible rules for support bounding:

1. $s(\{\langle\rangle\}) - s(\{\langle a \rangle\}) - s(\{\langle b \rangle\}) + s(\{\langle a \rangle, \langle b \rangle\}) \geq 0$
2. $s(\{\langle a \rangle\}) - s(\{\langle a \rangle, \langle b \rangle\}) \geq 0$
3. $s(\{\langle b \rangle\}) - s(\{\langle a \rangle, \langle b \rangle\}) \geq 0$
4. $s(\{\langle a \rangle, \langle b \rangle\}) \geq 0$

With this rules it is trivial to see that $s(csp) \in [1, 1]$. Thus, the support of this conjunction can be *derived* from its subconjunctions.

Algorithm 1: CSP miner Algorithm

Data: Sequence Database: $\mathcal{D}$; minimal frequency threshold ϵ
Result: The set of frequenct CSPs : $\mathcal{C}$

```
1  begin
2      k ← 1;
3      C ← ∅;
4      F_1 ← mine_sequence(k);
5      C ← F_1;
6      while F_k not empty do
7          foreach s ∈ F_k do
8              foreach x <_{inv prefix} s with x ⋠ s do
9                  generate_CSP(x, s, C);
10         k++;
11         F_k ← mine_sequence(k);
12     return C;
13 end
```

5 Algorithm

In this section we give a depth-first algorithm for the conjunctive sequence pattern mining problem. The basic idea of the algorithm is to alternate between a sequence mining task and a generation of all possible conjunctions. The algorithm exploits two properties: (i) in order to compute the frequency of conjunctions, we only need to compute the cardinal of the intersection set of the tidlists present in every sequence contained in the conjunction, (ii) to compute frequency bounds for a CSP Y, all frequencies for conjunction X such that $\{\langle()\rangle\} \preceq X \prec Y$ must be known. Since we are using a depth-first algorithm, many of these frequencies may be unavailable. In order to solve this problem we invert the order in wich we traverse the research space as previously described in [9]. If the algorithm runs with the derivability on, the bounds are computed for each candidate CSP.

The algorithm is illustrated with a toy database in Figure 1. Suppose $\epsilon = 1$, first all sequences of length 1 are mined, then CSPs are generated while traversing in an inverted depth-first manner the research space. When there is no more possible CSP generation, the algorithm generates level 2 sequences and restart CSP generation with the newly added sequences. The algorithm returns when there are no more frequent sequences to be mined.

6 Experimentations

We have performed tests using the CSP miner algorithm on synthetic and real-world datasets. The goal of the experimentations are: (i) to verify the validity and feasability of our CSP mining approach in the case of normal extraction and non-derivability and (ii) to compare our algorithm results with a naive mining approach.

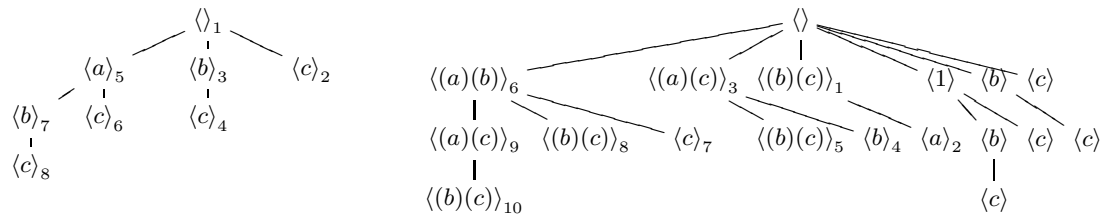


Fig. 1. Algorithm trace for database $\mathcal{D} = \{(a)(b)(c)\}$ The generations of CSPs is first done for sequences of size 1 then for sequences of size 2.

6.1 Experimental method

All experiments were performed on a Core-Duo 2.16 Ghz MacBook Pro with 2Gb of main memory, running Mac OS X 10.5.2. The conjunctive sequence pattern miner is implemented in C++ and based on the DMTL library[1] and the SPADE algorithm. We used two data sets in our experiments: a synthetic data set that was generated with QUEST[2] software and a real-world dataset containing 8 UNIX computer users logs from Purdue University over the course of up to 2 years.

The synthetic data set $C_{200}T_{2.5}S_{10}I_{10K}$ that was generated for our performance studies contains 200000 sequences based on 10000 items.

The *UNIX User Data* data set contains 9 sets of sanitized user data drawn from the command histories of 8 UNIX computer users at Purdue University. For this data set, mining CSPs and then extracting sequential association rules could be very helpful in this case for network intrusion systems. Details of the data sets are given in Table 1.

Table 1. Details on data sets used in the different experiments. # it / trans: average number of items per transaction; # trans. / sequence: average number of transactions per sequence.

Data set	# of sequences	# of items	# it / trans.	# trans. / sequence
$C_{200}T_{2.5}S_{10}I_{10K}$	200K	10K	2.5	10
UNIX User Data	11116	2016	1	39

6.2 Results

Synthetic dataset, validity and feasability. We first studied the effects of support values on the number of extracted CSPs, the overall runtime for our algorithm

[1] http://sourceforge.net/projects/dmtl
[2] http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/

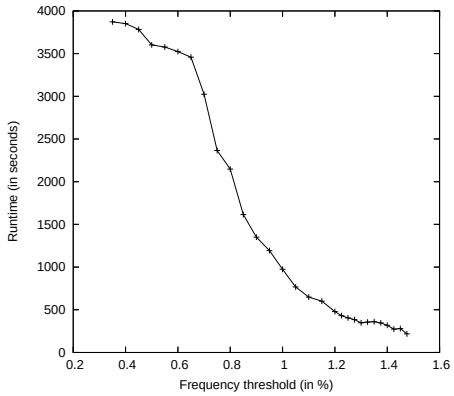
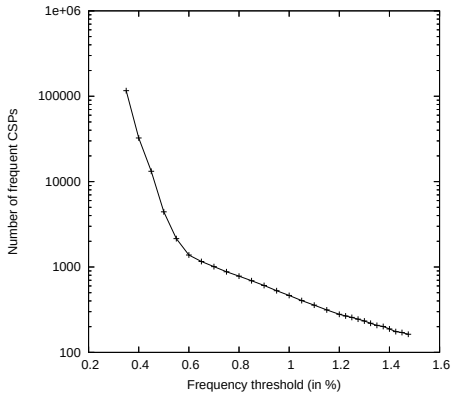
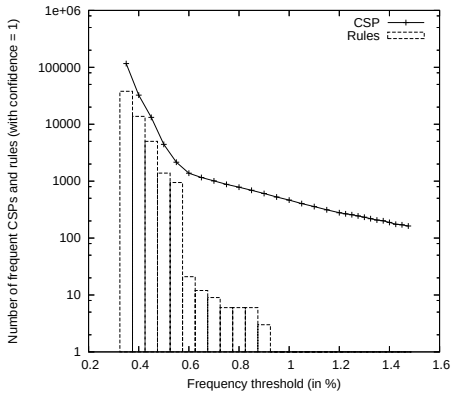


Fig. 2. Runtime experiments carried out on the data set $C_{200}T_{2.5}S_{10}I_{10K}$.



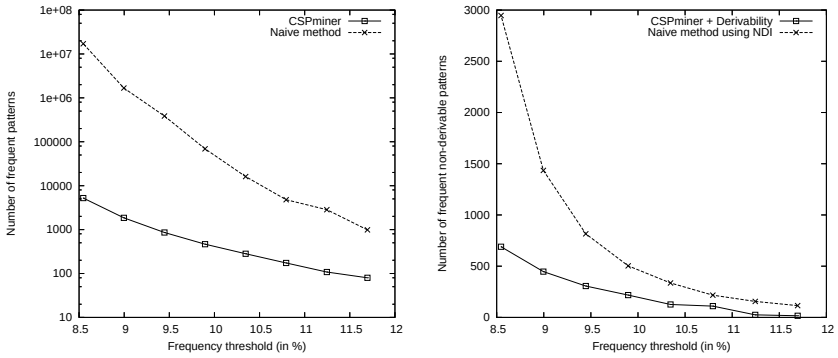
(a) Number of frequent CSPs



(b) Number of frequent CSPs and rules extracted

Fig. 3. Experiments carried out on the data set $C_{200}T_{2.5}S_{10}I_{10K}$.

and the number of extracted sequential association rules with confidence = 1. The effects of low support values on the overall runtime are exposed in Figure 2. The runtime is still very acceptable: for a support of 1.15%, the CSP miner algorithm needs 10 minutes to complete the extraction. Furthermore, our method seems to be rather immune to small support values and is still able to mine CSP until value 0.35% with acceptable overall runtime as 1 hour and 4 minutes are needed to complete the task. Figure 3(a) illustrates the number of extracted CSPs and show that our algorithm is capable of generating a very large set of frequent CSPs. The number of extracted sequential association rules with confidence = 1 w.r.t to the frequent CSPs is presented in Figure 3(b). There is no rules until reaching support value 0.95%. The explanation is that until this support values, the only frequent CSPs are of length 1 (simple sequences) which cannot be used to generate sequential association rules.



(a) Frequent patterns (b) Frequent non-derivable patterns

Fig. 4. Experiments carried out on the data set *UNIX User Data*.

UNIX User Data set and non-derivability. We used the *UNIX User Data* in these experiments to compare our approach with a naive method. The idea is to make a post-processing step after classical sequence mining. Every frequent sequence is associated with an item value and the original data set is transformed in order to keep only frequent sequences as items making the problem solvable by itemset mining. We did 2 types of itemset mining, (*i*) plain frequent itemset mining (using the *Apriori* algorithm) and (*ii*) Frequent Non-Derivable itemset mining[1]. Figure 4(a) illustrate the difference between our mining approach and the naive method. The worst case is when we are mining at a frequency of 8.54%, the CSP miner extracts 5211 CSPs while the naive method extracts more than 17 million conjunctions. This is due to the fact that the naive method do not make any comparison between every sequence in the conjunction sets extracted while our CSP miner do this kind of comparison in order to cope with the definition 4.

[1] The implementations were taken from http://www.adrem.ua.ac.be/∼goethals/software/

For example conjunction $\{\langle(a)(b)\rangle, \langle(a)\rangle\}$ will be mined and considered valid by the naive method while discarded by the CSP miner because $\langle(a)\rangle \preceq \langle(a)(b)\rangle$. Non-derivability issues for this data set are presented in Figure 4(b) and in Table 2, comparing these derivability results with the mined CSPs from Figure 4(a) nicely shows the improvements caused by the deduction rules and the high ratio shows that the non-derivable CSP representaion is indeed a good condensed representation. However, the derivability ratio stays higher for the naive method for two reasons: first, this is mainly due to the high number of frequent conjunctions mined by the naive method (more than 1 million conjunctions starting from frequency 9%), second, the incompleteness of anti-monotonicity for derivability for CSPs, discussed in Section 4, make it very hard to compete with the full anti-monotonicity of derivability for itemsets as used in the naive method.

Table 2. Details on *UNIX User Data* data set with derivability ratios for CSP miner and the naive method. # CSP: number of frequent CSPs; # Conj.: number of frequent conjunctions (naive method).# NDCSP: number of frequent non-derivable CSPs. # NDconj.: number of frequent non-derivable conjunctions (naive method).

Frequency	# CSP	# Conj.	# NDCSP	# NDconj.	CSP ratio	Conj. ratio
11.69%	80	984	16	115	20%	88.31%
11.24%	108	2840	25	156	23.14%	94.50%
10.79%	174	4784	110	217	36.78%	95.46%
10.34%	282	16146	126	336	44.68%	97.91%
9.89%	466	68991	218	503	53.21%	99.27%
9.44%	858	385107	306	816	64.29%	99.78%
8.99%	1856	1677387	447	1435	75.92%	99.91%
8.54%	5211	17127924	690	2947	86.75%	99.98%

7 Conclusions

We have introduced a new definition of equivalence classes for Sequential Patterns and investigated its computational complexity. We used these classes to exhibit a theorem stating that the lower bound for the frequency of sequential patterns is always equal to 0. This result underly that any frequency-based condensed representation is impossible for sequential patterns. Furthermore, we used the equivalence classes definition to define a new mining problem: the *Conjunctive Sequence Pattern mining* problem. We have also shown that unlike for sequences, we can compute lower and upper bounds on the pattern frequency leading to a concise representation close to the non-derivable itemsets representation. Furthermore, this new pattern is appealing as it can be used to exhibit sequential association rules.

References

1. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using bitmap representation. In *Proceedings of the 8th SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 02)*, pages 439–435, Alberta, Canada, 2002.
2. José L. Balcázar and Gemma C. Garriga. Horn axiomatizations for sequential data. *Theor. Comput. Sci.*, 371(3):247–264, 2007.
3. Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Min. Knowl. Discov.*, 7(1):5–22, 2003.
4. T. Calders. Itemset frequency satisfiability: Complexity and axiomatization. *Theoretical Computer Science*, 2007.
5. T. Calders and B. Goethals. Minimal k-free representations of frequent sets. In *PKDD proceedings*, pages 71–82, 2003.
6. T. Calders and B. Goethals. Non-derivable itemset mining. *Data Mining and Knowledge Discovery*, 14(1):171–206, 2007.
7. T. Calders, C. Rigotti, and J.F. Boulicaut. A survey on condensed representations for frequent sets. *Constraint Based Mining, Springer-Verlag, LNAI*, 3848:64–80, 2006.
8. Toon Calders and Bart Goethals. Mining all non-derivable frequent itemsets. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *PKDD*, volume 2431 of *Lecture Notes in Computer Science*, pages 74–85. Springer, 2002.
9. Toon Calders and Bart Goethals. Depth-first non-derivable itemset mining. In *SDM*, 2005.
10. K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*. Springer-Verlag, 1990.
11. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology (EDBT 96)*, pages 3–17, Avignon, France, 1996.
12. R. Agrawal R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE 95)*, pages 3–14, Tapei, Taiwan, 1995.
13. Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining closed sequential patterns in large databases. In Daniel Barbará and Chandrika Kamath, editors, *SDM*. SIAM, 2003.
14. G. Dong Z. Xing, J. Pei and P. S. Yu. Mining sequence classifiers for early prediction. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM'08)*, Atlanta, GA, April 24-26 2008.
15. Mohammed Javeed Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.