Towards Bounding Sequential Patterns*

Chedy Raïssi
INRIA
Nancy Grand Est, France
chedy.raissi@inria.fr

Jian Pei
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada
jpei@cs.sfu.ca

ABSTRACT

Given a sequence database, can we have a non-trivial upper bound on the number of sequential patterns? The problem of bounding sequential patterns is very challenging in theory due to the combinatorial complexity of sequences, even given some inspiring results on bounding itemsets in frequent itemset mining. Moreover, the problem is highly meaningful in practice, since the upper bound can be used in many applications such as space allocation in building sequence data warehouses.

In this paper, we tackle the problem of bounding sequential patterns by presenting, for the first time in the field of sequential pattern mining, strong combinatorial results on computing the number of possible sequential patterns that can be generated at a given length k. We introduce, as a case study, two novel techniques to estimate the number of candidate sequences. An extensive empirical study on both real data and synthetic data verifies the effectiveness of our methods.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; G.2.1 [**Discrete Mathematics**]: Combinatorics—*Counting problems, generating functions*

General Terms

Theory, Algorithms

*We are grateful to Drs. Toon Calders, Loïc Cerf, Marc Plantevit and Prof. Bruno Crémilleux for the inspiring and constructive discussions. We thank the anonymous reviewers for their comments and suggestions. Jian Pei's research is supported in part by an NSERC Discovery Grant and a BCFRST NRAS Endowment Research Team Program project. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Keywords

Sequential Pattern Mining, Combinatorics

1. INTRODUCTION

Sequence data is widely used in many applications. Consequently, mining sequential patterns and other types of knowledge from sequence data has become an important data mining task. The main emphasis has been on developing efficient mining algorithms and effective pattern representation [5]. However, an important fundamental problem still remains open: given a sequence database, can we have an upper bound on the number of sequential patterns in the database?

Any non-trivial answer to the question can find immediate applications in many sequence mining scenarios. For example, one issue in sequential pattern mining is that the runtime cannot be well estimated. That is, when a sequential pattern mining algorithm starts to run on a sequence database and keeps generating sequential patterns, one cannot tell how many more patterns will be generated and when the algorithm will stop. The progress of the mining algorithm cannot be estimated. If an upper bound on the number of sequential patterns can be obtained, the upper bound can be used to estimate the progress of the mining.

As another application example, building sequence data warehouses and conducting OLAP on sequences have found applications in, for example, web search [27], customer trajectory analysis [15], workflow analysis [9], and bioinformatics [4]. Space is one of the central issues in building a sequence data warehouse. How much space does a data warehouse need on a given sequence database?

The problem of bounding sequential patterns is very challenging due to the combinatorial complexity of sequences. Unlike bounding frequent itemsets [8], where each item can appear at most once in an itemset, an item may appear multiple times in a sequential pattern, which make the possible combinations much more numerous and complicated. To the best of our knowledge, no solid result has been reported in literature. Moreover, the difficulty of the problem has been well recognized. For example, Zaki [25] affirmed, "It is difficult to derive a closed-form expression for the exact number of k-sequences".

In this paper, we tackle the problem of bounding sequential patterns. We make two significant contributions. First, we present, for the first time in the field of sequential pattern mining, strong combinatorial results on bounding the number of possible sequential patterns that can be generated for a given length k. Second, we introduce, as a case

study, two novel techniques to estimate the number of candidate sequences. Fundamentally, our method is different from the results on bounding frequent itemsets [8], since the Kruskal-Katona theorem [11, 12], which is the major tool in bounding frequent itemsets, cannot be applied globally to sequences. We build different combinatorial tools based on the Whitney numbers of the poset of sequences and on prefix-itemset partitions. Using these combinatorial results, any sequence mining algorithm can *efficiently* compute upper bounds on the number of candidates to be searched. An extensive empirical study on both real data and synthetic data verifies the effectiveness of our methods. We believe that the results reported in this paper are a major breakthrough in the theory and foundation of sequential pattern mining, and may have a broad impact on other pattern mining problems.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 briefly reviews the preliminaries needed in our development. Section 4 introduces our new combinatorial results. In Section 5, we show how to apply our combinatorial results in different ways to derive an upper bound on the number of candidate sequences. An experimental study is reported in Section 6. We conclude our work and discuss several interesting directions in Section 7.

2. RELATED WORK

Since Agrawal and Srikant [2] introduced the problem of sequential pattern mining, many studies focused on developing efficient algorithms for sequential pattern mining, such as AprioriAll [2], GSP [21] and SPADE [25]. Several effective representations of sequential patterns have been proposed [24, 22]. However, to the best of our knowledge, no previous work obtains non-trivial results on estimating the number of candidates for sequential patterns.

A problem highly related to bounding sequential patterns is to bound frequent itemsets. A seminal paper by Geerts *et al.* [8] shed light on the pure combinatorial problem that lies behind the estimation of candidates in frequent itemset mining [1]. Using a well-known extremal combinatorics result, the Kruskal-Katona theorem [11, 12], Geerts *et al.* [8] provided hard and tight combinatorial upper bounds on the number of candidates. This result is also a theoretical basis for what is known as the *inverse mining problem* [19, 20], where the same Kruskal-Katona theorem is used to characterize properties of length distributions of frequent itemsets and their synthetic generation.

One may expect that the Kruskal-Katona theorem can be also used to tackle the sequential pattern bounding problem. Unfortunately, Leck [13] affirmed that there is no theorem of the Kruskal-Katona type for sequential patterns. Therefore, in this paper, we explore other fundamentally different approaches to tackle the problem of bounding sequential patterns.

3. PRELIMINARIES

Let $\mathcal{I} = \{i_1, i_2 \dots i_m\}$ be a finite set of *items*. An *itemset* $X \subseteq \mathcal{I}$ is a non-empty subset of $\mathcal{I}$. A *sequence* S over $\mathcal{I}$ is an ordered list $\langle X_1 \cdots X_n \rangle$, where X_i $(1 \leq i \leq n)$ is an itemset. The *length* of a sequence S, denoted by $|S|$, is $\sum_{i=1}^{n} |X_i|$. A *k-sequence* is a sequence of length k. We denote by $\mathbb{T}(\mathcal{I})$ 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), where $SID \in \{1, 2, \dots, |\mathcal{D}|\}$ is an identifier and $T \in \mathbb{T}(\mathcal{I})$ a sequence over $\mathcal{I}$.

Definition 1 (Subsequence). *A sequence $S' = \langle X'_1 \cdots X'_n \rangle$ is a **subsequence** of another sequence $S = \langle X_1 \dots X_m \rangle$, denoted by $S' \preceq S$, if $n \leq m$ and there exist $1 \leq i_1 < i_2 < \cdots < i_n \leq m$ such that $X'_j \subseteq X_{i_j}$ $(1 \leq j \leq n)$. S is said to be a **supersequence** of S'. S' is called a **proper subsequence** of S, denoted by $S' \prec S$, if $S' \preceq S$ and $S' \neq S$.*

Immediately, we have the following property.

Property 1. *The set $\mathbb{T}(\mathcal{I})$ with the binary relation $\preceq$ is a partially ordered set (i.e., it is reflexive, antisymmetric and transitive).*

Following with Property 1, we call $(\mathbb{T}(\mathcal{I}), \preceq)$ the **partially ordered sequence set**.

Definition 2 (Whitney numbers). *In the context of the partially ordered sequence set $(\mathbb{T}(\mathcal{I}), \preceq)$, the **Whitney numbers** w_i $(i \geq 1)$ are the numbers of i-sequences.*

Definition 3 (Concatenation and merge). *For sequences $S = \langle X_1 \cdots X_n \rangle$ and $S' = \langle X'_1 \cdots X'_m \rangle$, we define the **concatenation** operator $\circ$ such that $S \circ S' = \langle X_1 \cdots X_n X'_1 \cdots X'_m \rangle$. $S \circ S'$ is called a **concatenation supersequence** of S. We also define the **merge** operator $\diamond$ such that $S \diamond S' = \langle X_1 \cdots (X_n \cup X'_1) \cdots X'_m \rangle$. $S \diamond S'$ is called a **merge supersequence** of S.*

Definition 4 (**Prefix equivalence**). *Two sequences $S = \langle X_1 \cdots X_n \rangle$ and $S' = \langle X'_1 \cdots X'_n \rangle$ are said to be **prefix equivalent**, denoted by $S \equiv_\pi S'$, if $|S| = |S'|$ and for $(1 \leq i < n)$, $X_i = X'_i$.*

*Let $\mathcal{C}$ be a set of sequences. We denote by $[S]_\pi(\mathcal{C})$ the **set of the last itemsets of the prefix equivalence class** of S, which is the set of the last itemsets in the sequences in $\mathcal{C}$ equivalent to S, i.e., $[S]_\pi(\mathcal{C}) = \{X | S \equiv_\pi S' \circ X, S' \circ X \in \mathcal{C}\}$.*

For instance, $\langle \{a\}\{a,b\}\{a\} \rangle \equiv_\pi \langle \{a\}\{a,b\}\{b\} \rangle$, but $\langle \{a\}\{a,b\}\{a\} \rangle \not\equiv_\pi \langle \{a\}\{a,b\}\{b\}\{a\} \rangle$ because they do not have the same length. Let $\mathcal{C} = \{\langle \{a\}\{a\} \rangle, \langle \{a\}\{b\} \rangle, \langle \{a\}\{c\} \rangle, \langle \{a\}\{d\} \rangle\}$. The set of last itemsets of the prefix equivalence class for sequence $\langle \{a\}\{a\} \rangle$ is $[\langle \{a\}\{a\} \rangle]_\pi(\mathcal{C}) = \{\{a\}, \{b\}, \{c\}, \{d\}\}$.

Definition 5 (Support and frequency). *The **support** of a sequence S in a sequence database $\mathcal{D}$ is $Support(S, \mathcal{D}) = |\{(SID, T) \in \mathcal{D} | S \preceq T\}|$. The **frequency** of S in $\mathcal{D}$ is $freq_S^{\mathcal{D}} = \frac{Support(S, \mathcal{D})}{|\mathcal{D}|}$.*

Given a *minimum frequency threshold* σ, *sequential pattern mining* finds all sequences S such that $freq_S^{\mathcal{D}} \geq \sigma$, which are called the *sequential patterns*. We denote by $F_{\mathcal{D},\sigma} = \{S | freq_S^{\mathcal{D}} \geq \sigma\}$ the set of all sequential patterns, and by $F_{\mathcal{D},\sigma}^k$ the set of sequential patterns of length k.

Example 1 (Running Example). *In this paper, we use the sequence database $\mathcal{D}_{ex}$ in Table 1 as a running example. It contains 8 data sequences with $\mathcal{I} = \{a, b, c, d\}$. Given a minimum frequency threshold $\sigma = \frac{5}{8}$, $F_{\mathcal{D},\sigma} = \{\langle \{a\} \rangle, \langle \{b\} \rangle, \langle \{c\} \rangle, \langle \{d\} \rangle, \langle \{a\}\{a\} \rangle, \langle \{a\}\{b\} \rangle, \langle \{b\}\{a\} \rangle, \langle \{a,b\} \rangle, \langle \{a\}\{c\} \rangle, \langle \{a\}\{d\} \rangle, \langle \{c\}\{a\} \rangle, \langle \{a,c\} \rangle, \langle \{c\}\{b\} \rangle, \langle \{b\}\{b\} \rangle, \langle \{b\}\{c\} \rangle, \langle \{a\}\{a\}\{a\}) \rangle, \langle \{a\}\{a\}\{b\} \rangle, \langle \{a\}\{a,b\} \rangle, \langle \{a\}\{a\}\{c\} \rangle, \langle \{b\}\{a\}\{b\} \rangle, \langle \{c\}\{a\}\{b\} \rangle, \langle \{c\}\{a,b\} \rangle\}$.*

$\mathcal{D}_{ex} =$

S_1	$\langle\{a,b,c\}\{a\}\{a,b\}\{d\}\rangle$
S_2	$\langle\{a,b,c\}\{a,b,c,d\}\{d\}\{a,b,c\}\rangle\rangle$
S_3	$\langle\{a,b\}\{d\}\{c\}\rangle$
S_4	$\langle\{b\}\{a\}\{c\}\{a,b\}\{c\}\{b\}\{b\}\{c\}\{d\}\rangle$
S_5	$\langle\{a,b,c,d\}\{a\}\{a,b\}\rangle$
S_6	$\langle\{a\}\{c\}\{d\}\{b\}\{c\}\{b\}\{a\}\{b\}\{c\}\rangle$
S_7	$\langle\{a,c\}\{a\}\{a,b,c\}\rangle$
S_8	$\langle\{a\}\{a,b\}\{a,c\}\rangle$

Table 1: The sequence database used as the running example

4. COMBINATORIAL RESULTS

To find an upper bound on the number of sequential patterns, we start from calculating an upper bound on the number of potential sequential patterns of length k for every positive integer k. We tackle this problem by calculating the Whitney numbers for partially ordered sequence set.

The number of potential sequential patterns of length k is closely related to the number of different combinations made using the sequence extensions introduced in Definition 3. However, a direct estimation is difficult because it has to take into account the redundancy of some combinations. For instance, $\langle\{a\}\{b,c,d\}\{a,b\}\rangle$ can be constructed in many different ways: $\langle\{a\}\{b\}\rangle\diamond\langle\{c,d\}\{a,b\}\rangle$, $\langle\{a\}\{b,c\}\rangle\diamond\langle\{d\}\{a,b\}\rangle$, $\langle\{a\}\rangle\circ\langle\{b,c,d\}\{a,b\}\rangle$, etc. Developing a combinatorial formula that takes into account those redundancies is very difficult, if possible at all.

A simple observation can help to deal with the problem. A k-sequence S can only contain at most k itemsets. In fact, the number of possible k-sequences can be calculated by the number of concatenation extensions. For instance, for $\mathcal{I} = \{a, b\}$, the number of 2-sequences is the number of sequences that are obtained from the following two ways.

- The concatenation extension from the empty sequence using a 2-itemset: $\langle\emptyset\rangle \circ \langle\{a, b\}\rangle$; and
- The sequences of 2 length-1 itemsets, which are concatenation extensions of all possible sequences with 1-itemsets: $\langle\{a\}\rangle \circ \langle\{a\}\rangle$, $\langle\{a\}\rangle \circ \langle\{b\}\rangle$, $\langle\{b\}\rangle \circ \langle\{a\}\rangle$ and $\langle\{b\}\rangle \circ \langle\{b\}\rangle$.

Clearly, this way of enumerating the sequences avoids the redundancy problem described earlier. The correctness was established as part of the PrefixSpan algorithm [17]. Using this observation, counting the number of k-sequences can be described by the following recurrence relation:

$$w_k = \sum_{i=0}^{k-1} w_i \binom{n}{k-i}, \tag{1}$$

where $n = |\mathcal{I}|$, $w_0 = 1$, $w_1 = n$, and $k \geq 0$[1].

When $|\mathcal{I}|$ and k are both small, the above recurrence relation may be efficient for calculation. However, we would like to obtain an *explicit* formula. From the combinatorial point of view, this formula may help us understand the relations between counting sequential patterns and some other well-known combinatorial problems. The understanding may lay down new ways of handling the sequential pattern mining problem.

[1]Note that: $\binom{n}{k} = 0$ if $k > n$

In order to obtain such a formula, we want to represent the Whitney numbers w_k in a generating function [23, 7][2]

$$S(x) = \sum_k w_k x^k \tag{2}$$

From Equation (2), it follows that

$$w_k = \frac{S^k(0)}{k!}, \tag{3}$$

where $S^k(x)$ is the k^{th} derivative of $S(x)$.

Obtaining the generating function in our case is far from trivial. One of the popular methods to find generating functions is to compute the discrete convolution of two formal power series, in other words, their *Cauchy product*. Recall that for two formal power series $\sum_k a_k x^k$ and $\sum_k b_k x^k$, their Cauchy product is $\sum_{k=0}^{\infty} z_k x^k$ with $z_k = \sum_{i=0}^{k} a_i b_{k-i}$. As noted by Wilf [23]: *"It is certainly this product rule that accounts for the wide applicability of series methods in combinatorial problems"*.

In our case, we use one known formal power series, which can be trivially derived from the binomial theorem:

$$\sum_{i=0}^{n} \binom{n}{i} x^{n-i} = (1+x)^n = D(x). \tag{4}$$

We apply the Cauchy product with $S(x)$:

$$\begin{aligned} \sum_k z_k x^k &= S(x) \cdot D(x) \\ &= (\sum_k w_k x^k) \cdot (\sum_{i=0}^{n} \binom{n}{i} x^{n-i}) \\ &= \sum_{k=0}^{\infty} \sum_{i=0}^{k} w_i \binom{n}{k-i} x^k \end{aligned} \tag{5}$$

Thus, we have,

$$z_k = \sum_{i=0}^{k} w_i \binom{n}{k-i} \tag{6}$$

Two cases are possible for z_k, depending on the value of k. First, if $k = 0$, then $z_0 = w_0$. Second, if $k > 0$, then, from Equation (1), we have

$$\begin{aligned} z_k &= w_k + \sum_{i=0}^{k-1} w_i \binom{n}{k-i} \\ &= 2w_k \end{aligned} \tag{7}$$

Using (5), we can exhibit the generating function $S(x)$:

$$\begin{aligned} \sum_k z_k x^k &= S(x) \cdot (1+x)^n \\ &= z_0 + \sum_{k \geq 1} 2 w_k x^k \\ &= w_0 + 2 \sum_{k \geq 1} w_k x^k \\ &= 1 + 2(S(x) - 1) \end{aligned} \tag{8}$$

Finally,

$$S(x) = \frac{1}{2 - (1+x)^n} \tag{9}$$

[2]For a brief introduction to generating functions, please refer to `http://en.wikipedia.org/wiki/Generating_function`.

From this generating function, we can deduce an expression on the number of k-sequences that can be formed using $n = |\mathcal{I}|$ different items. From (9) it follows

$$\begin{aligned} S(x) &= \tfrac{1}{2}\left(\tfrac{1}{1-\frac{1}{2}(1+x)^n}\right) \\ &= \tfrac{1}{2}\left(\tfrac{1}{1-Z}\right) \\ &= \tfrac{1}{2}\sum_{i\geq 0} Z^i \end{aligned}$$

$$\text{with } Z = \frac{1}{2}(1+x)^n,$$

$$= \tfrac{1}{2}\sum_{i\geq 0}\frac{1}{2^i}(1+x)^{ni} \qquad (10)$$

Therefore, the k^{th} derivative of $S(x)$ is:

$$\begin{aligned} S^k(x) &= \sum_{i\geq 0}\frac{\frac{ni!}{(ni-k)!}(1+x)^{ni-k}}{2^{i+1}} \\ &= k!\sum_{i\geq 0}\frac{\frac{ni!}{(ni-k)!k!}(1+x)^{ni-k}}{2^{i+1}} \\ &= k!\sum_{i\geq 0}\frac{\binom{ni}{k}(1+x)^{ni-k}}{2^{i+1}} \qquad (11) \end{aligned}$$

From Equation (3), we have

$$w_k = \sum_{i\geq 0}\frac{\binom{ni}{k}}{2^{i+1}} \qquad (12)$$

We do not discuss the convergence case for the series $S(x)$ with $x = 0$. In fact, the analytic nature of the generating function does not interest us here. Thus, we limit our view of a generating function to serving as only a formal power series, i.e., as an algebraic object rather than as an analytic one. The multiple links between the previous expression and well-known combinatorial problems are discussed in Section 7.

EXAMPLE 2. *Let $\mathcal{I} = \{a, b\}$. The number of possible 2-sequences can be calculated using the recurrence relation in Equation (1):*

$$\begin{aligned} w_2 &= w_0\binom{2}{2} + w_1\binom{2}{1} \\ &= 1 + 4 = 5 \end{aligned}$$

The same result can be obtained with Equation (12): $w_2 = \sum_{i\geq 0}\frac{\binom{2i}{2}}{2^{i+1}} = 5$.

Likewise, the number of possible 3-sequences can be computed through the recurrence relation in Equation (1):

$$\begin{aligned} w_3 &= w_0\binom{2}{3} + w_1\binom{2}{2} + w_2\binom{2}{1} \\ &= 0 + 2 + 10 = 12 \end{aligned}$$

Using the formula in Equation (12), we have $w_3 = \sum_{i\geq 0}\frac{\binom{2i}{3}}{2^{i+1}} = 12$. The set of all 3-sequences is

$\{\ \langle\{a\}\{a\}\{a\}\rangle, \langle\{a\}\{a\}\{b\}\rangle, \langle\{a\}\{b\}\{a\}\rangle, \langle\{a\}\{b\}\{b\}\rangle,$
$\langle\{b\}\{a\}\{a\}\rangle, \langle\{b\}\{a\}\{b\}\rangle, \langle\{b\}\{b\}\{a\}\rangle, \langle\{b\}\{b\}\{b\}\rangle,$
$\langle\{a,b\}\{a\}\rangle, \langle\{a,b\}\{b\}\rangle, \langle\{a\}\{a,b\}\rangle, \langle\{b\}\{a,b\}\rangle\ \}$

5. ESTIMATION OF CANDIDATES

In this section, we use the combinatorial results to compute the upper bounds on the number of candidates in sequential pattern mining, which are naturally the upper bounds on the number of sequential patterns. We discuss two techniques.

5.1 A Basic Upper Bound

The first technique relies on the recurrence relation in Equation (1). Instead of using the Whitney numbers w_i $(0 \leq i < k)$ to compute the number of candidate k-sequences, we use the number of sequential patterns of every length i. Formally, we define C_k, an upper bound on the number of k-sequence candidates as follows.

$$C_k = \sum_{i=0}^{k-1}|F^i_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{k-i} \text{ with } |F^0_{\mathcal{D},\sigma}| = 1 \qquad (13)$$

EXAMPLE 3. *Consider our running example using the database $\mathcal{D}_{ex}$ in Table 1. Suppose $\sigma = \frac{5}{8}$. The set of length-1 sequential patterns in $\mathcal{D}_{ex}$ is $F^1_{\mathcal{D},\sigma} = \{\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle, \langle\{d\}\rangle\}$. Accordingly, an upper bound on the number of length-2 candidates that can be generated from these 1-sequences is*

$$\begin{aligned} C_2 &= |F^0_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{2} + |F^1_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{1} \\ &= |F^0_{\mathcal{D},\sigma}|\cdot 6 + |F^1_{\mathcal{D},\sigma}|\cdot 4 \\ &= 6 + 16 = 22 \end{aligned}$$

The number 22 includes the 6 possible 2-sequences with only 1 itemset: $\{\langle\{a,b\}\rangle, \langle\{a,c\}\rangle, \langle\{a,d\}\rangle, \langle\{b,c\}\rangle, \langle\{b,d\}\rangle, \langle\{c,d\}\rangle\}$, and the 16 2-sequences with 2 itemsets, such as $\langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{d\}\{a\}\rangle$, etc. Unsurprisingly, any breadth-first search algorithm based on the Apriori paradigm generates exactly those 22 sequences. In such a case, the upper bound is tight and realizable.

Similarly, we have

$$\begin{aligned} C_3 &= |F^0_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{3} + |F^1_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{2} + |F^2_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{1} \\ &= |F^0_{\mathcal{D},\sigma}|\cdot 4 + |F^1_{\mathcal{D},\sigma}|\cdot 6 + |F^2_{\mathcal{D},\sigma}|\cdot 4 \\ &= 4 + 24 + 44 \\ &= 72 \end{aligned}$$

Notice that this basic upper bound is effective for sequence databases of a small number of items, that is, $n = |\mathcal{I}|$ is small, since it only relies on multiplying binomial coefficients. However, for real-world data sets containing thousands of items, this bound may be very loose.

5.2 Prefix-based Bound

The bounding method discussed in Section 5.1 is easy to implement in any breadth-first search sequence mining algorithm. However, it only uses very little information in bounding, and can be improved substantially. In sequential pattern mining, an algorithm often can access to more information than just the number of sequential patterns of a certain length. Often, all sequential patterns up to a certain length k are available. We develop a prefix-based bound to make use of this extra information to achieve a better estimation by locally applying the Kruskal-Katona theorem.

THEOREM 1 (l-BINOMIAL REPRESENTATION [11, 12]). *Given two positive integers m and l, there exists a unique*

representation of m in the form

$$m = \binom{a_l}{l} + \binom{a_{l-1}}{l-1} + \ldots + \binom{a_t}{t}$$

where $a_l > a_{l-1} > \ldots > a_t \geq t \geq 1$.

For instance, when $m = 26$ and $l = 4$, we have $26 = \binom{6}{4} + \binom{5}{3} + \binom{2}{2}$.

DEFINITION 6 (SHADE). *An itemset $X \subseteq \mathcal{I}$ is a **k-subset** if $|X| = k$. Let $\mathcal{A}$ be a set of k-subsets of $\mathcal{I}$, where $k < |\mathcal{I}|$. The set $\bigtriangledown_p \mathcal{A} = \{D \subseteq \mathcal{I} : |D| = k + p, \forall \text{ k-subset } D' \subset D \rightarrow D' \in \mathcal{A}\}$ is called the **shade** of $\mathcal{A}$, where $p \geq 1$.*

*Let $\mathcal{B}$ be a set of k-sequences such that $\mathcal{B} \subseteq \mathbb{T}(\mathcal{I})$. The set $\bigtriangledown_p \mathcal{B} = \{S \in \mathbb{T}(\mathcal{I}) : |S| = k+p, \forall S' \prec S, |S'| = k \rightarrow S' \in \mathcal{B}\}$ is called the **shade** of $\mathcal{B}$. In other words, $\bigtriangledown_p \mathcal{B}$ consists of all $(k+p)$-sequences that can be constructed by using operators $\circ$ and $\diamond$ on the set of sequences $\mathcal{B}$.*

For instance, let $\mathcal{I} = \{a, b, c, d\}$ and $\mathcal{A} = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}\}$ then $\bigtriangledown_1 \mathcal{A} = \{\{a, b, c\}, \{a, b, d\}\}$ and $\{a, c, d\} \notin \bigtriangledown_1 \mathcal{A}$ because $\{c, d\} \notin \mathcal{A}$.

The next theorem, the Kruskal-Katona theorem, gives a tighter upper bound on the size of the shade for a set of k-subsets.

THEOREM 2 (KRUSKAL-KATONA THEOREM [11, 12]). *Let $\mathcal{A}$ be a set of k-subsets of $\mathcal{I}$ ($k < |\mathcal{I}|$). $|\mathcal{A}|$ can be written as a k-binomial representation:*

$$|\mathcal{A}| = m = \binom{a_k}{k} + \binom{a_{k-1}}{k-1} + \ldots + \binom{a_t}{t}.$$

The cardinality of $\bigtriangledown_p \mathcal{A}$ is bounded by

$$|\bigtriangledown_p \mathcal{A}| \leq \binom{a_k}{k+p} + \binom{a_{k-1}}{k-1+p} + \ldots + \binom{a_t}{t+p}$$

No other bounds can be tighter.

For instance, let $\mathcal{I} = \{a, b, c, d\}$ and $\mathcal{A} = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}\}$. Then, $|\mathcal{A}| = 5 = \binom{3}{2} + \binom{2}{1}$. Moreover, $|\bigtriangledown_1 \mathcal{A}| \leq \binom{3}{3} + \binom{2}{2} = 2$, which is a tight upper bound because $\bigtriangledown_1 \mathcal{A} = \{\{a, b, c\}, \{a, b, d\}\}$.

To understand the prefix-based technique, recall that a candidate sequence can be generated through the sequence extension operators. However, with the recurrence relation in Equation (13), some of the candidate sequences that are counted are *obviously infrequent.*

EXAMPLE 4. *To count the number of candidates C_3 in Example 3, $|F^1_{\mathcal{D},\sigma}|\binom{|F^1_{\mathcal{D},\sigma}|}{2}$ is used to compute the number of 1-sequences extended with itemsets of size 2. In this case, $\langle\{b\}\{a, d\}\rangle$, $\langle\{b\}\{b, d\}\rangle$,$\langle\{b\}\{c, d\}\rangle$ are counted as possible candidates, however, $\langle\{b\}\{d\}\rangle$ is not frequent. In fact, the number of sequences of length 1 extended with itemsets of cardinality 2 should be constrained by the sequences of length 2 and with 2 itemsets: $\langle\{b\}\{a\}\rangle$, $\langle\{b\}\{b\}\rangle$, $\langle\{b\}\{c\}\rangle$, which are from the prefix equivalence class of sequence $\langle\{b\}\{a\}\rangle$.*

In order to reflect this condition and generalize it to k-sequences, we will use *locally* the Kruskal-Katona theorem, since it computes efficiently a tight upper bound on the shade of the set of the last itemsets for a given prefix equivalence class. Let us look at an example.

Procedure PB(): regrouping the sequences in prefix equivalence classes.

Data: $F^k_{\mathcal{D},\sigma}$
Result: Prefix-based upper bound value for the number of candidate $k+1$-sequences

```
PB ← 0;
foreach set A = [⟨S⟩]_π(F^k_{D,σ}) do
    local ← Compute the upper bound ▽1 A;
    PB ← PB + local;
PB ← PB + |F^k_{D,σ}| · |F^1_{D,σ}|;
Return PB;
```

EXAMPLE 5. *Let PB_k be the prefix-based upper bound on the size of k-sequences candidates. To compute PB_3, we need to compute the bounds on the sequences generated from the length-2 sequential patterns with extension operators $\circ$ and $\diamond$, respectively.*

The $\diamond$-extension case: there are 4 prefix equivalence classes in $F^2_{\mathcal{D},\sigma}$: 1 prefix equivalence class based on the sequences containing 1 itemset with cardinality 2, and 3 prefix equivalence classes based on the sequences of 2 itemsets:

$$\begin{array}{lll} (i) & \mathcal{U} = & \{\langle\{a, b\}\rangle, \langle\{a, c\}\rangle\} \\ (ii) & \mathcal{V} = & \{\langle\{a\}\{a\}\rangle, \langle\{a\}\{b\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a\}\{d\}\rangle\} \\ (iii) & \mathcal{W} = & \{\langle\{b\}\{a\}\rangle, \langle\{b\}\{b\}\rangle, \langle\{b\}\{c\}\rangle\} \\ (iv) & \mathcal{X} = & \{\langle\{c\}\{a\}\rangle, \langle\{c\}\{b\}\rangle\} \end{array}$$

For each of these equivalence classes, we compute an upper bound based on the Kruska-Katona theorem. This theorem can be applied because the elements of the set $[\langle S\rangle]_\pi(\mathcal{V})$ are sets. For instance, from (ii), $|[\langle\{a\}(a)\rangle]_\pi(F^2_{\mathcal{D},\sigma})| = \binom{4}{1} = 4$. The cardinality of $\bigtriangledown_1 \mathcal{V}$ is bounded by

$$|\bigtriangledown_1 \mathcal{V}| \leq \binom{4}{1+1} = 6$$

Similarly, $|\bigtriangledown_1 \mathcal{U}| \leq 0$, $|\bigtriangledown_1 \mathcal{W}| \leq 3$ and $|\bigtriangledown_1 \mathcal{X}| \leq 1$. Thus, the upper bound on the number of sequences that can be generated using the operator $\diamond$ is $0 + 6 + 3 + 1 = 10$.

The $\circ$-extension case: Each of the sequences in $F^2_{\mathcal{D},\sigma}$ can be extended with one of the frequent items present in $F^1_{\mathcal{D},\sigma} = \{\langle\{a\}\rangle, \langle\{b\}\rangle, \langle\{c\}\rangle, \langle(d)\rangle\}$. In this example, the upper bound on the number of sequences that can be generated using the operator $\circ$ is $|F^2_{\mathcal{D},\sigma}| \cdot |F^1_{\mathcal{D},\sigma}| = 11 \cdot 4 = 44$.

In conclusion, $PB_3 = 10 + 44 = 54$. That is, the maximum number of candidate patterns of length 3 that can be generated is at most 54, smaller than C_3 in Example 3.

The only requirement for this approach is to regroup the sequences in prefix equivalence classes. This is usually implicitly done thanks to the data structures used in the sequential pattern mining algorithms, such as prefix-trees (i.e., trie structures). The pseudocode of the procedure is given in Procedure PB().

Further Improvements

The prefix-based upper bound works well with any breadth-first search algorithms. Still, some further improvements are possible.

One may notice that handling the case of the operator $\circ$ is quite trivial as it involves multiplying the number of frequent k-sequences with the number of frequent 1-sequences.

However, this way allows some false positives since not all 2-sequences of 2 items are frequent. Recall that any mining algorithm has access to the sequences in the database. Instead of blindly multiplying with the cardinality of the frequent 1-sequences, we can multiply only with the cardinality of the 2-sequences that can really result in sequence expansions. Let us illustrate this improvement using an example.

EXAMPLE 6. *Suppose that we want to extend the sequences from the prefix equivalence class* $\mathcal{A} = \{\langle\{a,b\}\rangle, \langle\{a,c\}\rangle\}$ *with the operator* $\circ$. *Based on Procedure* $PB()$, *this equivalence class generates* $|\mathcal{A}| \cdot |F^1_{\mathcal{D},\sigma}| = 8$ *candidate sequences, which are*

$$\{ \quad \langle\{a,b\}\{a\}\rangle, \langle\{a,b\}\{b\}\rangle, \langle\{a,b\}\{c\}\rangle, \langle\{a,b\}\{d\}\rangle, \\ \langle\{a,c\}\{a\}\rangle, \langle\{a,c\}\{b\}\rangle, \langle\{a,c\}\{c\}\rangle, \langle\{a,c\}\{d\}\rangle \quad \}$$

Yet, some of the sequences should not be generated as their sub-sequences are infrequent. For instance, $\langle\{b\}\{d\}\rangle$ *is infrequent. However,* $\langle\{a,b\}\{d\}\rangle$ *is counted as a possible candidate sequence.* $\langle\{a,b\}\rangle$ *should be extended only with the items in the set* $\{a,b,c\}$, *since only* $\langle\{a\}\{a\}\rangle$, $\langle\{a\}\{b\}\rangle$, $\langle\{a\}\{c\}\rangle$, $\langle\{b\}\{a\}\rangle$, $\langle\{b\}\{b\}\rangle$, *and* $\langle\{b\}\{c\}\rangle$ *are frequent (notice how item* d *is not expanded here). Applying the same process to each element of the equivalence classes yields the following*

$$\begin{aligned} |\{\langle\{a,b\}\rangle\}| \cdot 3 &+ |\{\langle\{a,c\}\rangle\}| \cdot 2 + |\{\langle\{a\}\{a\}\rangle\}| \cdot 4 + \\ |\{\langle\{a\}\{b\}\rangle\}| \cdot 3 &+ |\{\langle\{a\}\{c\}\rangle\}| \cdot 2 + |\{\langle\{a\}\{d\}\rangle\}| \cdot 0 + \\ |\{\langle\{b\}\{a\}\rangle\}| \cdot 3 &+ |\{\langle\{b\}\{b\}\rangle\}| \cdot 3 + |\{\langle\{b\}\{c\}\rangle\}| \cdot 2 + \\ |\{\langle\{c\}\{a\}\rangle\}| \cdot 2 &+ |\{\langle\{c\}\{b\}\rangle\}| \cdot 2 = 26 \end{aligned}$$

The improved prefix-based upper bound in this case is $PB_3^{imp} = 10 + 26 = 36$.

The implementation of this improvement bears negligible overhead in practice because all the needed sequences along with their extensions are already present in the data structures used by many sequence mining algorithms.

6. EMPIRICAL STUDY

In this section, we report an extensive empirical evaluation of our bounding techniques using real and synthetic data sets. All experiments were performed on an Apple MacBook computer with 2.8 Ghz Intel Core 2 Duo CPU and 4 Gb main memory, running Mac OS X 10.6.6 operating system. The bounding procedures were implemented in C++ along with a version of the algorithm SPADE using the Data Mining Template Library [26]. All the mathematical computation was processed using the GNU Multiple Precision Arithmetic Library[3]. The source code and the data sets are available at `http://www.loria.fr/~raissi/`.

Data sets. In our experiments, we used one real data set and a group of synthetic data sets generated by the QUEST software[4]. The *Amazon* data set consists of a collection of 100,000 product reviews from the website `amazon.com`. This is a sample of the data set used in [10]. Each review is a data sequence that has been lemmatized and grammatically filtered to remove uninteresting terms. Each sentence in a review is transformed into an itemset. The data set is very dense and contains 32,140 items. The average sequence length is 8. For synthetic data sets, various kinds of parameters and distributions were used to generate different data sets. We used the following convention to name the synthetic data sets: Cx means that the data set contains $x \times 1,000$ sequences, Ty means that the average number of items in an itemset is y, and Iz means that the data set contains $z \times 1,000$ distinct items.

[3] `http://gmplib.org/`

[4] `http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/`

Amazon data set, $\sigma = 0.15\%$.

Length k	Cand. gen. time	C_k	PB_k	PB_k^{opt}
1	**16.4534**	0	0	0
2	**117.57**	0.000003	0.001984	0.001984
3	**778.726**	0.000025	0.013382	0.045356
4	**24.5185**	0.000006	0.002798	0.004168
5	**0.0618551**	0.000009	0.001266	0.002395

C100T4I10 data set, $\sigma = 8\%$.

Length k	Cand. gen. time	C_k	PB_k	PB_k^{opt}
1	**16.1838**	0	0	0
2	**77.9223**	0.000005	0.000124	0.000124
3	**285.861**	0.000005	0.006093	0.032533
4	**352.146**	0.000006	0.240952	0.169538
5	**283.433**	0.000006	0.039826	0.340521
6	**121.09**	0.000006	0.035317	0.329862
7	**32.4553**	0.0000010	0.019776	0.147073
8	**5.18174**	0.0000010	0.007598	0.039192
9	**0.624778**	0.0000012	0.003519	0.007164
10	**0.0670125**	0.0000010	0.002802	0.001828

Table 2: Candidate generation time vs. upper bound computation time (in seconds).

Efficiency. The first batch of experiments focused on efficiency. As previously discussed, the upper bound computation depends on the number of sequential patterns. How does the computation time change when the number of sequential patterns increases? Using the *Amazon* data set, we varied the minimal support threshold and recorded for each bounding method its total computation time and the total number of sequential patterns. 240 thresholds were tried. Figure 1 shows these results. The experimental results verify the theoretical analysis. The three bounding methods *scale linearly* with respect to the number of sequential patterns.

We also compared the runtime of the bounding methods with respect to the candidate generation runtime in SPADE. If the bounding procedures are too slow comparing to the candidate generation time, then bounding candidates of sequential patterns would not help much in improving sequential pattern mining efficiency. Table 2 shows the results. Clearly, the computation cost for the upper bounds is negligible comparing to the candidate generation process.

Accuracy. Our second batch of experiments focused on the accuracy of the bounds. Figure 2 shows the results on two different synthetic data sets. As discussed in Section 5, the basic upper bound is much looser. For instance, on data set $C500T4I10$ with a minimal frequency threshold $\sigma = 10\%$, at level $k = 3$, the number of generated candidates by SPADE is 2,265 but the basic bound is 167,567,925,000. At the same level, the prefix-based bound is 2,599, which is very close to the actual number of generated candidates. This observation is further confirmed by the rest of our

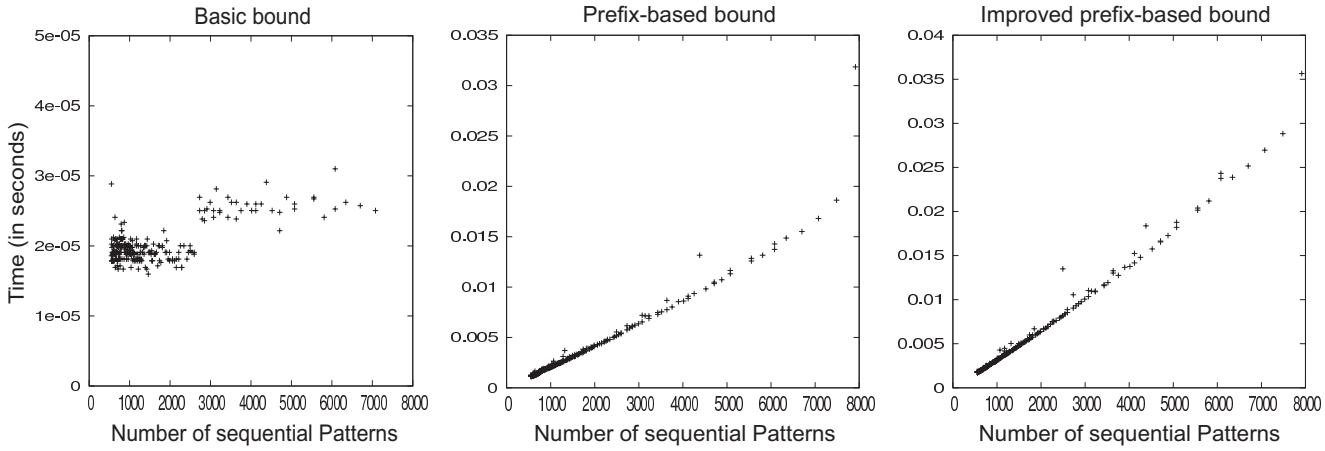

Figure 1: Time needed to compute the three upper bounds vs. the number of frequent sequences.

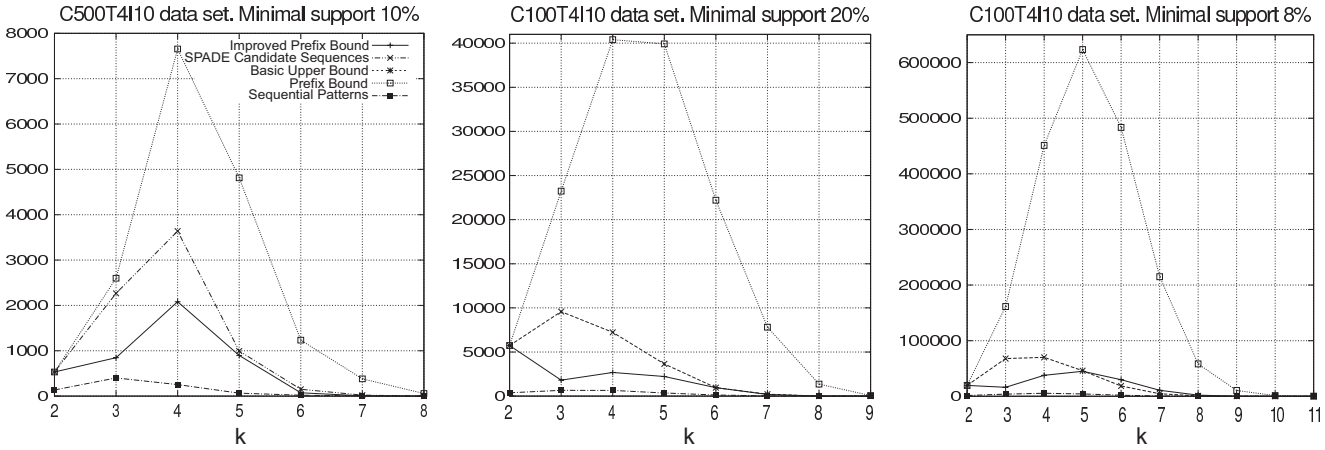

Figure 2: Upper bounds accuracy comparison with SPADE candidate generation.

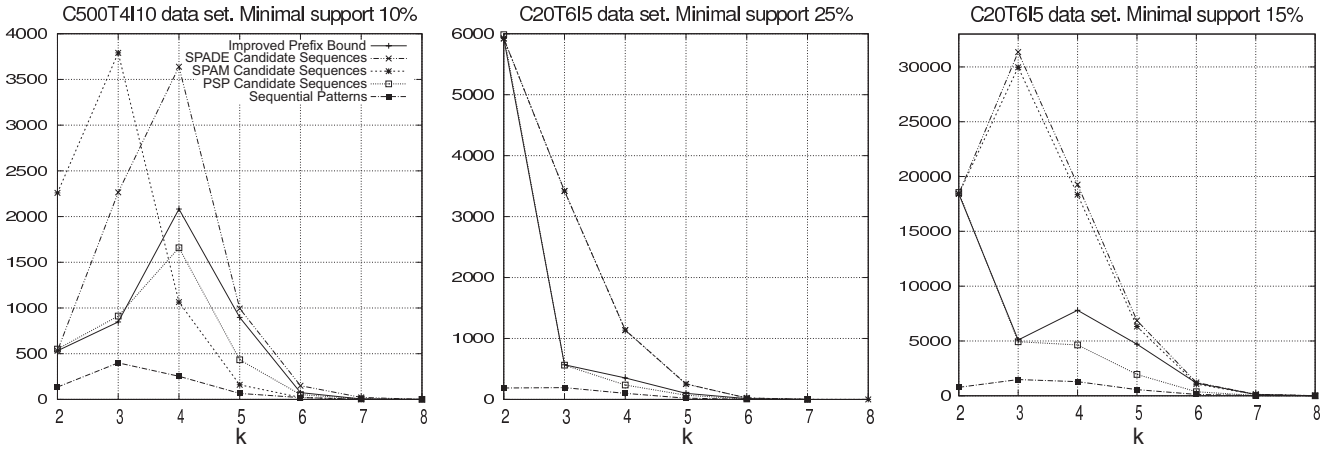

Figure 3: Optimized prefix-bound accuracy comparison with SPADE, SPAM and PSP candidate generations.

experiments, and indicates that the prefix-based bound is viable and useful. Surprisingly, the improved prefix-based bound is 847, dramatically less than the number of candidates generated by SPADE, and much closer to the number of sequential patterns. In fact, in more than 89% of the cases in our experiments, the improved prefix-based bound is lower than the number of candidates generated by SPADE, and is closer to the actual number of sequential patterns.

To further test the effectiveness of the improved prefix-based bound, we compared it with some other representative sequential pattern mining methods. We selected two algorithms that have radically different candidate generation strategies: SPAM [3] and PSP [16]. Please note that SPAM is a depth-first search method and is reported to outperform PrefixSpan. The results are shown in Figure 3. Our improved prefix-based bound is lower than what is actually generated by the SPADE and SPAM algorithms. Two explanations are possible. First, SPAM is a depth-first algorithm and thus cannot fully use the Apriori principle in pruning unfruitful candidates. Second, the candidate generation process in SPADE produces some false positive candidates. Recall that the candidate generation method in SPADE is based on a prefix equivalence constraint. For example, let $S_1 = \langle\{a\}\{b\}\{a\}\rangle$ and $S_2 = \langle\{a\}\{b\}\{b\}\rangle$, then three candidates, $\langle\{a\}\{b\}\{a\}\{b\}\rangle, \langle\{a\}\{b\}\{b\}\{a\}\rangle$, and $\langle\{a\}\{b\}\{a,b\}\rangle$, are generated. However, no prior check is done to see if $\langle\{a,b\}\rangle$ is actually frequent. This kind of candidates are quite often generated in many sequential pattern mining algorithms. Our improved prefix-based bound avoids such candidates in counting.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we reported a breakthrough in the theory and foundation of sequential pattern mining. We presented, for the first time in the field of sequential pattern mining, strong combinatorial results on computing the number of possible sequences that can be generated at a given length k. Two novel techniques were developed to estimate the number of candidate sequences. The effectiveness and efficiency of our methods were well verified by an extensive empirical study on both real data and synthetic data.

Motivated by the theoretical challenge of computing the Whitney numbers for sequential patterns, we provided in this paper novel ways of bounding sequential patterns. However, our findings are not solely restricted to this unique purpose. In this section, we discuss as future work several other applications in the domains of combinatorics, data mining and machine learning that can benefit from this study.

Bridging combinatorial problems and sequence mining problems. Closed form expressions and generating functions act like identification cards for combinatorial problems. Using 12, we notice that for $n = 2$ (i.e., 2 items), the first few terms of the sequence of Whitney numbers are $1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, \ldots$ which are known as the *"Pell numbers"* (A000129 in OEIS[5]). In fact, counting the number of sequences on $\mathcal{I} = \{a, b\}$ is similar to counting the number of 132-avoiding two-stack sortable permutations [6], which is a special type of pattern-avoiding permutations. We believe our work may help to investigate combinatorial techniques associated with pattern-avoidance problems and applying them for constrained sequence mining [18].

Generalized spectrum kernel. Leslie *et al.* [14] introduced a new sequence-similarity kernel, the *spectrum kernel*, to deal with classification of protein sequences with support-vector machines (SVMs) and other kernel methods. The basic idea of applying SVMs on sequence data is to map sequences into feature spaces by means of kernel functions and compute the maximum-margin hyperplane to separate two classes. The k-spectrum of a sequence is the set of all length-k contiguous subsequences that it contains. The feature map is indexed by all possible length-k subsequences (i.e., k-mers) from $\mathcal{I}$ and its dimensionality is $|\mathcal{I}|^k$. Notice that this kernel works for simple sequences (i.e., ordered list of items). To generalize this kernel to sequential patterns we have to, first, be able to count the size of the feature map, and, second, develop an efficient computation method based on the *"kernel trick"*, that is, mapping sequences into an inner product space. For the first task, it is easy to notice that for a given length k, the size of the feature map is equal to w_k defined in Equations (1) and (12). For the second task, efficient computation could be carried by a traversal of a trie data structure like the one used by several sequential pattern mining algorithms. So far, the overall complexity of computing the kernel of 2 sequences remains an open problem. We believe that based on our work, further research on the generalization of the sequence kernels could be applied in natural language processing applications where sequential patterns may represent different sentences or grammatical groups (for instance, a nominal group could be represented in a sequence by its own itemset).

Sequence inverse mining problem and privacy issues. From the privacy point of view, data owners may not want to share their original data sets. Instead, they may consider sharing some anonymized or synthetic data sets that respect the same pattern distribution. However, one concern is how many data sets exist that respect the same pattern distribution, and how much they can be different from each other. Can an attacker "enumerate" or "characterize" such data sets sufficiently precisely? In frequent pattern mining, the performance of frequent itemset mining algorithms depends on the length distribution of the patterns. To improve the experimental procedures, data miners may want to generate data sets (i.e., benchmarks) with a given distribution on the number of patterns. This approach was successfully applied to itemset mining in [19, 20] and is mainly based on the Kruskal-Katona theorem. With the theoretical results in this paper, the reverse mining problem for sequential patterns can be investigated with a strong theoretical basis. For instance, suppose that $\mathcal{I} = \{a, b\}$ and we want to generate a data set that contains exactly 2 sequential patterns of length 1, 7 patterns of length-2, and 1 patterns of length 3. The upper bound feasability condition [19] can be answered simply using Equation (1). In this case, the answer is "no" because, with $n = 2$ and $w_2 = 5$, one cannot generate 7 different 2-sequences. However, the lower bound feasibility condition remains an open problem.

[5]On-Line Encyclopedia of Integer Sequences: `http://oeis.org`.

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large database. In *Proceedings of the International Conference on Management of Data (ACM SIGMOD 93)*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE 95)*, pages 3–14, Taipei, Taiwan, 1995.

[3] 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.

[4] M. Cornell, N. W. Paton, S. Wu, C. A. Goble, C. J. Miller, P. Kirby, K. Eilbeck, A. Brass, A. Hayes, and S. G. Oliver. Gims - a data warehouse for storage and analysis of genome sequence and functional data. In *BIBE*, pages 15–22, 2001.

[5] G. Dong and J. Pei. *Sequence Data Mining*. Springer, USA, 2007.

[6] E. S. Egge and T. Mansour. 132-avoiding two-stack sortable permutations, fibonacci numbers, and pell numbers. *Discrete Appl. Math.*, 143:72–83, September 2004.

[7] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[8] F. Geerts, B. Goethals, and J. V. den Bussche. A tight upper bound on the number of candidate patterns. In N. Cercone, T. Y. Lin, and X. Wu, editors, *ICDM*, pages 155–162. IEEE Computer Society, 2001.

[9] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive rfid data sets. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 83–, Washington, DC, USA, 2006. IEEE Computer Society.

[10] N. Jindal and B. Liu. Opinion spam and analysis. In M. Najork, A. Z. Broder, and S. Chakrabarti, editors, *WSDM*, pages 219–230. ACM, 2008.

[11] G. Katona. A theorem of finite sets. *Theory of Graphs, Proc. Coll. held at Tihany, Hungary, September, 1966 (Akadémiai Kiadó, Budapest, 1968), reprinted in Classic Papers in Combinatorics, Ed. I. Gessel and G.-C. Rota, BirkhŁuser, Boston, 1987.*, pages 187–207, 1966.

[12] J. B. Kruskal. The number of simplices in a complex. *Mathematical Optimization Techniques (R. Bellman ed.), Univ. of California Press, Berkeley, Los Angeles*, pages 251–278, 1963.

[13] U. Leck. Nonexistence of a kruskal-katona type theorem for subword orders. *Combinatorica*, 24(2):305–312, 2004.

[14] C. S. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[15] E. Lo, B. Kao, W.-S. Ho, S. D. Lee, C. K. Chui, and D. W. Cheung. Olap on sequence data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 649–660, New York, NY, USA, 2008. ACM.

[16] F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD 98)*, pages 176–184, Nantes, France, 1998.

[17] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, and U. Dayal. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of 17th International Conference on Data Engineering (ICDE 01)*, pages 215–224, Heidelberg, Germany, 2001.

[18] J. Pei, J. Han, and W. Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *J. Intell. Inf. Syst.*, 28:133–160, April 2007.

[19] G. Ramesh, W. Maniatty, and M. J. Zaki. Feasible itemset distributions in data mining: theory and application. In *PODS*, pages 284–295. ACM, 2003.

[20] G. Ramesh, M. J. Zaki, and W. Maniatty. Distribution-based synthetic database generation techniques for itemset mining. In *IDEAS*, pages 307–316. IEEE Computer Society, 2005.

[21] 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.

[22] P. Tzvetkov, X. Yan, and J. Han. Tsp: Mining top-k closed sequential patterns. *Knowl. Inf. Syst.*, 7(4):438–457, 2005.

[23] H. S. Wilf. *Generatingfunctionology*. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[24] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large databases. In D. Barbará and C. Kamath, editors, *SDM*. SIAM, 2003.

[25] M. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

[26] M. J. Zaki, N. Parimi, N. De, F. Gao, B. Phoophakdee, J. Urban, V. Chaoji, M. A. Hasan, and S. Salem. Towards generic pattern mining. In B. Ganter and R. Godin, editors, *ICFCA*, volume 3403 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2005.

[27] B. Zhou, D. Jiang, J. Pei, and H. Li. Olap on search logs: an infrastructure supporting data-driven applications in search engines. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1395–1404, New York, NY, USA, 2009. ACM.