Mining Statistically Significant Sequential Patterns

Cécile Low-Kam
Montreal Heart Institute
Université de Montréal
Montréal, QC
clowkam@gmail.com

Chedy Raïssi
INRIA Nancy - Grand Est
Villers-lès-Nancy
F-54600, France
chedy.raissi@inria.fr

Mehdi Kaytoue
INSA-Lyon, CNRS
LIRIS UMR5205
F-69621, France
mehdi.kaytoue@insa-lyon.fr

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

***Abstract*—Recent developments in the frequent pattern mining framework uses additional measures of interest to reduce the set of discovered patterns. We introduce a rigorous and efficient approach to mine statistically significant, unexpected patterns in sequences of itemsets. The proposed methodology is based on a null model for sequences and on a multiple testing procedure to extract patterns of interest. Experiments on sequences of replays of a video game demonstrate the scalability and the efficiency of the method to discover unexpected game strategies.**

***Keywords*—Sequential pattern; null model; significance test;**

I. Introduction

Sequential pattern mining [1] is an important data mining task with many real applications. Most of the existing studies, such as [14], [23], [33], [6], [25], focused on efficient algorithms and effective pattern representations. In the existing work, absolute or relative frequency (also known as support) is used as the sole criterion in selecting frequent patterns. While frequency often serves as a good preliminary filter to remove noise patterns of very low popularity, in many applications, one has to find relevant patterns whose interestingness is defined in a statistical way, and cannot be specified using only a support threshold. A pattern of high frequency may not be interesting if it is statistically expectable from other patterns. At the same time, a pattern of low frequency may be interesting if it is statistically unexpected. Since a low support threshold often leads to a huge number of patterns, asking a user to select from patterns extracted using a low support threshold is overwhelming and impractical. This is a problem common not to only sequential patterns, but to frequent patterns in general.

To echo this challenge, several recent studies [20], [30], [3], [12] try to find patterns (i.e., itemsets or sequences) using some alternative interestingness measures or sampling representative patterns. A general idea, which is a framework of *finding unexpected patterns*, is to extract patterns whose characteristic on a given measure, such as the frequency, or more rarely the length [29], strongly deviates from its expected value under a null model. The frequency of a pattern is considered as a random variable, whose distribution under the null model has to be calculated or approximated. Then, the significance of the pattern is assessed through a statistical test that compares the expected frequency under the null model to the observed frequency. One of the key-points of this family of approaches is to *choose an appropriate null model*. It will ideally be a *trade-off* between adjustment to the data and simplicity: the model should capture some characteristics of the data, to integrate prior knowledge, without overfitting, to allow for relevant patterns discovery. A simple model, with low-order dependency, often results in faster computations and clear interpretation of the unexpected patterns.

Specifically, Gallo *et al.* [9], Hämäläinen and Nykänen [13], and Mampaey *et al.* [19] extracted itemsets of unexpected frequencies in non-sequential data. On sequence data, finding over-represented substrings in strings has been extensively studied in statistics, with interesting applications in biology, as these patterns were found to coincide with binding sites in DNA sequences [24]. Here, every element in a string is a simple item (no itemsets). When itemsets are allowed in sequence data, the order dimension introduces serious complexity in computing the expected frequency, since a pattern can have numerous and overlapping occurrences, and sequences can have different lengths. To the best of our knowledge, Gwadera and Crestani [12] gave the only approach that extends the unexpected pattern finding framework to sequence data, which ranks subsequences of itemsets with respect to their significance given a null model. Their model is a mixture model obtained by conditioning on the length of the sequence combined with a maximal entropy model. This approach, while theoretically sound, may become inefficient with a large set of items, and does not include a complete statistical testing procedure for global significance.

In this paper, we develop a rigorous and efficient framework to mine statistically significant, unexpected patterns in sequences of itemsets. We make several contributions.

First, we develop an approach that allows us to avoid considering overlapping occurrences or conditioning on the length of the sequence. This method, which is combinatorial in nature, permits the development of an elegant dynamic computation procedure based on the expected frequency of the prefix of each pattern, and therefore considerably speeds up the computations. Second, we investigate the theoretical properties of the expected frequency under the null model and relate them to a very well-known concise representation, *closed sequential patterns* [33], to obtain a more parsimonious and less redundant set of patterns. In addition, we provide and discuss multiple procedures to test for the significance of the extracted sequences combined with correction procedures to control the error rate at a global level. Last, we evaluate our approach on synthetic and real large-scale sequence data. We discuss in length the use of our algorithm SigSpan in an electronic-sport use case. We mine several large sequence bases of replays of Starcraft 2, a real-time strategy video game (Blizzard Entertainment). Our assessment shows an effective and scalable method. Helped by professional gamers, our framework is able to detect unexpected game strategies.

The rest of the paper is organized as follows. Section II reviews the related work. Section III briefly reviews the preliminaries needed in our development. Section IV introduces our null model. In Sections IV,V and VI, we discuss the computation of the expected support of a sequence, the significance tests and the algorithmic details. An experimental study is reported in Section VII. We conclude our work and discuss several interesting directions in Section VIII.

II. Related Work

A number of works have explored various notions of statistical *significance* for itemsets and have proposed novel and efficient methods for their extraction. In [13], an approach is developed to mine for statistically significant association rules, under an independence assumption between the condition and the consequence of the rule. The expected frequency for the condition and consequence is derived analytically, based on the marginal frequencies of the dataset. In [9], the authors use two null models to calculate the expected frequencies. The first is an independence model, where items are mutually independent and their probabilities of occurrence are their observed marginal relative frequencies. The second one is obtained by considering a distinct distribution for each transaction, where each of the items has the same transaction-dependent probability of occurrence. This second model consider large transactions as less interesting, because two such transactions are likely to share items by chance. As the expected frequency is anti-monotone (the expected frequency of an itemset is greater than the expected frequencies of its subsets), the authors only consider closed itemsets (itemsets having no superset with the same observed frequency). They then introduce the MINI algorithm to mine for non-redundant unexpected itemsets, by penalizing itemsets overlapping with itemsets with the highest p-values. An alternative to the independence model is the maximum entropy model used in [21]. A related framework is developed in [19], where unexpected itemsets given a maximum entropy model are iteratively used to update the model, resulting in a large coverage of the database and minimal redundancy between the itemsets.
In statistics, mining for over-represented patterns in sequences of items has been mainly developed for applications in biology, to find binding sites in DNA [24]. Because of the structure of DNA, where coding regions are usually made of consecutive bases, the patterns considered in these approaches are usually words (consecutive items), and the frequence considered is the total number of occurrences along the sequence. Under a Markov model, in [24] and [27], expected frequencies are computed given an exhaustive statistic for the model, in [22], by means of a convenient sequence associated to each word, and in [7], through generating functions. Note that [7] also considers words formed of items with variable-length gaps between them, the so-called *"hidden patterns"*. In [26], the objective is to find specific patterns called *structured motifs* of unexpected frequencies in a set of DNA sequences given a Markov model. Structured motifs are formed of two boxes of consecutive nucleotides separated by a gap of variable length, but bounded. The probability of occurrence for one such motif at a given position is approximated by considering the past up to a fixed order. [34] extend this last work by using the inclusion-exclusion principle, but only for structured motifs with a fixed length gap. In both approaches, the probabilities of occurrence are then compared to observed frequencies through a binomial test, as all sequences of the base are of the same length.

In the data mining community, a similar approach was developed by [11] to determine interesting frequent episodes in event sequences, i.e. subsequences with variable length gaps in sequences of items. Under an independence model, the authors use a sliding window to count all possible occurrences of an episode. Because of the dependency between overlapping occurrences, significance bounds for the frequency are set by using an asymptotic approximation for the variance and the Central Limit Theorem. In [16], the interest of subsequences is assessed by the difference between their observed and predicted frequencies, instead of the associated p-values. The most promising subsequences are then used to update a hidden Markov model through dynamic programming. Other solutions to reduce the set of frequent patterns to a smaller set of informative patterns include Minimum Description Length (MDL)-based approaches [18], [31]. Both approaches are conceived as a framework to summarize data.

To the best of our knowledge, the only approach that extends the null model based pattern extraction framework to sequences of itemsets is [12]. In this work, the null model is obtained by combining two models at different levels: itemset-wise and sequence-wise. The sequence-wise model is a mixture model obtained by conditioning on the length of the sequence, as a subsequence is more likely to occur in a longer sequence. The weights of the mixture are determined by the proportion of sequences of the same length in the dataset. For all the elements of the mixture, the authors use the same independence model as in their previous work [11], replacing items with itemsets. For the itemset-wise model, the authors choose a maximal entropy model, like in [21], as it makes mimimal assumptions about dependency between items, while setting the probability of an empty itemset to zero, unlike a simple independence model. The General Iterative Scaling Algorithm [4] can be used to calculate the expected frequencies of the itemsets under the maximum entropy model. The expected frequency of a sequential pattern is then obtained by combining the frequencies of its itemsets using the sequence-wise model. Finally, sequential patterns are ranked by a two-step algorithm which first searches for frequent patterns, then for each pattern, computes the expected frequency and uses the z-score for ranking unexpected patterns. As previously described in [19], the main difficulty of this approach is in the inference of an itemset's probability which is infeasible to do for a large number of items. In fact, the author of [28] shows that querying the maximum entropy model is **PP**-hard.

III. Preliminaries

Let $\mathcal{I} = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be a finite set of items. An *itemset* X is a non-empty subset of $\mathcal{I}$. A *sequence* s over $\mathcal{I}$ is an ordered list $\langle s_1, \ldots, s_\ell \rangle$, where $\ell > 0$, s_i is an itemset for $1 \leq i \leq \ell$. ℓ is called the *length* of the sequence s, denoted by $|s| = \ell$. Please note that the length of a sequence is the number of itemsets in it, instead of the number of item occurrences.

We denote by $\mathbb{T}(\mathcal{I})$ the set of all possible sequences over $\mathcal{I}$. A sequence database $\mathcal{D}$ of size n over $\mathcal{I}$ is a set of n sequences over $\mathcal{I}$. Note that the sequences in a database can

s_1	{a,b,c} {a,b} {b}
s_2	{a} {a,c} {a}
s_3	{a,b} {b,c}

TABLE I: A sequence database.

	X_1			X_2			X_3		
	X_{11}	X_{12}	X_{13}	X_{21}	X_{22}	X_{23}	X_{31}	X_{32}	X_{33}
s_1	1	1	1	1	1	0	0	1	0
s_2	1	0	0	1	0	1	1	0	0
s_3	1	1	0	0	1	1			

TABLE II: The Bernoulli binary representation of the sequence database in Table I.

have different lengths. We denote by $\ell_{\max}$ the length of the longest sequence of $\mathcal{D}$, that is, $\ell_{\max} = \max_{s\in\mathcal{D}}\{|s|\}$.

Definition 1 (Subsequence): A sequence $s = \langle s_1, \ldots, s_k\rangle$ is a **subsequence** of sequence $s' = \langle s'_1, \ldots, s'_{k'}\rangle$, denoted by $s \preceq s'$ if $k \leq k'$ and there exist $1 \leq r_1 < r_2 < \cdots < r_k \leq k'$ such that $s_j \subseteq s'_{r_j}$ for all $1 \leq j \leq k$. We also say that s is **contained** in s', s' **supports** s. We call s' a **supersequence** of s. The **greater prefix** of a sequence s of size k denotes the subsequence $\langle s_1 \ldots s_{k-1}\rangle$.

Definition 2 (Support and frequency): The **support** of a sequence s in a database $\mathcal{D}$, denoted by $Support(s, \mathcal{D})$ is the number of sequences in $\mathcal{D}$ that support s, that is,

$$Support(s, \mathcal{D}) = |\{s' \mid s' \in \mathcal{D}, s \preceq s'\}| \quad (1)$$

The **frequency** of s, denoted by $Freq(s, \mathcal{D}) = \frac{Support(s,\mathcal{D})}{n}$, is the relative support.

We often omit the sequence database $\mathcal{D}$ if it is clear from context.

Definition 3 (Sequential pattern mining): Given a minimal frequency threshold $0 < minFreq \leq 1$, the **problem of sequential pattern mining** is to find all sequences s such that $Support(s, \mathcal{D}) \geq minFreq$. Every sequence s such that $Support(s, \mathcal{D}) \geq minFreq$ is called a **sequential pattern**. We also say s is to be **frequent**.

Definition 4 (Frequent closed sequence): A sequential pattern s is a **frequent closed sequence** if there does not exist a proper supersequence s' such that $s \prec s'$ and $Support(s, \mathcal{D}) = Support(s', \mathcal{D})$. Frequent closed sequences directly provide the frequency of their subsequences and no additional processing is needed to extract this information. This concept of *closure* can be seen as a lossless compression of frequent patterns.

We illustrate definitions in the following running example.

Example 1: Consider the sequence database $\mathcal{D}$ in Table I on the alphabet $\mathcal{I} = \{a, b, c\}$. The size of $\mathcal{D}$ is $n = 3$. The longest sequence length is $\ell_{\max} = 3$. Sequence $\langle\{a, b\}\{b\}\rangle$ is contained in both s_1 and s_3, but not in s_2. Therefore, $Support(\langle\{a, b\}\{b\}\rangle) = 2$. The greater prefix of $\langle\{a, b\}\{b\}\rangle$ is $\langle\{a, b\}\rangle$. Given $minFreq = \frac{2}{3}$, $\langle\{a, b\}\{b\}\rangle$ is a sequential pattern. The set of frequent closed sequences is $\{\langle\{a\}\rangle, \langle\{c\}\rangle, \langle\{a\}\{c\}\rangle, \langle\{a, b\}\{b\}\rangle, \langle\{a, c\}\{a\}\rangle\}$.

The anti-monotonicity is an important property of sequential patterns.

Property 3.1 (Anti-monotonicity): For any sequences s and s' such that $s \preceq s'$, $Support(s) \geq Support(s')$. If s is infrequent, so is s'.

IV. A Null Model

Let $X = \langle X_1, X_2, \ldots\rangle$ be a stationary stochastic process of random vectors, where each vector

$$X_i = (X_{i1}, \ldots, X_{im}) \quad (2)$$

contains $m = |\mathcal{I}|$ random variables of Bernoulli distribution (one for each item). This model corresponds to the unexpected sequential pattern framework introduced and discussed in Section I. Then, each sequence $s \in \mathcal{D}$, written under a binary representation, is an independent realization x of X, of maximal length $\ell_{\max}$.

Example 2: Table II shows the Bernoulli binary representation of the database $\mathcal{D}$ in Table I. For example, for $i = 1, 2, 3$, variable X_{i1} indicates the presence or absence of item a in the corresponding itemset.

This binary representation is more convenient from a statistical perspective. The null model for sequences of itemsets is then naturally decomposed into a model for items in each itemset, and a model for itemsets in sequence.

We choose a global independence model. We assume that itemsets X_1, X_2, …, X_ℓ are independent and identically distributed (i.i.d.). Therefore,

$$P(X) \quad = \quad P(X_1)P(X_2)\cdots P(X_\ell) \quad (3)$$

We further assume that items are also independent. Thus,

$$P(X_i) \quad = \quad P(X_{i1})\cdots P(X_{im}) \quad (4)$$

for all $i = 1, \ldots, \ell$. The indicator random variables $X_{i1}, \ldots, X_{im}$ are independent Bernoulli distributed, of parameter $p_1, \ldots, p_m$, the marginal observed probabilities of each item in the dataset.

Using the independence models for itemsets is not new, Gallo *et al.* [10] and Gionis *et al.* [9] successfully apply it. Such an independence model allows useful interpretation as it can help identifying frequent co-occurrences of items that cannot be explained by their marginal frequencies. However, under independence assumptions, the null itemset has probability $(1-p_1) \times ... \times (1-p_m)$, a non-null quantity unless one of the items is present in each itemset. The maximum entropy (or log-linear) model [12] relaxes this constraint at the expense of much more costly computation. However, under the log-linear model, only one entry is removed from the sample space, with minimal effect on the remaining probabilities. Therefore, we choose the independance model which also respects the marginal frequencies of items and has a lower complexity.

V. Expected Frequency

Let $x = \langle x_1 \ldots x_k\rangle = \langle(x_{11}, \ldots, x_{1m}) \ldots (x_{k1}, \ldots, x_{km})\rangle$ be a sequential pattern of k itemsets, under the binary representation. For example, if $\mathcal{I} = \{a, b, c\}$, sequential pattern

X_1	X_2	X_3	X_4
x_1	x_2	.	.
x_1	x_2^c	x_2	.
x_1	x_2^c	x_2^c	x_2
x_1^c	x_1	x_2	.
x_1^c	x_1	x_2^c	x_2
x_1^c	x_1^c	x_1	x_2

TABLE III: Possible positions of itemsets of x in S.

$\langle\{a,b\}\{b\}\rangle$ can be rewritten as $\langle(1,1,0)(0,1,0)\rangle$. Denote by

$$\mu_{x_j} = P\left(X_{jr} = x_{jr}, r = 1,\ldots,m, x_{jr} = 1\right) \quad (5)$$

the probability of itemset x_j under the null model, for all $j \in \{1,\ldots,k\}$. For example, the event *"occurrence of itemset* $\{a,b\}$*"* means that either $\{a,b\}$ or $\{a,b,c\}$ was observed, and the associated probability is $\mu_{\{1,1,0\}} = P(X_{.1} = 1, X_{.2} = 1)$.

Example 3: Using the marginal frequencies of items in the dataset of Table II, under the independence model, we have $P(a) = \mu_{\{1,0,0\}} = P(X_{.1} = 1) = \frac{3}{4}$, as the item a appears in 6 itemsets out of 8, $P(b) = P(X_{.2} = 1) = \frac{5}{8}$ and $P(c) = P(X_{.3} = 1) = \frac{3}{8}$. Therefore $P(\{a,b,c\}) = \mu_{\{1,1,1\}} = \frac{45}{256}$.

The expected frequency of sequential pattern x in a single sequence, given the model defined above, is

$$p_\ell(x) = P\left(\exists\, X_{i_1},\ldots,X_{i_k}, 1 \le i_1 < \ldots < i_k \le \ell, x_1 \preceq X_{i_1}\ldots, x_k \preceq X_{i_k}\right), \quad (6)$$

where $x_j \preceq X_{i_j}$ if $X_{i_j r} = 1$ for all $r \in \{1,\ldots,m\}$ such that $x_{jr} = 1$. In other words, probability $p_\ell(x)$ is the probability that there exist k ordered itemsets in sequence X containing the itemsets of x. It depends on ℓ, since the longer the sequence, the greater the chance it supports x.

When the probability of all itemsets under the null model is known, computing the expected frequency $p_\ell(x)$ is a combinatorial problem, where we have to enumerate all possible occurrences of x in a sequence, while adjusting for multiple occurrences in a same sequence. We adopt a counting strategy where all mutually exclusive occurrences of x are listed according to the first occurrence of each of its itemsets.

To fix the ideas, we describe this enumeration for a sequential pattern $x = \langle x_1 x_2\rangle$ of length 2 and a random sequence $X = \langle X_1 \cdots X_4\rangle$ of length 4, such that X supports x. For each itemset x_j, we denote by x_j^c the set of all possible itemsets that do not contain x_j. Note that this set includes itemsets that contain some items of x_j but not all of them. We use a meta character "." to represent the set of all possible itemsets. Then X can be written as one and only one of the lines of Table III. The first possible configuration is that X_1 supports x_1 and X_2 supports x_2 (the first line of Table 2). Note that x_1 and x_2 may occur further in the sequence. The next possible configuration is that X_1 supports x_1, X_2 does not support x_2 but X_3 does (the second line). This case does not overlap with the previous one. The last possible case is that X_2 and X_3 do not support x_2, but X_4 does (the third line). A similar technique is used when the first occurrence of x_1 is in X_2 (the fourth and fifth lines) and in X_3 (the sixth line). Thus, all possible occurrences are counted, without redundancy.

This procedure is easily extended to the general case of a pattern of length k and a sequence of length ℓ, where $\ell \ge k$. First, note that if $\ell < k$, $p_\ell(x) = 0$. Then, for $1 \le j \le k$, let $q_{x_j}(i)$ be the probability that the first occurrence of itemset x_j is at the i^{th} position of the sequence. Then,

$$\begin{cases} q_{x_j}(1) = P\left(x_j \preceq X_1\right), \\ q_{x_j}(i) = P\left(x_j \preceq X_i, x_j \not\preceq X_{i'}, 1 \le i' < i\right), \\ \text{for } \ 2 \le i \le \ell. \end{cases} \quad (7)$$

Using the notation defined in Equation (5), we have

$$\begin{cases} q_{x_j}(i) = \left(1-\mu_{x_j}\right)^{i-1}\mu_{x_j}, \\ \text{for } \ 1 \le i \le \ell. \end{cases} \quad (8)$$

By stationarity and independence of the stochastic process, $q_{x_j}(i)$ is also the probability that x_j first occurs at the i^{th} position after x'_j, for any itemset x'_j. Note that for each itemset, this quantity can be computed by recurrence, since for $i \ge 2$,

$$q_{x_j}(i) = \left(1-\mu_{x_j}\right) q_{x_j}(i-1). \quad (9)$$

Example 4: From Table II, $q_{\{1,1,1\}}(1) = \mu_{\{1,1,1\}} = \frac{45}{256} \approx 0.18$, $q_{\{1,1,1\}}(2) = (1-\mu_{\{1,1,1\}})\mu_{\{1,1,1\}} = (1-\frac{45}{256}) \times \frac{45}{256} \approx 0.14$, and $q_{\{1,1,1\}}(3) = (1-\mu_{\{1,1,1\}})q_{\{1,1,1\}}(2) \approx 0.12$.

Listing all possible occurrences, the probability for x to occur in S is then

$$p_\ell(x) = \sum_{i_1 \in I_1, i_2 \in I_2, \ldots, i_k \in I_k} q_{x_1}(i_1) q_{x_2}(i_2) \ldots q_{x_k}(i_k), \quad (10)$$

where $I_j = \{1,2,\ldots,\ell + j - k - 1 - i_{j-1}\}$ is the set of all possible positions for itemset x_j. As first occurrences events are mutually exclusive, there are no compensating terms in this sum, unlike the inclusion-exclusion formula. This is the major advantage of the counting strategy presented above.

Example 5: The probability of occurrence of $\langle\{a,b,c\}\{a,b\}\rangle$ in a sequence of length $\ell = 3$ can be calculated as follows. First, we note that $\mu_{\{1,1,1\}} = \frac{45}{256} \approx 0.18$ and $\mu_{\{1,1,0\}} = \frac{15}{32} \approx 0.47$. Then $q_{\{1,1,1\}}(2) = (1 - \frac{45}{256}) \times \frac{45}{256} = \frac{9495}{65536} \approx 0.14$ and $q_{\{1,1,0\}}(2) = (1 - 15/32) \times \frac{15}{32} = \frac{255}{1024} \approx 0.25$. Therefore,

$$\begin{aligned} p_3(\{1,1,1\}\{1,1,0\}) &= q_{\{1,1,1\}}(1)q_{\{1,1,0\}}(1) \\ &\quad + q_{\{1,1,1\}}(1)q_{\{1,1,0\}}(2) \\ &\quad + q_{\{1,1,1\}}(2)q_{\{1,1,0\}}(1) \\ &\approx 0.18 \times 0.47 + 0.18 \times 0.25 \\ &\quad + 0.14 \times 0.47 \approx 0.20. \end{aligned} \quad (11)$$

The probability of occurrence $p_\ell(x)$ of each pattern x under the independence model can be calculated explicitly using Equation (10).

In the next section, we describe a dynamic strategy that calculates the support of any pattern given the support of its greater prefix, to reduce computational cost.

VI. SigSpan: An Algorithm for Unexpected Sequential Patterns Extraction

In this section, we propose an algorithm that integrates dynamic expected support computations and comparisons to observed support to extract patterns of unexpected frequency.

A. Dynamic Programming

For $x = \langle x_1 \dots x_k \rangle$, let $Q_x(i)$ be the probability that the first occurrence of x ends at the i^{th} position of the sequence. Let Q_x be the vector of length $\ell - k + 1$ such that $Q_x = (Q_x(i))_{|x| \leq i \leq \ell}$. Then,

$$p_\ell(x) \quad = \quad \sum_{i=|x|}^{\ell} Q_x(i), \tag{12}$$

by reordering the terms of sum (10).

Example 6: The probability that the first occurrence of $\langle \{a,b,c\}\{a,b\} \rangle$ ends at position 2 is

$$Q_{\{1,1,1\}\{1,1,0\}}(2) \quad = \quad q_{\{1,1,1\}}(1) q_{\{1,1,0\}}(1). \tag{13}$$

The probability that it ends at position 3 is

$$\begin{aligned} Q_{\{1,1,1\}\{1,1,0\}}(3) \quad &= \quad q_{\{1,1,1\}}(1) q_{\{1,1,0\}}(2) \\ &\quad + q_{\{1,1,1\}}(2) q_{\{1,1,0\}}(1). \end{aligned} \tag{14}$$

Summing these two quantities gives (11)

We now turn to the computation of $Q_x(i)$, for all $1 \leq i \leq \ell - k + 1$:

- if $k = 1$: then x is reduced to a single itemset and $Q_x = (q_x(i))_{1 \leq i \leq \ell}$ is obtained as in (8).
- if $1 < k \leq \ell$: then if x_- denotes $\langle x_1 ... x_{k-1} \rangle$, the largest prefix of x, then for $k \leq i \leq \ell$,

$$Q_x(i) \quad = \quad \sum_{j=|x|-1}^{i-1} Q_{x_-}(j) q_{x_k}(i-j), \tag{15}$$

which allows for using previous calculations for greater prefix x_-. Therefore, the expected support computation framework is easily integrated in the prefix tree structure for frequent patterns extraction of Section III.

Probability (15) is calculated as follows: let

$$M_{x_-} = \begin{bmatrix} Q_{x_-}(|x|-1) & Q_{x_-}(|x|) & \cdots & Q_{x_-}(\ell-1) \\ 0 & Q_{x_-}(|x|-1) & \cdots & Q_{x_-}(\ell-2) \\ 0 & 0 & \cdots & Q_{x_-}(\ell-3) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{x_-}(|x|-1) \end{bmatrix}. \tag{16}$$

Then, Q_x is obtained as the matrix product of this matrix and the vector

$$(q_{x_k}((\ell - |x| + 1)), ..., q_{x_k}(2), q_{x_k}(1))^T \tag{17}$$

Given the support of its greater suffix, the calculation of the expected support of x is reduced to its two last itemsets' positions enumeration, resulting in a complexity in $O(\ell^2)$. Replacing ℓ with the maximal length $\ell_{\max}$, and noting that $p_\ell(x)$ is obtained as a partial sum on the terms of $p_{\ell_{\max}}(x)$ for all $\ell \leq \ell_{\max}$, the final complexity of computing $p_\ell(x)$ for all $\ell \in \{1, ..., \ell_{\max}\}$ is $O(\ell_{\max}^2)$ for each x.

The expected frequencies are then compared to observed frequencies to extract unexpected sequential patterns.

B. Statistically Significant Patterns

For a sequential pattern x, and $i = 1, ..., n$, let Z_x^i be the Bernoulli random variable such that:

$$\begin{cases} Z_x^i = 1 & \text{if the } i^{th} \text{ sequence supports } x, \\ Z_x^i = 0 & \text{if not.} \end{cases} \tag{18}$$

If all sequences have the same length ℓ, $Z_x^i = 1$ with the probability $p_\ell(x)$ for all i and $Support(x)$ is the sum of these i.i.d. random variables. The expected support of x under the null model is $n \times p_\ell(x) = \mathbb{E}(Support(x))$, and $Support(x)$ has a binomial distribution.

In the general case, however, sequences of $\mathcal{D}$ have different lengths and $Support(x)$ is a sum of independent, non identically distributed variables of Bernoulli distribution $B(p_{\ell_i}(x))$, and of expectation $\mathbb{E}(Support(x)) = \sum_{i=1}^{n} p_{\ell_i}(x)$. The exact distribution of $Support(x)$ can be calculated with a simple procedure of quadratic complexity. Instead of the exact distribution, we use Hoeffding's concentration inequality ([15]) for sums of bounded, independent random variables:

$$\begin{aligned} &P\{Support(x) \geq Support_{obs}(x)\} \\ &\leq \exp(-\frac{2}{n}(Support_{obs}(x) - \mathbb{E}(Support(x)))^2), \end{aligned} \tag{19}$$

This gives an upper bound for the p-value associated to the sequential pattern x. The drawback of this method is that the bound is not tight for small-variance random variables. If we fix a critical threshold ε, if

$$Support_{obs}(x) - \mathbb{E}(Support(x)) \geq \sqrt{-\frac{n}{2} \log \varepsilon}, \tag{20}$$

then x is statistically over-represented at level ε. For example, a data analyst may consider that sequences are significant if $Support_{obs}(x) - \mathbb{E}(Support(x)) \geq 1000$ in a data set of size $100,000$. Then the analyst needs to choose ε such that $\varepsilon \geq \frac{1}{e^{20}}$.

As all frequent patterns are tested simultaneously, we apply a correction procedure for multiple testing. The Bonferroni correction controls the family-wise error rate by performing individual tests with a threshold of ε/N, where N is the number of tests. As Bonferroni correction is known to be very conservative, more recent and powerful methodologies propose to control *the false discovery rate* (FDR), *i.e.* the expected proportion of false positives. The Benjamini-Hochberg-Yekutieli procedure [2] allows for controlling the FDR at level ε for dependent tests by finding the largest q such that

$$\alpha_{(q)} \quad \leq \quad \frac{q}{N \sum_{q'=1}^{N} \frac{1}{q'}} \; \varepsilon, \tag{21}$$

where $\alpha_{(1)}, ..., \alpha_{(N)}$ are the N ordered p-values for all frequent patterns. If we use Hoeffding's approximation, we have a sequence of bounds $\beta_{(1)}, ..., \beta_{(N)}$ from (19) such that $\alpha_{(q)} \leq \beta_{(q)}$ for all $q = 1, ..., N$, and replacing $\alpha_{(q)}$ with $\beta_{(q)}$ in (21) still allows for controlling the FDR. As N is only known once all frequent patterns are discovered, both corrections are applied a posteriori.

Using the difference between the observed support and the expected one has a bias due to the scale of the observed

support. To overcome the issue, an alternative measure is to use the deviation in percentage, that is,

$$\frac{Support_{obs}(x) - \mathbb{E}(Support(x))}{\mathbb{E}(Support(x))}. \quad (22)$$

Algorithm 1: SigSpan

Data: A database DB, a minimal support $minSupp$ and a threshold ε
Result: $\mathcal{SF}$, a collection of unexpected sequential patterns

$\mathcal{SF} \leftarrow \emptyset$;
Compute probabilities of first occurrence $q_{x_k}(i)$;
/* mine frequent closed sequences */
$\mathcal{CF} \leftarrow ClosSpan(DB, minSupp)$;
/* test if the closed sequences are significantly over-represented */
for $\forall s \in \mathcal{CF}$ **do**
  /* Using dynamic programming */
  $es := ComputeExpectedSupport(s)$
  if *s is significant* **then**
    /* significance with Bonferroni test: $Support(s) - es \geq \sqrt{-\frac{|DB|}{2} \log \frac{\varepsilon}{|\mathcal{CF}|}}$, or FDR tests (see Equation 21) */
    $\mathcal{SF} \leftarrow \mathcal{SF} \cup s$
  end
end
return $\mathcal{SF}$

C. Closed Unexpected Sequential Patterns

Most of the algorithms for discovering sequential patterns use the antimonotonicity property of the frequence described in Section III. But the measure of interest defined in (20) is not anti-monotone: a supersequence of a non over-represented sequential pattern can be over-represented. However,

Lemma 6.1: The expected support is antimonotone.

Proof: For all $\ell \geq 2$, let x be a sequential pattern of length greater than 2, and x_- its greater prefix. We have defined a probabilistic framework, where the event "x occurs in sequence S" is included in the event "x_- occurs in S", therefore, $p_\ell(x) \leq p_\ell(x_-)$. ■

This property means that we can use the subset of closed unexpected sequential patterns as a summary of all the unexpected sequential patterns, without losing information: let us suppose that for a sequential pattern x,

$$Support_{obs}(x_-) = Support_{obs}(x), \quad (23)$$

where x_- is the greater prefix of x. As we are interested in over-represented sequential patterns, we also suppose that

$$\begin{aligned} Support_{obs}(x) &> \mathbb{E}(Support(x_-)) \\ &\geq \mathbb{E}(Support(x)). \end{aligned} \quad (24)$$

Then,

$$\begin{aligned} &Support_{obs}(x_-) - \mathbb{E}(Support(x_-)) \\ &\leq Support_{obs}(x) - \mathbb{E}(Support(x)). \end{aligned} \quad (25)$$

Therefore, we can restrict the search to the set of closed sequential patterns, as they are always more interesting in the sense of (20) than their subsequences.

Algorithm 1 lists the different steps of SigSpan in order to extract unexpected sequential patterns, given a minimum support $minSupp$, and a threshold ε for the p-values. The algorithm starts by computing the probabilities of first occurrence q_{x_k} then it calls the CloSpan algorithm to mine frequent closed sequences. Then, SigSpan does a depth-first traversal of the prefix tree of closed sequential patterns. For each pattern s, its support according to the model is computed using the support of its greater prefix. Therefore, we need to store the vector Q_{s_-} of the greater suffix of s. However, these vectors are discarded as we progress further on the tree. The different tests discussed in the previous section are evaluated for each closed sequence and if a sequence is significantly over-represented, it is stored in $\mathcal{SF}$ and outputted at the end.

VII. Experiments

In this section, we report an extensive empirical evaluation of our algorithm SigSpan using real and synthetic datasets. All experiments are performed on a 1.8 GHz Intel Core i5 with 8 GB main memory running Mac OS X 10.7.5. We started from the original C++ version of CloSpan to implement the algorithm SigSpan (compiled with g++ and -03 optimization). All the mathematical computations were processed using the uBlas Library[1]. The source code and the data sets are available at http://www.loria.fr/~raissi.

A. Synthetic Dataset

Finding injected patterns hidden in a generated dataset.

In order to assess the significance of our results, we check how SigSpan recognizes the unexpected sequence we introduced in a synthetic dataset on an alphabet of 1000 items. In this synthetic data set the items' probabilities distributions are generated following a Beta distribution and the length of the sequences follows a Poisson distribution to allow a maximal of flexibility in the datasets generation process[2]. We then introduce $\langle\{a,b,c\}\{a,b,c\}\rangle$ in 60% of the sequences. The independence assumption of the null model means that it should account for the high supports of items a, b, and c, but not for their co-occurrence in the pattern $\langle\{a,b,c\}\{a,b,c\}\rangle$. For each threshold, **true positives** are the number of sequences including $\langle\{a,b,c\}\{a,b,c\}\rangle$ that are considered unexpected by SigSpan, **false positives** are other unexpected patterns, **false negatives** are patterns including $\langle\{a,b,c\}\{a,b,c\}\rangle$ which are not outlined by the algorithm, and **true negatives** are the rest of the patterns. We draw a ROC curve on Figure 1 for different minimal frequency thresholds, with critical thresholds ε ranging from 10^{-100} to 10^{-1}, using our algorithm. This

[1] http://www.boost.org/libs/numeric/

[2] Due to space limitations, only the experiments with $|D| = 100000, \alpha = 0.001$, $\beta = 2$ and $\lambda = 7$ are reported. The data generator along with some other datasets is provided with the SigSpan downloadable implementation.

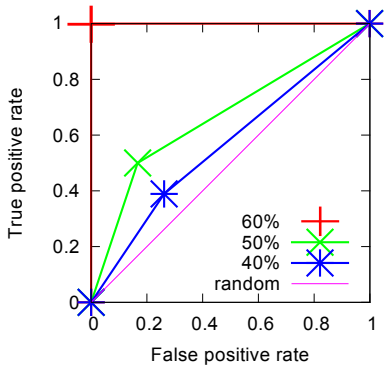


Fig. 1: ROC curve

Data	$\|\mathcal{D}\|$	$\|I\|$	l_{max}	l_{avg}	$max(\|s_i\|)$	$avg(\|s_i\|)$
PvP	6,668	1,161	57	11,47	17	2,13
PvT	18,754	3,656	70	19,03	23	2,48
PvZ	22,784	3,749	94	19,59	28	2,66
TvT	7,457	2,202	67	20,73	18	2,56
TvZ	23,638	4,493	75	22,52	29	2,55
ZvZ	9,554	1,690	60	14,18	16	2,08

TABLE IV: Real world datasets

curves represent the fraction of true positives with respect to the fraction of false positives and provide a measure for precision and recall. The best possible prediction would be the point (0, 1) corresponding to no false positives and no false negatives. Figure 1 shows that the performance of the algorithm depends on the minimal frequency threshold, but not on ε, as all points overlap for all curves. All three curves are above the random guess line, thus demonstrating the accuracy of the algorithm. As expected, the curve for $minFreq = 60\%$ performs better.

B. The StarCraft II Use Case

Context Opponents modeling aims at simulating player behaviors. This study domain started in the seventies and focused on chess strategies [8] before shifting in recent years to video games, one of the most lucrative leisure activities. Models of players behaviors are used within automated agents, to personalize a game environment for a specific user, to predict future actions of an opponent, etc. To tackle some of these problems, an important sub-problem is the automatic discovery of strategic patterns in a competitive video game environment [5]. We study one of the most competitive real-time strategy games (RTS) [17], StarCraft II (Blizzard Entertainment, 2010) which has its own world-wide players ranking system (ELO) and annual world cup competition series (WCS) with a US$1.6 millions prize pool for the year 2013 (among others premier tournaments).

A game of StarCraft II involves two players. Each player chooses a faction among Zerg (Z), Protoss (P) and Terran (T). As such, there are 6 different possible match-ups with different strategies of game. During a game, two players are battling on a map (aerial view), controlling buildings and units to gather supply, build an army with the final goal of winning by destroying the opponent's forces[3]. Such actions (training, building, moving, attacking) are done in real-time, see Figure 2 for an example. Each faction (Z,P,T) allows different units and buildings with distinctive weaknesses and strengths following a rock-paper-scissors principle. A strategy is hidden in large sequences of actions generated by players and called *replays*. Our goal is to mine significantly over-represented patterns to detect surprising strategies in this game.

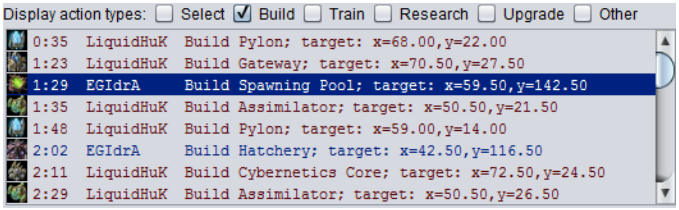


Fig. 2: An example of build-actions realized during a game.

Processing raw data We collected more that 300,000 replays from specialized repositories, and filtered them to keep the $90,678$ replays from professional players. We focus only on *buildings order* sequences (such as in Figure 2), the pillar elements of a strategy on which unit production type and attack timings depend [32]. All raw replays are partitioned into 6 datasets, one for each match-up. For each dataset, we derive one sequence for each replay: a sequence is seen as an ordered list of itemsets of buildings from B, where an itemset denotes a window of time of 30 seconds (the ordering of two items over a small period of time is insignificant in terms of strategy, hence the use of itemsets). Furthermore, both players actions are mixed in a sequence. Indeed, actions of both players are strongly correlated. We choose to distinguish both players within a sequence by adding the winner information. As such, an itemset of a sequence is a set of $(B \times \{winner, looser\})$. Finally, since timings are very important in a strategy, we encode the window's identifier for each item. As such, an itemset is a set of $\mathcal{I} = \mathbb{N} \times B \times \{winner, looser\}$. The first three itemsets built from the sample replay in Figure 2 are $\langle\{(1, pylon, looser)\}, \{(2, gateway, looser), (2, spanningpool, winner)\}, \{(3, assimilator, looser), (3, pylon, looser)\}\}\rangle$. Table IV summarizes the main characteristics of the different datasets. For example, the dataset labeled "PvT" encodes all sequences played by a Protoss player against a Terran player. The resulting datasets are described in Table IV. The sequences length distribution is given by Figure 3.

C. Quantitative Experiments

Performance and scalability Figures 5 and 4(a) give the execution times of SigSpan with different minimum supports. The closed pattern extraction time is separated from the statistical computation time. In general, we observed that the statistical model computation is moderate except for extremely low supports ($\leq 1\%$). The scalability tests reported by Figure 7 show that the total execution time grows linearly with the size of the data set when replicated (i.e. the same set of closed or

[3] http://en.wikipedia.org/wiki/Real-time˙strategy

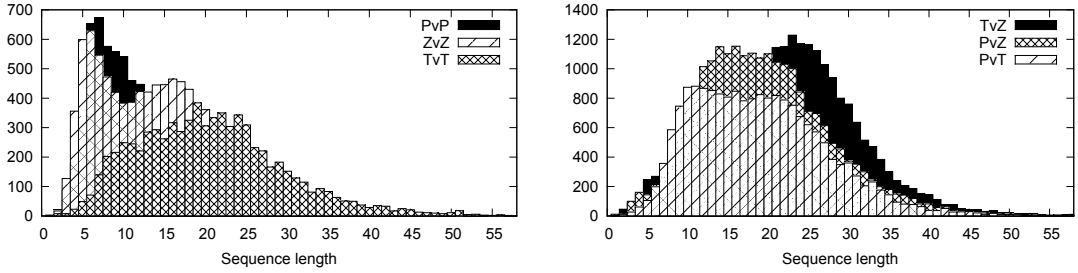


Fig. 3: Sequence length distribution of the StarCraft II datasets (sequence count in Y-axis).

significant patterns). For example, Figure 7(f) shows that the 200 times replicated ZvZ data set (2 millions of sequences) can be mined in less than 500 seconds with $minSupp = 0.5\%$.

Closed vs. significant counts when fixing ε and varying minimal frequency For each dataset, we set ϵ so that $Support_{obs}(x) - \mathbb{E}(Support(x)) \geq 500$ in equation (20). Figures 6 and 4 (c,d,e,f) reports the number of closed patterns and of significant closed patterns with several minimum supports. As expected, the number of closed patterns grows exponentially. Interestingly the number of closed significant patterns is several order of magnitude lower with low supports, and stays approximately constant. This is a very important remark as this collection of patterns can still be *humanly* processed and analyzed.

Closed vs. significant counts when fixing minimal frequency and varying ε. We now re-iterate the same experiment, this time fixing the frequency for our datasets[4]. Figure 8 reports these experiments. The number of frequent closed sequences remains constant. Both the number of unexpected sequences and ε decrease exponentially together.

D. Qualitative Interpretation

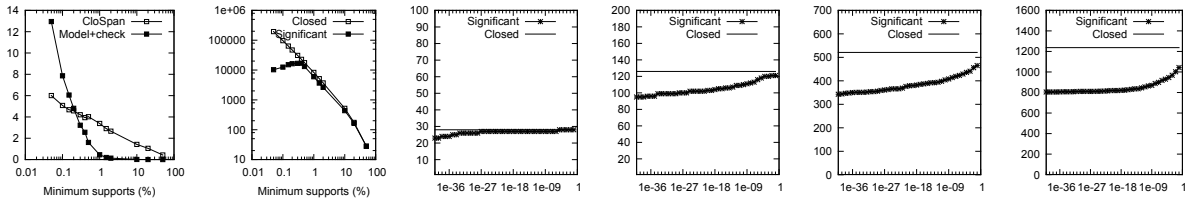
To assess the quality of the extracted patterns representing players tactics, we contacted a game expert. Our qualitative experiment is two-fold. Firstly, we show that classic and well-known strategy openings in StarCraft II are expected patterns. Second, we show that some risky strategies are unexpected with high potentials for professional players.

Known Strategies, Expected Sequences Like in chess, Starcraft II openings are generally well-known and codified (given a name). We would like to observe if known openings in StarCraft II are translated as expected sequences (i.e., least unexpected sequences). We started with the TvZ dataset and extracted all closed patterns with minimum support higher than 0.4%, i.e. about $700,000$ patterns. We sorted them according to the ratio given by Equation (22): the closer the ratio to zero, the more expected the pattern and finally selected the top-100 expected patterns. According to the expert, most of these patterns (with a ratio lower than 0.5) are either known openings or standard actions the players need to do to develop normally their economy (i.e. non significant). The first pattern is $\langle\{(5, winner, Barracks)\}, \{(7, winner, Barracks)\}\rangle$ is a classical opening where the player favors a special kind of unit composition that appears only in the *"Barracks"* building. Another known opening is translated and found in the pattern $\langle\{(6, looser, ComCenter)\}, \{(7, looser, ComCenter)\}\rangle$ (ranked 22). This strategy is called *"double expand"*. Consider now another scenario: the Terran (T) player has build a bunker (a defensive building). What is he expected to do next? We kept the $30,414$ patterns containing at least two items among one involving the bunker building. The most expected pattern is the following $\langle\{(6, winner, Bunker)\}, \{(7, winner, CommandCenter)\}\rangle$ with an observed support of 113, and expected support of 67 (out of $22,784$ sequences in the original dataset). For the expert, this pattern clearly indicates either a so-called *defensive-expand strategy* in the early stage of the game, or an *enemy rush* (an all-in strategy that is deadly if not countered, equivalent to a blitz attack in chess on the f7 or f2 square early in the game, putting the king in check[5]). In those examples, elements of the patterns are expected to occur together.

Surprising Strategies, Unexpected Sequences The so-called *DT-rush* is a risky Protoss gambling strategy that leads often to a win of the game when the opponent does not detected it (although it requires a strong technology coupled with long minutes of vulnerability). It is generally well detected and expected by the opponent given the information she gets by scouting the map. However, in the PvZ matchup, Protoss tends to use it, since Zerg needs some time to react to this piece of information. We mined the PvZ dataset accordingly with a minimum support of 0.04%. We kept only the patterns involving the building unlocking these invisible units (DT's). On 14 remaining patterns, 12 patterns are of size 1 (only one itemset), and involve the target building in late state of the game with expected values almost identical to the observed values. This is not surprising since all technologies are generally unlocked in mid/end game. The 2 last patterns however, involve the target building in the early stage of the game leading to a victory. Both patterns $\langle\{(6, winner, Canon)\}, \{(11, winner, DarkS)\}\rangle$ and $\langle\{(11, winner, Concil)\}, \{(12, winner, DarkS)\}\rangle$ have an observed support of 94 while an expected support of 3 and 17. They however do not correspond to known timings of the *DT rush* (which occurs earlier). These patterns highlight an unexpected strategy that the expert calls "delayed DT rush".

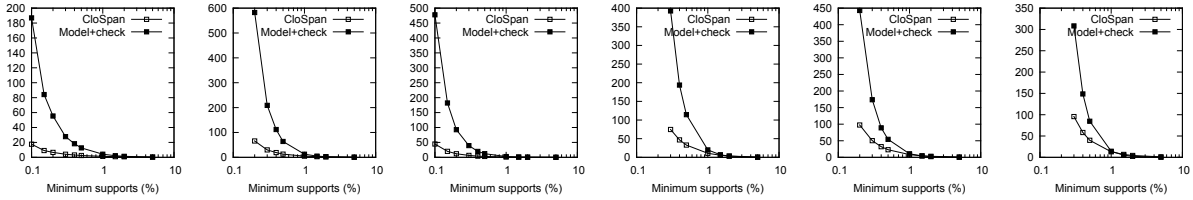
[4]Due to space limitations, only ZvZ dataset results are reported.

[5]http://en.wikipedia.org/wiki/Fast_chess



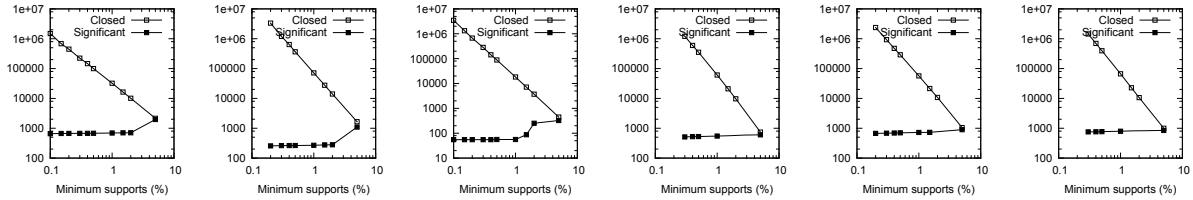
(a) Performance (b) Count (c) 50% (d) 25% (e) 10% (f) 5%

Fig. 4: Synthetic dataset



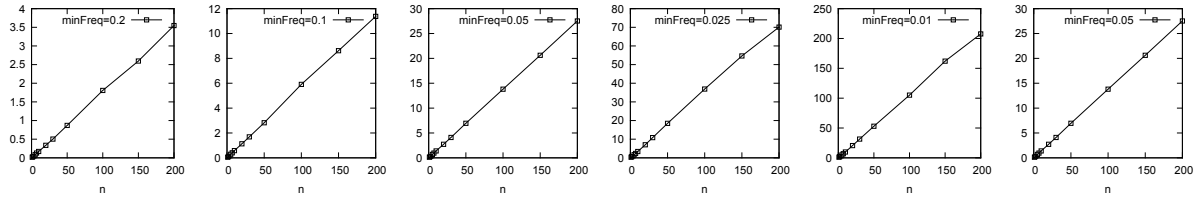
(a) PvP (b) TvT (c) ZvZ (d) PvZ (e) PvT (f) TvZ

Fig. 5: Performance (Y-axis in seconds)



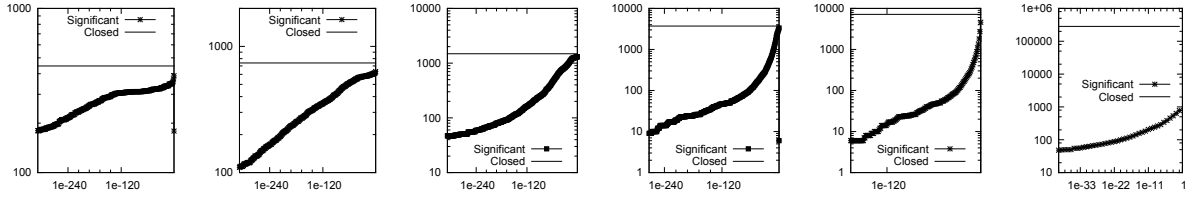
(a) PvP (b) TvT (c) ZvZ (d) PvZ (e) PvT (f) TvZ

Fig. 6: Pattern count



(a) 20% (b) 10% (c) 5% (d) 2.5% (e) 1% (f) 0.05%

Fig. 7: Scalability test with dataset ZvZ replicated n times with different minimum support (Y-axis in seconds)



(a) 5% (b) 4% (c) 3% (d) 2% (e) 1.5% (f) 0.3%

Fig. 8: Closed vs. significant pattern counts for different minimum support (subfigures) and ε parameters (X-axis).

We contrasted this result with the PvT dataset: Terran is the only faction that has a counter to the *DT-rush* early in the game, with almost no technology required. For a minimum support of 0.04%, the one and only pattern that involves the target building is $\langle\{9, looser, DarkS\}\rangle$ with an observed support of 95 and expected support of 94.7. This corroborates the fact given by the expert stating that *DT-rush* is not a viable strategy in the PvT matchup, but could be sometimes a surprising winning strategy in PvZ when delayed.

VIII. Conclusion

In this paper, we developed a new approach for extracting statistically significant sequential patterns. The proposed methodology is based on a null model for sequences of data and a combinatorial enumeration process of possible positions for the itemset of each subsequence. A statistical test is performed to determine over-represented sequential patterns, and an upper bound on the associated p-value is provided for greater efficiency. This probabilistic framework is seamlessly integrated to the tree structure of traditional frequent patterns mining algorithms. Correction procedures account for multiplicity of tests. Experimental results show the relevance of discovered patterns.

Acknowledgments. The third author was partially supported by the MI CNRS Mastodons program.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[2] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29:1165–1188, 2001.

[3] M. Boley, T. Horváth, and S. Wrobel. Efficient discovery of interesting patterns based on strong closedness. *Statistical Analysis and Data Mining*, 2(5-6):346–360, 2009.

[4] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. In *The Annals of Mathematical Statistics*, volume 43, pages 1470–1480, 1972.

[5] E. Dereszynski, J. Hostetler, A. Fern, T. Dietterich, T.-T. Hoang, and M. Udarbe. Learning probabilistic behavior models in real-time strategy games. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[6] B. Ding, D. Lo, J. Han, and S.-C. Khoo. Efficient mining of closed repetitive gapped subsequences from a sequence database. In Y. E. Ioannidis, D. L. Lee, and R. T. Ng, editors, *ICDE*, pages 1024–1035. IEEE, 2009.

[7] P. Flajolet, W. Szpankowski, and B. Vallée. Hidden word statistics. *J. ACM*, 53(1):147–183, 2006.

[8] J. Fürnkranz and M. Kubat. Machine learning in games: A survey. *Machines that learn to Play Games*, pages 11–59, 2001.

[9] A. Gallo, T. D. Bie, and N. Cristianini. Mini: Mining informative non-redundant itemsets. In *PKDD*, pages 438–445, 2007.

[10] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. In *KDD 2006*, pages 167–176, 2006.

[11] R. Gwadera, M. J. Atallah, and W. Szpankowski. Reliable detection of episodes in event sequences. *Knowl. Inf. Syst.*, 7(4):415–437, 2005.

[12] R. Gwadera and F. Crestani. Ranking sequential patterns with respect to significance. In *PAKDD (1)*, pages 286–299, 2010.

[13] W. Hämäläinen and M. Nykänen. Efficient discovery of statistically significant association rules. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 203–212, Washington, DC, USA, 2008. IEEE Computer Society.

[14] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In R. Ramakrishnan, S. J. Stolfo, R. J. Bayardo, and I. Parsa, editors, *KDD*, pages 355–359. ACM, 2000.

[15] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[16] S. Jaroszewicz. Interactive HMM construction based on interesting sequences. In *Proc. of Local Patterns to Global Models (LeGo'08) Workshop at the 12th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'08)*, pages 82–91, Antwerp, Belgium, 2008.

[17] M. Kaytoue, A. Silva, L. Cerf, W. Meira Jr., and C. Raïssi. Watch me playing, i am a professional: a first study on video game live streaming. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *WWW 2012 (Companion Volume)*, pages 1181–1188. ACM, 2012.

[18] H. T. Lam, F. Moerchen, D. Fradkin, and T. Calders. Mining compressing sequential patterns. In *SDM*, pages 319–330. SIAM / Omnipress, 2012.

[19] M. Mampaey, N. Tatti, and J. Vreeken. Tell me what i need to know: succinctly summarizing data with itemsets. In *KDD*, pages 573–581, 2011.

[20] M. Mampaey, J. Vreeken, and N. Tatti. Summarizing data succinctly with the most informative itemsets. *TKDD*, 6(4):16, 2012.

[21] H. Mannila and H. Toivonen. Multiple uses of frequent sets and condensed representations. In *In Proc. KDD Int. Conf. Knowledge Discovery in Databases*, pages 189–194. AAAI Press, 1996.

[22] G. Nuel. Pattern markov chains: optimal markov chain embedding through deterministic finite automata. *Journal of Applied Probability*, 1:226–243, 2008.

[23] J. Pei, J. Han, M. B. Asl, H. Pinto, Q. Chen, U. Dayal, and M. C. Hsu. Prefixspan mining sequential patterns efficiently by prefix projected pattern growth. In *Proc.17th Int'l Conf. on Data Eng.*, pages 215–226, 2001.

[24] B. Prum, F. Rodolphe, and . de Turckheim. Finding words with unexpected frequencies in DNA sequences. *J. R. Statist. Soc. B*, 57:205–220, 1995.

[25] C. Raïssi, T. Calders, and P. Poncelet. Mining conjunctive sequential patterns. *Data Min. Knowl. Discov.*, 17(1):77–93, 2008.

[26] S. Robin, J.-J. Daudin, H. Richard, M.-F. Sagot, and S. Schbath. Occurrence probability of structured motifs in random sequences. *Journal of Computational Biology*, 9(6):761–774, 2002.

[27] S. Schbath. An efficient statistic to detect over- and under-represented words in DNA sequences. *Journal of Computational Biology*, 4:189–192., 1997.

[28] N. Tatti. Computational complexity of queries based on itemsets. *Inf. Process. Lett.*, 98(5):183–187, 2006.

[29] N. Tatti. Significance of episodes based on minimal windows. In *ICDM*, pages 513–522. IEEE Computer Society, 2009.

[30] N. Tatti and M. Mampaey. Using background knowledge to rank itemsets. *Data Min. Knowl. Discov.*, 21(2):293–309, 2010.

[31] N. Tatti and J. Vreeken. The long and the short of it: summarising event sequences with serial episodes. In Q. Yang, D. Agarwal, and J. Pei, editors, *KDD*, pages 462–470. ACM, 2012.

[32] B. G. Weber and M. Mateas. A data mining approach to strategy prediction. In P. L. Lanzi, editor, *In Proc. of the 2009 IEEE Symposium on Computational Intelligence and Games,Milano, Italy, 7-10 September*, pages 140–147. IEEE, 2009.

[33] 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.

[34] J. Zhang, B. Jiang, M. Li, J. Tromp, X. Zhang, and M. Q. Zhang. Computing exact P-values for DNA motifs. *Bioinformatics*, 23(5):531–537, 2007.