DETECTION OF SETS OF EPISODES IN EVENT SEQUENCES: ALGORITHMS, ANALYSIS AND EXPERIMENTS

by Robert Gwadera, Mikhail Atallah, Wojciech Szpankowski

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Detection of Sets of Episodes in Event Sequences: Algorithms, Analysis and Experiments

Mikhail Atallah[*]
Department of Computer Sciences
Purdue University
West Lafayette, Indiana
mja@cs.purdue.edu

Robert Gwadera[†]
Department of Computer Sciences
Purdue University
West Lafayette, Indiana
gwadera@cs.purdue.edu

Wojciech Szpankowski[‡]
Department of Computer Sciences
Purdue University
West Lafayette, Indiana
spa@cs.purdue.edu

ABSTRACT

In our previous paper [12] the problem of the reliable detection of an "abnormal" episode was investigated, where by episode we mean a particular ordered sequence occurring as a subsequence of a large event stream *within a window of size w*. Here we extend this work to the more difficult case of many pattern sequences, including the important special case of all permutations of the same sequence, which models the situation where the order of events in an episode does not matter, e.g., when it is unimportant whether buying suspicious item X occurs before or after buying suspicious item Y. The difficulties of carrying out a probabilistic analysis for an arbitrary set of patterns, stem from the possible overlap of the respective windows within which the patterns occur as subsequences, the fact that the different patterns typically do have common symbols or common subsequences or possibly common prefixes, and that they may have different lengths. We also report of extensive experimental results, carried out on the Wal-Mart transactions database, that show a remarkable agreement with our theoretical analysis [1].

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data mining*

General Terms

Algorithms

[*]Portions of this work were supported by Grants EIA-9903545, IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

[†]The work of this author was supported by the NSF Grant CCR-0208709

[‡]The work of this author was supported by the NSF Grant CCR-0208709 and NIH R01 GM068959-01.

[1]A full version of this paper is available as `http://www.cs.purdue.edu/homes/spa/papers/`

1. INTRODUCTION

Detecting subsequence patterns in event sequences is important in many applications, including intrusion detection, monitoring for suspicious activities, and molecular biology (e.g., see [1, 7, 11, 14]). Whether an observed pattern of activity is significant or not (i.e., whether it should be a cause for alarm) depends on how likely it is to occur fortuitously. A long enough sequence of observed events will almost certainly contain any subsequence, and setting thresholds for alarm is an important issue in a monitoring system that seeks to avoid false alarms.

The basic question is then: *when is a certain number of occurrences of a particular subsequence unlikely to be generated by randomness (i.e., indicative of suspicious activity)?* A quantitative analysis of this question helps one to set a *threshold* so that real "intrusions" are detected and false alarms are avoided. Setting the threshold too low will lead to too many false alarms, whereas setting the threshold too high can result in failure to detect. By knowing the most likely number of occurrences and the probability of deviating from it, we can set a threshold such that the probability of missing real suspicious activities is small. Such a quantitative analysis can also help to choose the size of the sliding window of observation. Finally even in a court case one cannot consider certain observed "bad" activity as a convincing evidence against somebody if that activity is quite likely to occur under given circumstances. Therefore it is very important to quantify such probabilities and present a universal and reliable framework for analyzing a variety of event sources.

Let T be an ordered sequence of events (time-ordered events in a computer system, transactions in a database, purchases made, web sites visited, phone calls made, or combinations of these). Systems designed to detect "bad things" in T usually do not look at the whole of T, they usually involve a sliding "window of observation" (of size, say, w) within which the analysis is confined. This is done for two reasons:

(i) T is usually too long, and without a limited window approach it would involve having to save too much state, and (ii) T can be so long (e.g., in a continuously monitoring system) that *any* subsequence (bad or good) would likely occur within it. As an example of the need to confine the analysis to such a limited sliding window, note that three failed login attempts (with failure due to wrong password) are significant if they occur in rapid succession, but quite innocuous if they occur within a one-month interval. In this study we do not use the notion of real calendar time such as a "one month interval", instead we use the number of events as a proxy for time. This is why our interval length w is not the difference between two time stamps, but rather the size of a (contiguous) substring of T.

In [6] Mannila et al. introduced the problem of finding frequent episodes in event sequences, subject to observation-window constraint, where an episode was defined as a partially ordered collection of events, that can be represented as a directed acyclic graph. In our paper [12] the problem of the reliable detection of an "abnormal" episode was investigated, where by episode we mean a particular sequence occurring as an ordered subsequence of a large event stream *within a window of a given fixed size.* This kind of occurrence is called a "serial episode" in the terminology of [6], and we henceforth adopt this terminology. The event stream was modeled as a memoryless source, the episode under consideration was fixed, and had a length m that was substantially smaller than the number of events n in the stream. The assumption that the window of observation has a fixed size w, $m \leq w \leq n$, is motivated by the fact that most practical event-monitoring systems have a "timeout" mechanism where ancient events drop out of consideration, effectively defining a window of observation. The memoryless assumption turned out to not limit the practical usefulness of the theory, as tested in [12] on sample data for which the memoryless assumption was clearly violated (it was speculated in [12] that this happy situation may have occurred because the deviations from the model tended to cancel each other out, as happens so often when, e.g., analysis carried out under the assumption of independence end up being quite accurate even when independence does not hold). The framework can readily be extended to other models, such as the Markov model of any order, however at a substantial practical cost in terms of computational and space complexities.

Our paper [12] laid out foundations, but did not consider the case of detecting more than one episode, i.e., when k sequences are being monitored for simultaneously in the observed event stream ($k > 1$). This paper builds on [12] by extending it to the case of many pattern sequences, including the important special case of all permutations of the same sequence.This kind of occurrence is called a "parallel episode" in the terminology of [6], and we henceforth adopt this terminology. This case models a situation where the order of events does not matter, e.g., when it is unimportant whether buying suspicious item X occurs before or after buying suspicious item Y. To appreciate the difficulty of this extension, consider the much-simplified case when there are only two pattern sequences and no symbol is common to both of them: Even in this case, a considerable source of difficulty for the probabilistic analysis is the possible overlap of the respective windows within which the patterns occur as subsequences. Add to this the fact that the different patterns typically do have common symbols or common subsequences or possibly common prefixes, that they may have different lengths, and the problem becomes fraught with nasty interactions that prevent any straightforward solution to the $k > 1$ case from consisting of a simple combination or iteration of the techniques developed in [12] for the case $k = 1$ (although without the techniques of [12] we could not have carried out the present analysis of the case $k > 1$). Thus, this paper provides a probabilistic analysis for the following cases: (i) a parallel episode; (ii) an arbitrary set of serial episodes. We can also distinguish a hybrid case where the patterns are obtained from one particular sequence by ignoring the ordering for specific positions in that sequence, i.e., they are partial, restricted permutations of the basic sequence, as opposed to fully permuting the whole sequence in case (i). This case can be easily reduced to case (ii).

More formally, consider an alphabet $\mathcal{A}$ of cardinality $|\mathcal{A}|$, an infinite event sequence $T = t_1 t_2, \ldots$ over $\mathcal{A}$ and a set of episodes $\mathbf{S}$ over $\mathcal{A}$ in one of the following forms: either a set of all distinct permutations of an episode $S = S[1]S[2], \ldots S[m]$ of length m (parallel episode) or an arbitrary set of episodes $\mathbf{S} = \{S_1, S_2, \ldots, S_{|\mathbf{S}|}\}$ where every S_i is a serial episode for $1 \leq i \leq |\mathbf{S}|$. The first case captures situations where the ordering of the events within the window of observation does not matter, e.g., for the two events "bought an unusual amount of fertilizer" and "bought an assault rifle" it may not matter which one occurred first. We use a positive integer $w \geq m$ to represent the length of the window of observation. We assume that $\mathbf{S}$ is given while the event sequence is generated by a memoryless (Bernoulli) or Markov source, however in this paper we focus on memoryless sources. Our interest is in finding $\Omega^{\exists}(n, w)$ that represents the number of windows containing *at least one* occurrence of any element in $\mathbf{S}$ when sliding the window along n consecutive events of T. Based on the observed value of $\Omega^{\exists}(n, w)$ our task is to decide whether a suspicious activity took place or not.

The main thrust of our approach is based on the observation that when searching for unusual patterns (e.g., overrepresented or underrepresented patterns) we must assure, in order to avoid too many false positives, that such patterns are not generated by randomness itself. Therefore, as the first step we study the probabilistic behavior of $\Omega^{\exists}(n, w)$. We compute the expected value of $\Omega^{\exists}(n, w)$, its variance, and then show that appropriately normalized $\Omega^{\exists}(n, w)$ converges in distribution to the standard normal distribution. This allows us to set an upper thresholds $\tau_u(w)$ (for overrepresented patterns). More precisely, for a given level β, we have $P\left(\frac{\Omega^{\exists}(n,w)}{n} \geq \tau_u(w)\right) \leq \beta$. That is, if one *observes* more than $\tau_u(w)$ occurrences of windows with suspicious subsequences, it is highly unlikely that such a number is generated by randomness (i.e., its probability is smaller than β). We also show how to select the window size w so that suspicious subsequences do not occur almost surely in a window. This is necessary to reliably set up the threshold. In [6] Mannila defined the frequency $fr(\alpha, s, win)$ of an episode α as the fraction of windows of length win in which the episode occurs in an event sequence s. Given a frequency threshold min_fr, [6] considered an episode to be frequent if $fr(\alpha, s, win) \geq min\_fr$. In the that framework of [6], our problem can be stated as follows. Given an episode α, what window size win and what frequency threshold min_fr should we choose to ensure that the discovered frequent episodes are meaningful? Observe that for an appropriately low frequency min_fr and large window size win the episode will certainly occur in random data.

We applied our theoretical results by running an extensive series of experiments on real data. We used a part of Wal-Mart sales data for the years 1999 and 2000. We first show that our formulas for the probability closely approximate the experimental data. Then we insert randomly some sequences, defined as "suspicious", and detect them through our threshold mechanism Figure 12. For the purpose of experiments we implemented algorithms for detecting sets of episodes including the unordered case. For the case of detecting ordered episodes we used a variant of the algorithm developed in [6] where algorithms for recognizing both serial and parallel episodes in event sequences, subject to the observation-window constraint, were introduced. Further discussion on algorithms for detecting serial episodes can be found in [2, 4, 5, 8]. For detecting a parallel episode we present an alternative, new on-line $O(n \log m)$ detection algorithm, that we used in experiments.

The problem of episode matching can be rephrased in terms of pattern matching as the subsequence pattern matching or hidden pattern matching [9]. In particular we consider the windowed subsequence pattern matching where by an occurrence we mean a string of the following form $s_1 g_1 s_2 g_2, \ldots g_{m-1} s_m$ where $g_1 \ldots g_{m-1} \in \mathcal{A}^*$ such that the total length of $s_1 g_1 s_2 g_2 \ldots g_{m-1} s_m$ is at most w. Kumar and Spafford [7] applied subsequence pattern matching to intrusion detection. Flajolet *et al.* [9] presented a precise statistical analysis of the subsequence problem. In [8] Boasson *et al.* introduced the *Window-Accumulated Subsequence Matching Problem (WASP)* as finding the number of size w windows of text t of length n which contain pattern $p = p_1 \ldots p_k$ of length $k \leq w$ as a subsequence, where t and p are from an alphabet A. Pattern matching problems are extensively discussed in [10, 13, 14].

Our current work builds on the presented research and provides the first probabilistic analysis that quantifies $\Omega^{\exists}(n, w)$ for an arbitrary set of episodes. The paper is organized as follows. In section 2 we present our main results containing theoretical foundation. Section 3 contains experimental results demonstrating applicability of the derived formulas. Proofs were omitted for space limitation and are included in the full version of the paper.

2. MAIN RESULTS

Given an alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ and a set of patterns $\mathbf{S} = \{S_1, S_2, \ldots S_{|\mathbf{S}|}\}$ where $S_i = S_i[1]S_i[2] \ldots S_i[m_i]$ and $S_i[j] \in \mathcal{A}$ for $1 \leq i \leq |\mathbf{S}|$, we are interested in occurrences of members of $\mathbf{S}$ as a *subsequence* within a window of size w in another sequence known as the event sequence $T = T[1]T[2] \ldots$. A valid occurrence of a single episode S_i in T corresponds to a set of integers $k_1, k_2, \ldots, k_{m_i}$ for $1 \leq i \leq |\mathbf{S}|$ such that the following conditions hold:

1. $1 \leq k_1 < k_2 < \ldots < k_{m_i}$;
2. $T[k_1] = S_i[1], T[k_2] = S_i[2], \ldots, T[k_{m_i}] = S_i[m_i]$;
3. $i_{m_i} - i_1 < w$.

The first two conditions above state that S_i is a subsequence of T, while the last condition guarantees that S_i is a subsequence of T within a window of length w. In various applications, it is of interest to estimate the number of windows of length w containing *at least one* occurrence of either member of $\mathbf{S}$ when sliding the window along n consecutive events in the event sequence T. We use $\Omega^{\exists}(n, w, \mathbf{S}, \mathcal{A})$ to denote this number, that can range from 0 to n.

Notation: Throughout the paper, because $\mathbf{S}$ and $\mathcal{A}$ are always implied, we simplify our notation by dropping $\mathbf{S}$ and $\mathcal{A}$ from the notation $\Omega^{\exists}(n, w, \mathbf{S}, \mathcal{A})$ and denoting it as $\Omega^{\exists}(n, w)$ instead (S and $\mathcal{A}$ are understood). We use the same notational simplification for all other variables that depend on $\mathbf{S}$ and $\mathcal{A}$. We also occasionally use index $m_i - k$ to mean "dropping the last k symbols of S_i", e.g., $P^{\exists}(w, m_1 - k, m_2)$ implies a pattern that is the prefix of S_1 of length $m_1 - k$ and that the second pattern is all of S_2.

Based on the observed value of $\Omega^{\exists}(n, w)$ our task is to decide whether a suspicious activity took place or not. In terms of $\Omega^{\exists}(n, w)$ we can define a threshold in two ways depending on what we consider to be the *unusual* activity [12]. Thus, for a given confidence level β (e.g., $\beta = 10^{-5}$) we define the *upper threshold* $\tau_u(w)$, when $\mathbf{S}$ is overrepresented in T, as follows

$$P\left(\frac{\Omega^{\exists}(n, w)}{n} \geq \tau_u(w)\right) \quad \leq \quad \beta.$$

Another interesting problem is the selection of monitoring system parameters, in particular the size of the window so one can properly design the system. We select w to avoid elements in $\mathbf{S}$ being almost surely in every window for the upper threshold $\tau_u(w)$.

Throughout the paper we assume that the event sequence is generated by a memoryless (Bernoulli) source, i.e., alphabet symbols are generated independently of each others with probability $P(a_i)$ for any $a_i \in \mathcal{A}, i = 1, 2, \ldots, |\mathcal{A}|$.

We need to analyze $\Omega^{\exists}(n, w)$ in order to find the threshold. We will prove here that appropriately normalized $\Omega^{\exists}(n, w)$ is normally distributed. We also find the mean and the variance of $\Omega^{\exists}(n, w)$. In our windowing method we start monitoring T by positioning the right end of the first window on an event in T corresponding to position 1 and while sliding the window n consecutive events to position n we update $\Omega^{\exists}(n, w)$. By assuming that T is infinite, we mean that no matter what window size w we select, there is enough past events available for the initial w consecutive windows.

We start with computing the mean value $\mathbf{E}[\Omega^{\exists}(n, w)]$. Clearly, it is equal to

$$\mathbf{E}[\Omega^{\exists}(n, w)] \quad = \quad nP^{\exists}(w)$$

where $P^{\exists}(w)$ is the probability that a window of size w contains at least one occurrence of either episode in $\mathbf{S}$ as a subsequence. Equivalently we use $P^{\exists}(w, m_1, m_2, \ldots m_{|\mathcal{A}|})$ in derivations where we need to refer to the lengths of patterns in $\mathbf{S}$. For the sake of the presentation we focus throughout the paper on the case where either $\mathbf{S} = \{S_1, S_2\}$, where each S_i being serial or $\mathbf{S}$ is the set of all permutations of an episode S but our our derivations will easily be seen to generalize to an arbitrary set of episodes $\mathbf{S}$.

2.1 Analysis of $P^{\exists}(w)$

Let $\mathcal{W}^{\exists}(w, m_1, m_2)$ be the *set* of all possible distinct windows of length w containing S_1 or S_2 (or both) at least once as a subsequence. $P^{\exists}(w, m_1, m_2)$ is therefore equal to the sum of the probabilities of all the elements of $\mathcal{W}^{\exists}(w, m_1, m_2)$:

$$P^{\exists}(w, m_1, m_2) \quad = \sum_{x \in \mathcal{W}^{\exists}(w, m_1, m_2)} P(x)$$

For $1 \leq i \leq |\mathcal{W}^{\exists}(w, m_1, m_2)|$, let $\mathcal{W}^{\exists}(w, m_1, m_2)[i]$ denote the i-th lexicographically smallest element of $\mathcal{W}^{\exists}(w, m_1, m_2)$. Then the above equation can be equivalently written as:

$$P^{\exists}(w) \quad = \sum_{i=1}^{|\mathcal{W}^{\exists}(w, m_1, m_2)|} P(\mathcal{W}^{\exists}(w, m_1, m_2)[i]).$$

In order to enumerate all the elements in $\mathcal{W}^{\exists}(w, m_1, m_2)$ we define a recursion to describe the structure of this set. Because in the memoryless model $P(\mathcal{W}^{\exists}(w, m_1, m_2)[i])$ is a product of individual probabilities of symbols, to any recursive formula for $\mathcal{W}^{\exists}(w, m_1, m_2)$ there corresponds a similar formula for $P^{\exists}(w, m_1, m_2)$ (and vice-versa). We now show that $P^{\exists}(w, m_1, m_2)$ for the set $\mathcal{A} = \{S_1, S_2\}$ satisfies the following recurrence

$$\begin{cases}
\textbf{if} \quad S_1[m_1] \neq S_2[m_2] \quad \textbf{then} \\
P^{\exists}(w, m_1, m_2) = \\
P(S_1[m_1])P^{\exists}(w-1, m_1-1, m_2) + \\
P(S_2[m_2])P^{\exists}(w-1, m_1, m_2-1) + \\
(1 - P(S_1[m_1]) - P(S_2[m_2]))P^{\exists}(w-1, m_1, m_2) \\
\quad \textbf{for} \quad w > 0, m_1, m_2 > 0 \\
\\
\textbf{if} \quad S_1[m_1] = S_2[m_2] \quad \textbf{then} \\
P^{\exists}(w, m_1, m_2) = \\
P(S_1[m_1])P^{\exists}(w-1, m_1-1, m_2-1) + \\
(1 - P(S_1[m_1]))P^{\exists}(w-1, m_1, m_2) \\
\quad \textbf{for} \quad w > 0, m_1, m_2 > 0 \\
\\
P^{\exists}(w, 0, 0) = 1 \quad \textbf{for} \quad w > 0 \\
P^{\exists}(0, m_1, m_2) = 0 \quad \textbf{for} \quad m_1, m_2 > 0 \\
P^{\exists}(1, m_1, 0) = 1 \quad \textbf{for} \quad m_1 > 0 \\
P^{\exists}(1, 0, m_2) = 1 \quad \textbf{for} \quad m_2 > 0 \\
P^{\exists}(0, 0, 0) = 1
\end{cases}$$

Indeed, consider a window of size w containing S_1 or S_2 as a subsequence. Then depending on whether the last symbols of S_1 and S_2 are equal or not there are two cases. If $S_1[m_1] \neq S_2[m_2]$ then there are three cases: either $S_1[m_1]$ is the last symbol in the window giving the term $P(S_1[m_1])P^{\exists}(w-1, m_1-1, m_2)$, or $S_2[m_2]$ is the last symbol in the window giving the term $P(S_2[m_2])P^{\exists}(w-1, m_1, m_2-1)$, or none of the above which leads to the term $(1 - P(S_1[m_1]) - P(S_2[m_2]))P^{\exists}(w-1, m_1, m_2)$. If $S_1[m_1] = S_2[m_2]$ then there are two cases depending on whether the last symbol of the window is equal to $S_1[m_1]$ or not. From the above discussion it is clear that the shape of the "recursion graph" is determined by interactions between symbols in S_1 and S_2, i.e., whether their symbols at pairs of positions are equal or not. Therefore in order to find a solution to $P^{\exists}(w, m_1, m_2)$ we have to enumerate all pairs of indices (i, j) such that $P^{\exists}(k, i, j)$ appears in the recursion tree (not all such pairs of indices qualify). This recursion graph is now described more formally (as stated earlier, in addition to depicting the recurrence, the graph also describes all elements of $\mathcal{W}^{\exists}(w, m_1, m_2)$).

$G(\mathbf{S}) = (V, E)$ is the edge-labeled graph defined as follows. The vertex set V is a subset of all the pairs (i, j), $0 \leq i \leq m_1$, $0 \leq j \leq m_2$. That subset, as well as E, are defined inductively as follows.

- $(0, 0)$ is in V.
- If (i, j) is in V, $i < m_1$, and $S_1[i+1] \neq S_2[j+1]$ then $(i+1, j)$ is also in V, and an edge from from (i, j) to $(i+1, j)$ labeled $S_1[i+1]$ exists in E.
- If (i, j) is in V, $j < m_2$, and $S_1[i+1] \neq S_2[j+1]$ then $(i, j+1)$ is also in V, and an edge from (i, j) to $(i, j+1)$ labeled $S_2[j+1]$ exists in E.
- If (i, j) is in V, $i < m_1$ and $j < m_2$, and $S_1[i+1] = S_2[j+1]$ then $(i+1, j+1)$ is also in V, and an edge from (i, j) to $(i+1, j+1)$ labeled $S_1[i+1]$ $(= S_2[j+1])$ exists in E.
- A self-loop from vertex (i, j) to itself exists and has label equal to (i) $\mathcal{A}$ if $i = m_1$ or $j = m_2$, (ii) $\mathcal{A} - S_1[i+1] - S_2[j+1]$ if $i < m_1$ and $j < m_2$.

The following observations, in which we do not count self-loops towards the in-degree and out-degree of a vertex, follow from the above definition of $G(\mathbf{S})$.

- The in-degree of vertex $(0, 0)$ is zero.
- The in-degree and out-degree of every vertex (i, j) is at most three; if $\mathbf{S}$ consisted of $|\mathbf{S}| > 2$ serial episodes then the in-degree and out-degree of any vertex would be at most $|\mathbf{S}| + 1$.
- $|V| = O(m_1 m_2)$ and $|E| = O(|\mathbf{S}| m_1 m_2)$.

Let $E(path)$ and $V(path)$ denote the sequence of consecutive edges and (respectively) vertices in any *path*, except that $V(path)$ does not include the last vertex on *path* (why this is so will become apparent below). In what follows, if vertex (i, j) is in $V(path)$, then we use $n_{i,j}(path)$ to denote the number of times the self-loop at (i, j) is used; if *path* is understood and there is no ambiguity, then we simply use $n_{i,j}$ rather than $n_{i,j}(path)$. Let an *end-vertex* be a vertex whose out-degree is zero. Let $\mathcal{R}$ be the set of all distinct simple paths (i.e., witout self-loops) from the start-vertex to any end-vertex. Let $P(E(path)) = \prod_{e \in E(path)} P(label(e))$ be the product of probabilities of the edge labels in $path \in \mathcal{R}$. Now let $\mathcal{L}_w$ be the set of all distinct paths of length w, *including self-loops*, from the start-vertex to all end-vertices (that is, self-loops do count towards path length). Then we have

$$\mathcal{W}^{\exists}(w, m_1, m_2) \quad = \quad \{E(path) : path \in \mathcal{L}_w\}.$$

Examples of $G(\mathbf{S})$ are shown in Figure 1 and in Figure 2.

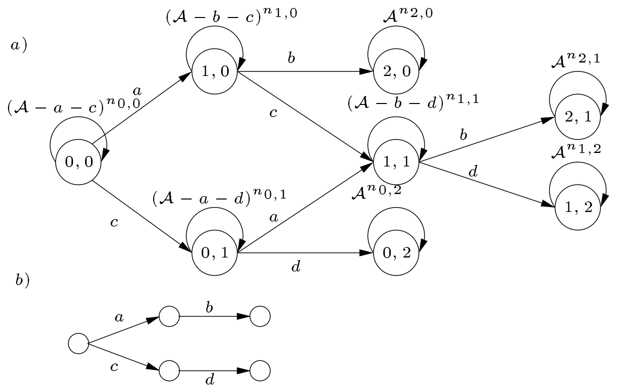

Figure 1: a) $G(\mathbf{S})$ for $\mathbf{S} = \{ab, cd\}$ represented by trie b)

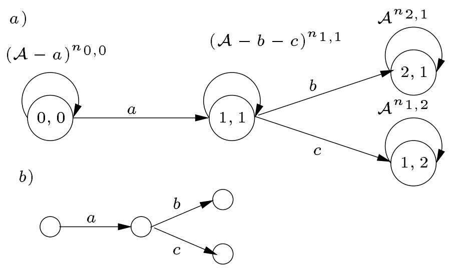

Figure 2: a) $G(\mathbf{S})$ for $\mathbf{S} = \{ab, ac\}$ represented by trie b)

THEOREM 1. *Consider a memoryless source with $P(S_i[k])$ being the probability of generating the k-th symbol of $S_i \in \mathbf{S} = \{S_1, S_2\}$. Let also*

$$P(E(path)) = \prod_{edge \in E(path)} P(label(edge)).$$

Then for m_1, m_2 and $w \geq m_i$ we have

$$P^{\exists}(w, m_1, m_2) = \sum_{path \in \mathcal{R}} P(E(path)) \sum_{g=0}^{w-|E(path)|} \sum_{\sum n_{i,j}(path)=g} \cdot \prod_{(i,j) \in V(path)} (1 - P(\{S_1[i+1]\} \cup \{S_2[j+1]\}))^{n_{i,j}(path)}.$$

In our experiments, we implemented an efficient dynamic programming algorithm based on Theorem 2.1. In section 3.1.2 we present evidence how Theorem 2.1 works well on real data by comparing it to the estimate $P_e^{\exists}(w) = \frac{\Omega^{\exists}(n,w)}{n}$ given the actual $\Omega^{\exists}(n, w)$. We also solved $P^{\exists}(w)$ for an important special case when $\mathbf{S}$ consist of all permutations of one pattern S, which is the case of a parallel episode. Using Theorem directly to design an algorithm in such an unordered case would be inefficient because we would then need to consider a graph having a disastrous $|V| = O(m^{m!})$.

In order to simplify the graph $G(\mathbf{S})$ that would result from all permutations, and bring its number of vertices down to a manageable size (quantified below), we exploit the structure of a set of all permutations to design a different graph. Notice that, for a parallel episode, every path in $\mathcal{R}$ is a permutation of symbols in S. In addition, the out-degree of a vertex is at most m if all symbols of S are different. Furthermore a transition from from $P^{\exists}(k, i, j, \ldots)$ to $P^{\exists}(k+1, i', j', \ldots)$ takes place for any symbol of S not seen so far since the order of symbols does not matter. This observations allow us to introduce a variant of $G(\mathbf{S})$ called $G(S!)$, which enumerates $\mathcal{W}^{\exists}(w, m_1, m_2)$ for the unordered case.

Let $G_{S!}(V, E)$ be a directed graph with the following structure.

- V is the set of all possible subsets of symbols of S of size $i = 0, 1, 2, m$. Each vertex is of the form $v(\{i_1, i_2, \ldots, i_m\})$ where $i_j = 1$ if the vertex contains i_j-th symbol in its subset or $i_j = 0$ otherwise. E is the set of symbols in S corresponding to an increase by one of the cardinality of subsets of symbols in adjacent vertices.
- The *in-degree* $d_{in}(v(\{i_1, i_2, \ldots, i_m\})) \leq m + 1$ for all vertices except for the start-vertex where $d_{in}(v(\{0, 0, \ldots, 0\})) = 0$.
- The *out-degree* $d_{out}(v(i, j)) \leq m + 1$ for all vertices except for the end-vertex where $d_{out}(v(\{1, 1, \ldots 1\})) = 0$ Notice, that vertex $v(\{1, 1, \ldots 1\})$ is the only end-vertex.
- Each vertex has a self-loop corresponding to lack of progress to the next vertex. There are no other cycles in the graph.
- $|V| = O(\sum_{i=0}^{m} \binom{m}{i})$ and $|E| = O(|S|^{m+1})$.

Let $\mathcal{R}$ be the set of all distinct permutations of S, corresponding to all distinct paths from the start-vertex the end-vertex excluding self-loops and the end-vertex. Let $\mathcal{L}$ be the set of all distinct paths, including the self-loops, from the start-vertex to the end-vertex of length w in $G_{S!}(V, E)$ then as in the case of $G_{\mathbf{S}}(V, E)$ the union of such paths enumerates $\mathcal{W}^{\exists}(w, m_1, m_2)$. Examples of $G_{S!}(V, E)$ are shown in Figure 3 and in Figure 4. The next theorem is an adoption of Theorem 2.1 to the unordered case.

THEOREM 2. *Consider a memoryless source with $P(S_k)$ being the probability of generating the k-th symbol of S where S is a parallel episode with*

$$P(S) = \prod_{i=1}^{m} P(S[i])$$

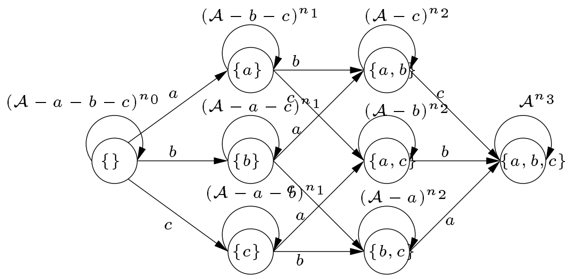

Figure 3: $G_{S!}(V, E)$ **for** $S = abc$ **and** $\mathcal{A} = \{a, b, c, d\}$

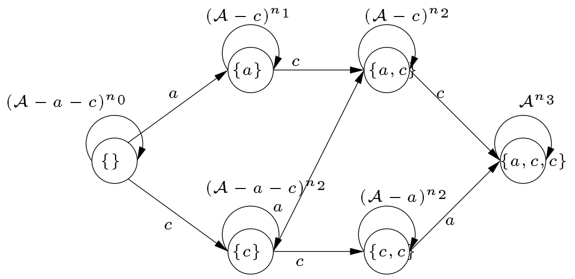

Figure 4: $G_{S!}(V, E)$ **for** $S = acc$ **and** $\mathcal{A} = \{a, b, c, d\}$

then for all m and $w \geq m$ we have

$$P^{\exists}(w) = \sum_{path \in \mathcal{R}} P(S) \sum_{g=0}^{w-m} \sum_{\sum n_{i_1,i_2,\ldots,i_m} = g} \cdot$$

$$\prod_{v(i_1,i_2,\ldots,i_m) \in V(path)} \left(1 - P\left(\bigcup_{i_j=1} S[i_j]\right)\right)^{n_{i_1,i_2,\ldots,i_m}} .$$

In our experiments we implemented an efficient dynamic programming algorithm based on Theorem 2 for computing $P^{\exists}(w)$. In order to verify the applicability of Theorem 2 for the Wal-Mart data we also designed an efficient $O(n \log(m))$ tree based algorithm for discovering occurrences of a parallel episode in an event sequence. In section 3.1.1 we presented how 2 works well on real where the formula for $P^{\exists}(w)$ agrees with the Wal-Mart transactions. The details of the $O(n \log(m))$ parallel episode detection algorithm are provided in section 2.2. When establishing the above formulas for $P^{\exists}(w)$ we also solved a related combinatorial problem on strings that is of independent interest. Namely, given $\mathcal{A}$, w and $\mathbf{S}$: Find the cardinality of $\mathcal{W}^{\exists}(w, m_1, m_2)$ that we denote $C^{\exists}(w, m_1, m_2)$. The formula for $C^{\exists}(w, m_1, m_2)$ has the following form.

THEOREM 3. *The number of distinct windows of size w containing at least one occurrence of either member of set* $\mathbf{S} = \{S_1, S_2\}$ *is equal to*

$$C^{\exists}(w, m_1, m_2) = \sum_{path \in \mathcal{R}} \sum_{k=0}^{w-|V(path)|} \binom{k+m-1}{k} \cdot$$

$$\prod_{v(i,j) \in V(path)} \left(|\mathcal{A}| - \sum_{S_1[i+1] \cup S_2[j+1]} 1\right)^k |\mathcal{A}|^{w-|V(path)|-k}.$$

for $m_1 = |S_1|$ and $m_2 = |S_2|$. Proofs of all theorems presented in this section are included in the full version of the paper.

2.2 Efficient algorithm for detection of a parallel episode

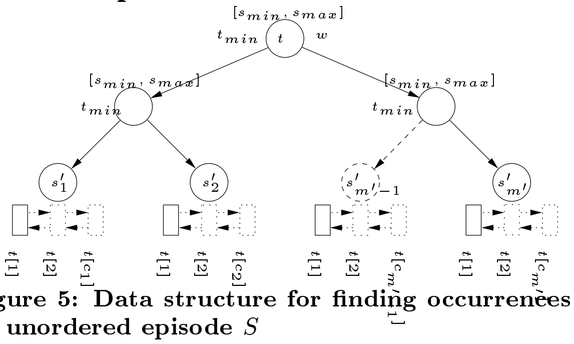

Figure 5: Data structure for finding occurrences of an unordered episode S

Let $S = S[1]S[2]\ldots S[m]$ be the input episode for the algorithm. Let $S' = \{s'_1, s'_2, \ldots s'_{m'}\}$ be a set of cardinality m' of all different symbols of pattern S. Let c_i for $i = 0, 1 \ldots m'$ be the number of times an alphabet symbol s'_i occurs in S. We build a binary tree over symbols in S' as leaves. Figure 5 shows the tree.

Each node in the tree contains the usual tree pointers: $parent$, $left$, $right$ and search key interval $[s_{min}, s_{max}]$ where s_{min} is the smallest key in the subtree and s_{max} is the largest key in the subtree. In addition, depending on the type a node keeps the following specific information:

- root: t event counter, counts the number of elapsed (scanned) events.
- internal node:
  - t_{min} the minimum time of the arrival for the subtree rooted at this node
- leaf node:
  - search key value s_i
  - *dlist*: doubly linked list containing one element for each symbol a_i in the pattern S. The purpose of the list is to keep track of the most recent occurrences of the symbol a_i sorted by arrival times. So the size of the list is one unless S contains more than one symbol a_i. Let $t_{a_i}[1], t_{a_i}[2] \ldots t_{a_i}[c_i]$ for $i = 0, 1 \ldots m'$ be the times of the occurrence of symbol a_i from the left to the right in the list then they must satisfy the condition $t_{a_i}[j] < t_{a_i}[j+1]$

for $j = 1, \dots c_i$. So the leftmost element of the list contains the oldest occurrence of a_i and the rightmost element contains the most recent occurrence of a_i.

The tree supports the following operations:

- **update(s)**: when a new symbol s arrives the time t is incremented by one and the search tree structure is used to find the proper leaf. If the search finds leaf s_i then the leftmost element of the doubly linked list with time $t_{s_i}[1]$ is removed and a new element with the current time t is attached to the right end of the list. Once the new value is attached to the list the value of the leftmost element, as the oldest one, is propagated upward as long as it is smaller then t_{min} of the internal node on the path to the root. This operation takes $\log(m)$ time.
- **exists**: if $t - t_{min} + 1 \leq w$ at the root node then at least one permutation of S occurs as a subsequence withing the window. This operation takes $O(1)$.

The time complexity for finding $\Omega^{\exists}(n,w)$ is $O(n \log m)$, because we perform n slides of the window each requiring $O(\log m)$. The presented tree structure can also handle a problem when instead of a window of size w we associate a time to live *ttl* with each symbol of the pattern S. In such a case each leaf stores the expiration time $expt = t + ttl$ and the each internal node stores the minimum expiration time $expt_{min}$ in its subtree. The *update*(s) operation remains the same and *exists* returns *true* if $expt_{min} > t$ at the root node.

2.3 Computation of threshold $\tau_u(w)$

Now we derive variance and normal limiting distribution of $\Omega^{\exists}(n,w)$. Observe that

$$\Omega^{\exists}(n,w) = \sum_{i=1}^{n} I_i^{\exists}$$

where

$$I_i^{\exists} = \begin{cases} 1 & \text{if any member of } \mathbf{S} \text{ occurs at least once as a} \\ & \text{subsequence in the window ending at position } i \text{ in } T; \\ 0 & \text{otherwise,} \end{cases}$$

where i is the relative position with respect to the first position ($i = 1$). Thus, we easily have

$$\begin{aligned} \mathbf{E}[I_i^{\exists}] &= P^{\exists}(w), \\ \mathbf{Var}[I_i^{\exists}] &= P^{\exists}(w) - (P^{\exists}(w))^2. \end{aligned}$$

In order to compute variance of $\Omega^{\exists}(n,w)$ we need $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,k)$, defined as the probability that two overlapping windows at respective position i and j for $|i-j| < w$ have $I_i = 1$ and $I_j = 1$. The variance can be expressed as follows

$$\begin{aligned} &\mathbf{Var}[\Omega^{\exists}(n,w)] = n\mathbf{Var}[I_1^{\exists}] + 2\textstyle\sum_{1 \leq i < j \leq n}^{n} \mathbf{Cov}[I_i^{\exists}, I_j^{\exists}] \\ &= n\mathbf{Var}[I_1^{\exists}] + \\ &2(n-w+1)\textstyle\sum_{k=1}^{w-1} \left[P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,k) - (P^{\exists}(w))^2\right] + \\ &+2(w-2)\left[\textstyle\sum_{k=2}^{w-1}\sum_{q=k}^{w-1} \left[P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,k) - (P^{\exists}(w))^2\right]\right]. \end{aligned}$$

The two terms involving $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,k)$ in the above formula represent correlation between windows (with $2(w-1)$ neighborhood), where $k = w - |i-j|$ represents the length of the overlap between windows at position i and j. $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,k)$ can be computed by an enumerative algorithm. One concludes, however, that $\mathbf{Var}[\Omega^{\exists}(n,w)] \sim n\sigma$ for some $\sigma > 0$. In view of the above, and using the fact that $\Omega^{\exists}(n,w)$ is the so called $w-1$-dependent sequence (i.e., $\Omega^{\exists}(n,w)$ depends on the last $w-1$ windows), we may apply Theorem 27.5 of [3] to establish the central limit theorem for $\Omega^{\exists}(n,w)$.

THEOREM 4. *The random variable $\Omega^{\exists}(n,w)$ obeys the Central Limit Theorem in the sense that its distribution is asymptotically normal. More precisely, for $a, b = O(1)$ we have*

$$\lim_{n\to\infty} P\left\{a \leq \frac{\Omega^{\exists}(n,w) - \mathbf{E}[\Omega^{\exists}(n,w)]}{\sqrt{\mathbf{Var}[\Omega^{\exists}(n,w)]}} \leq b\right\} = \frac{1}{\sqrt{2\pi}}\int_a^b e^{\frac{-t^2}{2}}dt$$

Using the above findings and the definition of $\mathbf{E}[\Omega^{\exists}(n,w)]$ we establishing the reliable threshold $\tau_u(w)$ as follows.

$$\begin{cases} P\left\{\frac{\Omega^{\exists}(n,w)}{n} \geq \tau_u(w)\right\} &= y(b,\infty) \\ \tau_u(w) &= P^{\exists}(w) + \frac{b\sqrt{\mathbf{Var}[\Omega^{\exists}(n,w)]}}{n} \\ y(a,b) &= \frac{1}{\sqrt{2\pi}}\int_a^b e^{\frac{-t^2}{2}}dt \end{cases}$$

Thus, for a given β we can compute $\tau_u(w)$ by selecting either b_0 such that $\beta = y(b_0, \infty)$. Observe that when a b are large (say order of 10) the above probability is small enough to be qualified as a *moderate large deviations.* This captures the nature of unusual episodes, as needed.

3. EXPERIMENTS

The purpose of our experiments was to test applicability of the analytical results for real sources subject of a possible monitoring systems. Therefore we selected Wal-Mart data available on the departmental Oracle server in the Department of Computer Sciences, Purdue University. The database contains part of Wal-Mart sales data for the years 1999 and 2000 in 135 stores. We selected one of the stores, one category of items of cardinality 35 and extracted 9.66 million records from the table *Item_Scan*, sorted by scan time, where each record has the following form

```
Table walmart.Item_Scan
        Visit_Nbr             Integer
        Item_Nbr              Integer
        Item_Quantity         Decimal(9,2)
        Total_Scan_Amount     Decimal(9,2)
        transaction_Date      Date Format
        Unit_Cost_Amount      Decimal(9,4)
        Unit_Retail_Amount    Decimal(9,2)
        Tax_Collect_Code      Char(1)
     Primary Index (Visit_Nbr);
```

Thus the alphabet size $|\mathcal{A}| = 35$. We divided our sources into training sets and testing sets. Training sets are data sets, which we consider to constitute normal behavior for the environment from which the data were drawn. Once the training data has been characterized, i.e. the probability model has been built, we can start monitoring unknown data called testing data. During the monitoring process the testing data are compared to expectations generated by the training data. In section 3.1 we tested how well the formulas for $P^{\exists}(w)$ worked on the Wal-Mart data. To accomplish this we estimated the actual probability of existence based on the actual number of windows $\Omega^{\exists}(n,w)$ as $P_e^{\exists}(w) = \frac{\Omega^{\exists}(n,w)}{n}$ and compared its value to the computed $P^{\exists}(w)$ for different values of w. We used the following error metric d

$$d = \left[\frac{1}{r} \sum_{i=1}^{r} \frac{|P_e^{\exists}(w_i) - P^{\exists}(w_i)|}{P_e^{\exists}(w_i)} \right] 100\%$$

where $w_1 < w_2 < \dots w_r$ are the tested window sizes. In section 3.2 we tested the detection properties of the threshold $\tau_u(w)$ as a function of the window length w. As a testing source for computing the probabilities of symbols in $|\mathcal{A}|$ for all experiments we used the first 9.56 million records leaving the last 100 thousand records for testing. All our algorithms have been implemented in C++ and run under Linux operating system.

3.1 Estimation of $P^{\exists}(w)$ and $\mathbf{E}[\Omega^{\exists}(n,w)]$

In all experiments in this section we used the same testing source of length $n = 10^5$ events.

3.1.1 *The case a parallel episode S*

In this experiment we set

$$S = \{item_0, item_4, item_5, item_6, item_9, item_{10}, item_{17}\}$$

Then for $w = 10, 15, 20, \dots 180$ we ran the tree based detection algorithm presented in section 2.2 to find $\Omega^{\exists}(n,w)(10^5, w)$ (the actual number of occurrences of S in the stream of length 10^5). Then we computed $P_e(w)$ for each $\Omega^{\exists}(n,w)(10^5, w)$ and compared $P_e(w)$ to the analytically computed $P^{\exists}(w)$ using our algorithm based on Theorem 2. The result is shown in Figure 6, which indicated an exceptionally close fit between $P^{\exists}(w)$ and $P_e^{\exists}(w)$ with the difference d of order 2%. One of the two reasons of such a surprising closeness may be the fact that in a case of a parallel episode we neglect some correlations between consecutive symbols of S. Another reason may be a closeness of the time ordered purchases of the item category considered in experiments to the ideal Markov source. We also plotted the actual $\Omega^{\exists}(n,w)$ and the computed $\mathbf{E}[\Omega^{\exists}(n,w)]$ in Figure 7.

3.1.2 *The case of a set of two serial episodes $\mathbf{S} = \{S_1, S_2\}$*

In this experiment we set

$$\begin{aligned} S_1 &= \{item_0, item_4, item_5, item_6, item_9, item_{10}, item_{17}\} \\ S_2 &= \{item_0, item_6, item_5, item_4, item_{10}, item_9, item_{17}\} \end{aligned}$$

where S_2 is a permuted version of S_1. This case reflects a situation when a pattern of interest is only partially restricted and the serial case is too restrictive but the parallel case too relaxed. Once we created the set $\mathbf{S}$ we ran an algorithm for finding $\Omega(10^5, w)$ for the set. Then we computed $P_e(w)$ for each $\Omega(10^5, w)$ and compared $P_e(w)$ to the analytically computed $P^{\exists}(w)$ using our algorithm based on Theorem 2.1. The result is shown in Figure 8, which indicated a very close

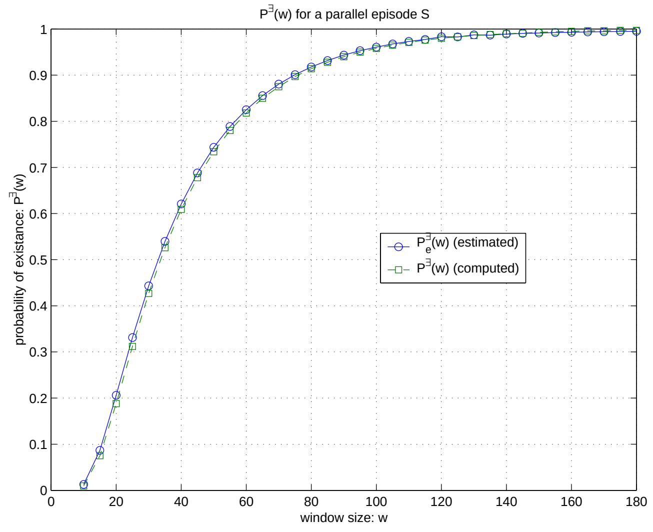

Figure 6: $P_e^{\exists}(w) = \frac{\Omega^{\exists}(n,w)}{n}$ and $P^{\exists}(w)$ for a parallel episode S, using Wal-Mart data

fit between $P^{\exists}(w)$ and $P_e^{\exists}(w)$ with d of order 13% but not so close as in the case parallel ($d = 2\%$). The reason may be the fact in this partially restricted case some correlations between the symbols come into play and the two sets S_1 and S_2 do not represent all of the serial correlations between the symbols in $S_1 \cup S_2$. We also plotted the actual $\Omega^{\exists}(n,w)$ and the computed $\mathbf{E}[\Omega^{\exists}(n,w)]$ in Figure 9.

3.1.3 *Comparison of the three cases: S_1 parallel, $\{S_1, S_2\}$ serial and S_1 serial*

In this experiment we investigate the relationship between the formulas for $P(w)$ and the experimental $P_e(w)$ for the episode S in the three cases: parallel, partially ordered (S_2 is a permutation of S_1) and serial in [12]. For the first two cases we use the results obtained in the previous experiments. For the third case we ran an algorithm for discovering a serial episode in the event sequence for $w = 10, 15, 20, \dots 180$ as in the previous experiment to create $\Omega^{\exists}(n,w)(10^5, w)$ at the same points. For computing $P^{\exists}(w)$ we used the algorithm we designed for sets since the single serial episode is just a singleton set and we know that the Theorem 2.1 generalizes to an arbitrary set of episodes. The results for $P^{\exists}(w)$ are shown in Figures 10 and the corresponding estimates are shown in Figure 11. The figures clearly indicate that the serial and parallel cases of an episode S establish the lower and upper bound on the probability of existence of S in window of size w. Based on our paper [12] we can prove that $P^{\exists}(w) = 1 - \theta(\rho^w)$ for w large and $\rho < 1$.

3.2 Threshold $\tau_u(w)$

This experiment demonstrates the application of the upper threshold $\tau_u(w)$ in for a a parallel episode S. It also shows the relationship between $\tau_u(w)$ and w in detecting an unusual behavior. We set

$$S = \{item_8, item_{12}, item_{14}, item_{15}, item_{19}, item_{20}, item_{26}\}$$

and consider the parallel case of S. We selected a part of the scans of length $n = 50000$ as the test source. In order

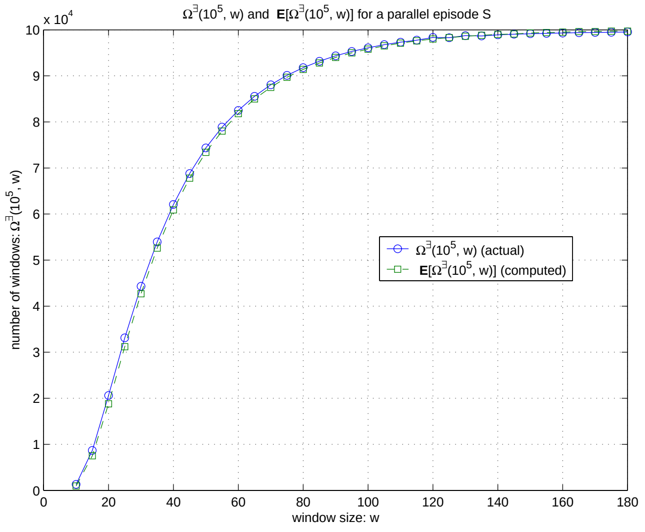

Figure 7: $\Omega^{\exists}(n,w)$ and $\mathbf{E}[\Omega^{\exists}(n,w)]$ for a parallel episode S, using Wal-Mart data

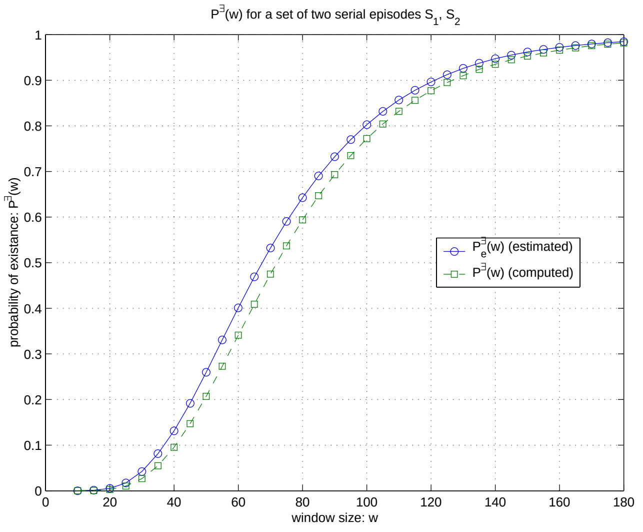

Figure 8: $P_e^{\exists}(w) = \frac{\Omega^{\exists}(n,w)}{n}$ and $P^{\exists}(w)$ for a set $\mathbf{S} = \{S_1, S_2\}$ of serial episodes, using Wal-Mart data

to establish the threshold $\tau_u(w) = P^{\exists}(w) + \frac{4\sqrt{\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w)]}}{n}$ we estimated $\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w)]$ using the sample variance estimator defined below of the scans.

$$\sqrt{\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w)]} = \sqrt{\frac{\sum_{i=1}^{10}(\Omega^{\exists}(n,w)_i - \mathbf{E}[\Omega^{\exists}(n,w)])^2}{10}}.$$

In order to use the estimator we selected 10 fragments of length 50000 in the source sequence as the training sets. We repeated the threshold computation for three values of $w = 40, 30, 25$ for the same episode S. We simulated an attack by inserting the episode S as a string into the testing source until we exceeded the threshold. We normalized the number of insertion by n. Figure 12 presents results. We conclude that the if w decreases then the number of attacks causing $\frac{\Omega^{\exists}(n,w)}{n}$ to rise above the $\tau_u(w)$ decreases as well making the monitoring system more sensitive to attacks. From the monitoring system point of view this means that $P^{\exists}(w)$ should be selected "as small as possible" because as $P^{\exists}(w)$ grows than it is harder and harder to draw a line between a normal and unusual behavior. Also the growth of $\tau_u(w)$ is exponential as w increases, which is cause by the exponential growth of $P^{\exists}(w)$ in the formula for $\tau_u(w)$.

4. CONCLUSIONS AND EXTENSIONS

We presented the exact formulas for the probability of existence $P^{\exists}(w)$, for an arbitrary set of serial episodes and the case of a parallel episode S. For the purpose of our experiments we implemented the formulas by efficient dynamic programming algorithms, proving the formulas are applicable in practice. We did not present those algorithms for space limitation. We also designed a $O(n\log(m))$ tree based algorithms for discovering a parallel episode S of length m in an event sequence T of length n. In experiments on Wal-Mart transactions we showed that the formulas exactly approximated the real life data. We also showed that as the window size grows w sensitivity of a monitoring system drops requiring more attacks to exceed the threshold.

An obvious extension of this work is to use the graphs $G_S(V,E)$, $G_{S!}(V,E)$, whose paths from the start-vertices to end-vertices of length w correspond to the set $W(w)$. Let $\mathcal{W}^{\exists}(w)[i][j]$ be the j-th symbol in $\mathcal{W}^{\exists}(w)[i]$ where $j = 1, 2, \ldots, w$. Then for the first order Markov source the probability that a window of length w contains at least one occurrence of a pattern S of length m is equal to:

$$P^{\exists}(w) = \sum_{i=1}^{|\mathcal{W}^{\exists}(w)|} P(\mathcal{W}^{\exists}(w)[i][1])P(\mathcal{W}^{\exists}(w)[i][2]|\mathcal{W}^{\exists}(w)[i][1])\ldots P(\mathcal{W}^{\exists}(w)[i][w]|\mathcal{W}^{\exists}(w)[i][w-1])$$

where $\mathcal{W}^{\exists}(w)[i][l]$ is the l-the symbol of the i-th member of $\mathcal{W}^{\exists}(w)$ in lexicographic order and $P(\mathcal{W}^{\exists}(w)[i][2]|\mathcal{W}^{\exists}(w)[i][1])$ is the conditional probability. However the realization of the Markov model involves a substantial computational cost.

5. REFERENCES

[1] M. D. A. Wespi, H. Debar and M. Nassehi. Fixed vs. variable-length patterns for detecting suspicious process behavior. *Computer Security*, 2000.

[2] A. Apostolico and M. Atallah. Compact recognizers of episode sequences. *Information and Computation*, 174.

[3] P. Billingsley. *Probability and measure.* John Wiley, New York, 1986.

[4] L. G. D. G. G. Das, R. Fleischer and J. Kärkkäinen. Episode matching. In *Lecture Notes in Computer Science*, 1997.

[5] M. R. G. Kucherov. Matching a set of strings with variable length don't cares. *Theoretical Computer Science*, 1997.

[6] H. T. H. Mannila and A. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1997.

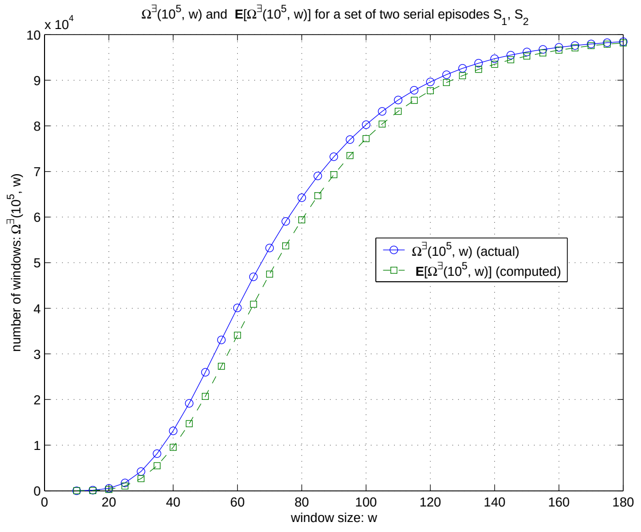

Figure 9: $\Omega^{\exists}(n, w)$ and $\mathbf{E}[\Omega^{\exists}(n, w)]$ for a set $\mathbf{S} = \{S_1, S_2\}$ of serial episodes, using Wal-Mart data

[7] S. Kumar and E. Spafford. A pattern-matching model for intrusion detection. In *Proceedings of the National Computer Security Conference*, 1994.

[8] I. G. L. Boasson, P. Cegielski and Y. Matiyasevich. Window-accumulated subsequence matching problem is linear. In *PODS*, 1999.

[9] W. S. P. Flajolet, Y. Guivarc'h and B. Vallée. Hidden pattern statistics. In *ICALP 2001*, 2001.

[10] B. S. P. Nicodème and P. Flajolet. Motif statistics. *Lecture Notes in Computer Science*, 1999.

[11] P. Pevzner. *Computational Molecular Biology: An Algorithmic Approach.* MIT Press, 2000.

[12] W. S. R. Gwadera, M. Atallah. Reliable detection of episodes in event sequences. In *ICDM*, 2003.

[13] M. Régnier and W. Szpankowski. On pattern frequency occurrences in a markovian sequence. *Algorithmica*, 1998.

[14] W. Szpankowski. *Average Case Analysis of Algorithms on Sequence.* John Wiley, 2001.

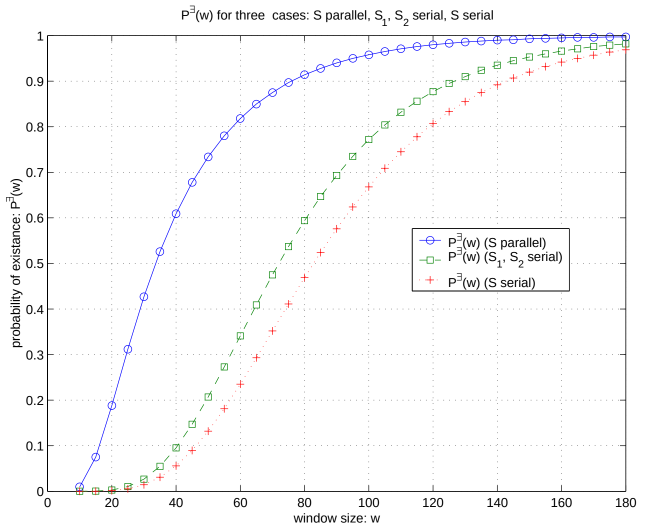

Figure 10: $P^{\exists}(w)$ for three cases: S parallel, $\{S_1, S_2\}$ serial and S serial, using Wal-Mart data

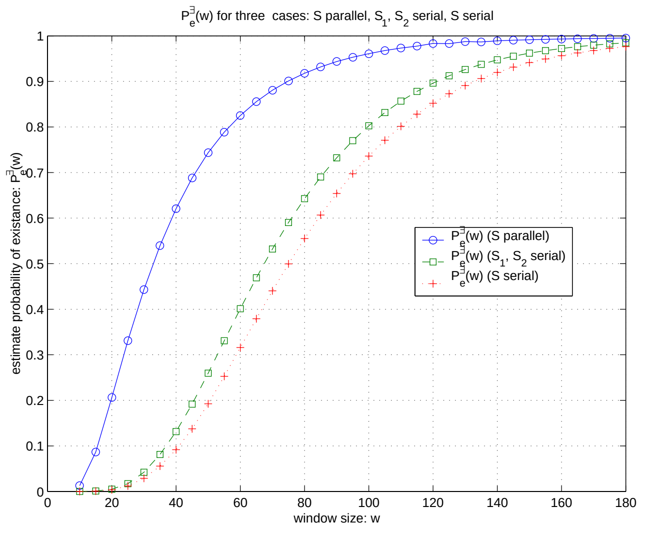

Figure 11: $P_e^{\exists}(w) = \frac{\Omega^{\exists}(n,w)}{n}$ for three cases: S parallel, $\{S_1, S_2\}$ serial and S serial, using Wal-Mart data

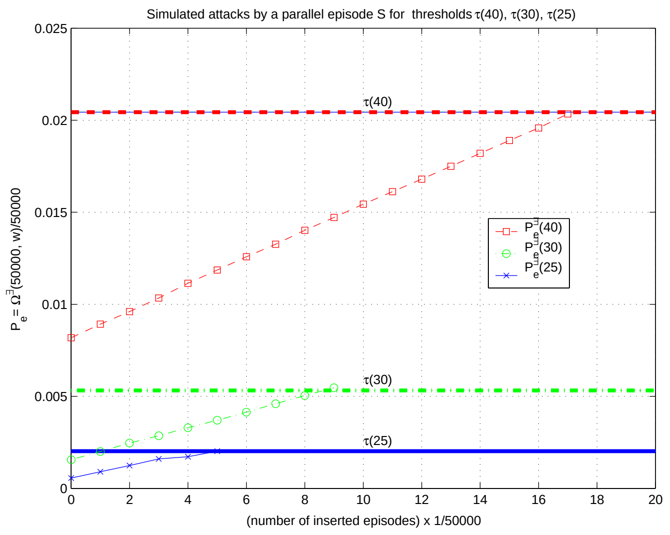

Figure 12: The upper threshold $\tau_u(w)$ as a function of w for a parallel episode S, using the Wal-Mart database