Reliable Detection of Episodes in Event Sequences

March 19, 2004

Robert Gwadera, Mikhail J. Atallah*, Wojciech Szpankowski†
Department of Computer Science
Purdue University
W. Lafayette, IN 47907
{gwadera,mja,spa}@cs.purdue.edu

Abstract

Suppose one wants to detect "bad" or "suspicious" subsequences in event sequences. Whether an observed pattern of activity (in the form of a particular subsequence) is significant and 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. Suppose a long sequence T of observed events contains a suspicious subsequence pattern S within it, where the suspicious subsequence S consists of m events and spans a window of size w within T. We address the fundamental problem: is a certain number of occurrences of a particular subsequence unlikely to be generated by randomness itself (i.e., indicative of suspicious activity)? If the probability of an occurrence generated by randomness is high and an automated monitoring system flags it as suspicious anyway, then such a system will suffer from generating too many false alarms. This paper quantifies the probability of such an S occurring in T within a window of size w, the number of distinct windows containing S as a subsequence, the expected number of such occurrences, its variance, and establishes its limiting distribution that allows to set up an alarm threshold so that the probability of false alarms is very small. We report on experiments confirming the theory and showing that we can detect bad subsequences with low false alarm rate.

Key Terms: Episode pattern matching, hidden pattern matching, overrepresented and underrepresented patterns, data mining, probabilistic analysis.

*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 and NIH R01 GM068959-01.

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 [9, 12, 17, 18]). 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.

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 an episode over $\mathcal{A}$ in one of the following forms: either a sequential pattern $S = s_1 s_2, \ldots s_m$ of length m, or a set of patterns $\mathbf{S} = \{S_1, S_2, \ldots, S_{|\mathbf{S}|}\}$, or the set of all distinct permutations of S; this last case captures situations where the ordering of the events within the window of observation does not matter, e.g., for the two events "bought a large number of bullets" 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 an episode is given while the event sequence T is generated by a memoryless (Bernoulli) or Markov source. However in this paper we focus on the case of the single pattern episode S and we assume T is a memoryless sources. In [10] Mannila *et al.* introduced the problem of discovering frequent episodes in event sequences, where an episode was defined as a collection of events occurring together within a certain time interval. In terminology of [10] an episode in the form of a sequential pattern S was defined as a *serial* episode and the set of all permutations of an episodes S was defined as a *parallel* episode.

Our interest is in finding $\Omega^{\exists}(n, w, m)$ that represents the number of windows containing *at least one* occurrence of S when sliding the window along n consecutive events of T. Based on the observed value of $\Omega^{\exists}(n, w, m)$ 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 that such patterns are not generated by randomness itself in order to avoid too many false positives. Therefore, as the first step we study the probabilistic behavior of $\Omega^{\exists}(n, w, m)$. We compute the expected value of $\Omega^{\exists}(n, w, m)$, its variance, and then show that appropriately normalized $\Omega^{\exists}(n, w, m)$ converges in distribution to the standard normal distribution. This allows us to set either an upper thresholds $\tau_u(w, m)$ (for overrepresented patterns) or a lower threshold $\tau_\ell(w, m)$ (for underrepresented patterns) depending on the definition of unusual activity. More precisely, for a given level β, we have either $P\left(\frac{\Omega^{\exists}(n,w,m)}{n} \geq \tau_u(w, m)\right) \leq \beta$ or $P\left(\frac{\Omega^{\exists}(n,w,m)}{n} \leq \tau_\ell(w, m)\right) \leq \beta$, respectively. That is, if one *observes* more than $\tau_u(w, m) \cdot n$ occurrences (upper threshold) or fewer than $\tau_\ell(w, m) \cdot n$ occurrences (lower threshold) 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 [10] the frequency of an episode α $fr(\alpha, s, win)$ was defined as the fraction of windows of length win in which the episode occurs in an event sequence s. Given a frequency threshold min_fr, they considered an episode to be frequent if $fr(\alpha, s, win) \geq min\_fr$. In the framework of Mannila's work, 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, since for an appropriately low frequency min_fr and large window size win the episode will certainly occur in random data.

We verify our theoretical results by running an extensive series of experiments. One set of experiments is performed on an on-line version of *War and Peace*, as an example of English text source. In another experiment we use web-logs obtained from `http://www.cs.washington.edu/ai/adaptive-data/` that contains user accesses to the music machines web site (currently at `http://machines.hyperreal.org`) from 1/01/99 through 4/30/99. We first show that our formula for the probability approximates very well the experimental one. Then we insert randomly some sequences, defined as "suspicious", and detect them through our threshold mechanism.

Our problem can be rephrased in terms of pattern matching as the subsequence pattern matching or hidden pattern matching [7]. 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 [9] applied subsequence pattern matching to intrusion detection. Apostolico and Atallah [2] designed a fast algorithm to detect subsequences in a text, while Flajolet *et al.* [7] presented a precise statistical analysis of the subsequence problem. In [4] 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. Our work builds on the above mentioned research and provides the first probabilistic analysis that quantifies $\Omega^{\exists}(n, w, m)$.

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. Derivations of theoretical results are presented in Section 4 using analytic tools of analysis of algorithms such as generating functions and complex asymptotic (cf. [15, 16]).

2 Main Results

Given an alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_{|\mathcal{A}|}\}$ and a pattern $S = s_1 s_2, \ldots s_m$ of length m, we are interested in occurrences of 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 S in T corresponds to a set of integers $i_1, i_2, \ldots, i_m$ such that the following conditions hold:

1. $1 \leq i_1 < i_2 < \ldots < i_m$;
2. $t_{i_1} = s_1, t_{i_2} = s_2, \ldots, t_{i_m} = s_m$;
3. $i_m - i_1 < w$.

The first two conditions above state that S is a subsequence of T, while the last condition guarantees that S 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 S when sliding the window along n consecutive events in the event sequence T; we use $\Omega^{\exists}(n, w, m, S, \mathcal{A})$ to denote this number, that can range from 0 to n.

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

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

1. *Upper threshold* ($\tau_u(w, m)$): when S is overrepresented in T we quantify it by setting
$$P\left(\frac{\Omega^{\exists}(n, w, m)}{n} \geq \tau_u(w, m)\right) \quad \leq \quad \beta.$$
2. *Lower threshold* ($\tau_\ell(w, m)$): when S is underrepresented in T we quantify it by setting
$$P\left(\frac{\Omega^{\exists}(n, w, m)}{n} \leq \tau_\ell(w, m)\right) \quad \leq \quad \beta.$$

The case number 2 above corresponds to a situation when for a normal behavior of T we must see at least a certain number of windows containing S and if that number drops suddenly then it can suggest an intentional suppression of S.

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 S being almost surely in every window for the upper threshold $\tau_u(w,m)$, or to avoid S being almost surely in none of the windows for the lower threshold $\tau_\ell(w,m)$.

In order to find a reliable threshold that minimizes the number of false positives, we compare the *observed* value of $\Omega^\exists(n,w,m)$ to a threshold that was computed for the probabilistic model so that the observed value is very *rare*. Throughout the paper we assume that the event sequence is generated by a memoryless (Bernoulli) source, i.e., 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 $\Omega^\exists(n,w,m)$ in order to find the threshold. We will prove here that appropriately normalized $\Omega^\exists(n,w,m)$ is normally distributed. We also find the mean and the variance of $\Omega^\exists(n,w,m)$. 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 $\Omega^\exists(n,w,m)$. By assuming that T is infinite, we mean that no matter what window size w we select, there is enough past events available for n consecutive windows.

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

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

where $P^\exists(w,m)$ is the probability that a window of size w contains at least one occurrence of the episode S of size m as a subsequence. The probability of existence $P^\exists(w,m)$ satisfies the following recurrence

$$\begin{cases} P^\exists(w,m) &= (1-p_m)P^\exists(w-1,m) + p_m P^\exists(w-1,m-1) & w>0,\ m>0, \\ P^\exists(w,0) &= 1 & w>0, \\ P^\exists(0,m) &= 0 & m>0, \\ P^\exists(0,0) &= 1 & . \end{cases}$$

Indeed, consider a window of size w. Observe that either the last symbol of the pattern, s_m, does not occur at the w-th position of the window or it does occur. In the former situation S must occur within the window of size $w-1$ leading to the term $(1-p_m)P^\exists(w-1,m)$ of the above recurrence. The latter situation provides the second term of the recurrence.

In Section 4 we solve the above recurrence using generating functions. Then we apply Cauchy's residue theorem to obtain an asymptotic expansion of $P^\exists(w,m)$ for fixed m and large w. We summarize our results in the next theorem.

Theorem 1 *Consider a memoryless source with p_i being the probability of generating the i-th symbol of S and $q_i = 1-p_i$. Let also*

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

- *Then for all m and $w \geq m$ we have*

$$P^\exists(w,m) = P(S)\sum_{i=0}^{w-m}\ \sum_{\sum_{k=1}^{m} n_k = i}\ \prod_{k=1}^{m} q_k^{n_k}. \tag{1}$$

- *Let now m be fixed and assume $i \neq j$ implies $p_i \neq p_j$. Then as $w \to \infty$*

$$P^\exists(w,m) = 1 - P(S)\sum_{i=1}^{m}\frac{(1-p_i)^w}{p_i}\prod_{j\neq i}^{m}\frac{1}{p_j-p_i} + O(r^{-w}) \tag{2}$$

where $r > (1-p_{\max})^{-1}$.

Notice that the asymptotic approximation reveals the anticipated fact about the behavior of $P^{\exists}(w,m)$, i.e., that $P^{\exists}(w,m) = 1$ as $w \to \infty$, and the rate of convergence is exponential. Furthermore, Theorem 1 can be used to identify the maximum value $w_{\max}$ of the window size. It seems reasonable to select $w_{\max}$ so that the pattern S does *not* occur in such a window almost surely or with high probability. Therefore, for a give $\delta \in (0,1)$ we find $w_{\max}$ so that $P^{\exists}(w,m) < 1-\delta$. Since $0 < \delta < 1$ we can use our asymptotic formula (2). We first approximate $P^{\exists}(w,m)$ by its leading term $p_{min} = \min_{0 \le i \le m}\{p_i\}$, that is,

$$P^{\exists}(w,m) \quad \sim \quad 1 - P(S)\frac{(1-p_{min})^w}{p_{min}} \prod_{j \neq j_{min}}^{m} \frac{1}{p_j - p_{min}}$$

This leads to

$$w_{\max} \quad \approx \quad \frac{\log(\delta) - \log\left(\frac{P(S)}{p_{min}} \prod_{j \neq min}^{m} \frac{1}{p_j - p_{min}}\right)}{\log(1-p_{min})}$$

which can be estimated from observed data.

When establishing the above formulas for $P^{\exists}(w,m)$ we also solved two related combinatorial problems on strings that are of independent interest. Namely, given $\mathcal{A}$, w and S:

1. Construct the set $\mathcal{W}^{\exists}(w,m)$ of all windows as strings of length w over $\mathcal{A}$ containing all possible occurrences of S as a subsequence. In Theorem 3 we show the enumeration formula.

2. Find the cardinality of $\mathcal{W}^{\exists}(w,m)$ that we denote $C^{\exists}(w,m)$. In Theorem 4 we prove that

$$C^{\exists}(w,m) \quad = \quad \sum_{k=0}^{w-m} \binom{k+m-1}{k} (|\mathcal{A}|-1)^k |\mathcal{A}|^{w-m-k}.$$

We refer to Section 4 for the solution for those problems.

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

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

where

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

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

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

In order to compute variance of $\Omega^{\exists}(n,w,m)$ we need $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k)$, 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

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

The two terms involving $P^{\exists}_{(I_i^{\exists}\cap I_j^{\exists})}(w,m,k)$ 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. $P^{\exists}_{(I_i^{\exists}\cap I_j^{\exists})}(w,m,k)$ can be computed from Theorem 3.

One concludes, however, that $\mathbf{Var}[\Omega^{\exists}(n,w,m)] \sim n\sigma$ for some $\sigma > 0$. In view of the above, and using the fact that $\Omega^{\exists}(n,w,m)$ is the so called $w-1$-dependent sequence (i.e., $\Omega^{\exists}(n,w,m)$ depends on the last $w-1$ windows), we may apply Theorem 27.5 of [3] to establish the central limit theorem for $\Omega^{\exists}(n,w,m)$.

Theorem 2 *The random variable $\Omega^{\exists}(n,w,m)$ obeys the Central Limit Theorem in the sense that its distribution is asymptotically normal. More precisely, for $a, b = O(1)$ we have*

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

for m and w fixed.

The above findings are foundations for establishing reliable thresholds $\tau_u(w,m)$ or $\tau_\ell(w,m)$ for n large. Let

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

Thus, for a given β we can compute $\tau_u(w,m)$ or $\tau_\ell(w,m)$ by selecting either b_0 such that $\beta = y(b_0,\infty)$ or by selecting a_0 such that $\beta = y(-\infty,a_0)$, respectively. Observe that when a and 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 Experimental Results

The purpose of our experiments was to test the applicability of the analytical results we derived for sources that are apparently not memoryless, i.e., they do not satisfy the assumptions under which the formulas for memoryless source were derived. We of course do not need to test the formulas for memoryless sources, because we already know the equations hold in such cases. Therefore we ran the experiments for an English text source and also for web access data.

An English text is of course not memoryless. As an example consider string "th" which occurs more frequently than "tz" or "ts". So in this example the letter "h" will occur more likely if the previous letter was "t". However, English text can be modeled well by a Markov source.

The web accesses are also not memoryless not only because of hierarchical structure but also because of correlations between links. For example a person looking for a product in an on-line store will most likely visit all manufacturers of the search product.

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, which in our case means our 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.

Thus, the main focus of our experiments was to test how well the formula for $P^{\exists}(w,m)$ works for apparently non memoryless sources. To accomplish this we estimated the actual probability of existence based on the actual number of windows $\Omega^{\exists}(n,w,m)$ as $P_e^{\exists}(w,m) = \frac{\Omega^{\exists}(n,w,m)}{n}$ and compared its value to the computed $P^{\exists}(w,m)$ for different values of w. We used the following error metric d

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

where $w_1 < w_2 < \ldots w_r$ are the tested window sizes.

We used an algorithm, based on dynamic programming, for finding windowed subsequences. We also implement the dynamic programming solution to $P^{\exists}(w,m)$ given in chapter 2. We converted the sources appropriately to a special text file format that was used by the algorithm implemented in C++ and run under Linux.

3.1 English text source

The text source we used is an on-line version of *War and Peace* by Leo Tolstoy from `www.friends-partners.org/newfriends/culture/literature/war_and_peace/war-peace_intro.html`. The work consists of 15 books. Each book has over 20 chapters each of which consists of over 5000 letters. We preprocessed the chapters in order to remove all symbols but 26 letters of the English alphabet without distinguishing between upper and lower case letters.

In the first experiment we compared the analytically computed $P^{\exists}(w,m)$ (cf. Theorem 1) with its estimator $P_e^{\exists}(w,m) = \frac{\Omega^{\exists}(n,w,m)}{n}$. We used chapter 1-5 as a training set for estimation of $p_1, p_2, \ldots p_m$ based on the symbol frequencies. We set $S = gwadera$ and for selected values of $w \in [13, 600]$ we ran the algorithm for finding $\Omega^{\exists}(n,w,m)$ in chapter 6 of length $n = 6881$ as the testing source. Figures 1 and 2 illustrate the results showing two main facts: $P^{\exists}(w,m)$ approaches 1 as w goes to infinity and $P^{\exists}(w,m)$ very closely approximates the actual $P_e^{\exists}(w,m)$ (d is of order 12%).

In the next experiment we demonstrated the application of $\tau_u(w,m)$. To accomplish it we estimated variance of $\Omega^{\exists}(n,w,m)$ denoted $\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w,m)]$. For this purpose we randomly chose 8 chapters and shortened them to the same length $n = 8000$ letters creating 8 training sources denoted $T_1, T_2, \ldots, T_8$ and

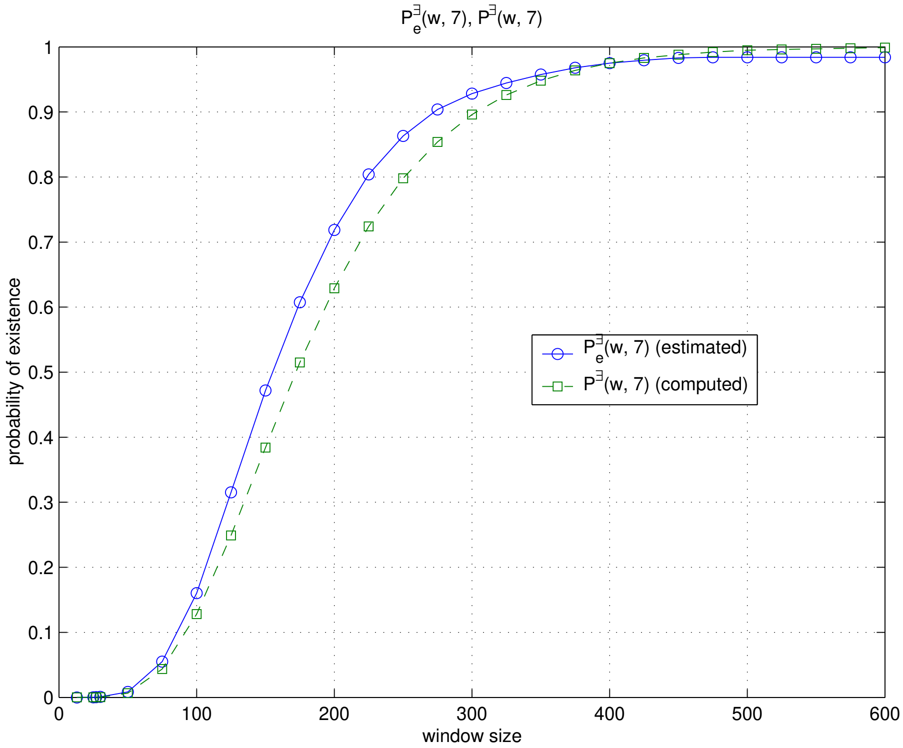


Figure 1: $P_e^{\exists}(w,7) = \frac{\Omega^{\exists}(6881,w,7)}{n}$ and $P^{\exists}(w,7)$ for $S = gwadera$

one testing source T_9. We used the following sample variance estimator

$$\sqrt{\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w,m)]} \quad = \quad \sqrt{\frac{\sum_{i=1}^{8}(\Omega^{\exists}(n,w,m)_i - \mathbf{E}[\Omega^{\exists}(n,w,m)])^2}{8}}.$$

for $S = wojciech$ and $w = 100$. In particular, we set $\tau_u(w,m) = P^{\exists}(w,m) + \frac{5\sqrt{\widehat{\mathbf{Var}}[\Omega^{\exists}(n,w,m)]}}{n}$. We used such a trained model to monitor T_9 as the testing set. To verify our threshold experimentally we artificially kept injecting $S = wojciech$ as a subsequence into different places in T_9. After each insertion we ran the algorithm for finding $\Omega^{\exists}(n,w,m)$ and checked whether we exceeded $\tau_u(w,m)$. To make it more interesting we considered two values of gaps between inserted symbols of S: $gap = 0$ and $gap = 11$. In other words we injected S as $s_1 g^{gap} s_2 g^{gap} \ldots g^{gap} s_m$, where $g \in \mathcal{A}^+$. The results are shown in Figure 3. The horizontal dash-dot line shows $P^{\exists}(100,8) = 3 \cdot 10^{-3}$ for no insertions. The solid line shows $\tau_u(100,8) = 1.45 \cdot 10^{-2}$. We can see that if $gap = 0$, then we need only two episodes to exceed $\tau_u(100,8)$ versus three if $gap = 11$. This makes sense if we notice that if the episode is stretched to the window boundaries ($gap = 11$) then it is more noise-like compared to the case when $gap = 0$, which suggests an intentional action (attack) and should be detected early.

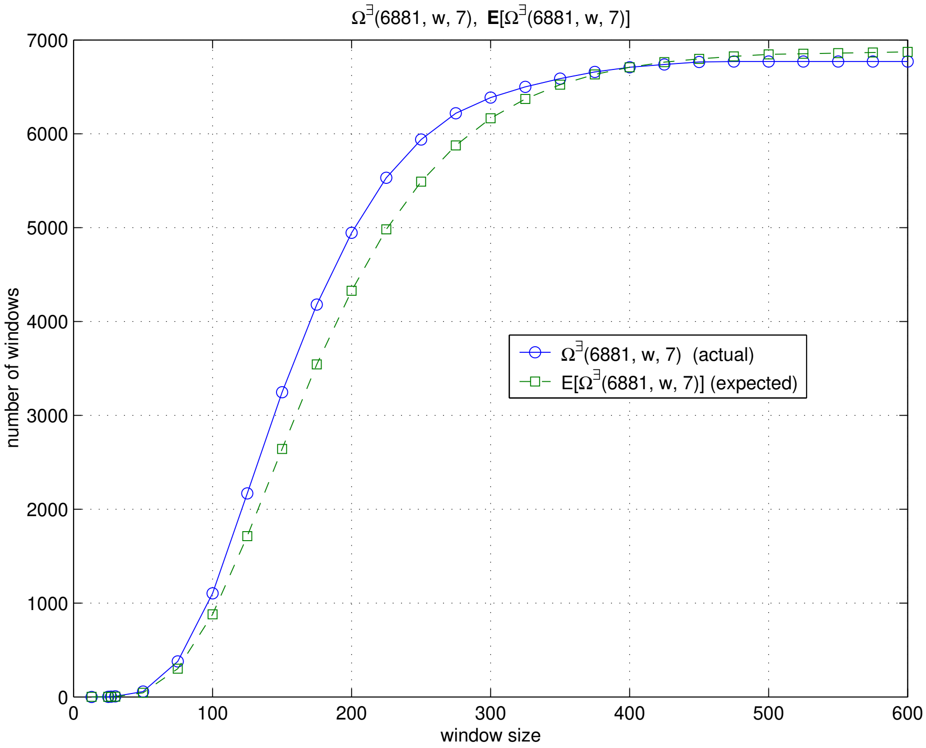


Figure 2: $\Omega^{\exists}(6881, w, 7)$ and $\mathbf{E}[\Omega^{\exists}(6881, w, 7)]$ for $S = gwadera$

3.2 Web access data

We used logs of user accesses to the music machines web site (currently at `http://machines.hyperreal.org`), which record accesses from 1/01/99 through 4/30/99 . The logs have been anonymized with respect to originating machines. That is, in each hit, the IP address of the machine generating the server request has been converted to a random looking number. All hits from one machine on a particular day are labeled with the same number. In the experiments we focused on `http://machines.hyperreal.org/manufacturers/` web page containing links to manufacturers of music instruments. Each link corresponds to an alphabet symbol and the alphabet size was $|A| = 81$. The training and testing sequences were created by considering only unique accesses made by the same originating machine. If a given host made many accesses to the same manufacturer per session then we treated it as one access and consider the first access only.

In the first experiment we compared the computed $P^{\exists}(w, m)$ with its estimator $P_e^{\exists}(w, m) = \frac{\Omega^{\exists}(n,w,m)}{n}$. We created three sources T_1, T_2, T_3 each of length $n = 22000$. The training set established T_1, T_2 and the testing source was T_3. We set $S = \{Akai, ARP, Korg, Moog, Yamaha, Casio, Sequential\}$ and for selected values of $w \in [25, 500]$ we ran the algorithm for finding $\Omega^{\exists}(n, w, m)$ on T_3. Figures 4 and 5 illustrate the results. $P^{\exists}(w, m)$ still provides a good approximation of $P_e^{\exists}(w, m)$ ($d = 14\%$). The reason the value of d is bigger than for the text source is the fact that the web accesses are a more memory dependent source than English text. Therefore the Markov model seems to be more suitable for the web access source.

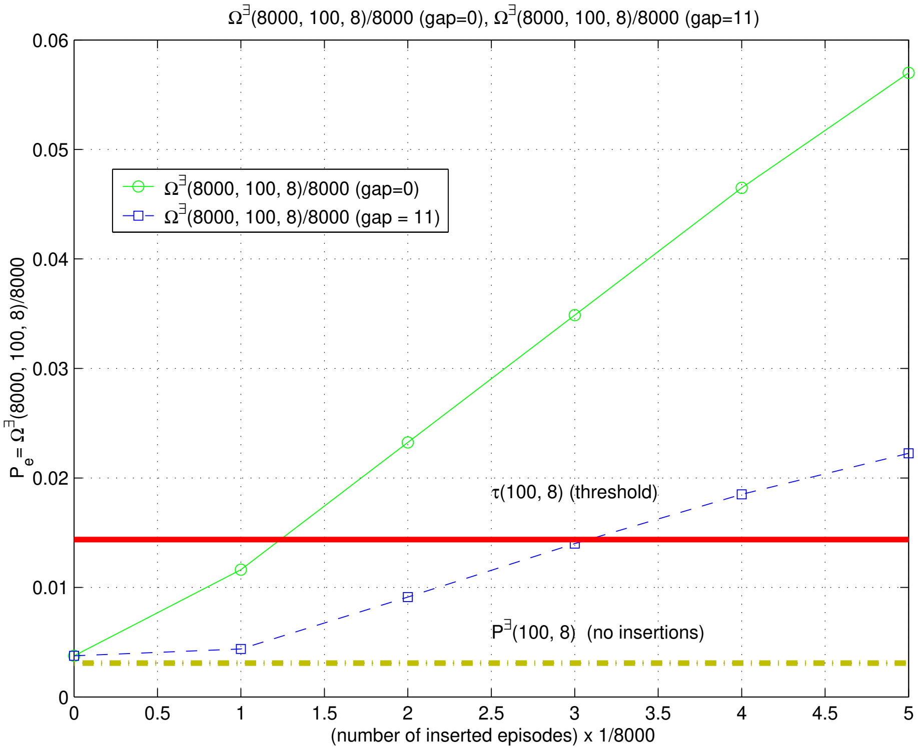


Figure 3: Detection of artificially inserted pattern *wojciech*

4 Derivations of Analytical Results

In this section we provide derivations and proofs of the findings we claimed in Section 2. We also present some new results.

4.1 Set of Windows Containing S as a subsequence

Let $\mathcal{W}^{\exists}(w,m)$ be the *set* of all distinct windows of length w containing S as a subsequence. $P^{\exists}(w,m)$ is therefore equal to the sum of the probabilities of all the elements of $\mathcal{W}^{\exists}(w,m)$, as follows:

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

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

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

We will now show that a recursive formula for enumerating the elements of $\mathcal{W}^{\exists}(w,m)$ has the form below. Recall that the notation $\mathcal{W}^{\exists}(a,b)$ when $b < m$, means the set of windows of size a that contain

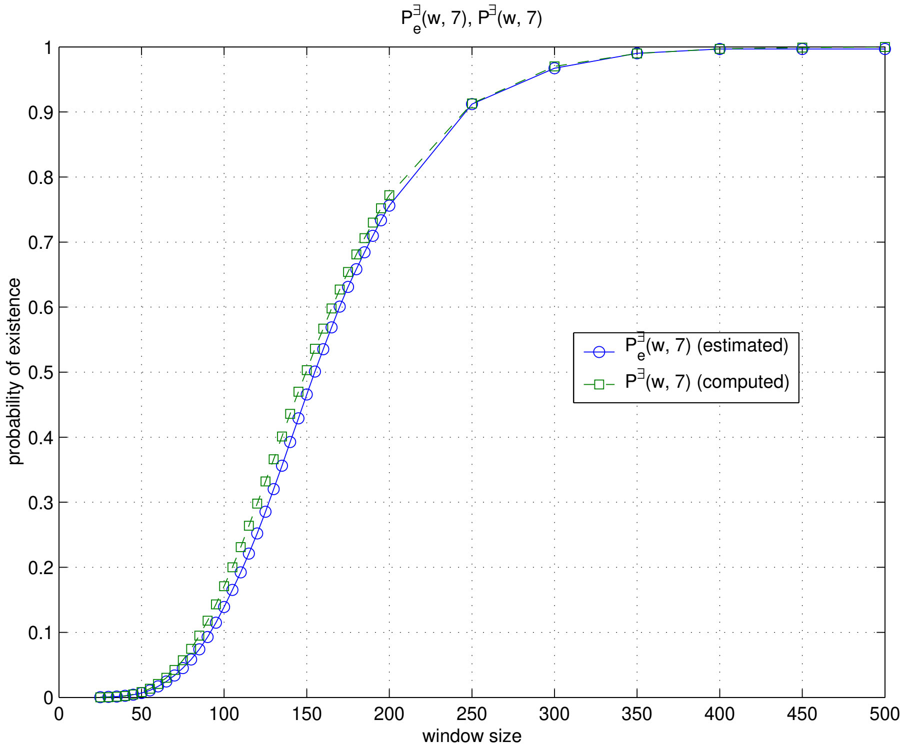


Figure 4: $P_e^\exists(w,7) = \frac{\Omega^\exists(22000,w,7)}{22000}$ and computed $P^\exists(w,7)$ for the web access data

the b-prefix of S (= the string consisting the the first b symbols of S).

$$\begin{cases} \mathcal{W}^\exists(w,m) &= (\mathcal{A} - \{s_m\}) \times \mathcal{W}^\exists(w-1,m) \cup \{s_m\} \times \mathcal{W}^\exists(w-1,m-1) & w > 0 \cap m > 0, \\ \mathcal{W}^\exists(w,0) &= \mathcal{A}^w & w > 0, \\ \mathcal{W}^\exists(0,m) &= 0 & m > 0, \\ \mathcal{W}^\exists(0,0) &= 1. \end{cases}$$

That the elements generated at each level of the recursion are distinct can be seen by noting that we divide $\mathcal{W}^\exists(w,m)$ into two subsets: Strings that have s_m as their last symbol, and strings that have other symbols than s_m as their last symbol. We now turn our attention to showing that we do generate all strings of $\mathcal{W}^\exists(w,m)$. Consider all $\binom{w}{m}$ positions of a window where S may occur as a subsequence. We claim that the recursion considers all positions where the m respective symbols of S can occur, and that it considers these m-tuples of positions in a particular order: Decreasing lexicographic order of those tuples, that is, tuple $(i_1, i_2, \ldots, i_m)$ is considered before tuple $(i_1^{'}, i_2^{'}, \ldots, i_m^{'})$ if the former is lexicographically larger than the latter.

Simply observe that $\mathcal{W}^\exists(w,m)$ can be split into two disjoint subsets:

- Windows having s_m at their last position. Because the last window symbol is fixed as s_m for all of them, their enumeration effectively becomes that of the windows o size $w-1$ that contain the $(m-1)$-prefix of S (= the string consisting the the first $m-1$ symbols of S). This latter enumeration is what we mean by the notation $\mathcal{W}^\exists(w-1,m-1)$.

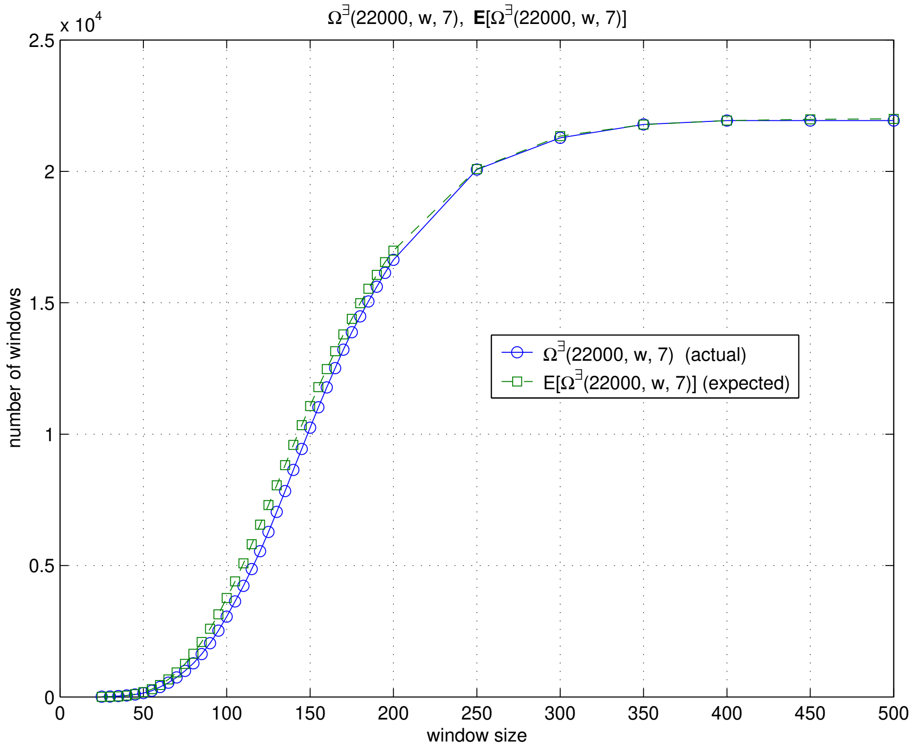


Figure 5: $\Omega^{\exists}(22000, w, 7)$ and $\mathbf{E}[\Omega^{\exists}(22000, w, 7)]$ number of occurrences for the web access data

- Windows not having s_m at their last position. Because the last symbol cannot be considered part of an occurrence of S, their enumeration effectively becomes that of the windows of size $w-1$ that contain S. This latter enumeration is what we mean by the notation $\mathcal{W}^{\exists}(w-1, m)$.

From the above, it is straightforward to obtain the following (we omit the details of the derivation).

Theorem 3 *The set of all distinct windows of length w, that contain a string S of length m as a subsequence, can be enumerated as follows.*

$$\mathcal{W}^{\exists}(w,m) = \bigcup_{\sum_{k=1}^{m+1} n_k = w-m} (\mathcal{A}-s_1)^{n_1} \times \{s_1\} \times (\mathcal{A}-s_2)^{n_2} \times \{s_2\} \times \ldots \times (\mathcal{A}-s_m)^{n_m} \times \{s_m\} \times \mathcal{A}^{n_{m+1}}$$

To visualize the set $\mathcal{W}^{\exists}(w,m)$ we can use an automation. In Figure 6 the members of $\mathcal{W}^{\exists}(w,m)$ are strings consisting of edge symbols of all unique (at least one edge different) paths of length w from the start state, 0 to the ending state, m.

Based on Theorem 3 we can divide the elements of $\mathcal{W}^{\exists}(w,m)$ into equivalence classes $\mathcal{V}^{\exists}$ with respect to the ordered sequences of $(n_1, n_2, \ldots, n_{m+1})$ for which $\sum_{i=1}^{m+1} n_i = w-m$. The number of such ordered partitions is $\binom{w-m+m+1-1}{w-m} = \binom{w}{m}$. It is equivalent to the number of positions of S as a subsequence in the window of length w. Thus $|\mathcal{V}^{\exists}| = \binom{w}{m}$.

Example. Let $\mathcal{A} = \{a, b\}$ $S = ba$ and $w = 3$. We generate $\mathcal{W}^{\exists}(3,2)$ in Table 1 and compute $P^{\exists}(w,m)$. From (3) we obtain $P^{\exists}(3,2) = P(S)(p_a + p_b + p_b + p_a) = 2P(S)$.

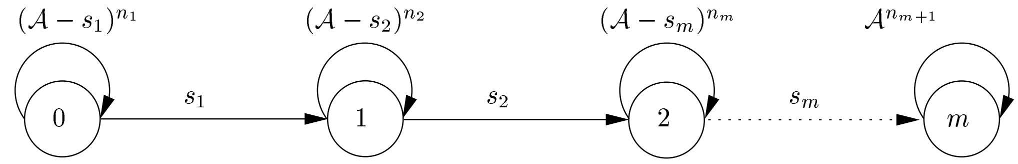


Figure 6: Graphical interpretation of the solution to $\mathcal{W}^{\exists}(w,m)$

i	$W(3,2)[i]$	n_1	n_2	n_3
2	$\underline{ba}a$	0	0	1
3	$\underline{ba}b$	0	0	1
4	$\underline{b}b\underline{a}$	0	1	0
1	$a\underline{ba}$	1	0	0

Table 1: Enumeration of $W(3,2)$ for $\mathcal{A}=\{a,b\}$ and $S=ba$ using Theorem 3

4.2 Evaluation of $C^{\exists}(w,m)$

Recall that $C^{\exists}(w,m)$ denotes the cardinality of $\mathcal{W}^{\exists}(w,m)$.

The recurrence for $C^{\exists}(w,m)$ follows directly form the one for $\mathcal{W}^{\exists}(w,m)$. Namely,

$$\begin{cases} C^{\exists}(w,m) &= (|\mathcal{A}|-1)C^{\exists}(w-1,m)+C^{\exists}(w-1,m-1) & w>0 \cap m>0, \\ C^{\exists}(w,0) &= |\mathcal{A}|^{w} & w>0, \\ C^{\exists}(0,m) &= 0 & m>0, \\ C^{\exists}(0,0) &= 1. & \end{cases}$$

We use the method of generating functions to find the solution for $C^{\exists}(w,m)$. For an in-depth discussion of generating functions see, for example, [16]. We leave m as a free variable and define the following family of generating functions

$$W_m(x) \quad = \quad \sum_{w=0} C^{\exists}(w,m)x^{w}$$

where x is a complex number. From the above recurrence we obtain

$$\begin{cases} W_m(x) &= (|\mathcal{A}|-1)\sum_{w=1} C^{\exists}(w-1,m)x^{w}+\sum_{w=1} C^{\exists}(w-1,m-1)x^{w} & m>0, \\ W_0(x) &= \sum_{w=0} C^{\exists}(w,0)x^{w} & m=0. \end{cases}$$

We now work with $W_m(x)$ for $m>0$.

$$\begin{aligned} W_m(x) &= (|\mathcal{A}|-1)\sum_{w=1} C^{\exists}(w-1,m)x^{w}+\sum_{w=1} C^{\exists}(w-1,m-1)x^{w} \\ &= (|\mathcal{A}|-1)x\sum_{w=1} C^{\exists}(w-1,m)x^{w-1}+x\sum_{w=1} C^{\exists}(w-1,m-1)x^{w-1} \\ &= (|\mathcal{A}|-1)x\sum_{w=0} C^{\exists}(w,m)x^{w}+x\sum_{w=0} C^{\exists}(w,m-1)x^{w} \\ &= (|\mathcal{A}|-1)xW_m(x)+xW_{m-1}(x). \end{aligned}$$

We represent $W_m(x)$ in the form of a first order recurrence with respect to m.

$$\begin{aligned}
W_m(x)(1-(|\mathcal{A}|-1)x) &= xW_{m-1}(x) \\
W_m(x) &= \frac{x}{(1-(|\mathcal{A}|-1)x)}W_{m-1}(x) \\
&= x^m\frac{1}{(1-(|\mathcal{A}|-1)x)^m}W_0(x).
\end{aligned}$$

Using the fact that

$$W_0(x) = \sum_{w=0}|\mathcal{A}|^w x^w = \frac{1}{(1-|\mathcal{A}|x)}$$

we obtain

$$W_m(x) = x^m\frac{1}{(1-(|\mathcal{A}|-1)x)^m}\frac{1}{(1-|\mathcal{A}|x)}.$$

Denoting by $[x^w]f(x)$ the coefficient at x^w of $f(x)$, we find

$$C^{\exists}(w,m) = [x^w]W_m(x).$$

Since

$$[x^w]\frac{1}{(1-(|\mathcal{A}|-1)x)^m} = \binom{w+m-1}{w}(|\mathcal{A}|-1)^w,$$

and

$$[x^w]\frac{1}{(1-(|\mathcal{A}|-1)x)^m}\frac{1}{(1-|\mathcal{A}|x)} = \sum_{k=0}^{w}\binom{k+m-1}{k}(|\mathcal{A}|-1)^k|\mathcal{A}|^{w-k}$$

we finally obtain

$$[x^w]W_m(x) = \sum_{k=0}^{w-m}\binom{k+m-1}{k}(|\mathcal{A}|-1)^k|\mathcal{A}|^{w-m-k}.$$

Theorem 4 *The number of all windows of length w over an alphabet $\mathcal{A}$ which contains at least one occurrence of a pattern of length m does not depend on the symbols of the pattern and is equal to:*

$$C^{\exists}(w,m) = \sum_{k=0}^{w-m}\binom{k+m-1}{k}(|\mathcal{A}|-1)^k|\mathcal{A}|^{w-m-k}.$$

4.3 Evaluation of $P^{\exists}(w,m)$

Recall that $P^{\exists}(w,m)$ is the probability that a window of size w contains at least one occurrence of the episode S of size m as a subsequence. The recurrence for $P^{\exists}(w,m)$ follows directly form the one for $\mathcal{W}^{\exists}(w,m)$. In particular,

$$\begin{cases}
P^{\exists}(w,m) = (1-p_m)P^{\exists}(w-1,m)+p_mP^{\exists}(w-1,m-1) & w>0\cap m>0, \\
P^{\exists}(w,0) = 1 & w>0, \\
P^{\exists}(0,m) = 0 & m>0, \\
P^{\exists}(0,0) = 1. &
\end{cases}$$

As before we use the method of generating functions to find the solution for $P^{\exists}(w,m)$. Let

$$W_m(x) \quad = \quad \sum_{w=0} P^{\exists}(w,m)x^w.$$

From the above recurrence we find

$$\begin{cases} W_m(x) = q_m \sum_{w=1} P^{\exists}(w-1,m)x^w + p_m \sum_{w=1} P^{\exists}(w-1,m-1)x^w & m>0, \\ W_0(x) = \sum_{w=0} P^{\exists}(w,0)x^w & m=0. \end{cases}$$

where $q_m = 1 - p_m$. We now work with $W_m(x)$ for $m > 0$

$$\begin{aligned} W_m(x) &= q_m \sum_{w=1} P^{\exists}(w-1,m)x^w + p_m \sum_{w=1} P^{\exists}(w-1,m-1)x^w \\ &= q_m x \sum_{w=1} P^{\exists}(w-1,m)x^{w-1} + p_m x \sum_{w=1} P^{\exists}(w-1,m-1)x^{w-1} \\ &= q_m x \sum_{w=0} P^{\exists}(w,m)x^w + p_m x \sum_{w=0} P^{\exists}(w,m-1)x^w \\ &= q_m x W_m(x) + p_m x W_{m-1}(x). \end{aligned}$$

We represent $W_m(x)$ in the form of the first order recurrence with respect to m

$$\begin{aligned} W_m(x)(1-q_m x) &= p_m x W_{m-1}(x) \\ W_m(x) &= \frac{p_m x}{(1-q_m x)} W_{m-1}(x) \\ &= \prod_{i=1}^{m} p_i x^m \prod_{i=1}^{m} \frac{1}{(1-q_i x)} W_0(x) \end{aligned}$$

Using the fact that

$$\sum_{w=0} x^w \quad = \quad \frac{1}{(1-x)},$$

we obtain

$$\begin{aligned} W_m(x) &= \prod_{i=1}^{m} p_i x^m \prod_{i=1}^{m} \frac{1}{(1-q_i x)} \frac{1}{(1-x)} \\ &= P(S) x^m \prod_{i=1}^{m} \frac{1}{(1-q_i x)} \frac{1}{(1-x)}. \end{aligned}$$

But $P^{\exists}(w,m) = [x^w]W_m(x)$, and since

$$\prod_{i=1}^{m} \frac{1}{(1-q_i x)} \quad = \quad \prod_{i=1}^{m} \sum_{w=0} q_i^w x^w,$$

and

$$\begin{aligned} [x^w] \prod_{i=1}^{m} \sum_{w=0} q_i^w x^w &= \sum_{\sum_{k=1}^{m} n_k = w} q_1^{n_1} q_2^{n_2} \dots q_m^{n_m} \\ &= \sum_{\sum_{k=1}^{m} n_k = w} \prod_{k=1}^{m} q_k^{n_k} \end{aligned}$$

we use the partial sum property to derive the following

$$[x^w]\prod_{i=1}^{m}\sum_{w=0} q_i^w x^w \frac{1}{(1-x)} \quad = \quad \sum_{i=0}^{w} \sum_{\sum_{k=1}^m n_k = i} \prod_{k=1}^{m} q_k^{n_k},$$

we finally obtain

$$\begin{aligned}[x^w]W_m(x) &= P(S)[x^{w-m}]\prod_{i=1}^{m}\sum_{w=0} q_i^w x^w \frac{1}{(1-x)} \\ &= P(S)\sum_{i=0}^{w-m} \sum_{\sum_{k=1}^m n_k = i} \prod_{k=1}^{m} q_k^{n_k}.\end{aligned}$$

This proves formula (1) in Theorem 1.

4.3.1 Asymptotic approximation of $P^{\exists}(w,m)$

Now we estimate $P^{\exists}(w,m)$ asymptotically as $w \to \infty$ and m fixed, that is, we prove (2) of Theorem 1. In our previous derivations we obtained

$$W_m(z) = P(S)z^m \prod_{i=1}^{m} \frac{1}{(1-q_i z)} \frac{1}{(1-z)}.$$

Observe that the exact value of $P^{\exists}(w,m)$ is equal to the coefficient of $W_m(x)$ at x^w which – we recall – we denote as $[x^w]W_m(x)$. By the Cauchy coefficient theorem (cf. [16]) we know that

$$P^{\exists}(w,m) = [z^w]W_m(z) = \frac{1}{2\pi i}\oint W_m(z)z^{-w-1}dz$$

where z is a complex variable and the integration is over a small circle around $z = 0$. To evaluate this integral we use another Cauchy result known as the Cauchy residue theorem [16]. For this we enlarge the circle around $z = 0$ so that it contains all singularities $W_m(z)$. In our case, the radius r of such a circle must satisfy $r > (1-p_{\max})^{-1}$. Then

$$P^{\exists}(w,m) = -\sum_{p} Res[W_m(z)z^{-w-1}, z = p] + O(r^{-w})$$

where $Res[f(z), z = a]$ is the residue of $f(z)$ at $z = a$. We recall that if $f(z) = \frac{\phi(z)}{\varphi(z)}$, where $\phi(z)$ and $\varphi(z)$ are analytic functions in $z = a$ subject to $\varphi(z) = 0$, $\varphi^{'}(z) \neq 0$ and $\phi(z) \neq 0$, then a is a pole of $f(z)$ and

$$Res\left[\frac{\phi(z)}{\varphi(z)}, z = a\right] \quad = \quad \frac{\phi(a)}{\varphi^{'}(a)}.$$

Therefore,

$$Res[W_m(z)z^{-w-1}, z = 1] = -P(S)1^{m-w-1}\prod_{i=1}^{m}\frac{1}{(1-q_i)} = -1.$$

Similarly for $z = \frac{1}{q_i}$ we have

$$\begin{aligned}Res\left[W_m(z)z^{-w-1}, z = \frac{1}{q_i}\right] &= (-1)\frac{1}{q_i}P(S)\left(\frac{1}{q_i}\right)^{m-w-1}\prod_{j\neq i}^{m}\frac{1}{\left(1-\frac{q_j}{q_i}\right)}\frac{1}{1-\frac{1}{q_i}} \\ &= P(S)\frac{(1-p_i)^w}{p_i}\prod_{j\neq i}^{m}\frac{1}{p_j - p_i}.\end{aligned}$$

Putting everything together we obtain

$$\begin{aligned} P^{\exists}(w,m) &= -Res[W_m(z)z^{-w-1}, z=1] - \sum_{i=0}^{m} Res\left[W_m(z)z^{-w-1}, z=\frac{1}{q_i}\right] \\ &= 1 - P(S)\sum_{i=1}^{m}\frac{(1-p_i)^w}{p_i}\prod_{j\neq i}^{m}\frac{1}{p_j-p_i} + O(r^{-w}). \end{aligned}$$

Finally, we propose a dynamic programming algorithm for computing $P^{\exists}(w,m)$. Let $Q[i,j]$ denote the product $\prod_{k=1}^{j} q_k^{n_k}$ such that $\sum_{k=1}^{j} n_k = i$ then

$$\begin{cases} Q[i,j] &= \sum_{k=0}^{i} Q[i-k,j-1]\cdot q_j^k \quad 1<j\leq m, \quad 1<i\leq w-m \\ Q[i,1] &= q_1^i \\ Q[0,j] &= 1 \qquad\qquad\qquad\qquad\quad 1\leq j\leq m \\ P^{\exists}(w,m) &= P(S)\sum_{i=0}^{w-m} Q[i,m] \end{cases}$$

The time complexity of the algorithm is $O((w-m)^2\cdot m)$ and is equal to the space required to build the table $Q[w-m,m]$. Let $v[a:b]$ denote the substring of a string v between indexes a and b such that $a<b$ and $1\leq a,b\leq w$. Let $p[1:m]$ be an array with probabilities $p_1,p_2,\ldots,p_m$ of the symbols in S.

Algorithm 1: Computation of $P^{\exists}(w,m)$

---

input : w, m, $p[1:m]$
output: $P^{\exists}(w,m)$
begin
  for $j=1$ ***to*** m **do**
    $Q[0,j]=1;$
  $P^{\exists}(w,m)=1;$
  for $i=1$ ***to*** $w-m$ **do**
    $Q[i,1]=(1-p[1])^i;$
    for $j=2$ ***to*** m **do**
      $Q[i,j]=0;$
      for $k=0$ ***to*** i **do**
        $Q[i,j]=Q[i,j]+Q[i-k,j-1]*(1-p[j])^k;$
    $P^{\exists}(w,m)=P^{\exists}(w,m)+Q[i,m];$
  $P^{\exists}(w,m)=P(S)*P^{\exists}(w,m);$
end

4.4 Variance

We need to establish a precise formula for variance $\mathbf{Var}[\Omega^{\exists}(n,w,m)]$. First observe that

$$\mathbf{Var}[\Omega^{\exists}(n,w,m)] = \sum_{i=1}^{n}\mathbf{Var}\left[I_i^{\exists}\right] + 2\sum_{1\leq i<j\leq n}\mathbf{Cov}[I_i^{\exists},I_j^{\exists}],$$

where trivially

$$\mathbf{Cov}[I_i^{\exists}, I_j^{\exists}] \quad = \quad \mathbf{E}[I_i^{\exists}, I_j^{\exists}] - \mathbf{E}[I_i^{\exists}] \cdot \mathbf{E}[I_i^{\exists}]$$

and

$$\mathbf{E}[I_i^{\exists}, I_j^{\exists}] = \begin{cases} 0 & \text{if} \quad |i-j| \geq w \\ P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k) & \text{if} \quad |i-j| < w. \end{cases}$$

This leads to

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

Note that the formula for variance depends on a windowing method, i.e., on all possible configurations of overlaps of considered windows. To compute $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k)$ we define $\mathcal{W}^{\exists}(w,m,k)_{(I_i^{\exists} \cap I_j^{\exists})}$ as the set of all possible pairwise overlapping windows on $k = w - |i-j|$ symbols such that $I_i^{\exists} = 1$, $I_j^{\exists} = 1$ for $i < j$ and $|i-j| < w$. In other words, $\mathcal{W}^{\exists}(w,m,k)_{(I_i^{\exists} \cap I_j^{\exists})}$ can be enumerated as all $2w-k$ length strings consisting of two windows $\mathcal{W}^{\exists}(w,m)[r]$ and $\mathcal{W}^{\exists}(w,m)[q]$ of length w, that overlap on k positions. The overlap is between the last k symbols of $\mathcal{W}^{\exists}(w,m)[r]$ and the first k symbols of $\mathcal{W}^{\exists}(w,m)[q]$ for $1 \leq q,r \leq C^{\exists}(w,m)$.

Let $\mathcal{W}^{\exists}(w,m,k)_{(I_i^{\exists} \cap I_j^{\exists})}[l]$ be the l-th string of $\mathcal{W}^{\exists}(w,m,k)_{(I_i^{\exists} \cap I_j^{\exists})}$. Then we can express $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k)$ as follows

$$P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k) \quad = \quad \sum_{l=1}^{|\mathcal{W}^{\exists}(w,m,k)|} P\left(\mathcal{W}^{\exists}(w,m,k)_{(I_i^{\exists} \cap I_j^{\exists})}[l]\right).$$

Now we present an exact algorithm for computing $P^{\exists}_{(I_i^{\exists} \cap I_j^{\exists})}(w,m,k)$. The idea of the algorithm is to enumerate all pairs of sets of windows, which overlap on k symbols, i.e. the last k symbols of the first set of windows are equal to the first k symbols of the second set of windows. From Theorem 3 it is known that all elements in $\mathcal{W}^{\exists}(w,m)$ can be divided into a set of equivalence classes $\mathcal{V}^{\exists}$. Let $\mathcal{V}^{\exists}[i]$ be the i-th element of $\mathcal{V}^{\exists}$. Our algorithm generates elements in $\mathcal{V}^{\exists}$ and finds all overlaps on k symbols between them.

Let $\mathcal{V}^{\exists}[i]$ and $\mathcal{V}^{\exists}[j]$ be candidates for the overlap

$$\begin{aligned} \mathcal{V}^{\exists}[i] \quad &= \quad \overline{s_1}^{n_1^i} \times s_1 \times \overline{s_2}^{n_2^i} \times s_2 \ldots \overline{s_m}^{n_m^i} \times s_m \times \mathcal{A}^{n_{m+1}^i}, \\ \mathcal{V}^{\exists}[j] \quad &= \quad \overline{s_1}^{n_1^j} \times s_1 \times \overline{s_2}^{n_2^j} \times s_2 \ldots \overline{s_m}^{n_m^j} \times s_m \times \mathcal{A}^{n_{m+1}^j}, \end{aligned}$$

where $\overline{s_i} = \mathcal{A} - s_i$, $i,j \leq |\mathcal{V}^{\exists}|$, $\sum_{r=1}^{m+1} n_r^i = w - m$ and $\sum_{r=1}^{m+1} n_r^j = w - m$.

Algorithm 2: Computation of $P^{\exists}_{(I_i^{\exists}\cap I_j^{\exists})}(w,m,k)$ for $\mathbf{Var}[\Omega^{\exists}(n,w,m)]$

input : w, m, k, $P(a_1), P(a_2), \ldots, P(a_{|\mathcal{A}|})$

output: $P^{\exists}_{(I_i^{\exists}\cap I_j^{\exists})}(w,m,k)$

begin

  for $i = 1$ ***to*** $\binom{w}{m}$ **do**

    for $j = 1$ ***to*** $\binom{w}{m}$ **do**

      $Overlap[1:k] = \mathcal{V}^{\exists}[i][w-k+1:w] \cap \mathcal{V}^{\exists}[j][1:k]$;

      if $Overlap[1:k] \neq \emptyset$ **then**

        $P^{\exists}_{(I_i^{\exists}\cap I_j^{\exists})}(w,m,k) += P(\mathcal{V}^{\exists}[i][1:w-k]) * P(Overlap[1:k]) * P(\mathcal{V}^{\exists}[j][k+1:w]))$;

end

Note that unlike Algorithm 1 for $P^{\exists}(w,m)$, which is fast even for large w Algorithm 2, is not practical for large w. When w is large we use the sample variance estimator. An alternative approach for large w is to improve Algorithm 2 by cutting the search space (through elimination of certain pairs of sets of windows).

5 Conclusions and extensions

We presented an exact formula for $P^{\exists}(w,m)$, the probability that an episode S of length m occurs in a window of length w in an event sequence T over the alphabet $\mathcal{A}$ for the memoryless model. In addition we gave an asymptotic approximation of $P^{\exists}(w,m)$, which shows that, for appropriately large w, $P^{\exists}(w,m)$ asymptotically tends to one as expected. By providing an efficient dynamic programming method for computing $P^{\exists}(w,m)$ we showed its applicability to real-time monitoring systems. In the experiments, we chose two apparently non-memoryless sources (the English alphabet and the web access data) and showed that, even for these cases, $P^{\exists}(w,m)$ closely approximated the estimated $P^{\exists}_e(w,m)$. This seems to be yet another one of those intriguing situations where an equation derived under a certain set of assumptions holds in practical examples for which those assumptions are clearly violated. Based on the formula for $P^{\exists}(w,m)$ we proposed a reliable episode detection method, where as a measure of normal behavior we used $\Omega^{\exists}(n,w,m)$, the number of windows which contain at least one occurrence of a defined "bad" episode. Reliability of $\Omega^{\exists}(n,w,m)$ as a normal-behavior measure stems from the fact that, for a given S and $\mathcal{A}$, we can analytically select the window length w to minimize false alarms. We proved that $\Omega^{\exists}(n,w,m)$ has the Gaussian distribution. Knowing $\mathbf{E}[\Omega^{\exists}(n,w,m)]$, $\mathbf{Var}[\Omega^{\exists}(n,w,m)]$ or their estimates, and for a given confidence level β, we showed how to set the upper threshold $\tau_u(w,m)$ and the lower threshold $\tau_\ell(w,m)$. In experiments we tested $\tau_u(w,m)$ by artificially injecting bad episodes into the testing source and observed that $\tau_u(w,m)$ did indeed provide a sharp detection of intentional (bad) episodes.

An obvious extension of this work is to use Theorem 3 to compute $P^{\exists}(w,m)$ for Markov source of any order. Let $\mathcal{W}^{\exists}(w,m)[i][j]$ be the j-th symbol in $\mathcal{W}^{\exists}(w,m)[i]$ where $j = 1,2,\ldots,w$. Then for the first order source the probability that a window of length w contains at least one occurrence of a pattern S of length m is equal to:

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

where $\mathcal{W}^{\exists}(w,m)[i][l]$ is the l-the symbol of the i-th member of $\mathcal{W}^{\exists}(w,m)$ in lexicographic order and $P(\mathcal{W}^{\exists}(w,m)[i][2]|\mathcal{W}^{\exists}(w,m)[i][1])$ is the conditional probability. This approach is, however, not very computationally efficient.

References

[1] A. Aho and M. Corasick (1975), Efficient String Matching: An Aid to Biblographic Search *Programming Techniques.*

[2] A. Apostolico and M. Atallah (2002), Compact Recognizers of Episode Sequences, *Information and Computation*, 174, 180-192.

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

[4] L. Boasson, P. Cegielski, I. Guessarian, and Y. Matiyasevich (1999), Window-Accumulated Subsequence Matching Problem is Linear, *Proc. PODS*, 327-336.

[5] M. Crochemore and W. Rytter (1994), *Text Algorithms*, Oxford University Press, New York.

[6] G. Das, R. Fleischer, L. G asieniec, D. Gunopulos, and J. Kärkkäinen (1997), Episode Matching, In *Combinatorial Pattern Matching, 8th Annual Symposium*, *Lecture Notes in Computer Science* vol. 1264, 12–27.

[7] P. Flajolet, Y. Guivarc'h, W. Szpankowski, and B. Vallée (2001), Hidden Pattern Statistics, ICALP 2001, Crete, Greece, LNCS 2076, 152-165.

[8] G. Kucherov, M. Rusinowitch (1997), Matching a Set of Strings with Variable Length Don't Cares, *Theoretical Computer Science* 178, 129–154.

[9] S. Kumar and E.H. Spafford (1994), A Pattern-Matching Model for Intrusion Detection, *Proceedings of the National Computer Security Conference*, 11–21.

[10] H. Mannila, H. Toivonen, and A. Verkamo (1997), Discovery of frequent episodes in event sequences *Data Mining and Knowledge Discovery*, 1(3), 241-258.

[11] P. Nicodème, B. Salvy, and P. Flajolet (1999), Motif Statistics, *European Symposium on Algorithms*, *Lecture Notes in Computer Science*, No. 1643, 194–211.

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

[13] M. Régnier and W. Szpankowski (1998), On pattern frequency occurrences in a Markovian sequence *Algorithmica*, 22, 631-649.

[14] I. Rigoutsos, A. Floratos, L. Parida, Y. Gao and D. Platt (2000), The Emergence of Pattern Discovery Techniques in Computational Biology, *Metabolic Engineering*, 2, 159-177.

[15] R. Sedgewick and P. Flajolet (1995), *An Introduction to the Analysis of Algorithms*, Addison-Wesley, Reading, MA.

[16] W. Szpankowski (2001), *Average Case Analysis of Algorithms on Sequence*, John Wiley, New York.

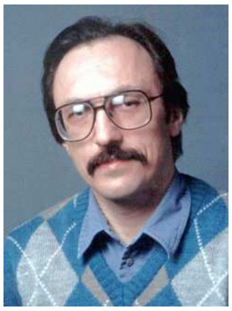
[17] M. Waterman (1995), *Introduction to Computational Biology*, Chapman and Hall, London.

[18] A. Wespi, H. Debar, M. Dacier, and M. Nassehi (2000), Fixed vs. Variable-Length Patterns For Detecting Suspicious Process Behavior, *J. Computer Security*, 8, 159-181.

[19] S. Wu and U. Manber (1995), Fast Text Searching Allowing Errors, *Comm. ACM*, 35:10, 83–991.

Robert Gwadera received the M.S. degree in Electrical and Computer Engineering from Technical University of Gdansk, Poland in 1995. In 2003 he received the M.S. in Computer Sciences from Purdue University. He is currently working towards his PhD in Computer Science at Purdue University. His research interests are data mining, databases and information security.

Mikhail ("Mike") Atallah obtained his PhD from the Johns Hopkins University in 1982, and joined the Purdue University Computer Science Department where he was promoted to Associate Professor in 1986, to Professor in 1989, and to Distinguished Professor in 2004. His current research interests are in information security (in particular, software security, secure protocols, and watermarking). He received a Presidential Young Investigator Award from the National Science Foundation in 1985. A Fellow of the IEEE, he has served on the editorial boards of SIAM Journal on Computing, IEEE Transactions on Computers, and many other journals, and has also served on the Program Committees of many conferences and workshops. He was Keynote and Invited Speaker at many national and international meetings. In June 2001 he co-founded Arxan Technologies Inc., a startup in the software security products space, that has secured funding from top-tier venture capital firms.



Wojciech Szpankowski received the M.S. degree and the PhD degree in Electrical and Computer

Engineering from Technical University of Gdansk, Poland in 1976 and 1980, respectively. Currently, he is Professor of Computer Science at Purdue University. During 1992/1993 he was a Professeur Invite INRIA, France, in the Fall of 1999 he was a Visiting Professor at Stanford University. His research interests cover design and analysis of algorithms, bioinformatics and multimedia compression (information theory). He has published over 170 papers on these topics. In 2001 he published his book "Average Case Analysis of Algorithms on Sequences", John Wiley & Sons. He has been a guest editor for several journals. He is managing editor of "Theoretical Computer Science and Discrete Mathematics", and he is on the editorial boards of "Theoretical Computer Science" and " Foundation and Trends in Communications and Information Theory". In 2003 he chaired NSF Workshop on Information Theory and Computer Science Interface, Chicago. He is Fellow of the IEEE.