New Generation Computing, 25(2007)61-93
Ohmsha, Ltd. and Springer

Efficient Discovery of Unusual Patterns in Time Series

Stefano LONARDI
Department of Computer Science & Engineering
University of California
Riverside, CA 92521, USA
stelo@cs.ucr.edu
Jessica LIN
Department of Information and Software Engineering
George Mason University
Fairfax, VA 22030, USA
jessica@ise.gmu.edu
Eamonn KEOGH and Bill 'Yuan-chi' CHIU
Department of Computer Science & Engineering
University of California
Riverside, CA 92521, USA
{eamonn,ychiu}@cs.ucr.edu

Received 15 March 2006
Revised manuscript received 15 June 2006

Abstract The problem of finding a specified pattern in a time series database (i.e., query by content) has received much attention and is now a relatively mature field. In contrast, the important problem of enumerating all surprising or interesting patterns has received far less attention. This problem requires a meaningful definition of "surprise", and an efficient search technique. All previous attempts at finding surprising patterns in time series use a very limited notion of surprise, and/or do not scale to massive datasets. To overcome these limitations we propose a novel technique that defines a pattern surprising if the frequency of its occurrence differs substantially from that expected by chance, given some previously seen data. This notion has the advantage of not requiring the user to explicitly define what is a surprising pattern, which may be hard, or perhaps impossible, to elicit from a domain expert. Instead, the user gives the algorithm a collection of previously observed "normal" data. Our algorithm uses a suffix tree to efficiently encode the frequency of all observed patterns and allows a Markov model to predict the expected frequency of previously unobserved patterns. Once the suffix tree has been constructed, a measure of surprise for all the patterns in a new database can be determined in time and space linear in the size of

the database. We demonstrate the utility of our approach with an extensive experimental evaluation.

Keywords: Time Series, Suffix Tree, Novelty Detection, Anomaly Detection, Markov Model, Feature Extraction.

§1 Introduction

The problem of efficiently locating previously defined patterns in a time series database (i.e., query by content) has received much attention and may now be essentially regarded as a solved problem (e.g., References[1, 11, 16, 21, 29, 31, 32, 52, 61]).

In contrast, the problem of enumerating all surprising or interesting patterns has received far less attention. The utility of such an algorithm is quite obvious. It would potentially allow a user to find surprising patterns in a massive database without having to specify in advance what a surprising pattern looks like. This problem should not be confused with the relatively simple problem of outlier detection. Hawkins' classic definition of an outlier is "... an observation that deviates so much from other observations as to arouse suspicion that it was generated from a different mechanism".[23]

Here, however, we are not interested in finding individually surprising datapoints. Instead we are looking for surprising *patterns*, i.e., combinations of datapoints whose structure and frequency somehow defies our expectations. The problem is referred to under various names in the literature, including novelty detection[13] and anomaly detection.[58]

The problem clearly requires a meaningful definition of "surprise". The literature contains several such definitions for time series; however they are all too limited for a useful data-mining tool. Consider, for example, the notion introduced by Shahabi *et al.*,[51] which define *surprise* in time series as "*...sudden changes in the original time series data, which are captured by local maximums of the absolute values of (wavelet detail coefficients)*". However, it is not difficult to think of very surprising patterns that defy this rule.

For example, consider the time series in Fig. 1. Here the beginning of each normal heartbeat is considered very surprising, but the temporary absence of a heartbeat is considered to be the least surprising subsection of the time series!

Several other definitions of surprise for time series exist, but all suffer from similar weaknesses.[10, 13, 58, 59] To overcome these limitations we propose a novel definition that defines a pattern surprising if the frequency of its occurrence differs substantially from that expected by chance, given some previously seen data. This notion has the advantage of not requiring the user to explicitly define what is a surprising pattern, which may in any case be impossible to elicit from a domain expert. Instead, the user gives the algorithm a collection of previously observed data, which is considered "normal". The measure of surprise of a newly observed pattern is considered relative to this data collection, and thus eliminates the need for a specific model of normal behavior.

Note that unlike all previous attempts to solve this problem, the measure

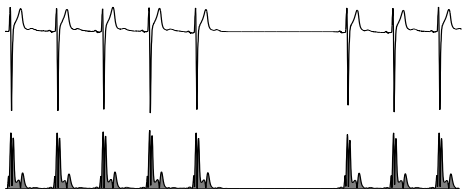

Fig. 1 The time series at the top is a normal healthy human electrocardiogram with an artificial "flatline" added. The sequence at the bottom indicates how surprising local subsections of the time series are under the measure introduced in Shahabi *et al.*. In this case the beginning of each normal heartbeat is very surprising, but the "flatline" is the least surprising part of the time series, a very unintuitive result

of surprise of a pattern is not tied exclusively to its structure. Instead, it depends on the departure of the frequency of the pattern from its expected frequency. This is the crucial distinction of our approach from all the others.

For example consider the "head and shoulders" pattern shown in Fig. 2. The existence of this pattern in a stock market time series should not be considered surprising since it is known to occur (even if only by chance). However, if it occurred ten times this year, as opposed to occurring an average of twice a year in previous years, our measure of surprise will flag the pattern as being surprising. Intuitively, the pattern would also be surprising if its frequency of occurrence is less than expected. Once again our definition would flag such patterns.

Our definition of surprise would be of little utility without a technique that allowed efficient determination of the expected frequency of a pattern. We demonstrate how a suffix tree can be used to efficiently encode the frequency of all observed patterns. Since it is possible that a pattern observed in the new data was not observed in the training data, we propose a technique based on Markov models to calculate the expected frequency of previously unobserved patterns. Once the suffix tree has been constructed, the measure of surprise for all the patterns in a new database can be determined in linear time and linear space with respect to the size of the database.

The rest of the paper is organized as follows. In Section 2 we introduce some notations needed in the rest of the document. In Section 3 we discuss

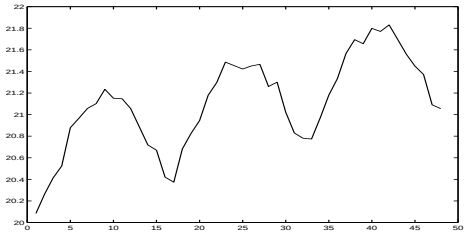

Fig. 2 An Example of the Classic "Head and Shoulders" Pattern

techniques to discretize time series, a necessary preprocessing step for our approach. In Sections 4 and 5 we introduce Markov models and we show how to encode them in suffix trees. In Section 6 we show how to efficiently compute the expected frequency of patterns by using suffix trees. We present our algorithm in Section 7. In Section 8 we demonstrate the utility of our approach with detailed empirical evaluations. We wait until Section 9, when the reader's intuitions about the problem are more fully developed to discuss related work. Finally, in Section 10 we offer conclusions and directions for future research. With respect to conference papers[34, 41] where we first reported the preliminary results presented here, we have expanded the section on experimental results, but more importantly we carefully detailed the linear-time algorithm using the suffix tree and proved its time complexity.

§2 Notation

We use Σ to denote a nonempty *alphabet* of *symbols*. We will set $a = |\Sigma|$ to be the cardinality of the alphabet. A *string* over Σ is an ordered sequence of symbols from the alphabet. Given a string x, the number of symbols in x defines the *length* $|x|$ of x. Henceforth, we will use the variable n to denote the length of x, i.e., $|x| = n$.

Let us decompose a text x into uvw, i.e., $x = uvw$ where u, v and w are strings over Σ. Strings u, v and w are called *substrings*, or *words*, of x. Moreover, u is called a *prefix* of x, and w is called a *suffix* of x.

We write $x_{[i]}$, $1 \leq i \leq |x|$, to indicate the i-th symbol in x. We use $x_{[i,j]}$ as shorthand for the substring $x_{[i]}x_{[i+1]} \ldots x_{[j]}$ where $1 \leq i \leq j \leq n$, with the convention that $x_{[i,i]} = x_{[i]}$. Substrings in the form $x_{[1,j]}$ corresponds to the prefixes of x, and substrings in the form $x_{[i,n]}$ to the suffixes of x.

We say that a string y has an *occurrence* at position i of a text x if $y_{[1]} = x_{[i]}$, $y_{[2]} = x_{[i+1]}$, $\ldots$, $y_{[m]} = x_{[i+m-1]}$, where $m = |y|$. For any substring y of x, we denote by $f_x(y)$ the number of occurrences of y in x.

Given a set of strings $\mathcal{X} = \{x_1, x_2, \ldots, x_k\}$, $k > 1$, the *colors* of y are the members of the subset of $\mathcal{X}$ such that each contains at least one occurrence of y. The number of colors of y is denoted by $c(y)$.

Throughout this document, variables y and w usually indicate substrings of the text x. Unless otherwise specified, we assume the generic term m as the length of any of these words.

We denote by X the time series database $X = X_1X_2 \ldots X_k$ under study, where each X_i is real number and k is the length of the time series. We use variable R to refer to the *reference* time series database, that is to say, the time series which has been annotated as normal, or "unsurprising" by a user.

§3 Dimensionality Reduction and Discretization

For concreteness, we recall the notion that defines the level of surprise.

Definition 3.1
A time series pattern P, extracted from database X is *surprising* relative to

a database R, if the frequency of its occurrence is greatly different from that expected by chance, assuming that R and X are created by the same underlying process.

In order to compute this measure, we must calculate the probability of occurrence for the patterns of interest. Here we run into the familiar paradox that the probability of a particular real number being chosen from any distribution is zero[19]. Since a time series is an ordered list of real numbers, the paradox clearly applies. The obvious solution is to discretize the time series into some finite alphabet Σ. Using a finite alphabet allows us to avail of Markov models to estimate the expected probability of occurrence of a previously unseen pattern. The problem of discretizing (symbolizing, tokenizing, quantizing) time series into a finite alphabet has received much attention in diverse fields, including astronomy, medicine, chemistry, etc. (see e.g., Reference[14] for an exhaustive overview). The representation has also captured the attention of the data mining community who use discretized time series to support similarity search[24] and to enable change point detection.[21]

While there are literally hundreds of papers on discretizing time series, none of the techniques are suitable for our purposes. In particular, the following limitations thwart our efforts to create a time/space efficient and robust algorithm.

- Most symbolic approximations require access to the entire dataset, before the approximation can be created.[21,24] This is untenable for large datasets, and impossible if we wish to monitor streaming data.
- Almost all time series datasets are very high dimensional. This is a challenge because all non-trivial data mining algorithms degrade exponentially with dimensionality. None of the symbolic representations that we are aware of allow dimensionality reduction.[2,21,24] There is some reduction in the storage space required, since fewer bits are required for each value; however, the intrinsic dimensionality of the symbolic representation is the same as the original data.
- For many of the symbolic approximations in the literature, it is not clear that the approximation accurately reflects the true shape and structure of the original time series. This is in contrast to real valued approximation techniques such as wavelets[11] and Fourier transforms,[16,32] in which the fidelity of approximation can be explicitly quantified (by measuring the reconstitution error).

In contrast to the above, we describe novel symbolic representation that can be calculated with single pass over the data. The approximation allows dimensionality reduction, and its fidelity to the original sequence can be meaningfully visualized and qualified. We call our representation SAX (**S**ymbolic **A**ggregate Appro**X**imation).

Our discretization technique is not applied to the entire time series. Since we are interested in locally surprising patterns, we convert the real-valued time

series to a symbolic representation at a local scale. We achieve this by extracting subsequences with a sliding window, performing the conversion, and concatenating the resultant symbols back into global strings r and x. Sliding windows are commonly used in time series data mining, for example, for indexing,[11,32,61] for rule discovery[12] and for anomaly detection.[13]

The technique presented in the following allows a time series of arbitrary length k to be reduced to a string of arbitrary length n, ($n < k$, typically $n \ll k$). As an intermediate step between the original time series and our discrete representation of it, a dimensionality-reduced version of the data will be created. We will utilize the Piecewise Aggregate Approximation (PAA),[31,61] which we review in the next paragraph.

Dimensionality Reduction. First, the time series X of length k is encoded in a n-dimensional space by a vector $\bar{X} = [\bar{x}_1, \ldots, \bar{x}_n]$. The i-th element of $\bar{X}$ is calculated by the following equation

$$\bar{x}_i = (n/k) \sum_{j=(i-1)(k/n)+1}^{i(k/n)} x_j \tag{1}$$

In order to reduce the time series from k dimensions to n dimensions, the data is divided into n equal sized "frames". The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. The representation can be visualized as an attempt to approximate the original time series with a linear combination of box basis functions as shown in Fig. 3.

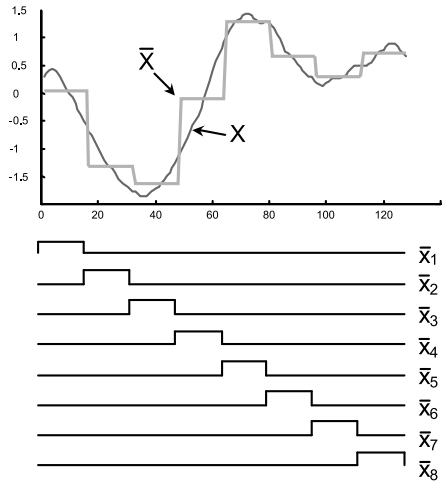

Fig. 3 The PAA representation can be readily visualized as an attempt to model a sequence with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions

The PAA dimensionality reduction is intuitive and simple, yet has been shown to rival more sophisticated dimensionality reduction techniques like Fourier transforms and wavelets.[11,31,61] In addition it has several advantages over its rivals, including being much faster to compute, and being able to support many different distance functions, including weighted distance functions,[36] arbitrary Minkowski norms,[61] and even dynamic time warping.[21,47]

Discretization. Having transformed a time series database into the PAA, we can apply a further transformation to obtain a discrete representation. Among all possible mappings from real numbers to symbols, we require a discretization that will produce symbols with equiprobability. If we did not enforce a symmetric distribution, this would imply that some words generated by this source would be very frequent and some would be very rare simply because they contain an occurrence of a frequent or a rare symbols. More precisely, strings should be flagged when there are surprising higher-order correlations between adjacent symbols. Enforcing each symbol to have the same probability also maximizes the entropy and the information encoded on a limited alphabet.

The objective of having each symbol to be equiprobable is easily achieved since we observed experimentally that normalized time series have a Gaussian distribution. We illustrate this observation in Fig. 4, where we took a pool of 64 diverse time series datasets from the UCR time series archive, and randomly extracted 10,000 subsequences of length 128 from it. We then plotted them in a normal probability plot, a classic statistical tool used to visually evaluate if a data source comes from a normal distribution. If the data is Gaussian, we expect to see that the data points approximately follow the diagonal line in the plot, which is exactly what we observe.

Given that the normalized time series have Gaussian distribution, we can determine the "breakpoints" that will produce equal-sized areas under Gaussian curve.

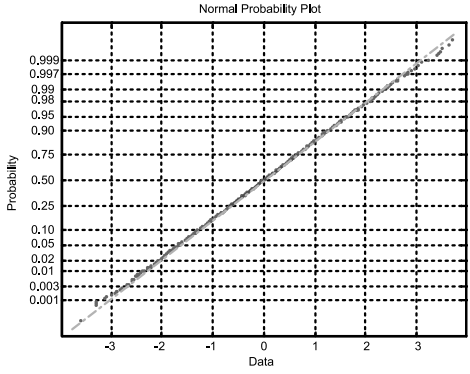

Fig. 4 A normal probability plot of the distribution of values from subsequences of length 128 from 64 different datasets. The highly linear nature of the plot strongly suggests that the data came from a Gaussian distribution

Definition 3.2
Breakpoints are a sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a N(0,1) Gaussian curve from β_i to β_{i+1} is $1/a$ (β_0 and β_a are defined as $-\infty$ and $+\infty$, respectively).

These breakpoints may be determined by looking them up in a statistical table. For example, Table 1 gives the breakpoints for values of a from 3 to 10.

Once the breakpoints have been obtained, we can discretize a time series in the following manner. We first obtain a PAA of the time series. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol `a`, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol `b`, etc. Figure 5 illustrates the idea.

Note that in this example the three symbols, `a`, `b` and `c` are approximately equiprobable as we desired. We call the concatenation of symbols that represent a subsequence a *word*.

The use of fixed breakpoints for all datasets may seem unintuitive. It might be argued that one could adaptively learn the best location for the breakpoints by computing the empirical Probability Density Function (PDF) of the observed values and dividing this PDF into k bins of equal probability mass. In fact, this was our original plan. However, to our surprise we found that the

Table 1 A lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions

a	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

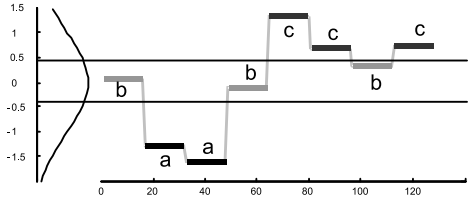

Fig. 5 A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into symbols. In the example above, with $k = 128, n = 8$ and $a = 3$, the time series is mapped to the word `baabccbc`

learned PDF was always very close to a Gaussian for a vast range of datasets, including data from medicine, finance, robotics, meteorology, networking, human motion, music and space telemetry. Nevertheless, an adaptive version of SAX remains a possibility for a pathological dataset that warrants it.

Definition 3.3
Let α_i denote the i-th element of the alphabet, i.e., $\Sigma = \{\alpha_1, \alpha_2, \ldots\}$. Given a time-series X of length k, and its PAA approximation $\bar{X}$, the corresponding word $x = x_{[1]}x_{[2]} \ldots x_{[n]}$ is obtained as follows:

$$x_{[i]} = \alpha_j, \quad \text{iff} \quad \beta_{j-1} \leq \bar{x}_i < \beta_j.$$

We have now defined the two representations required for our pattern discovery (the PAA representation is merely an intermediate step required to obtain the symbolic representation).

As noted earlier, for most symbolic approaches in the literature, it is not clear how well these discrete representations approximate the true structure of the original time series. Since we are ultimately interested in surprising patterns in the original time series, this is a critical issue. We can evaluate the quality of the SAX representation in several ways. As a simple sanity check, we can visually compare it to the four most common real valued representations in the literature.[32] Figure 6 shows one such comparison. To make this comparison fair, we ensured that the number of bits used by all five representations is exactly the same. While this is a subjective test, SAX does seem to be at least competitive with the other approaches.

An objective test can be designed as follows. We begin by defining a distance measure between two time series in the SAX space.

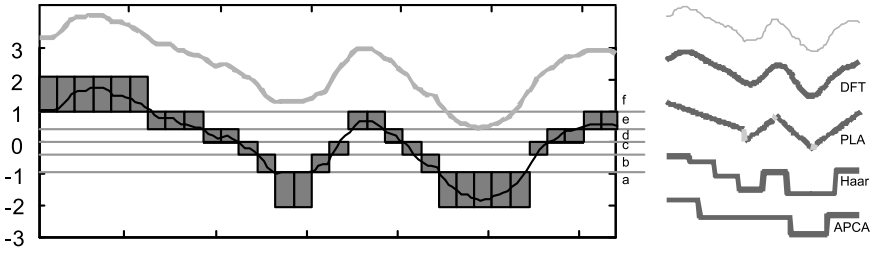

Fig. 6 A visual comparison of our approach and the four most common time series data mining representations. (left) A raw time series of length 128 is transformed into the word **ffffffeeeddcbaabceedcbaaaaacddee**. (right) The four most common time series data mining representations. From bottom to top; Adaptive Piecewise Constant Approximation, Haar wavelet, Piecewise Linear Approximation, Discrete Fourier Transform.

Table 2 The error rate of one-nearest-neighbor on 9 time series classification problems using Euclidean distance and 4 symbolic representations of time series. The fact that SAX has an error rate that is far lower than its symbolic rivals, and almost indistinguishable from that obtained on the raw data strongly suggests that SAX is accurately capturing the underlying structure of data, a fact which confirms it choice as the symbolic representation to support anomaly detection.

	Euclidean	SAX	IMPACTS	SDA	SLS
Word Spotting	4.78	6.80	25.75	34.87	45.33
Sign language	28.70	27.62	50.30	56.23	34.21
GUN	5.50	5.92	34.92	44.33	48.91
Nuclear Trace	11.00	11.00	21.12	43.95	38.42
Leaves	33.26	35.91	70.76	59.83	60.23
4-Faces	6.25	7.27	45.64	47.95	37.54
Control Chart	7.5	7.3	54.2	39.1	14.12
2-Patterns	1.04	1.05	23.5	34.6	04.74
CBF	0.16	0.12	41.2	53.5	05.23

$$\textsc{SaxDist}(\bar{Q},\bar{C}) \equiv \sqrt{(k/n)\sum_{i=1}^{n}(dist(\bar{q}_i,\bar{c}_i))^2}, \qquad (2)$$

where $dist(q,c)$ can be determined by examining a lookup table (see e.g.,[41]). The distance measure SAXDIST is designed to be an approximation of the Euclidean metric, the distance measure of choice for the vast majority of time series data mining and indexing algorithms (see, e.g., References[9,11,12,16,31,32,36,56,61]).

While we can directly measure the fidelity of SAX to the original signal (by measuring the reconstruction error), the other discretization methods do not have a way to directly measure their fidelity. However they all have distance measures defined on them, so we can indirectly measure the fidelity of their approximation by considering their classification error rates. The idea is that no matter what features the discretization method was specialized in extracting (i.e., the extrema, or the rate of change, etc.), we would hope that these features would be strongly correlated with the class labels on at least some problems.

We measured the error rates using leaving-one-out evaluation with the one nearest neighbor algorithm on all the classification datasets in the UCR archive. For fairness, each of the symbolic approaches was compressed to one 1/16 the size of the original data. We compared to the Euclidean distance on the raw data, since this simple approach has been shown to be an extremely competitive technique.[33] Table 2 contains the results.

Note that the results for SAX and the full raw data are not significantly different, but that the three other symbolic approaches perform very badly, often close to the default rate. These results strongly suggest that using SAX we can capture the "essence" of the time series of interest.

§4 Markov Models

We now turn the attention to the probabilistic model that will be used to

compute the expected number of occurrences of each pattern. Consider strings generated by a stationary Markov chain of order $M \geq 1$ on the finite alphabet Σ. Let $x = x_{[1]}x_{[2]}\ldots x_{[n]}$ be an observation of such a random process and $y = y_{[1]}y_{[2]}\ldots y_{[m]}$ an arbitrary but fixed pattern over Σ with $m < n$.

The stationary Markov chain is completely determined by its *transition* matrix

$$\Pi = (\pi(y_{[1,M]}, c))_{y_{[1]},\ldots,y_{[M]},c\in\Sigma}$$

where

$$\pi(y_{[1,M]}, c) = \mathbf{P}(X_{i+1} = c \,|X_{[i-M+1,i]} = y_{[1,M]})$$

are called *transition probabilities*, with $y_{[1]}, \ldots, y_{[M]}, c \in \Sigma$ and $M \leq i \leq n-1$. The vector of the *stationary probabilities* μ of a stationary Markov chain with transition matrix Π is defined as the solution of $\mu = \mu\Pi$. Thus, the unique stationary distribution μ is defined by the equation

$$\begin{aligned}\mu(y_{[1,M]}) &= \mathbf{P}(X_{[i-M+1,i]} = y_{[1,M]}) \\ &= \sum_{a\in\Sigma} \mu(ay_{[1,M-1]})\pi(ay_{[1,M-1]}, y_{[M]})\end{aligned}$$

where $M \leq i \leq n-1$.

We now introduce the random variable associated with the occurrences of a generic word y. We define $Z_i, 1 \leq i \leq n-m+1$ to be 1 if y occurs in x starting at position i, 0 otherwise. We set $Z_y = \sum_{i=1}^{n-m+1} Z_{i,y}$ so that Z_y is the random variable for the total number of occurrences $f_x(y)$.

In the stationary M-th order Markovian model, the expectation of Z_i, which represents the probability that y occurs at a given position i, is given by

$$\begin{aligned}E(Z_i) &= \mathbf{P}(X_{[i,i+m-1]} = y) \\ &= \mu(y_{[1,M]}) \prod_{i=1}^{m-M} \pi(y_{[i,i+M-1]}, y_{[i+M]}).\end{aligned}$$

The expected count of the occurrences y under the Markov model is therefore

$$\begin{aligned}E(Z_y) &= (n-m+1)E(Z_i) \\ &= (n-m+1)\mu(y_{[1,M]}) \prod_{i=1}^{m-M} \pi(y_{[i,i+M-1]}, y_{[i+M]})\end{aligned} \tag{3}$$

because the distribution of the Z_i's does not depend on i.

When the true model is *unknown*, the transition and stationary probabilities have to be estimated from the observed sequence x. Let y be a substring of x, where $m = |y| \geq M+2$. The transition probability can be estimated by the *maximum likelihood estimator*[49)]

$$\hat{\pi}(y_{[1,M]}, c) = \frac{f_x(y_{[1,M]}c)}{f_x(y_{[1,M]})} \tag{4}$$

and the *stationary probability* by the maximum likelihood estimator

$$\hat{\mu}(y_{[1,M]}) = \frac{f_x(y_{[1,M]})}{n - M + 1}. \tag{5}$$

Substituting in equation (3) for the estimators (4) and (5) we obtain an estimator of the expected count of y

$$\hat{E}(Z_y) = \frac{\prod_{i=1}^{m-M} f_x(y_{[i,i+M]})}{\prod_{i=2}^{m-M} f_x(y_{[i,i+M-1]})}. \tag{6}$$

A frequently used choice for the order of the chain is $M = m - 2$, called *maximal model*. When $M = m-2$, the estimator of the expected counts becomes

$$\hat{E}(Z_y) = \frac{f_x(y_{[1,m-1]}) f_x(y_{[2,m]})}{f_x(y_{[2,m-1]})}.$$

A precise relationship between the expectation of y and the expectation of its prefix and suffix is established in the following fact. The theorem derives immediately from equation (6).

Lemma 4.1
Let y be a substring of x and $w_1 = y_{[2,m]}$, $w_2 = y_{[1,m-1]}$. Then

$$\hat{E}(Z_y) = \frac{f(y_{[1,M+1]})}{f(y_{[2,M+1]})} \hat{E}(Z_{w_1}) = \hat{E}(Z_{w_2}) \frac{f(y_{[m-M,m]})}{f(y_{[m-M,m-1]})}$$

From an algorithmic perspective, Lemma 4.1 allows one to compute $\hat{E}(Z_y)$ incrementally from the $\hat{E}$ value of a prefix or a suffix of y. To simplify the notation, in the following we will use $\hat{E}(w)$ instead of $\hat{E}(Z_w)$.

§5 Suffix Trees

A simple method to count the number of occurrences of each substring in a sequence would be to create a look-up table. However, more time- and space-efficient data structure to organize a dictionary of words are known. One of these structures is the *suffix tree* (see, e.g., Reference[22] and references therein). The suffix tree is a digital search tree that represents a set of strings over a finite alphabet Σ. It has n leaves, numbered 1 to n. Each internal node, other than the root, has at least two children and each edge is labeled with a nonempty substring of x. No two edges outgoing from a node can have labels beginning with the same character. The tree has the property that for any leaf i, the concatenation of the labels on the path from the root to the the leaf i spells out exactly the suffix of x that starts at position i, that is, $x_{[i,n]}$. The substrings of x can be obtained by spelling out the words from the root to any internal node of the tree or to any position in the middle of an edge. The position in the tree corresponding to each substring y of x is called the *locus* of y. When a locus corresponds to a node of the suffix tree, we call it *proper*.

In order to achieve overall linear-space allocation, the labels on the edges are described implicitly: for each word, it suffices to save an ordered pair of integers indexing one of the occurrences of the label in the text. Each edge label requires thus constant space, which, in conjunction with the fact that total number of nodes and edges is bounded by $O(n)$, results in the overall linear space for the tree.

Several clever $O(n \log |\Sigma|)$ constructions are available (see, e.g., Reference[44, 57]). More recent linear-time algorithms are by Ukkonen[54] which is on-line, and by Farach[17] which is optimal for large alphabets. The large majority of these constructions exploit the presence of *suffix links* in the tree. The existence of suffix links is based on the following fundamental fact.

Lemma 5.1
If $w = ay, a \in \Sigma$ has a proper locus in T_x, then so does y.

Accordingly, suffix links are maintained in the tree from the locus of each string ay to the locus of its suffix y, for all $a \in \Sigma$.

Having built the tree, some additional processing make it possible to count and locate all the distinct instances of any pattern in $O(m)$ time, where m is the length of the pattern. In fact, the computation of the statistics of all substrings of a string is a direct application of suffix trees. We first need some definitions. We define the *leaf-list* $LL(u)$ of a node u as the ordered set of indices stored in the leaves of the subtree rooted at u. We refer to the unique string on the path from the root to a node u of the tree as the *path-label* $L(u)$ of u. Vertex u is also the proper locus of $L(u)$. Some strings do not have a proper locus because their paths end in the middle of an arc. We can think about this non-proper loci to correspond to *implicit* nodes of the tree. Clearly, these implicit nodes are non-branching (unary).

Given a word w, we denote by $<w>$ its proper locus, if it exists. If instead w ends in the middle of an arc then $<w>$ denotes the node corresponding to the shortest extension of w that has a proper locus. Clearly, $L(<w>) = w$.

By the structure of a suffix tree, the number of occurrences $f_x(w)$ of any string w is given by the number of leaves in the subtree rooted at $<w>$, that is, $f(w) = |LL(<w>)|$. In Fig. 7, the number of occurrences is stored in the internal nodes. The algorithm that annotates the tree with the value $f(w)$ takes linear time and space in the size of x.

The suffix tree for a single string can be easily extended to represent all the suffixes of a set of sequence $\{x_1, x_2, \ldots, x_k\}$. These suffixes are collected in a tree called *generalized suffix tree* (see Reference[22]). Now the leaves of the tree must have two indices, namely the index of the suffix and the index of the sequence, which we call *color*. The generalized suffix tree can also be built in linear time and space.

Once the tree is built, one can annotate each branching node $<w>$ with the number of colors $c(w)$. We recall that given strings $\{x_1, x_2, \ldots, x_k\}$ and a substring w, $c(w)$ is defined as the number of distinct sequences that contain at least one occurrence of w. An algorithm for computing the number of colors

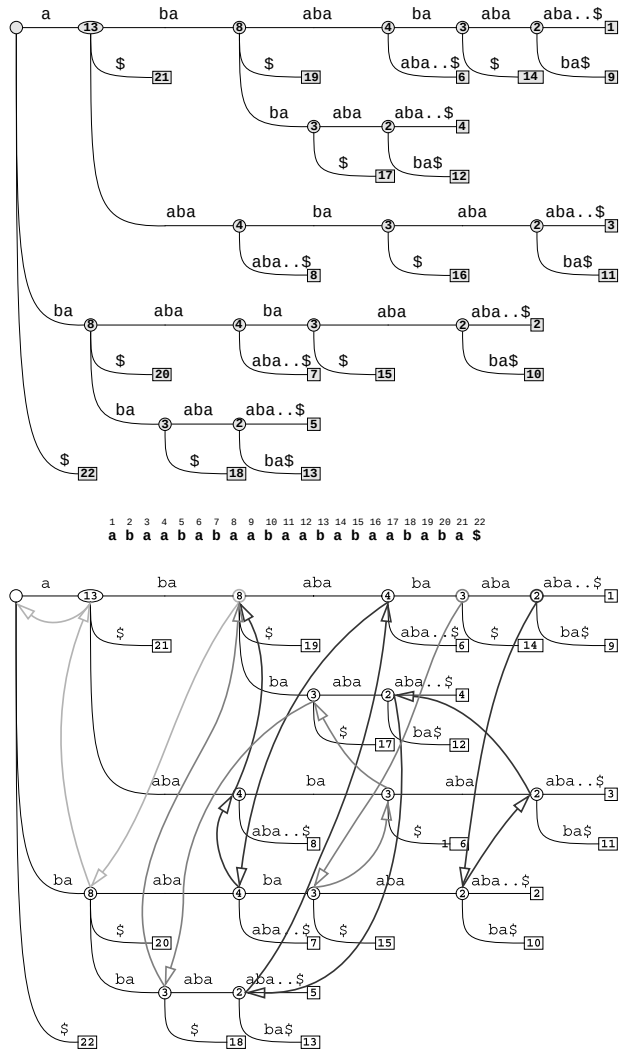

Fig. 7 TOP: The suffix tree T_x for the string $x =$ `abaababaabaababaababa$`, with internal nodes storing the number of occurrences. BOTTOM: The same suffix tree augmented with suffix links. The suffix links for the leaves are omitted for clarity (the path is $1 \rightarrow 2 \rightarrow \ldots \rightarrow 22 \rightarrow$ root)

for all the nodes of the suffix tree has been proposed by Hui[25] and takes linear time.

§6 Computing Scores by Comparing Trees

Let now r be the reference sequence, and x the sequence under analysis obtained by discretizing the time series R and X, respectively. In this section we show a preprocessing phase which takes r and x as input, and produces a

generalized suffix tree T for $\{x, r\}$ where each branching node $u \in T$ which is a locus for a substring $L(u)$ of x is annotated with the measure of surprise of the corresponding word.

The algorithm PREPROCESS described below computes the scores for all the substring of x with respect to the reference string r. Although the algorithm does not require an upper bound on the size of the substrings, it is reasonable to assume that we will rarely consider substrings longer than $\log_{|\Sigma|} n$ symbols. It is well known that words longer than $\log_{|\Sigma|} n$ symbols have an expected count whose limit tends to a constant instead of growing to infinity when n goes to infinity. These long words are therefore of limited interest to us.

In the first step of the preprocessing phase we build the generalized suffix trees T for $\{x, r\}$ and we annotate the internal nodes with the number of occurrences and the number of colors. This step requires linear time and space.

In the second step, we visit each branching node u of T in a breadth-first order. We do not visit implicit nodes of the tree because the frequency of substrings changes only at branching nodes and never in the middle of an arc. The same property is exploited in References[3, 4, 42] to detect usual words in DNA.

For a string $w = L(u)$, if w has two colors (i.e., it occurs both in x and in r), then we compute directly the score, assuming $\alpha f_r(w)$ to be the expected number of occurrences of w in the reference string, where $\alpha = \dfrac{|x| - m + 1}{|r| - m + 1}$. The scale factor α takes care of of adjusting the occurrences based on the length of x and r. For example, if r is two times longer than x we scale the number of occurrence observed in r by roughly $1/2$.

If otherwise the substring w has one color and it comes from x then we look for the Markov order $M(w)$ in the interval $[1, \ldots, |w| - 2]$ such that all the strings $w_{[j,j+M(w)]}$ occur in r, for $j = 1, \ldots, |w| - M(w)$. In other words, we look for the longest set of strings from r that cover w as it is done for the estimator of the expectation for Markov chains (see Section 4). This strategy corresponds to the idea of trying first the higher Markov orders, and falling back to lower orders whenever the information to compute the estimator of the expectation are insufficient. If every possible choice does not meet the requirements, we use the probability of the symbols from r (stored in a table) to compute the estimate.

Finally, we set the surprise score $z(w)$ to be the difference between the observed number of occurrences $f_x(w)$ and $\hat{E}(w)$. The preprocessing algorithm is sketched in Table 3.

After this manuscript was completed, it was brought to our attention that a quite similar strategy to estimate the probability of a substring was used in Reference[26]. In that paper, the authors compare several strategies (for example References[28, 39]) to estimate the probability of a string y using the number of occurrences of shorter strings contained in y. The conclusion of their study is that the most accurate prediction is the method called MOLC, which is very similar to the technique employed here.

The time complexity depends on the time taken to compute $\hat{E}(w)$. If the algorithm was implemented as in Table 3, the time complexity would be

Table 3 Outline of the preprocessing algorithm for the computation of the scores obtained comparing the the reference string r against the string under analysis x

suffix_tree PREPROCESS (**string** r, **string** x)
  let T = GENERALIZED_SUFFIX_TREE(x, r)
  let $\alpha = \dfrac{|x| - m + 1}{|r| - m + 1}$
  ANNOTATE_$f(w)(T)$
  ANNOTATE_$c(w)(T)$
  visit T in breadth-first traversal, for each node u **do**
    let $w = L(u), m = |w|$
    if w has two colors **then**
      let $\hat{E}(w) = \alpha f_r(w)$
    else if w comes from x **then**
      find the largest M in $[1, m-2]$
        such that $\prod_{j=1}^{m-M} f_r(w_{[j,j+M]}) > 0$
      if such M exists **then**
        let $\hat{E}(w) = \alpha \dfrac{\prod_{j=1}^{m-M} f_r(w_{[j,j+M]})}{\prod_{j=2}^{m-M} f_r(w_{[j,j+M-1]})}$
      else /* $M = 0$ */
        let $\hat{E}(w) = (|x| - m + 1) \prod_{i=1}^{m} \dfrac{f_r(y_{[i]})}{|r|}$
    let $z(w) = f_x(w) - \hat{E}(w)$
    store $z(w)$ in the node u
  return T

superlinear. To compute efficiently $\hat{E}(w)$ we need to exploit Lemma 4.1 and the suffix links of Lemma 5.1.

During the breadth-first traversal, as long as the number of colors is two, then $\hat{E}(w) = \alpha f_r(w)$ and $M(w) = |w| - 2$. This process requires $O(1)$ per node. When the number of colors drops to one, the value of $M(w)$ need to be updated.

Assume that we have already computed $M(w)$ for all the strings of a size m (recall that we are traversing breadth-first). In order to get $M(wv)$, where $|w| = m$ and v is the string on the edge to the next branching node, it suffices to know $M(w)$ and $M(w_{[2,m]}v)$. The following fact is easy to prove, and it is left to the reader.

Fact 6.1
Given two strings w and v

$$M(wv) = \min\{M(w), M(w_{[2,m]}v)\}$$

To find $M(w_{[2,m]}v)$, one need to start from the node $<wv>$ and traverse one suffix link to the node $<w_{[2,m]}v>$. Node $<w_{[2,m]}v>$ exists because of Lemma 5.1. Observe that $w_{[2,m]}$ has length $m-1$, and therefore it must have been processed before w. A careful ordering of the traversal can make sure that

$M(w_{[2,m]}v)$ will be available before $M(wv)$ is needed.

Given that $M(w)$ and $M(w_{[2,m]}v)$ can be obtained in $O(1)$, $M(wv)$ can also be found in constant time. Once $M(wv)$ is found, the value of $\hat{E}(wv)$ is computed using Lemma 4.1 by taking the $\hat{E}(\cdot)$ from the string from which we obtained the minimum, and multiplying by the appropriate factor. When M becomes zero, the computation of $\hat{E}(w)$ involves multiplying the probabilities of the symbols in w. This can be done in $O(1)$ by precomputing in linear time a vector that contains the probabilities of all the prefixes of x.[3,4,42] Since we spend $O(1)$ in each node, we can state the following.

Theorem 6.1
The algorithm PREPROCESS runs in linear time and linear space.

§7 TARZAN Algorithm

Having reviewed extensive material on discretization, Markov models and suffix trees, we now give a concise description of the proposed algorithm, which we call TARZAN.[*1] The basic algorithm is sketched in Table 4.

The inputs are the reference database R, the database to be examined X, and the parameters which control the discretization process. The algorithm begins by discretizing the data to the desired granularity. The two resultant strings are passed to the PREPROCESS algorithm which constructs the annotated suffix tree T. After this has been accomplished, the surprise of each substring found in x can be determined. Those substrings which have surprising ratings exceeding a certain user defined threshold (as defined by the absolute value of $z(w)$) can be returned and examined by the user.

The length l_2 of the sliding window is connected with the feature window length l_1 and the alphabet size a (which have been discussed in Section 3). We suggest choosing $l_2 < \log_{|\Sigma|} |x|$ because words longer than $\log_{|\Sigma|} |x|$ have extremely small expectations and belong to a different probabilistic regime. In

Table 4 Outline of the TARZAN algorithm: l_1 is the feature window length, a is the alphabet size for the discretization, l_2 is the scanning window length and c is the threshold

```
void TARZAN (time_series R, time_series X, int n, int a, real c)
  let x = DISCRETIZE_TIME_SERIES (X, n, a)
  let r = DISCRETIZE_TIME_SERIES (R, n, a)
  let T = PREPROCESS (r, x)
  for i = 1, |x| − l2 + 1
    let w = x[i,i+l2−1]
    retrieve z(w) from T
    if |z(w)| > c then print i, z(w)
```

[*1] TARZAN is not an acronym. It is a pun on the fact that the heart of the algorithm relies on comparing two suffix trees, *"tree to tree"*. Tarzan (R) is a registered trademark owned by Edgar Rice Burroughs, Inc.

fact, scores $z(w)$ are asymptotically Gaussian distributed when $|w| < \log_{|\Sigma|} |x|$ and Poisson distributed for longer words.[49] The threshold on the score c can be determined by gathering statistics about the distribution of the scores and/or assuming the distribution of the scores to be normal.

§8 Experimental Evaluation

Empirically we would like to demonstrate two features of our proposed approach.

- **Sensitivity (High True Positive Rate):** the algorithm can find truly surprising patterns in a time series.
- **Selectivity (Low False Positive Rate):** the algorithm does not find spurious "surprising" patterns in a time series.

We compare our approach with the TSA-tree Wavelet based approach of Shahabi *et al.*[51] and to the Immunology (IMM) inspired work of Dasgupta and Forrest,[13] which are the only obvious candidates for comparison. More details about these approaches are contained in Section 9.

We begin with a very simple experiment as a reality check. We constructed a reference dataset by creating a sine wave with 1,000 datapoints and adding some Gaussian noise. We then built a test dataset using the same parameters as the reference set; however, we also inserted three artificial anomalies of different types. We compared all three approaches under consideration. The results are shown in Fig. 8. We used a feature window of length $l_1 = 32$ for TARZAN and IMM, and an alphabet of size $a = 3$ for TARZAN.

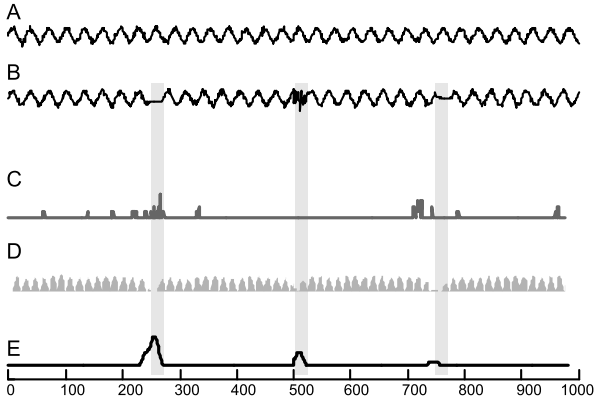

Fig. 8 A comparison of three anomaly detection algorithms on the same task. A) The training data, a slightly noisy sine wave. B) A time series containing three synthetic "anomalies", their locations are denoted by the gray bars C) The IMM anomaly detection algorithm failed to find the last two anomalies, and introduced some false alarms. D) The TSA-Tree approach is also unable to detect the anomaly. E) TARZAN shows a strong peak for the duration of the anomalies

The IMM approach was unable to find the last two anomalies, and it introduced some false alarms. Unlike the two other approaches, this method has a stochastic component. On some runs it appeared to detect the one of the three anomalies, but it always produce several false alarms of equal magnitude. The TSA approach also failed to find the anomalies.[*2] As noted by the authors, surprising patterns might exist at different scale. The graphic above shows the results at level 4' (see the original paper[51]) for details). However, similar results are seen at other levels. In contrast to the other techniques, TARZAN shows a strong peak for the duration of the anomalies. Note that for consistency with the other techniques, we flipped the results for TARZAN upside down, so the low expectation for the anomalies shows as peaks.

Objective Tests on Real Datasets. We performed additional extensive testing in medical and biomechanical datasets, including gait data, electrocardiograms, electromyelograms, and electroencephalograms. While the results looked subjectively correct, it takes some ingenuity to find datasets that allow testing.

In the following, we will consider some experiments in real datasets, where objective ground truth is available. Having shown above that the IMM algorithm and TSA-Tree approach are not realistic competitors, we exclude them from some of the experiments below, when this improves clarity. All datasets used in this paper are freely available from the UCR archive.

1) Stress Test Database. This database consists of twelve half-hour ECG recordings of normal ambulatory ECG recordings. At a randomly chosen time, the volunteers were subject to a loud unexpected sound, and their ECGs were annotated with the time of the occurrence of the sound.[45] We conducted a simple experiment as follows. We use a training sequence that contains just the first 20 seconds of the data, and then used TARZAN to find the most surprising pattern in the rest of the dataset. Figure 9 shows the results. We only used the first 20 seconds of normal heartbeats as our reference section because we wanted to determine whether we could obtain good results with limited training data.

Note that in Fig. 9(C) there is a strong double peak in surprise level just after data point 1,000, following by a zero surprise for the interval from about 1,150 to 1,250. These results make perfect sense, because visual inspection confirms that only the RST waves are anomalous, and the PQR waves look just the same before and after the loud noise event.

2) Power Demand Database. We consider a dataset that contains the power demand for a Dutch research facility for the entire year of 1997.[55] The data is sampled over 15 minute averages, and thus contains 35,040 points. The nice feature of this dataset is that although it contains great regularity, as shown in Fig. 10, it also contains regions that could objectively be said to be surprising or anomalous. In particular, there are several weeks on which one or more days were national holidays, and thus the normal pattern of five weekday peaks,

[*2] The authors define surprise as an absolute value, but visualize the level of surprise in their paper without taking the absolute value. For clarity and consistency we have graphed absolute values.

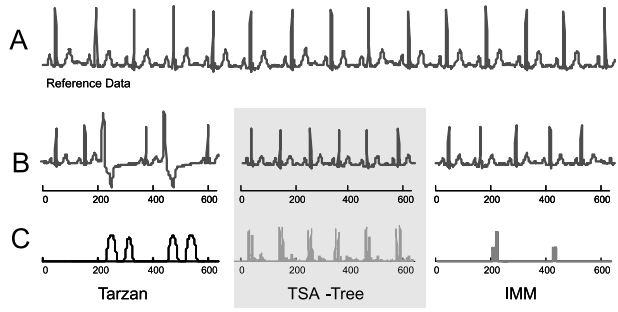

Fig. 9 An experiment in detecting anomalous electrocardiograms. A) The first 20 seconds of a normal heartbeat were used as training data. B) The TARZAN algorithm was then used to search the rest of the dataset to find the most surprising pattern, which we realigned to begin at point 1,000. C) The level of surprise returned by TARZAN, TSA-Tree and IMM for the subsequence shown above. The TARZAN results show a clear peak at the ECGS just after the volunteer was subjected to the unexpected noise. The TSA-Tree is unable to detect the anomaly, whereas IMM

Fig. 10 The first three weeks of the power demand dataset. Note the repeating pattern of a strong peak for each of the five weekdays, followed by relatively quiet weekends

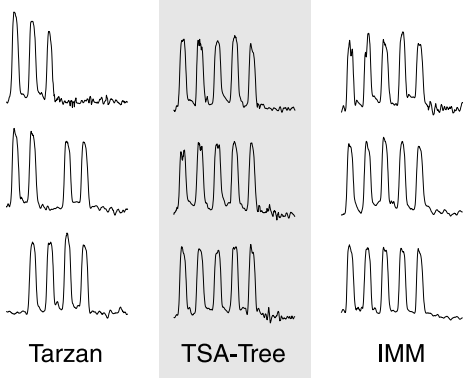

Fig. 11 The Three Most Surprising Weeks in the Power Demand Dataset, as Determined by TARZAN, TSA-Tree and IMM

followed by a relatively flat weekend, is disturbed.

We used from Monday January 6^{th} to Sunday March 23^{rd} as reference data. This time period is devoid of national holidays. We processed the remainder of the year with TARZAN, with a window size equivalent to four hours ($l_1 = 16$ datapoints), and an alphabet of size $a = 4$. Due to the size of the dataset we will just show the three most surprising sequences found by each algorithm. For each of the three approaches we show the entire week (beginning Monday), in which the three largest values of surprise fell. The results are shown in Fig. 11.

Both TSA-tree and IMM returned sequences that appear to be normal workweeks; however, TARZAN returned three sequences that correspond to the weeks that contain national holidays in the Netherlands. In particular, from top to bottom, the week spanning both December 25^{th} and 26^{th}, and the weeks containing Wednesday April 30^{th} (Koninginnedag, "Queen's Day") and May 19^{th} (Whit Monday) respectively. These results present strong visual evidence that TARZAN is able to find surprising patterns in time series.

3) Video Dataset. This experiment was conducted on a dataset of time series that were extracted from video surveillance footage.[9] For concreteness, we will briefly discuss how the data was extracted.

A Canon ZR40 camcorder with the shutter at 1/60, video size of 720x480 pixels, and frame rate of 30 frames per second was used to record the actions of a female actor. Only the actor's right hand was tracked; to facilitate this she wore a red glove on that hand only. A frame that has good color visibility was selected from the video sequence. The selected frame is then used to calculate Hue (H), Saturation (S), and Value (V) from each pixel. A region of the color to be tracked (red in this case) is also selected and this forms the region of interest (ROI). The HSV from each pixel of the selected frame, along with the mean and covariance from the ROI of the selected frame form the input to find probability distribution for the ROI over the whole image.[20] The probability distribution uses a multivariate Gaussian and results in a probability matrix of the size of the image. This resultant matrix is converted to a binary image by thresholding, and then the resultant binary image is used to compute the centroid position of the hand. The overall dataset is very high quality; however, there is some noise and dropouts due to occlusion etc. We extracted a time series from the video by tracking the center of mass for the right hand. Each frame of the original video becomes a datapoint in a time series. We only consider the Y-axis in the experiment that follows. The entire dataset consists of just over 200,000 datapoints.

The actor was asked to perform a variety of actions fifty times in a row, with a short pause in-between. The actions consisted of two classes, "innocuous", for example, pointing to photograph on the wall, and "threatening", for example, drawing a replicate firearm from a holster and aiming it. We asked the video technician who created the dataset to give us the most uneventful session as a reference time series, and used TARZAN to explore the rest of the data. We found the three most surprising events, according to our algorithm, and asked the

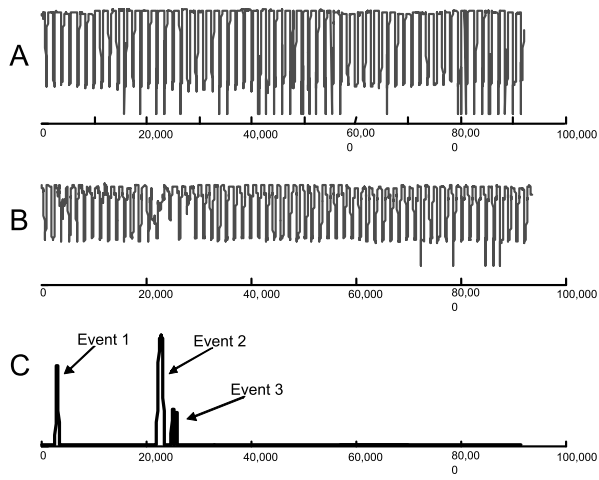

Fig. 12 An experiment in detecting anomalous subsequences from a video surveillance dataset. A) The first 3 minutes of the video capture sequence were used as training data. B) The TARZAN algorithm was then used to search the rest of the dataset to find the most surprising pattern, which we realigned to begin at point 1. Only a 3-minute subsection of the entire dataset is shown C) The level of surprise returned by TARZAN for the subsequence shown above. It shows a clear peak at three locations. An explanation of these events is discussed in the main text

technician to show us the corresponding video sequences. Figure 12(C) shows the small subsection of the dataset that contains the three most surprising events.

Inspection of video provides an immediate intuitive explanation of the results, as shown in Figs. 13 and 14. The sequence in B is supposed to record the actor drawing a replica gun from a hip mounted holster, aiming it at a target, and returning it to the holster. The entire sequence of events is repeated approximately fifty times. However, watching the video we discovered that at about ten seconds into the video, the actor misses the holster when returning the gun. An off-camera (inaudible) remark is made, and the actor looks toward the video technician, and convulses with laughter, at one point (frame 385) she is literally bent double with laughter. This event was the Event 1 discovered by TARZAN.

The actor quickly regains her composure, and the 50 seconds are uneventful. At that point, the actor looses concentration, draws the gun, but only raises it half way before returning it to the holster; she pauses briefly to regain her composure, then realizes that the video technician did not notice her mistake. She calls out to him, then mimics the mistake she just made. This occurs exactly where TARZAN discovers the Events 2 and 3. The video technician tells the actor to "just carry on, ignore it", and the rest of the video shoot is uneventful.

4) Aerospace Anomaly Detection Benchmarks. We were fortunate enough

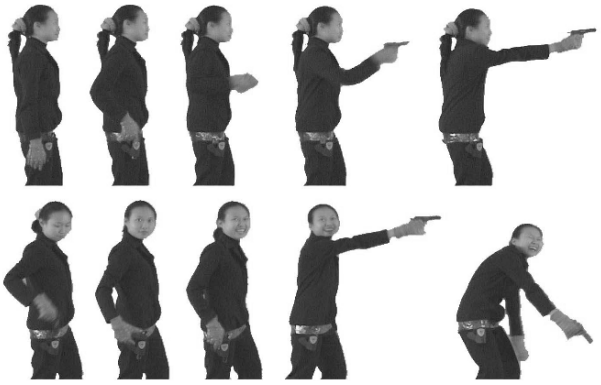

Fig. 13 Stills from the video shoot experiment. (Above) A typical sequence starts with the actors' hands by her sides. She removes the gun from the holster and aims it at a target. The sequence concludes with the actor returning the gun to the holster and her hand to her side (not shown). (Below) The stills corresponding to the Event 1, flagged by TARZAN. When returning the gun to the holster, the actor misses. She looks at the video technician, who has made a remark, she smiles and attempts to continue, but she is not looking at the target, she briefly convulses with laughter before regaining composure.

to obtain a set of anomaly detection benchmarks from the Aerospace Corp (TAC). The Aerospace Corp has a unique and crucial responsibility. It is charged with providing engineering assessments for the engineering discipline specialists who make the critical *go/no-go* decision moments before the launch of every unmanned space vehicle launched by the DoD. The cost of a false positive, allowing a launch in spite of a fault, or a false negative, stopping a potentially successful launch, can be measured in the tens of millions of dollars, not including the cost in morale and other more intangible detriments to the U.S. defense program. In the hours before a space launch TAC must monitor telemetry from more than 1,000 sensors. This is currently done having experienced engineers visually inspect incoming traces; naturally they would like to augment the engineer's abilities with a tool that could draw attention to potential anomalies.

As a sanity check for potential anomaly detection algorithms, TAC has created two benchmark tasks, called *Impulse* and *Sine*, respectively. The *Impulse* problems are designed to model switching states of electrical systems, and the *Sine* problems are designed to model valve systems. Each problem consists of one reference dataset, and ten datasets that had an artificial anomaly introduced at a fixed location. Examples of the anomalies include "one cycle additive Gaussian noise added", "two cycles frequency halved" and "with one impulse advanced 90°", etc. Figure 15 shows a subset of the data.

In our experiments on these benchmarks we did the following. For the two rival approaches we spent one hour of human time, and one hour of genetic algorithm CPU time to find the best parameter settings. For our approach

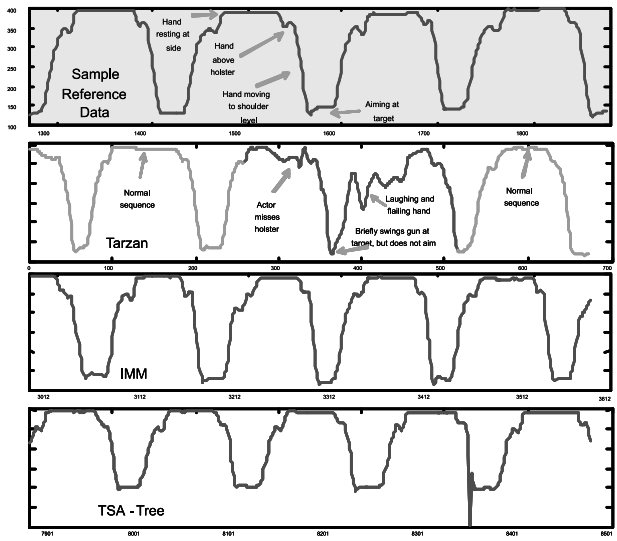

Fig. 14 The Y-axis time series corresponding to the events shown in Fig. 13. The top two sequences are zoom-ins of sequence B of Fig. 12. Note that as traditional in video coordinates, the top of the frame has a Y-value of zero, and the values increase moving down the frame (Above) Four typical examples of normal gun draw event. (Below) The most surprising subsequences found by the three algorithms under consideration. The subsequence found by TARZAN has an intuitive explanation. It is not obviously apparent that the subsequence found by IMM corresponds to a truly surprising subsequence. The subsequence returned by TSA-Tree contains a dropout (i.e., an outlier) at time 8357. As discussed above, outliers are a simple special case of anomaly. If we remove the outliers by simple smoothing, the results for TARZAN are unchanged, but TSA-tree finds an another subsequence which is not obviously surprising.

we showed a TAC engineer a one-page description of the TARZAN algorithm, and asked her to set the parameters (without actually allowing her to run any experiments). We only count TARZAN's annotation a success if doubling and halving the parameters also finds the anomalies. Each of the test problems is of length 1,000 and has its anomaly at time 500. We count an algorithm a success if it locates its strongest anomaly anywhere between time 450 and 550 (so a random algorithm has a one in ten chance of being correct). Table 5 summarizes the results.

In fairness we note that one of the Sine problems has missing values that are coded as NaNs. Because SAX codes these as a special character, this was a trivial problem for TARZAN to solve. Both the other techniques failed on this problem, because neither has a mechanism for dealing with missing values, however they could probably be extended for this eventuality.

The previous experiments demonstrate the ability of TARZAN to find sur-

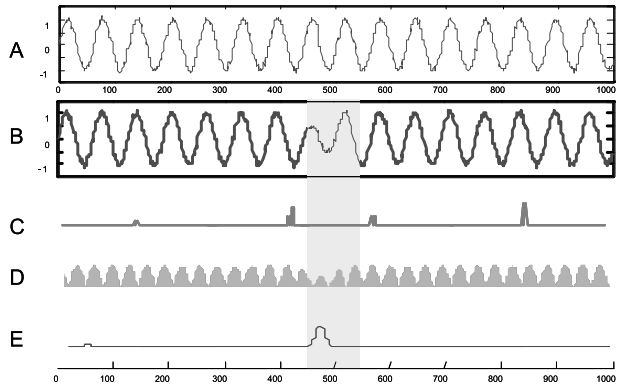

Fig. 15 A comparison of three anomaly detection algorithms on one of the Aerospace Benchmark problems. A) The training data, a slightly noisy sine wave. B) A time series containing a synthetic anomaly, its location is denoted by the gray bar. The anomaly is called "L-1M: Sine with amplitude decrease" C) The IMM anomaly detection algorithm failed to find the anomaly. D) The TSA-Tree approach is also unable to detect the anomaly. E) TARZAN shows a strong peak for the duration of the anomaly.

Table 5 The precision of the three algorithms under consideration on two sets of artificial anomaly-detection problems created by The Aerospace Corp.

	Impulse (true positives)	*Sine(true positives)*
TARZAN	10	9
TSA-Tree	2	3
IMM	3	4

prising patterns; however, we also need to consider TARZAN's selectivity. If even a small fraction of patterns flagged by our approach are false alarms, then, as we attempt to scale to massive datasets, we can expect to be overwhelmed by innumerable spurious "surprising" patterns.

In designing an experiment to show selectivity we are faced with the problem of finding a database which has been carefully annotated by independent means. Because using a real data set for this task would always be open to subjective post-hoc explanations of results, we will conduct the experiment on random walk data.[16,32)]

By definition, random walk data can contain any possible pattern. In fact, as the size of a random walk dataset goes to infinity, we should expect to see every pattern repeated an infinite number of times.[19)] We can exploit this property to test our algorithm. Suppose we fix a test dataset X_{RW} to be a random walk dataset of length 128. If we train TARZAN on another short random walk dataset, we should expect that the test data would be found surprising, since the chance that similar patterns exist in the short training database are very small. However as we increase the size of the training data, the overall measure

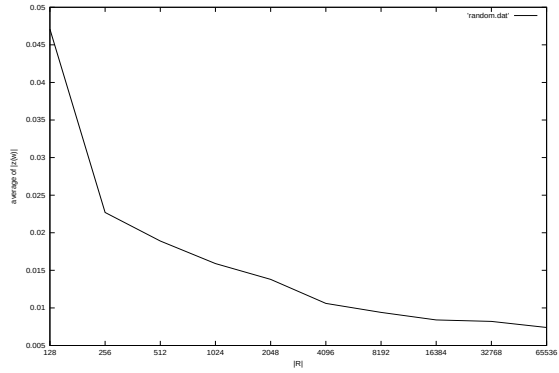

Fig. 16 The average of $|z(w)|$ of all patterns w found by TARZAN in a small fixed random walk database X_{RW} when trained on increasing large random walk datasets. With more experience, TARZAN finds the patterns less surprising

of surprise of the test data should decrease, since it is more likely that similar data was encountered. To restate, the intuition is that the more experience our algorithm has seen random walk data, the less surprising our particular section of random walk X_{RW} should appear.

We will not consider IMM and TSA-tree in this experiment. IMM is not defined for arbitrary large random walk datasets (see Section 9), and TSA-tree does not learn from experience, and thus will have a constant level of surprise. We tested TARZAN with increasing large reference datasets length 128 to 65,536. We used a feature window length $l_1 = 12$ and an alphabet size $a = 4$.

The results are shown in Fig. 16. Note that with little training data, the average value of $|z(w)|$ is quite high. In other words, the patterns in X_{RW} appear surprisingly rare because TARZAN has not seen enough training data to build a general enough model of what to expect when confronted with more random walk data. However, with more experience, TARZAN finds the patterns in X_{RW} to be less and less surprising. These results strongly suggest that TARZAN is able to improve its selectivity as it sees more data.

§9 Related Work

The task of finding surprising patterns in data has been an area of active research, which has long attracted the attention of researchers in biology, physics, astronomy and statistics, in addition to the more recent work by the data mining community. The problem, and closely related tasks are variously referred to as the detection of "Aberrant Behavior",[38] "Novelties",[13,43] "Faults",[59] "Surprises",[10,51] "Deviants",[27] "Temporal Change",[6,18] "Bursts",[37,62] and "Outliers".[23]

The problem of detecting surprising patterns in discrete domains has received the most attention, especially in the context of network intrusion.[60] However, this problem differs from the task discussed in this paper in that the data is already discrete, it arrives at arbitrary intervals, and typically scalability to

large datasets is not an important issue. In addition, while the data may be correlated, they are atomic units (for example, Unix commands, or ATM transactions), whereas time series data are intrinsically continuous. Various approaches to this problem have been suggested; many of them reduce to the idea that surprising patterns should be less compressible than unsurprising ones.[10)]

In the context of the analysis of biosequences, the search for unexpectedly frequent or infrequent substrings is only one component of the broader quest for interesting patterns of more general kinds. Along these lines, patterns and families thereof have been variously characterized, and criteria, algorithms and software developed in correspondence. Without pretending to be exhaustive, we mention TEIRESIAS,[50)] GIBBS SAMPLER,[40)] MEME,[5)] WINNOWER,[30,48)] PROJECTION,[53)] WEEDER,[46)] VERBUMCULUS,[3,4,42)] among others.

Detecting surprising patterns in time series has received much less attention. There has been some work on novelty detection in time series using neural networks,[7,58)] however we do not consider this approach in detail since scalability is clearly an issue.

A simple approach to monitoring time series is to place a restriction on the maximum and minimum tolerable values, and to sound an alarm if the signal ever moves out of this envelope of acceptable behavior. This practice is referred to as *limit-checking*. While trivial to implement, the method is known to have poor sensitivity. A more sophisticated approach involves *discrepancy-checking*. The idea here is to use the data observed thus far to predict future values. Any discrepancy between the two that exceeds a certain tolerance can be denoted as surprising.[8)] Of course, this technique is only as good as the prediction algorithm, and time series prediction is a notoriously difficult problem. Thus, this method tends to have poor selectivity, producing many false alarms. In interesting work by Decoste,[15)] the author integrated both *limit-checking* and *discrepancy-checking* in a single framework, however scalability to massive datasets remains an issue, and the measure of surprise of an alarm can only be judged relative to the prediction model, which must be correctly specified by a domain expert.

Jagadish *et al.*[27)] introduced a technique for mining deviants in time series; however, deviants are *"...points with values that differ greatly from that of surrounding points"*, and thus this work may be considered more of a generalization of classic outlier detection.[23)]

In Reference[51)] and several follow up papers, Shahabi *et al.* suggest a method to find both trends and "surprises" in large time series datasets. The authors achieve this by using a wavelet-based tree structure (TSA-Tree) that can represent the data at different scales, e.g., the weather trend in last month vs. last decade. However, the definition of surprise used seems limited to dramatic shifts in the signal. In particular, this approach is not suitable for detecting unusual data patterns that hide inside the normal signal range. For example, the system would not be able to detect if we give it an EEG time series that we had flipped upside down, since the wavelet-based "surprise" features are invariant to this transformation of the data.

The immunological based approach of Dasgupta and Forrest,[13] is inspired by the negative selection mechanism of the immune system, which discriminates between *self* and *non-self*. In this case *self* is the model of the time series learned from the reference dataset, and *non-self* are any observed patterns in the new dataset that do not conform to the model within some tolerance. A major limitation of the approach is that it is only defined when the space of *self* is not exhaustive. However, if you examine enough random walk data (or financial data, which is closely modeled by random walk[16]), *self* rapidly becomes saturated with every possible pattern, and thus *non-self* is the null set, and nothing encountered thereafter is considered surprising.

§10 Conclusions

In this paper we introduced TARZAN, an algorithm that detects surprising patterns in a time series database in linear space and time. Our definition of surprising is general and domain independent, and flags a pattern to be surprising if the frequency with which we observe it differs greatly from that expected given previous experience. We compared it to two other algorithms on both real and synthetic data, and found it to have much higher sensitivity and selectivity.

Although the ability of TARZAN to find surprising patterns without user intervention is a great advantage, we intend to investigate the possibility of incorporating user feedback and domain based constraints. A current trend in reducing the number of parameters in data mining algorithms[35] could enlist TARZAN among the parameter-light methods. TARZAN is certainly not parameter-free. The discretization and the dimensionality reduction have a few key parameters, as well as TARZAN itself.

We conclude by noting that here we have concentrated the attention solely on the intricacies of finding the surprising patterns, without addressing the many meta-questions that arise. For example, the possible asymmetric costs of false alarms and false dismissals, and the actionability of discovered knowledge.[18] We intend to address these issues in future work.

Acknowledgements

A preliminary version of this work was presented at the *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '02)*, Edmonton, Alberta, Canada, and included in its *Proceedings*, pp. 550-556, ACM press (2002). A portion of Section 3 of this paper was presented at the *8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* and included in its *Proceedings*, pp. 2-11 (2003).

This project was supported in part by NSF CAREER IIS-0447773, NSF CAREER IIS-0237918, and NSF DBI-0321756.

The authors would like to thank Bhrigu Celly, Chotirat Ann Ratanamahatana and Victor Zordan for their help with the video data experiment, and Michalis Vlachos and Ahmet Bulut for their useful comments and suggestions.

References

1) Agrawal, R., Faloutsos, C., and Swami, A, "Efficient Similarity Search in Sequence Databases," in *Proc. of the 4th Int'l Conference on Foundations of Data Organization and Algorithms*, pp. 69–84, Chicago, IL, 1993.
2) Andre-Jonsson, H. and Badal D.Z., "Using Signature Files for Querying Time-series Data," in *Proc. of Principles of Data Mining and Knowledge Discovery, 1st European Symposium*, pp. 211–220, Trondheim, Norway, 1997.
3) Apostolico, A., Bock, M.E. and Lonardi, S., "Monotony of Surprise and Large-scale Quest for Unusual Words," *J. Comput. Bio., 10, 3–4*, pp. 283–311, 2003.
4) Apostolico, A., Bock, M. E., Lonardi, S. and Xu, X., "Efficient Detection of Unusual Words," *J. Comput. Bio., 7, 1/2*, pp. 71–94, Jan., 2000.
5) Bailey, T.L. and Elkan, C., "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization," *Machine Learning, 21, 1/2*, pp. 51–80, 1995.
6) Blockeel, H., Furnkranz, J., Prskawetz, A. and Billari, F.C., "Detecting Temporal Change in Event Sequences: An Application to Demographic Data," in *Proc. Principles of Data Mining and Knowledge Discovery*, pp. 29–41, 2001.
7) Borisyuk, R., Denham, M., Hoppensteadt, F., Kazanovich, Y. and Vinogradova, O., "An Oscillatory Neural Network Model of Sparse Distributed Memory and Novelty Detection," *BioSystems, 58*, pp. 265-272, Jan., 2000.
8) Brutlag, J.D., "Aberrant Behavior Detection in Time Series for Network Service Monitoring," in *Proc. of the 14th Systems Administration Conference, Berkeley, CA, Dec., 3–8, 2000*, pp. 139-146, The USENIX Association.
9) Celly, B., Ratanamahatana, C.A., and Zordan, V., "A Novel Technique for Indexing Video Surveillance Data," submitted for publication, 2003.
10) Chakrabarti, S., Sarawagi, S., and Dom, B., "Mining Surprising Patterns Using Temporal Description Length," in *Proc. 24th Int. Conf. Very Large Data Bases*, pp. 606–617, 1998.
11) Chan, K., and Fu, A.W., "Efficient Time Series Matching by Wavelets," in *Proc. of the 15th IEEE Int'l Conference on Data Engineering*, pp. 126–133, 1999.
12) Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P., "Rule Discovery from Time Series," in *Proc. of the 4th International Conference of Knowledge Discovery and Data Mining, AAAI Press*, pp. 16–22, 1998.
13) Dasgupta, D., and Forrest, S., "Novelty Detection in Time Series Data Using Ideas from Immunology," in *Proc. of The International Conference on Intelligent Systems*, 1999.
14) Daw, C.S., Finney, C.E.A. and Tracy, E.R., "Symbolic Analysis of Experimental Data," *Review of Scientific Instruments*, 2001.
15) Decoste, D., "Mining Multivariate Time-series Sensor Data to Discover Behavior Envelopes," in *Proc. Knowledge Discovery and Data Mining*, pp. 151–154, 1997.
16) Faloutsos, C., Ranganathan, M., and Manolopoulos, Y., "Fast Subsequence Matching in Time-series Databases," *SIGMOD Record (ACM Special Interest Group on Management of Data), 23, 2*, pp. 419–429, June, 1994.
17) Farach, M., "Optimal Suffix Tree Construction with Large Alphabets," in *Proc. 38th Annual Symposium on Foundations of Computer Sciencepp*, pp. 137–143, Oct., 1997.

18) Fawcett, T., and Provost, F., "Activity Monitoring: Noticing Interesting Changes in Behavior," in *Proc. Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,*, pp. 53–62, 1999.

19) Feller, W., *An Introduction to Probability Theory and its Applications*, Wiley, New York, 1968.

20) Forsyth, D. and Ponce, J., *Computer Vision: A Modern Approach*, Prentice-Hall, 2001.

21) Ge, X. and Smyth, P., "Deformable Markov Model Templates for Time-series Pattern Matching," in *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pp. 81–90, 2000.

22) Gusfield, D., *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.

23) Hawkins, D.M., *Identification of Outliers, Monographs on Applied Probability & Statistics*, Chapman and Hall, London, 1980.

24) Huang, Y.-W. and Yu, P., "Adaptive Query Processing for Time-series Data," in *Proc. Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (S. Chaudhuri and D. Madigan eds.)*, pp. 282–286, ACM Press, Aug., 1999.

25) Hui, L.C.K., "Color Set Size Problem with Applications to String Matching," in *Proc. of the 3rd Annual Symposium on Combinatorial Pattern Matching (Tucson, AZ, 1992)*, (A. Apostolico, M. Crochemore, Z. Galil and U. Manber, eds.), *LNCS644*, Springer-Verlag, Berlin, pp. 230–243.

26) Jagadish, H.V., Kapitskaia, O., Ng, R.T. and Srivastava, D., "One-dimensional and Multi-dimensional Substring Selectivity Estimation," VLDB Journal 9, 3, pp. 214–230, 2000.

27) Jagadish, H.V., Koudas, N. and Muthukrishnan, S. "Mining Deviants in a Time Series Database," in *Proc. 25th International Conference on Very Large Data Bases*, pp. 102–113, 1999.

28) Jagadish, H.V., Ng, R.T. and Srivastava, D., "Substring Selectivity Estimation," in *Proc. of PODS*, pp. 249–260, 1999.

29) Kalpakis, K., Gada, D. and Puttagunta, V., "Distance Measures for Effective Clustering of Arima Time-series," in *Proc. of the 2001 IEEE International Conference on Data Mining*, pp. 273–280, San Jose, CA, 2001.

30) Keich, U. and Pevzner, P.A., "Finding Motifs in the Twilight Zone," in *Annual International Conference on Computational Molecular Biology*, pp. 195–204, Washington, DC, Apr., 2002.

31) Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S., "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Journ. of Knowledge and Information Systems, 3, 3*, pp. 263–286, 2000.

32) Keogh, E., Chakrabarti, K., Pazzani, M. and Mehrotra, S., "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *SIGMOD Record (ACM Special Interest Group on Management of Data), 30*, pp. 2151–162, June, 2001.

33) Keogh, E. and Kasetty, S., "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 102-111, Edmonton, Alberta, Canada, 2002.

34) Keogh, E., Lonardi, S. and Chiu, B., "Finding Surprising Patterns in a Time Series Database in Linear Time and Space," in *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pp. 550–556, Edmonton, Alberta, Canada, July, 2002.

35) Keogh, E., Lonardi, S. and Ratanamahatana, C.A., "Towards Parameter-free Data Mining," in *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pp. 206–215, Seattle,WA, 2004.

36) Keogh, E. and Pazzani, M., "An Enhanced Representation of Time Series Which Allows Fast and Accurate Classifcation, Clustering and Relevance Feedback, in *Proc. 4th International Conference on Knowledge Discovery and Data Mining*, pp. 239–241, 1998.

37) Kleinberg, J., "Bursty and Hierarchical Structure in Streams," in *Proc. of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, Edmonton, Alberta, Canada, July, 2002.

38) Kotsakis, E. and Wolski, A., "Maps: A Method for Identifying and Predicting Aberrant Behaviour in Time Series," in *Proc. 14th Internat. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 2001.

39) Krishnan, P., Vitter, J.S. and Iyer, B., Estimating Alphanumeric Selectivity in the Presence of Wildcards," in *Proc. of SIGMOD*, pp. 282–293, 1996.

40) Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C., "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment," *Science, 262*, pp. 208–214, Oct., 1993.

41) Lin, J., Keogh, E., Lonardi, S. and Chiu, B., "A Symbolic Representation of Time Series with Implications for Streaming Algorithms," in *Proc. of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 2–11, San Diego, CA, 2003.

42) Lonardi, S., "Global Detectors of Unusual Words: Design, Implementation, and Applications to Pattern Discovery in Biosequences," *Ph.D. thesis*, Department of Computer Sciences, Purdue University, August 2001.

43) Ma, J. and Perkins, S., "Online Novelty Detection on Temporal Sequences," in *Proc. International Conference on Knowledge Discovery and Data Mining (KDD'03)*, Washington, DC, 2003.

44) Mccreight, E.M., "A Space-economical Suffix Tree Construction Algorithm," *J. Assoc. Comput. Mach., 23, 2*, pp. 262–272, Apr. 1976.

45) Moody, G., Muldrow, W. and Mark, R.G., "A Noise Stress Test for Arrhythmia Detectors," *Computers in Cardiology 11*, pp. 381–384, 1984.

46) Pavesi, G., Mauri, G. and Pesole, G., "An Algorithm for Finding Signals of Unknown Length in DNA Sequences," in *Proc. of the International Conference on Intelligent Systems for Molecular Biology*, pp. S207–S214, AAAI press, Menlo Park, CA, 2001.

47) Perng, C., Wang, H., Zhang, S. and Parker, S., "Landmarks: A New Model for Similarity-based Pattern Querying in Time Series Databases," in *Proc. of 16th International Conference on Data Engineering*, 2000.

48) Pevzner, P.A. and Sze, S.-H., "Combinatorial Approaches to Finding Subtle Signals in DNA Sequences, in *Proc. of the International Conference on Intelligent Systems for Molecular Biology*, pp. 269–278, AAAI press, Menlo Park, CA, 2000.

49) Reinert, G., Schbath, S. and Waterman, M.S., "Probabilistic and Statistical Properties of Words: An Overview," *J. Comput. Bio., 7*, pp. 1–46, 2000.

50) Rigoutsos, I. and Floratos, A., "Combinatorial Pattern Discovery in Biological Sequences: The TEIRESIAS Algorithm," *Bioinformatics, 14, 1*, pp. 55–67, 1998.

51) Shahabi, C., Tian, X. and Zhao, W., "Tsa-tree: A Wavelet-based Approach to Improve the Efficiency of Multi-level Surprise and Trend Queries," in *Proc. 12th International Conference on Scientific and Statistical Database Management*, 2000.

52) Struzik, Z.R., and Siebes, "A. Measuring Time Series Similarity through Large Singular Features Revealed with Wavelet Transformation,". in *Proc. of the 10th International Workshop on Database & Expert Systems Applications*, pp. 162–166, 1999.

53) Tompa, M. and Buhler, J., "Finding Motifs Using Random Projections," in *Annual International Conference on Computational Molecular Biology*, pp. 67–74, Montreal, Canada, Apr., 2001.

54) Ukkonen, E., "On-line Construction of Suffix Trees," *Algorithmica, 14, 3*, pp. 249–260, 1995.

55) Van Wijk, J.J. and Van Selow, E.R., "Cluster and Calendar-based Visualization of Time Series Data," in *Proc. IEEE Symposium on Information Visualization*, pp. 4–9, Oct., 25–26, 1999.

56) Vlachos, M., Kollios, G. and Gunopulos, D., "Discovering Similar Trajectories," in *18th IEEE International Conf. on Data Engineering*, San Jose, CA, 2002.

57) Weiner, P., "Linear Pattern Matching Algorithm," in *Proc. of the 14th Annual IEEE Symposium on Switching and Automata Theory*, pp. 1–11, Washington, DC, 1973.

58) Whitehead, B. and Hoyt, W.A., "A Function Approximation Approach to Anomaly Detection in Propulsion System Test Data," in *Proc. AIAA/SAE/ASME/ASEE 29th Joint Propulsion Conference*, Monterey, CA, June 1993.

59) Yairi, T., Kato, Y. and Hori, K., "Fault Detection by Mining Association Rules from House-keeping Data," in *Proc. of International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2001.

60) Yang, J., Wang, W. and Yu, P., "Info-miner: Mining Surprising Periodic Patterns," in *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 395–400, 2001.

61) Yi, B.K. and Faloutsos, C., "Fast Time Sequence Indexing for Arbitrary Lp Norms,". in *Proc. of the 26st Intl. Conference on Very Large Databases*, pp. 385–394, 2000.

62) Zhu, Y. and Shasha, D., *Efficient Elastic Burst Detection in Data Streams.*

Stefano Lonardi, Ph.D.: He is Assistant Professor at University of California, Riverside, CA. He received his "Laurea cum laude" from University of Pisa in 1994 and his Ph.D. in 2001 from the Department of Computer Sciences, Purdue University, West Lafayette, IN. He also holds a doctorate degree from University of Padua (1999). In 2005, he received the CAREER award from National Science Foundation. His recent research interest includes algorithms, computational molecular biology, and data mining.

Jessica Lin, Ph.D.: She is an assistant professor of Information and Software Engineering in George Mason University. She received her PhD in Computer Science from the University of California, Riverside. Her research interests include data mining and informational retrieval.

Eamonn Keogh: He is an assistant professor at the University of California, Riverside. His research interests are machine learning, information retrieval and techniques for solving similarity and indexing problems in time-series datasets.

Bill 'Yuan-chi' Chiu: He is a graduate student at University of California, Riverside. His research interests are evolutionary algorithms, reinforcement learning algorithms, physic-based character animation, and high-level behavior synthesis using motion-captured data.