GrammarViz 3.0: Interactive Discovery of Variable-Length Time Series Patterns

10

PAVEL SENIN, Los Alamos National Laboratory, University Of Hawai'i at Mānoa
JESSICA LIN and XING WANG, George Mason University
TIM OATES and SUNIL GANDHI, University Of Maryland, Baltimore County
ARNOLD P. BOEDIHARDJO, CRYSTAL CHEN, and SUSAN FRANKENSTEIN,
US Army Engineer Research and Development Center

The problems of recurrent and anomalous pattern discovery in time series, e.g., motifs and discords, respectively, have received a lot of attention from researchers in the past decade. However, since the pattern search space is usually intractable, most existing detection algorithms require that the patterns have discriminative characteristics and have its length known in advance and provided as input, which is an unreasonable requirement for many real-world problems. In addition, patterns of similar structure, but of different lengths may co-exist in a time series. Addressing these issues, we have developed algorithms for variable-length time series pattern discovery that are based on symbolic discretization and grammar inference—two techniques whose combination enables the structured reduction of the search space and discovery of the candidate patterns in linear time. In this work, we present GrammarViz 3.0—a software package that provides implementations of proposed algorithms and graphical user interface for interactive variable-length time series pattern discovery. The current version of the software provides an alternative grammar inference algorithm that improves the time series motif discovery workflow, and introduces an experimental procedure for automated discretization parameter selection that builds upon the minimum cardinality maximum cover principle and aids the time series recurrent and anomalous pattern discovery.

CCS Concepts: • **Information systems → Data mining**;

Additional Key Words and Phrases: Interactive data mining

ACM Reference format:
Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. 2018. GrammarViz 3.0: Interactive Discovery of Variable-Length Time Series Patterns. *ACM Trans. Knowl. Discov. Data.* 12, 1, Article 10 (February 2018), 28 pages.
https://doi.org/10.1145/3051126

This work is partially supported by the National Science Foundation, under Grant nos. 1218325 and 1218318.
Authors' addresses: P. Senin, Los Alamos National Laboratory, mailstop M888, Los Alamos, NM 87545; email: psenin@lanl.gov; J. Lin and X. Wang, Computer Science, George Mason University, Fairfax, Virginia; emails: {jessica, xwang24}@gmu.edu; T. Oates and S. Gandhi, Computer Science, University of Maryland, Baltimore County, Baltimore; emails: Marylandoates@cs.umbc.edu, sunilga1@umbc.edu; A. P. Boedihardjo, C. Chen, and S. Frankenstein, US Army Corps of Engineers, Washington, DC; emails: {arnold.p.boedihardjo, crystal.chen, susan.frankenstein}@usace.army.mil.

https://doi.org/10.1145/3051126

1 INTRODUCTION

The task of pattern mining in sequential data is an important problem that has many applications. Typically, two types of patterns receive attention from researchers and practitioners: (i) frequently occurring, highly similar patterns called time series motifs, and (ii) rare and unexpected patterns that are called time series anomalies.

The problem of frequent sequential pattern discovery spans multiple fields: it is a pre-cursor to association rule mining (Agrawal et al. 1993; Agrawal and Srikant 1995), the core of the biological sequence significance assessment (Durbin et al. 1998; Gionis and Mannila 2003; He 2006; Staden 1989; Buhler and Tompa 2002), and play an important role in speech recognition (Vitevitch et al. 1997) and image analysis (Julea et al. 2011). A substantial body of literature has been devoted to techniques that aid the discovery of frequent sequential patterns.

Anomalous (rare) sequential pattern detection is an equally important problem—the ability to discover anomalies efficiently is particularly important in a variety of application domains where anomalies convey critical and actionable information, such as in health care, equipment safety, security surveillance, and fraud detection. The anomaly detection problem has been studied in diverse research areas (Gupta et al. 2014). Despite the problem's simplicity at the abstract level, where an anomaly is defined as a pattern that does not conform to the underlying generative processes, the problem is difficult to solve in its most general form Chandola (2009).

In our previous work, we defined two concepts that extend and formalize the notion of frequent and anomalous patterns, namely "time series motifs," which are, informally stated, frequently repeated patterns (Lin et al. 2002; Patel et al. 2002), and "time series discord," the most unusual pattern in the observed signal (Keogh et al. 2005). Since then, a great deal of work has been proposed for the discovery of time series motifs (Grabocka et al. 2016; Mohammad and Nishida 2014; Beaudoin et al. 2008; Castro and Azevedo 2010; Chiu et al. 2003; Lin et al. 2002; Meng et al. 2008; Minnen et al. 2006, 2007; Mueen et al. 2009; Oates 2002; Patel et al. 2002; Tanaka and Uehara 2004; Tanaka et al. 2005; Tang and Liao 2008) and discords (Piciarelli et al. 2008; Yankov et al. 2008; Kawahara and Sugiyama 2009; Wei et al. 2006; Gupta et al. 2014). Moreover, compared with other techniques, time series discord has been shown to capture the most unusual subsequence within a time series that is likely to correspond to possible anomalies within the generative processes, as was confirmed in a recent extensive empirical study by Chandola et al., where the authors conclude that "..on 19 different publicly available data sets, comparing 9 different techniques time series discord is the best overall technique among all techniques" (Chandola 2009).

Most existing work on time series pattern discovery, however, suffers a significant limitation; that is, they require an input parameter, the expected pattern length m, to be defined. This input requirement restricts the search space on pattern length, allowing the search to finish in a reasonable amount of time. However, it also burdens the problem of proper length selection to the user.

A few algorithms were proposed to discover motifs of variable lengths (Minnen et al. 2006; Oates 2002; Tanaka et al. 2005; Nunthanid et al. 2011; Tang and Liao 2008); however, they either do so via post-processing, scale poorly, or quantize the whole data rather than considering overlapping subsequences, often resulting in inaccurate and incomplete patterns. Similarly, few algorithms were proposed for a variable length anomalous pattern discovery among which the WCAD algorithm—a parameter-free approximate anomaly discovery approach—proposed by Keogh et al. (2004) is closest to our technique. However, built upon the use of an off-the-shelf compressor, WCAD is computationally expensive and still requires the expected anomaly length to be specified.

In our previous work, we have addressed the aforementioned issue and proposed the first realistic algorithm for the variable-length pattern discovery problem (Li et al. 2012; Senin et al. 2015, 2014). We have shown that the frequent and anomalous patterns of different lengths often co-exist within the same dataset and that it is possible to discover them by using the combination of two

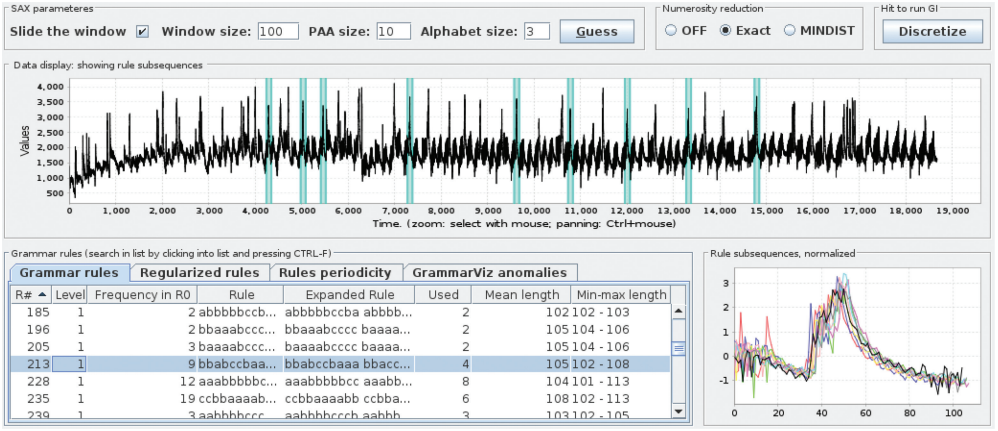

Fig. 1. A screenshot of GrammarViz 3.0 GUI—an interactive time series pattern mining tool implementing our algorithms. The approximate motif discovered in the insect dataset whose length vary from 102 to 108 is shown. The top panel shows the entire dataset span highlighting the motif occurrences, the bottom panels show the table of all patterns discovered in the dataset and superimposed, z-normalized subsequences corresponding to the selected grammar rule.

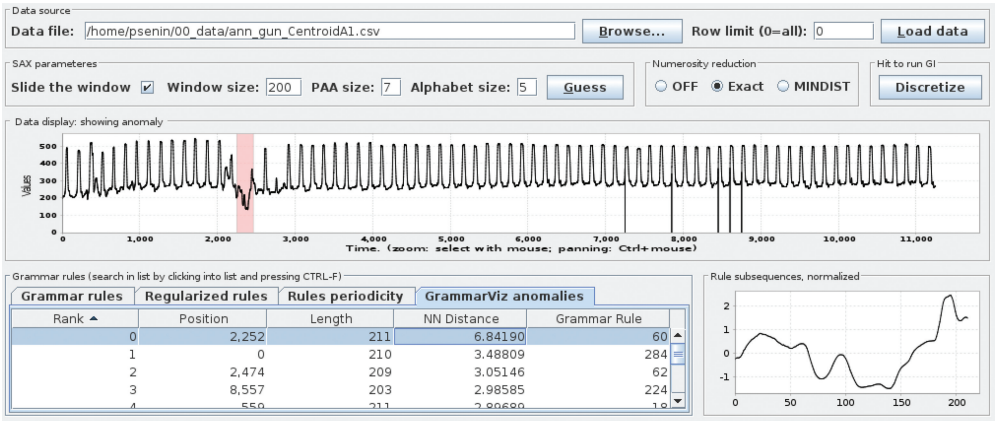

Fig. 2. An exact discord discovered using GrammarViz 3.0 in the Video telemetry dataset [Keogh et al. 2004]. The entire dataset highlighting the discord occurrence shown at the top panel, the bottom panels show the table of discovered discords and z-normalized anomalous subsequence.

techniques: time series discretization and grammatical inference (Li et al. 2012; Senin et al. 2015, 2014). Figure 1 shows an example of our motif discovery technique applied to the Insect dataset, where it discovers nine instances of the approximately similar pattern. Figure 2 shows an example of a time series discord discovery using our technique.

While the proposed techniques advance the state of the art in motif and discord discovery by allowing variable pattern lengths, many challenges remain. The most important ones arise from the nature of the employed techniques, namely discretization and grammar inference. Specifically, the efficiency and the effectiveness of our techniques depend on the proper (according to the task context) selection of discretization parameters. While the ability to discover variable-length patterns mitigates some of the parameter selection problem, the *ability to select nearly-optimal*

parameters is still largely unaddressed. The second challenge relates to the rule sets returned by the grammar inference algorithm. Grammatical inference effectively decomposes discretized time series into a large hierarchical structure describing long- and short-term correlations between discretized subsequences. Often, however, the amount of grammar rules, each of which may correspond to many overlapping time series subsequences, is too large and spurious to comprehend clearly. Thus, *an efficient pattern-pruning technique is highly desirable.*

In this work, we propose the following solutions for both the problems: A semi-automated discretization parameter optimization procedure, and two strategies for pattern numerosity reduction. Overall, in contrast with our previous work, in this article we

(1) introduce an implementation of the offline grammar inference algorithm called Re-Pair (Larsson and Moffat 1999) and show how its application improves the time series motif discovery;
(2) propose a motif summarization technique based on clustering of similar patterns;
(3) propose a greedy grammar rule redundancy pruning technique based on the minimum cardinality maximum cover principle;
(4) propose and discuss discretization parameter optimization heuristics based on the redundant rule elimination technique;
(5) demonstrate the implementation of the proposed technique in the interactive time series pattern mining workflow.

The rest of the article is organized as follows. Section 2 defines key terms and formally states the problem. Section 3 describes the techniques we build upon—the time series symbolic discretization and grammatical inference. We discuss the application of these techniques to the problem of time series variable length pattern discovery in Section 4. In Section 5, we show two solutions for pattern redundancy reduction and in Section 6, we describe our approach for automated discretization parameters selection. The proposed techniques are experimentally validated in Section 7. We conclude and discuss future work in Section 8.

2 NOTATION AND THE PROBLEM DEFINITION

To precisely state the problem at hand, and to relate our work to previous research, we shall define the key terms used throughout this article. We begin by defining our data type, time series:

Definition 2.1 (Time series). Time series $T = t_1, \ldots, t_m$ is a set of scalar observations ordered by time.

Since we focus on the detection of patterns which are the local structural features, we consider short subsections of time series called subsequences:

Definition 2.2 (Subsequence). Subsequence C of time series T is a contiguous sampling $t_p, \ldots, t_{p+n-1}$ of points of length $n << m$ where p is an arbitrary position, such that $1 \leq p \leq m - n + 1$.

Here, we introduce a notion of the *sliding window*. Typically, all subsequences of a time series T of the length m are extracted by sliding a window of the user-defined subsequence length n across T. This process, called *sliding window subsequence extraction* results in a set of $m - n + 1$ subsequences.

As it is well acknowledged in the literature, and as we have shown before in Lin et al. (2003), it is often meaningless to compare time series unless they are z-normalized:

Definition 2.3 (Z-normalization). Z-normalization is a process that brings the mean of a subsequence C to zero and its standard deviation to one and effectively enables the comparison of

time-series by their structural similarities (Goldin and Kanellakis 1995). A z-normalized representation of the time series T can be obtained by subtracting the mean of T from each value and dividing by its standard deviation:

$$T_{\text{znorm}} = \frac{T - \mu_T}{\sigma_T}. \tag{1}$$

Given two time series subsequences C and M, both of length n, the distance between them is a real number that accounts for how much these subsequences are different. The function which outputs this number when given C and M is called the *distance function* and denoted $Dist(C, M)$. One of the most commonly used distance functions is the Euclidean distance:

Definition 2.4 (Euclidean distance). Euclidean distance is defined as the square root of the sum of the squared differences between each pair of the corresponding data points in C and M:

$$Dist_{\text{Eucl}}(C, M) = \sqrt{(c_1 - m_1)^2 + (c_2 - m_2)^2 + \cdots + (c_i - m_i)^2 + \cdots + (c_n - m_n)^2}. \tag{2}$$

Some of our techniques are built upon determining if a given subsequence C is similar to other subsequences M under distance measure $Dist$. This notion is formalized in the definition of a match:

Definition 2.5 (Match). Given a positive real number t (i.e., threshold) and subsequences C and M, if $Dist(C, M) \leq t$ then subsequence M is a match to C.

When searching for patterns using a distance function, it is important to exclude self-matches, which are subsequences that overlap the subsequence currently being considered. Such self-matches can yield degenerate and unintuitive solutions as discussed in Keogh et al. (2005). For two subsequences C and M, we define a non-self-match:

Definition 2.6 (Non-self-match). Given a subsequence C of length n starting at position p of time series T, the subsequence M beginning at q is a non-self-match to C at distance $Dist(C, M)$ if $|p - q| \geq n$.

In this work, we discuss grammar inference-based algorithms designed to discover two types of time series patterns: frequent patterns, i.e., motifs, and rare or anomalous patterns, i.e., discords.

A time series motif is a frequently occurring pattern. Different definitions of motif exist in the literature. As an example, Lin et al. (2002, 2003) defined a motif as follows:

Definition 2.7 (Closest-Pair Time series motif). The closest-pair time series motif is the most similar pair of non-overlapping subsequences of length m (T_i^m, T_j^m), where $i < j$.

In this work, our definition of motif is not restricted to the closest pair, but rather, *a set* of similar (possibly overlapping) subsequences found in the dataset. This definition is closer to the one introduced in the original motif work (Patel et al. 2002). In addition, contrary to existing definitions, where similar subsequences in a motif must have the same length, this requirement is lifted in our definition, and, in addition to that we allow motifs to overlap.

Definition 2.8 (Approximate-Length Time series motif). The approximate-length time series motif is a set of similar subsequences $(T_{i_1}^{m1}, T_{i_2}^{m2}, \ldots, T_{i_k}^{mk})$ of lengths $m1, m2, \ldots, mk$, respectively, where $i_1 < i_2 < \cdots < i_k$.

As mentioned, one of the most effective methods for time series anomaly detection is via discord discovery. Formally, it is defined as follows:

Definition 2.9 (Time series discord). Given a time series T, the time series subsequence $C \in T$ is called the discord if it has the largest Euclidean distance to its nearest non-self-match (Keogh et al.

2005). Thus, time series discord is a subsequence within a time series that is maximally different to all the rest of subsequences in the time series, and therefore naturally captures the most unusual subsequence within the time series (Keogh et al. 2005).

2.1 Problem Definitions

The task of finding structural patterns in time series, i.e., time series motifs and discords, is defined as follows:

Given a time series T, find a subsequence pattern P that is the most frequently occurring structural phenomenon in the time series; and find a subsequence C that is the most (structurally) different from the rest of the observed subsequences.

Both tasks, however, are very difficult to solve in its general form without *a notion of the context* (Lin et al. 2007; Chandola 2009). The context is the information that can be induced from the structure of the dataset or specified as a part of the problem, and, essentially, defines a normal or expected signal behavior. The context information places constraints on both the search space and the results, making it possible to find a *meaningful* solution. Based on this rationale, we re-define the motif and anomaly discovery problems as: *Given a time series T and* some context , find the most frequent subsequence pattern $P \in T$, and find the most structurally different subsequence $C \in T$, both of which can be related to the context.

Traditionally, in motif and discord discovery (Lin et al. 2007; Chandola 2009), the context is provided by the user-defined pattern length, and the notion of the "most structurally different or similar" is defined as the largest Euclidean distance to the nearest non-self-match. Both constraints, while determining the task and the solution exactly, place severe restrictions on the result by assuming unrealistic *a priori* knowledge about the exact motif and anomaly lengths.

In this work, we address this issue in two ways. First, our algorithms allow the pattern length to vary in boundaries that are consistent with the time series context of correlating patterns (as it is determined by the grammar induction algorithm used). Second, we allow the user to control this context analysis process with our interactive graphical user interface (GUI) tool. Specifically, we expect the user to navigate the dataset via panning and zooming and to select the input time series fragment that satisfies some subjective criteria, such as the correspondence to the expected normal generative process. Given such a fragment, our tool finds the optimal parameter set which yields the most descriptive grammar about the fragment at the first step. It then applies these chosen parameters to the whole dataset, highlighting motifs and anomalies at the second step.

The learned context, however, depends on the time series span analyzed. Thus, if it contains long noisy or significantly different from the true or expected generative process structural fragments, the inferred (learned) context may fail to capture the true signal structure, and the technique will fail. Therefore, the proper input is crucial for the context learning success, and for this reason we implemented our techniques behind a GUI, which facilitates human-computer interaction, enabling high quality context selection through visual examination.

3 GRAMMAR-BASED TIME SERIES DECOMPOSITION

Before describing our approach in detail, consider the following example showing the context-free grammar properties used in our approach. Let

$$S = abc\ \ abc\ \ cba\ \ cba\ \ bac\ \ xxx\ \ abc\ \ abc\ \ cba\ \ cba\ \ bac$$

be the input string under analysis (e.g., derived from a time series and reflecting its structure). For reason that will become clearer later, the input string consists of a sequence of *words* (in this example, three-letter words or triplets). Each triplet is considered an atomic unit, or a *terminal*

Table 1. The SEQUITUR Grammar for the Input String S

Grammar rule	Expanded grammar rule
R0 → R1 xxx R1	abc abc cba cba bac xxx abc abc cba cba bac
R1 → abc abc cba cba bac	abc abc cba cba bac

Table 2. The RE-PAIR Grammar for the Input String S

Grammar rule	Expanded grammar rule
R0 → R4 xxx R4	abc abc cba cba bac xxx abc abc cba cba bac
R1 → abc abc	abc abc
R2 → cba cba	cba cba
R3 → R1 R2	abc abc cba cba
R4 → R3 bac	abc abc cba cba bac

in the sequence. The task is to find a hidden hierarchical structure in this sequence by grammar induction.

A careful look at the string shows that there are two repeated patterns *abc abc cba cba bac* separated by *xxx*. Ideally, we expect the grammar induction algorithm to reflect this, as shown in the grammar obtained by the SEQUITUR (Nevill Manning and Witten 1997) algorithm on the input S.

In addition to these two repeated sequences separated by *xxx*, an attentive reader shall notice that each of them consists of two sequences of paired words "abc abc" and "cba cba" that are followed by "bac"—a property that can potentially be used. Indeed, the second grammar inference algorithm implemented in our tool, called RE-PAIR, Larsson and Moffat (1999) exploits this property of the input string S in full.

As shown, both grammar induction algorithms successfully find repeated patterns in the input string and create hierarchical structures of grammar rules that reduce the length of the input string (i.e., compress it). Both grammars consist of grammar rules that are encoded by *non-terminals* revealing repeated patterns in the input. At the same time, the subsequence *xxx* is unique to S, and thus algorithmically *incompressible*, was not included in any of the grammar rules.

In our previous work (Li et al. 2012; Senin et al. 2015), we have shown that by the analysis of a grammar built upon time series discretization, it is possible to identify recurrent patterns, i.e., patterns that correspond to $R1$ in the SEQUITUR example and $R1$–$R4$ in the RE-PAIR example, as well as anomalous patterns, i.e., *xxx*, at the same time.

To illustrate this, suppose we annotate each word of the input string S with the number of rules that the word appears in, excluding the top-level rule R0. The input string S becomes the following in the case of the SEQUITUR application:

$$S = abc_1\ abc_1\ cba_1\ cba_1\ bac_1\ xxx_0\ abc_1\ abc_1\ cba_1\ cba_1\ bac_1.$$

Since all words except *xxx* occur in the grammar's single rule $R1$, they are all annotated with 1; since *xxx* is not a part of the rule, it is annotated with 0. In the case of RE-PAIR application, the annotation counts are different:

$$S = abc_3\ abc_3\ cba_3\ cba_3\ bac_1\ xxx_0\ abc_3\ abc_3\ cba_3\ cba_3\ bac_1.$$

Here, all occurrences of the words *abc* and *cba* have a count of 3 because they appear in the grammar's rules $R1$/$R2$, $R3$, and $R4$. The word *bac* has a count of 1 since it appears only in $R4$; whereas the word *xxx* has a count 0 because it is not a part of any rule ($R0$ is excluded from counting).

As empirically shown in our previous work (Senin et al. 2015), the annotation counts naturally reflect the algorithmic compressibility of terminal and non-terminal symbol sequences; specifically, we have shown that the compressibility correlates the frequency of a symbol's inclusion into the grammar rules. While most of the triplets in S are compressible to a higher degree depending on the grammar induction algorithm used, the triplet xxx_0 is algorithmically incompressible by any of the grammar induction algorithms and thus algorithmically random.

In turn, if the input string S is derived by discretizing a time series into a sequence of words, where each word corresponds to a time series subsequence extracted via a sliding window, then the subsequence in the time series that xxx represents is a potential anomaly, whereas other subsequences represent repeated patterns, i.e., time series motifs.

Note that when identifying potential motifs and anomalies we have not used any explicit distance computation between terminal or non-terminal symbols, grammar rules, or their corresponding (i.e., raw) subsequences. Moreover, note that the time series discretization technique SAX (Lin et al. 2003) and both grammatical inference algorithms Sequitur (Nevill Manning and Witten 1997) and Re-Pair (Larsson and Moffat 1999) also do not compute any distance (i.e., they do not explicitly measure how far apart objects are). Hence, unlike most of time series pattern discovery algorithms, our approach does not require any distance computation to discover and to rank multiple potential motifs and anomalies.

In the above example, potential motifs and anomalies are discovered and accounted for by a grammatical inference algorithm, which determines the pattern length automatically in the course of the grammar induction process—a property that contrasts with the rest of the time series pattern discovery algorithms that require the length of a potential pattern to be known in advance.

3.1 Time Series Symbolic Discretization

Since grammar induction algorithms are designed for discrete data, we begin by discretizing a continuous time series with SAX (Symbolic Aggregate approXimation) (Lin et al. 2003). In addition, since an anomaly is a local phenomenon, we apply SAX to subsequences extracted via a sliding window. SAX performs discretization by dividing a z-normalized subsequence into w equal-sized segments. For each segment, it computes a mean value and maps it to symbols according to a pre-defined set of breakpoints dividing the distribution space into α equiprobable regions, where α is the alphabet size specified by the user. This *subsequence discretization* process illustrated in Figure 3 outputs an ordered set of SAX words, where each word corresponds to the leftmost point of the sliding window, and which we process with numerosity reduction at the next step.

As an example, consider the sequence $S1$ where each word (e.g., *aac*) represents a subsequence extracted from the original time series via a sliding window and discretized with SAX (the subscript following each word denotes the starting position of the corresponding subsequence in the time series):

$$S1 = aac_1\ aac_2\ abc_3\ abb_4\ acd_5\ aac_6\ aac_7\ aac_8\ abc_9\ \ldots$$

In contrast to many SAX-based anomaly discovery techniques that store SAX words in a trie or a hash table for optimizing the search, and essentially throw away the ordering information, we argue that the sequential ordering of SAX words provides valuable *contextual information*, and is the key for allowing variable-length pattern discovery.

3.2 Numerosity Reduction

As we have shown in Lin et al. (2002), neighboring subsequences extracted via sliding window are often similar to each other. When combined with the smoothing properties of SAX, this phenomenon persists through the discretization, resulting in a large number of consecutive SAX words

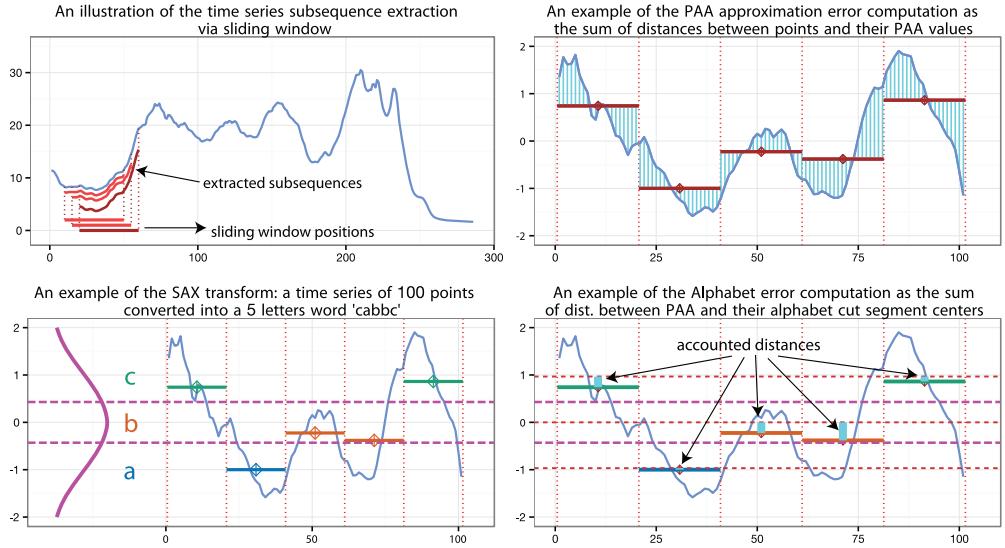

Fig. 3. The illustration of time series subsequence extraction via sliding window and SAX discretization (left panels) and the SAX transform approximation error computation (right panels, discussed in Section 6). The distance values considered are the vertical lines highlighted with cyan color.

that are identical. Later, these yield a large number of trivial matches, which significantly affect performance. To address this issue, we employ a numerosity reduction strategy: if in the course of discretization, the same SAX word occurs more than once consecutively, instead of placing every instance into the resulting string, we record only its first occurrence. Applied to $S1$, this process yields:

$$S2 = aac_1\ abc_3\ abb_4\ acd_5\ aac_6\ abc_9.$$

In addition to speeding up the algorithm and reducing its space requirements, the numerosity reduction procedure provides an important feature in this work—it naturally enables the discovery of variable-length motifs and anomalies as we show next.

3.3 Grammar Induction on SAX Words

Next, the reduced sequence of SAX words (from repetitions) is input into a grammar induction algorithm in order to build a context-free grammar. GrammarViz 3.0 implements two GI algorithms: Sequitur (Nevill Manning and Witten 1997), and Re-Pair (Larsson and Moffat 1999), which we shall briefly discuss in this section.

3.3.1 Sequitur Grammar Induction Algorithm. Sequitur is a linear time and space algorithm that derives the context-free grammar from a string incrementally. It is an *online algorithm* that does not need the access to the whole input sequence before building a grammar. Processing the input string from left to right, Sequitur builds the hierarchical structure of a context-free grammar by identifying and exploiting symbol correlations while maintaining the two constraints of uniqueness and utility at all times. Although simple in design, Sequitur has been shown to be competitive with the state of the art compression algorithms—the property that allows us to use the notion of Kolmogorov complexity for anomaly detection (Senin et al. 2015). In addition, Sequitur performance tends to improve with the growth of the input string size (Nevill-Manning and Witten 1997).

Table 3. The Quantitative Comparison of Sequitur and Re-Pair Grammars

		Sequitur grammar					Re-Pair grammar				
Dataset	Length	Rules	Intervals	Rule mean length	Rule frequency min–max	Rule cov.curve min–max	Rules	Intervals	Rule mean length	Rule frequency min–max	Rule cov.curve min–max
GPS Track	17,175	277	745	369.6	2–9	0–40	502	1,466	364.0	2–13	0–78
Dutch PD	35,039	251	1,413	829.1	2–38	1–55	229	2,249	825.6	2–46	0–75
ECG0606	2,299	60	218	139.5	2–13	2–20	61	320	137.9	2–18	1–29
ECG308	5,400	49	197	366.0	2–10	1–23	57	352	344.8	2–17	0–30
ECG15	15,000	115	526	365.9	2–24	0–21	108	618	347.8	2–32	0–24
ECG108	21,600	213	1,068	349.7	2–37	0–27	196	1,292	343.3	1–40	0–34
ECG300	536,976	2,376	35,740	386.3	2–766	1–133	1,654	36,935	387.2	1–1734	0–40
ECG318	586,086	1,260	29,075	489.0	2–1577	2–91	817	30,086	476.2	1–2073	0–32
Insect	18,667	411	1,967	146.4	2–32	1–27	356	2,128	143.5	1–54	0–27
NPRS43	18,019	729	3,300	135.6	2–62	1–40	660	3,488	134.1	2–111	0–44
NPRS44	24,125	950	4,501	135.9	2–66	0–39	822	4,697	134.3	2–139	0–44
TEK14	4,999	106	428	149.2	2–16	0–45	128	538	148.7	1–24	0–48
TEK16	4,999	92	346	150.1	2–17	0–40	118	497	147.8	1–18	0–40
TEK17	4,999	98	366	148.9	2–12	0–41	119	511	149.8	1–22	0–44
Video	11,251	150	1,064	167.3	2–62	0–27	123	1,320	161.4	1–64	0–30
Winding	2,499	79	198	131.9	2–7	1–18	231	550	127.9	2–7	2–46

Note: Re-Pair consistently produces more frequent rules than Sequitur.

When applied to a sequence of SAX words, Sequitur treats each word as an input string token and builds the context-free grammar's hierarchical structure. This structure recursively reduces all *digrams*—consecutive pairs of tokens (terminal or non-terminal)—occurring more than once in the input string to a single new non-terminal symbol.

3.3.2 Re-Pair Grammar Induction Algorithm. Re-Pair is a dictionary-based compression method invented in 1999 by Larsson and Moffat (1999). In contrast with Sequitur, Re-Pair is an *off-line algorithm* that requires the whole input sequence to be accessible before building a grammar. Similar to Sequitur, Re-Pair also can be utilized as a grammar-based compressor that discovers a compact grammar that generates the text (Charikar et al. 2005). As noted by the authors, when compared with online compression algorithms, the disadvantage of Re-Pair having to store a large message in memory for processing is "*illusory*" when compared with storing the growing dictionary of an online compressor, as the incremental dictionary-based algorithms maintain an equally large message in memory as a part of the dictionary (Larsson and Moffat 1999).

Re-Pair performs a recursive pairing step—finding the most frequent pair of symbols in the input sequence and replacing it with a new symbol—until every pair appears only once. The algorithm can be implemented in linear time and space (Larsson and Moffat 1999). Due to the simplicity and its off-line data processing paradigm, Re-Pair can be easily scaled over multiple CPUs, which is implemented in GrammarViz 3.0.

From our experience, both algorithms are suitable for recurrent and anomalous pattern discovery. The offline nature of the Re-Pair algorithm and its symbol-pairing design specificity typically result in creating grammars with much deeper hierarchy and larger *inter- or between*-rule length variance than those created with Sequitur, a property that sheds more light on the structure of the input data. At the same time, Sequitur grammar rules typically map to longer subsequences with higher *intra- or within*-rule length variance, which is useful in the variable length motif mining workflow. These distinctive properties are clearly illustrated in the examples above: for the same string S, Sequitur produced a single, five symbols long rule (Table 1), whereas Re-Pair created a grammar hierarchy of four rules whose length varies from 2 to 5 (Table 2). Table 3 and Figure 4 provide more evidence for these characteristic properties, as shown, Sequitur

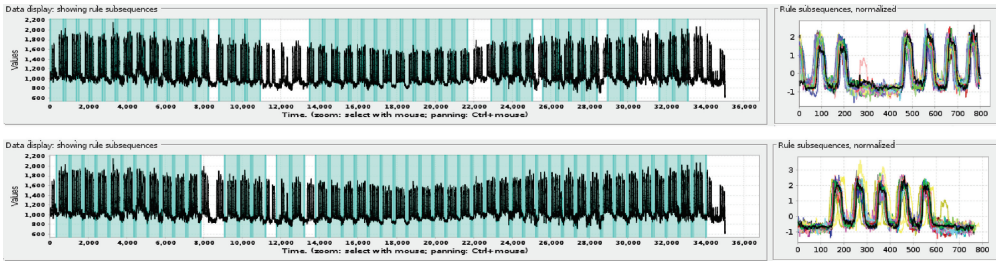

Fig. 4. The visual comparison of grammars built with SEQUITUR (top) and RE-PAIR (bottom) for the Dutch Power Demand dataset when using GrammarViz 3.0 GUI with sliding window of size 750, PAA size of 6, and Alphabet size of 3. The most frequent RE-PAIR rule covers 46 eight-day intervals with a pronounced pattern of five working days and also contrast abnormal weeks containing national holidays; the most frequent SEQUITUR rule covers only 38 eight-day intervals which are also more difficult to interpret. Right panels show superimposed time series intervals corresponding to the most frequent rule occurrences in the input time series).

consistently produces grammars with longer, less frequent rules, whereas RE-PAIR produces grammars with shorter rules but of higher occurrence frequency and deeper hierarchy (indirectly accounted for in the "Intervals" and the "Rule coverage curve" columns).

3.4 Variable Time Series Pattern Length

To reiterate the benefit of the numerosity reduction strategy and how it lends itself to variable-length pattern discovery with GI, consider the single grammar rule $R1$ generated by SEQUITUR from the string $S1$ as shown here:

Grammar rule	Expanded grammar rule
R0 → R1 abb acd R1	aac_1 abc_3 abb_4 acd_5 aac_6 abc_9
R1 → aac abc	aac abc

In this grammar, $R1$ concurrently maps to substrings of different lengths: $S1_{[1:3]}$ of length 3 (i.e., aac_1 aac_2 abc_3) and $S1_{[6:9]}$ of length 4 (i.e., aac_6 aac_7 aac_8 abc_9), respectively. The potential anomalous substring "abb_4 acd_5" has length 2. Since each SAX word corresponds to a *single point* of the input time series (a subsequence starting point), $R1$ maps to its subsequences of variable lengths.

3.5 Mapping Rules to Subsequences

As shown in the above example, by keeping the offsets of the SAX words throughout the procedures of discretization and grammar induction, our algorithm is able to map rules and SAX words back to their original time series subsequences.

4 PATTERN MINING WITH GRAMMAR-BASED TIME SERIES DECOMPOSITION

Previously in Li et al. (2012), we proposed GrammarViz, an algorithm for variable-length time series motif discovery that makes full use of the hierarchy in Sequitur's grammar. We showed the ability of the proposed algorithm to discover recurrent patterns of variable lengths. This is due to several properties of the algorithm, including the data smoothing capability of SAX, numerosity reduction that enables the discovery of variable-length patterns, and Sequitur's *utility* constraint that ensures that all of the grammar's non-terminals correspond to recurrent patterns. Figures 1 and 4 show the examples of time series motif discovery using our tool which implements the proposed algorithms.

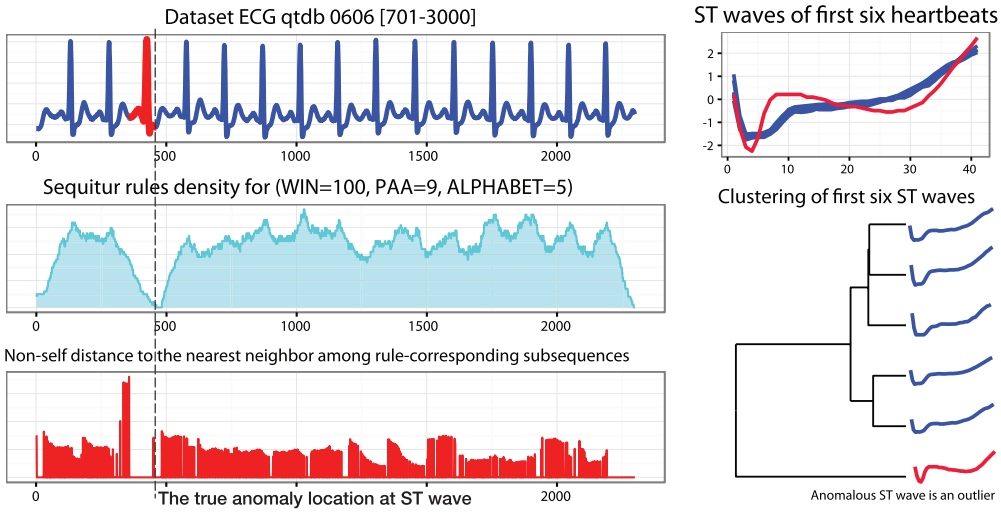

Fig. 5. Anomaly discovery in ECG0606 dataset. The left top panel shows the anomalous heartbeat location. The middle left panel shows that the rule density curve clearly identifies the true anomaly by its global minimum. The bottom left panel confirms that the RRA-reported discord has indeed the largest distance to its nearest non-self-match. The right top panel shows superimposed *z*-normalized subsequences of ST waves in the first six heartbeats, the anomalous wave is colored red; the bottom right panel shows their clustering with Euclidean distance.

In early GrammarViz 3.0 (i.e., GrammarViz 2.0) development iterations we also provided a pilot module demonstrating the potential of grammar-based time series decomposition for the time series anomaly discovery—the research direction that we later fully developed in Senin et al. (2015), where we formally introduced the notion of the grammar rule density curve, which is the key to our grammar-driven anomaly detection algorithms.

The rule density curve reflects the input time series compressibility and is built as follows: An empty array of length m (the length of the time series T) is first created. Each element in this array corresponds to a time series point and is used to keep count of the grammar rules that span (or "cover") the point. Second, since the locations of corresponding subsequences for all grammar rules are known, by iterating over all grammar rules the algorithm increments a counter for each of the time series points that the rule spans. After this process each element of the array contains a value indicating the total number of grammar rules that covers the corresponding time series point. The curve that corresponds to the array's values is the *rule density curve* that effectively marks each point of the input time series with its degree of normality/discordance with respect to a learned context.

Since each SAX string (which may or may not be a part of a grammar rule) corresponds to a subsequence starting at some position of the input time series, the points for which the rule density counters are the global minima correspond to the grammar symbols (terminals or non-terminals) whose inclusion in the grammar rules are minimal. These subsequences are algorithmically anomalous, and we consider the intervals in the rule density curve containing the minimal values potential anomalies. As an example, consider the rule density curves shown in the middle panel of Figure 5 where the global minima of the rule density curve pinpoints the true, annotated anomaly that is hard to detect even by visual inspection (Keogh et al. 2005).

The rule density curve properties can be explained by the algorithmic compressibility framework, where the *algorithmic (Kolmogorov) randomness* of a string has been defined through

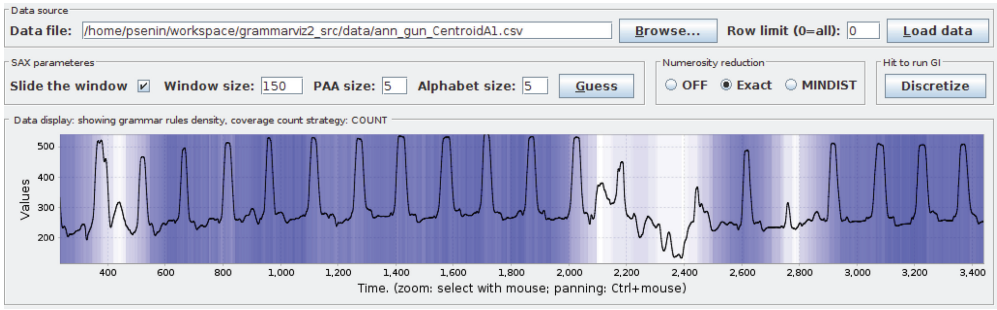

Fig. 6. Incorporating the rule-density-curve approach in GrammarViz 3.0. The varying degrees of shades in the background correspond to *rule density* curve values: the non-shaded (white) intervals pinpoint true anomalies, whereas darkly shaded intervals indicate overlaps between recurrent correlating patterns.

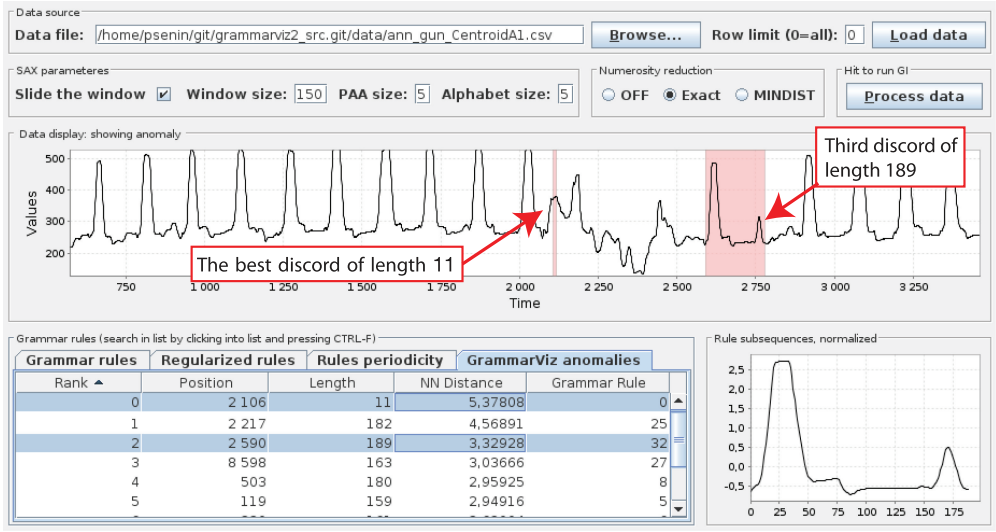

Fig. 7. Incorporating the RRA algorithm in GrammarViz 3.0. The screenshot shows its application to the video data telemetry. As shown, when configured with a window length of 150, RRA was able to detect multiple discords whose lengths vary from 11 to 189.

its incompressibility, i.e., the lack of algorithmically exploitable redundancy (Martin-Löf 1966; Grünwald 2007; Li and Vitányi 2013). Consider the role of the Sequitur *digram uniqueness* constraint, which ensures that none of the digrams processed by the algorithm (i.e., compressed into non-terminals) repeats itself. This property guarantees the *exhaustiveness of the search* for algorithmically exploitable redundancies in the input string, and consequently *asymptotically maximal compression* of the output string (Nevill-Manning and Witten 1997). Both properties put our approach within the Kolmogorov complexity framework based on the algorithmic compressibility and allow us to relate algorithmically incompressible subsequences to anomalies.

Figures 6 and 7 illustrate the implementation of our techniques for the rapid approximate time series anomaly discovery via rule density curve and for the exact time series discord discovery using Rare Rule Anomaly (RRA) algorithm, in which we rank anomaly *candidates* by their respective rule densities in search of exact, variable-length discords (Senin et al. 2015).

5 ORGANIZING GRAMMAR-BASED PATTERNS

As shown above, the grammar-based time series decomposition offers an advantageous capability to find patterns of different lengths. However, the hierarchical and the variable-length natures of the patterns mean that often they are numerous and overlapping. While this specificity aids time series anomaly discovery (Senin et al. 2015), the large number of overlapping time series motif candidates is difficult to examine visually. To simplify the navigation and to provide basic ranking mechanism, our tool implements interactive multiple-pattern selection capabilities and their ordering by the rule order, average rule length, rule frequency, rule use, and so on, and enables searching in the pattern table by the query. Nevertheless, the capacity of organizing rules in an intelligent way and a mechanism for their automated pruning are highly desirable.

In this section, we describe two simple heuristics designed to reduce the number of reported patterns. The first heuristic, performed via post-processing of candidate motifs, is based on the exclusion of highly similar, overlapping patterns found with clustering. In essence, we collect all subsequences corresponding to grammar rules into a mixed bag, eliminate overlapping patterns and organize the remaining patterns into a much smaller, *representative set* of motifs. The second heuristic is built similar to the greedy solution of minimum-cardinality set cover problem (an NP-hard problem) (Halperin and Karp 2005; Young 2008) and attempts to find the smallest set of rules that cover the most of the input time series in a greedy fashion.

5.1 Heuristic 1: Pruning Similar Overlapping Patterns

As the first step of this rule-reduction process, we pool all of the motif subsequences—that is, all subsequences that appear in at least one SEQUITUR rule—and sort them in ascending order based on their lengths. Next, based on the user-defined length-difference parameter, these subsequences are partitioned into several length groups. In the second step, we remove all the redundant subsequences within the same length group. A candidate subsequence is considered redundant if it overlaps with other subsequences. When there is an overlap found there are different ways to decide which subsequence to keep. A simple way is to remove the shorter one. Another method is to compare the importance *im* of points in each subsequence. Value *im* is set as the number of rules that point belongs to. Finally, we apply a Euclidean distance-based hierarchical clustering on the non-overlapping, similar-length subsequences to organize them into different pattern groups. Each pattern group (cluster) represents a motif. An example of clustered patterns is shown in Figure 8. There are only 25 pattern groups (motifs) after this post-processing procedure, compared to 252 from the original SEQUITUR rules.

Table 4 shows results of this rule-reduction technique application in our experimentation with 16 real datasets.

5.2 Grammar Cover

In order to formally describe our redundant rule pruning technique and later the automated parameters selection procedure, we shall define the grammar cover:

Definition 5.1 (Grammar cover). The grammar cover $Cover^G(T)$ of a grammar G inferred from a time series T is the ratio of two values—the number of time series points that are located within any of the subsequences corresponding to the grammar rules and the total length of time series.

5.3 Heuristic 2: Pruning Redundant Grammar Rules

While it is natural to consider the most frequent repeated time series intervals as a time series motif candidate, we observed that the most frequently occurred rules in $R0$ may not be the most informative due to the overlapping of their corresponding intervals. To mitigate this issue,

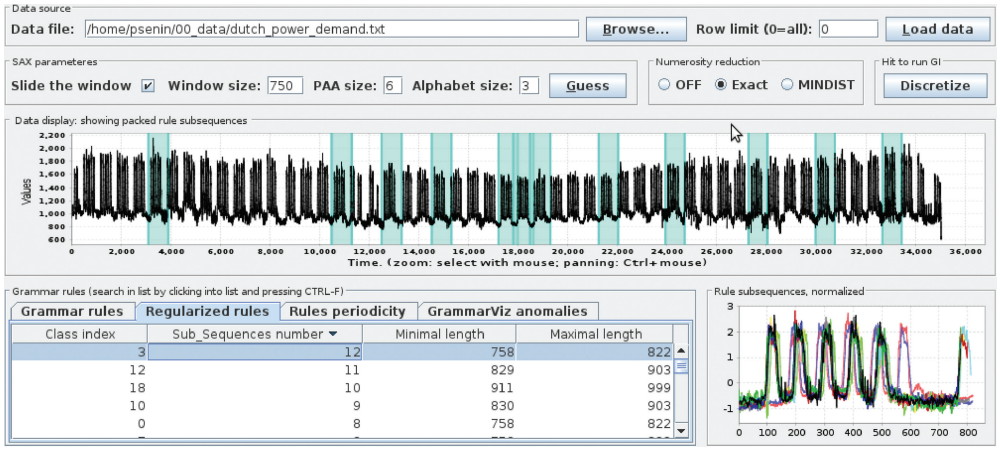

Fig. 8. Pattern groups after pruning and clustering.

Table 4. Comparison of Original Grammar, Organized Grammar, and Pruned Grammar

	SAX param.			Full Sequitur grammar			Organized grammar			Pruned grammar		
Dataset	Win	PAA	Alph	Rules	Max. freq.	Cover	Rules	Max. freq.	Cover	Rules	Max. freq.	Cover
GPS Track	350	15	4	278	9	0.9987	11	5	0.9310	29	9	0.9987
Dutch PD	750	6	3	252	38	1.0000	25	7	0.9707	9	38	1.0000
ECG0606	120	4	4	61	13	1.0000	14	7	0.9500	10	13	1.0000
ECG15	300	4	4	116	24	0.9996	25	8	0.9961	14	24	0.9996
ECG308	300	4	4	50	10	1.0000	17	8	0.9952	9	10	1.0000
ECG108	300	4	4	214	37	0.9972	14	8	0.9787	13	37	0.9972
ECG300	300	4	4	215	103	1.0000	49	9	0.9997	30	89	1.0000
ECG318	300	4	4	143	133	1.0000	49	12	0.9977	16	102	1.0000
Insect	128	4	4	412	8,156	1.0000	21	8	0.9939	72	32	1.0000
NPRS43	128	5	4	730	62	1.0000	3	5	0.9871	80	62	1.0000
NPRS44	128	5	4	951	66	0.9997	4	6	0.9926	42	66	0.9997
TEK14	128	4	4	107	16	0.9990	18	7	0.9864	14	12	0.9990
TEK16	128	4	4	93	17	0.9958	17	8	0.9668	13	17	0.9958
TEK17	128	4	4	99	12	0.9990	16	8	0.9830	16	12	0.9990
Video	128	4	4	151	62	0.9998	32	11	0.9859	13	62	0.9998
Winding	120	5	5	80	7	1.0000	15	5	0.8692	14	7	1.0000

we develop a grammar-rule pruning technique based on the greedy solutions (Young 2008) to minimum-cardinality set cover problem (Halperin and Karp 2005). It is a greedy algorithm that selects a minimal set of grammar rules that provide the maximum grammar cover.

The pruning algorithm is shown in Algorithm 1. Its input consists of the grammar G describing the input time series, and the time series length m. The algorithm outputs a reduced grammar covering the same time series. The pruning process is wrapped into the outer loop (lines 2–29) that breaks in two cases: (i) when the whole time series span is covered by selected grammar rules or (ii) when it is impossible to extend the cover (lines 12–14). Two consecutive over-the-rules loops are inside the main outer loop. In the first loop (lines 4–11), the algorithm finds the rule that extends the current cover at most. In the second loop (lines 16–27), the algorithm performs a check on whether the addition of the new rule makes any of the previously selected rules obsolete—in that case the redundant rule is removed from the selection and set aside. Upon finishing the pruning process, selected rules are partially expanded, following the exclusion of some of the initial rules (line 30), and the resulting grammar is returned.

ALGORITHM 1: PruneRedundantGrammarRules

```
Data: Grammar G (a set of rules), length of the input time series m
Result: Grammar GR
range ←array[0..m]; usedRules ← ∅;
while HasZeroes(range) do
    bestRule ← Null; bestDelta ← −Infinity;          /* search for a rule that extends */
    for r ∈ G do                                      /* current cover the most */
        if r ∉ usedRules & r ∉ removedRules then
            delta = CoverDelta(range , r);
            if delta > bestDelta then
                bestDelta = delta; bestRule = r;
            end
        end
    end
    if bestDelta < 0 then                             /* break the pruning process */
        break;                                        /* if unable to extend the cover */
    end
    usedRules ← r; continueSearch ← True;
    while continueSearch do                  /* check if the adding of a new rule makes */
        continueSearch ← False;                       /* the old selection redundant */
        for r ∈ usedRules do
            intervalsA ← RuleIntervals(r);
            intervalsB ← RuleIntervals(usedRules −r);
            if intervalsA covered by intervalsB then
                removedRules ← r;      /* remember the redundant with selection rule */
                usedRules ← usedRules −r;     /* remove redundant rule from selection */
                continueSearch ← True;
            end
        end
    end
    range ← UpdateRanges(range, r);
end
GR ← RulesAsGrammar(usedRules, G);
```

5.4 The Intuition Behind the Rule Pruning Algorithm

The intuition behind this algorithm is simple—since our task in hand is to find maximally repeated *and* minimally-overlapping subsequences (which we consider the most informative), at each iteration, as the best candidate we select the rule that covers the most of the uncovered-so-far time series span thus naturally provides much information about its structure.

A notable property of our grammar pruning algorithm is that it searches for a minimal subset of grammar rules that *has the same cover as the full grammar*, which relates it to a series of algorithms dealing with serial episodes mining from event sequences (Tatti and Vreeken 2012; Van Leeuwen and Vreeken 2014; Lam et al. 2014) that employ coding tables to find a minimal set of episodes for the observed sequence description in a succinct and characteristic manner. Also note that these and our technique are built upon the similar foundation—the Minimal Description Length (MDL) (Grünwald 2007) and Kolmogorov complexity (i.e., algorithmic compression) (Li and Vitányi 2013) formalisms.

6 AUTOMATED DISCRETIZATION PARAMETERS SELECTION

The problem of discretization parameter optimization for time series anomaly and frequent pattern discovery remains unsolved to the best of our knowledge. In this work, for the first time, we propose a semi-automated solution whose performance quality increases with the user participation. The proposed solution is based on the inherent discretization and grammar inference process properties, which we shall discuss.

6.1 Discretization Granularity Effect on the Grammar Size and Complexity

The discretization process discussed in the previous sections depends on three parameters: sliding window size, PAA size, and the alphabet size. To investigate the effect of these parameters on the grammar's quantitative characteristics, we conducted a series of exploratory experiments using 16 datasets shown in Table 3. These datasets represent a variety of generative processes: the GPS track dataset is transformed with a Hilbert space-filling curve log of the daily commute trajectories (Senin et al. 2015); the Dutch PD dataset represents the yearly energy consumption by the research facility (Van Wijk and Van Selow 1999); ECG datasets correspond to various phenomena observed in the ECG data (Goldberger et al. 2000); Insect dataset reflects the insect feeding behaviors (Mueen et al. 2009), NPRS data represent respiration rhythms (Keogh et al. 2005), TEK (Keogh et al. 2005) and Winding datasets are from industrial processes telemetry, and Video data is a telemetry stream from a video recording (Keogh et al. 2004).

First, we investigated the effect of discretization granularity on the time series approximation error. For this, we designed an error function that is the sum of two error values computed for each of the subsequences extracted via a sliding window. These error values are (i) the PAA approximation error and (ii) the SAX transform approximation error, which are schematically shown at Figure 3. As shown, the PAA approximation error is the normalized sum of Euclidean distances between time series points after z-normalization and their corresponding PAA values:

$$\text{Error}_{\text{PAA}}(C_{p,p+n}) = \frac{\sum_{i=p}^{i<p+n} \sqrt{(C_i - PAA_{C_i})^2}}{n}, \tag{3}$$

where C is the subsequence of the time series T extracted via sliding window of length n. The SAX approximation error is the normalized sum of distances between the PAA values and the centers of SAX alphabet cut segments to which this values belong:

$$\text{Error}_{\text{Alphabet}}(C_{PAA^k}) = \frac{\sum_{i=0}^{i<k} \sqrt{(C^i_{PAA^k} - CutCenter(C^i_{PAA^k}))^2}}{k}, \tag{4}$$

where k is the PAA size and C_{PAA^k} is the PAA transform of C. The value of the $CutCenter(C^i_{PAA^k})$ is taken from the table of precomputed values similar to the SAX cut lines table.

The intuition behind this error function design is that it accounts for an averaged distance between a time series point and its SAX representation over the whole time series. Naturally, as the values of PAA and Alphabet parameters of SAX transform increase, the approximation error decreases, as shown in Figure 9 where the surface-composing values are averaged for sliding windows ranging from 30 to 400.

At the same time, as the PAA and the Alphabet values increase, the amount of SAX words and their diversity also increase, causing the total number of rules in the resulting grammar to grow. Figure 10 indirectly shows that dependency by the strong negative correlation between the approximation error and the total number of rules in the grammar.

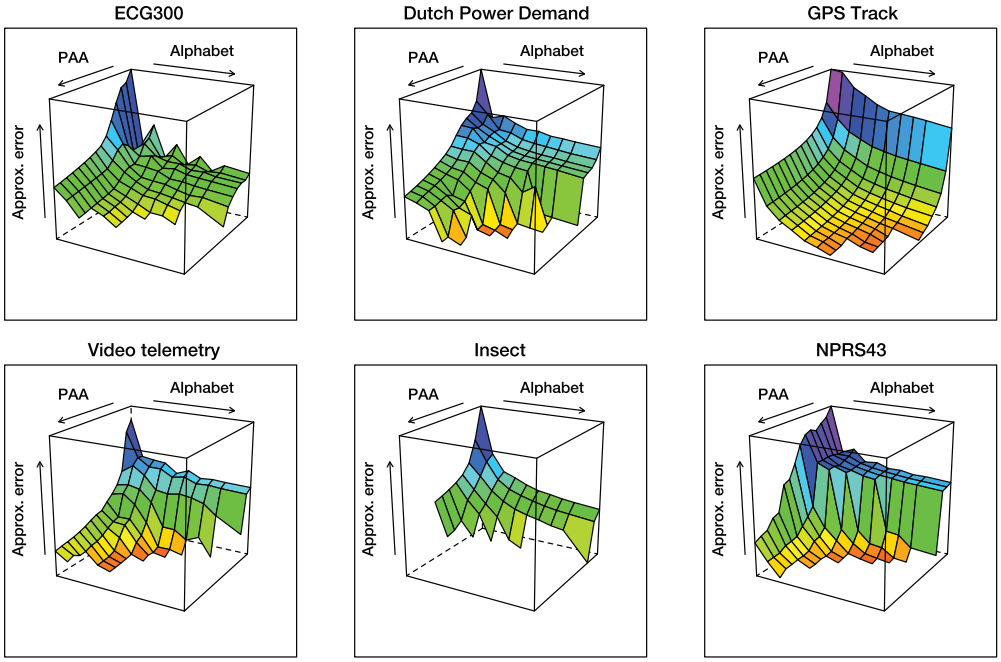

Fig. 9. The illustration of the discretization granularity effect on the approximation error. As the PAA and Alphabet parameters values increase, the approximation error decreases. The dependency is more pronounced for longer and more regular time series (top row plots).

6.2 The Grammar Reduction Coefficient

Above, we proposed a grammar pruning procedure that is designed to eliminate redundant grammar rules that do not contribute to the time series grammar cover. We propose to use the ratio of rules in the pruned grammar to the total number of rules as the discriminative function for selecting the set of optimal discretization parameters:

$$\text{Reduction Coefficient} = \frac{\text{Number of rules in the pruned grammar}}{\text{Number of rules in the full grammar}} \tag{5}$$

Specifically, in order to select the optimal discretization parameters set for a time series under analysis, we propose the following process:

- —First, a parameter learning interval is chosen from the input time series. If the input time series is short, its whole span can be used; if it is long (tens of thousands of points), it is advisable to select a shorter interval to speed-up the sampling process. In addition to that, it is advisable to choose an anomaly- and noise-free interval that reflects the expected generative process in order to avoid learning biases.
- —Second, a range of acceptable discretization parameter values is specified. For example, for a sliding window length, the acceptable range can be from 10 to a doubled length of a typical structural phenomenon observed in the time series. For PAA, a typical range can be from 2 to 50 (assuming that the sliding window length is more than maximal PAA value); and for the alphabet from 2 to 15.
- —Third, for each of the parameter combinations within the specified ranges, a grammar is inferred and pruned, in order to compute the reduction coefficient value. However, during

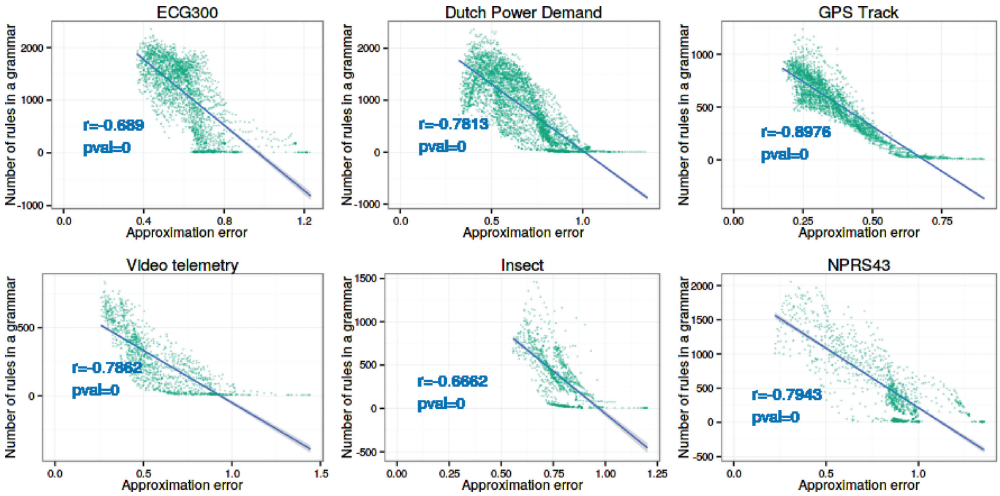

Fig. 10. The correlation between the approximation error and the total number of rules in a grammar. Larger grammars tend to have a small approximation error (which is linked to the high discretization granularity).

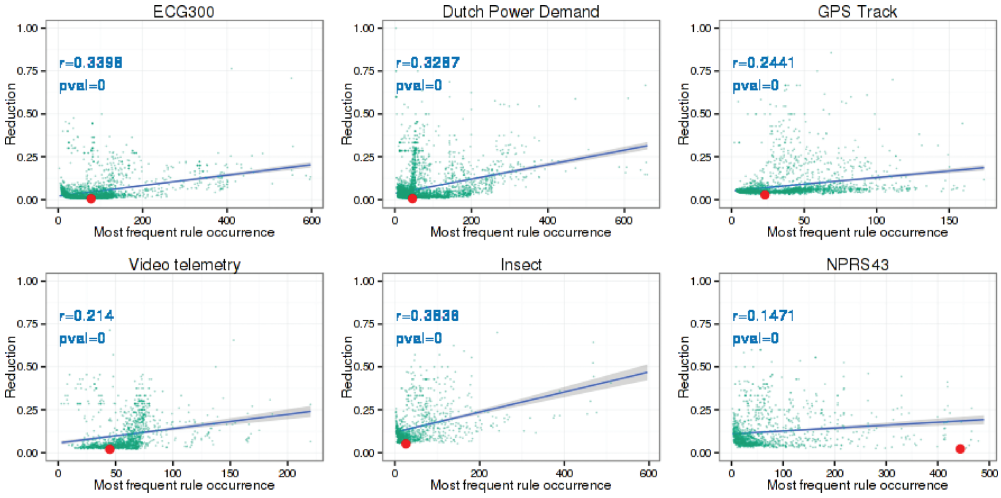

Fig. 11. The correlation between the most frequent rule occurrence and the reduction coefficient. Note, that while the correlation coefficient is positive in general, the minimal reduction value (marked by the red dot) is not the leftmost one and, from our empirical evaluation, tends to correspond to the value that approximately equals to the most frequently observed time series structural phenomenon (i.e., motif).

this sampling, any grammar whose cover falls below a fixed threshold—usually in a range from 0.9 to 1.0 depending on the expectation for an anomalous ranges fraction—shall be discarded as not describing the input time series in full and the corresponding parameters combination is marked as invalid.

—Finally, among all sampled and valid combinations, we select the one that yields the minimal value of the reduction coefficient as the optimal discretization parameter set.

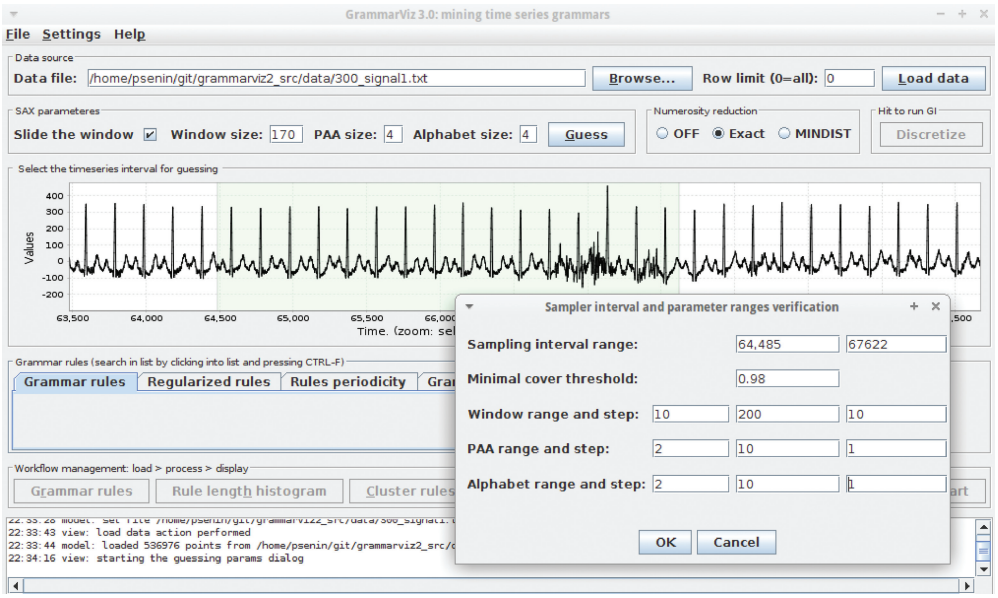

Fig. 12. The illustration of the parameters optimization module. After the user selected the short subsection of the very long time series as the sampler input, GrammarViz 3.0 runs a validation dialog in order to verify the interval selection, cover threshold, acceptable discretization parameter ranges, and the sampling grid step.

An explanation of our target function design and the parameter optimization process can be that the grammar reduction coefficient reflects a number of properties. First, it decreases when the denominator—a number of rules in the grammar—increases, which happens when the discretization parameters grow and *the approximation error decreases* (Figure 10). This growth, however, is limited by the cover threshold, which makes sure that the correlations between words are occurring and it is possible to describe the time series by the grammar's hierarchical structure in full. Second, the grammar reduction coefficient also decreases when the numerator—the number of the rules in the reduced grammar—decreases, which happens when *the small number of non-redundant rules describes the time series in full* (again, according to the desired cover threshold). Thus, by the design, the minimal grammar reduction coefficient value corresponds to a parameters set that allows to describe the input time series in full with the minimal approximation error, and which, at the same time, allows to reduce the grammar's hierarchical structure to only a few non-redundant rules capturing repeated structural phenomena within the input signal (Figure 11).

We implemented the parameters optimization workflow in GrammarViz 3.0 as shown in Figure 12. The typical interactive workflow scenario for the parameter optimization consists of few steps that can be repeated if necessary: loading the dataset into GrammarViz 3.0 GUI, exploring it using panning and zooming, selecting a dataset section that reflects the expected generative process, configuring the sampling grid density, and running the sampler.

6.3 Parameters Optimization Algorithm's Complexity

The automated parameters selection algorithm proposed in this work consists of two steps procedure run for each of the user's defined values of sliding window size (W), PAA reduction coefficient (P), and the SAX alphabet size (A). The first step of the procedure is the grammar in-

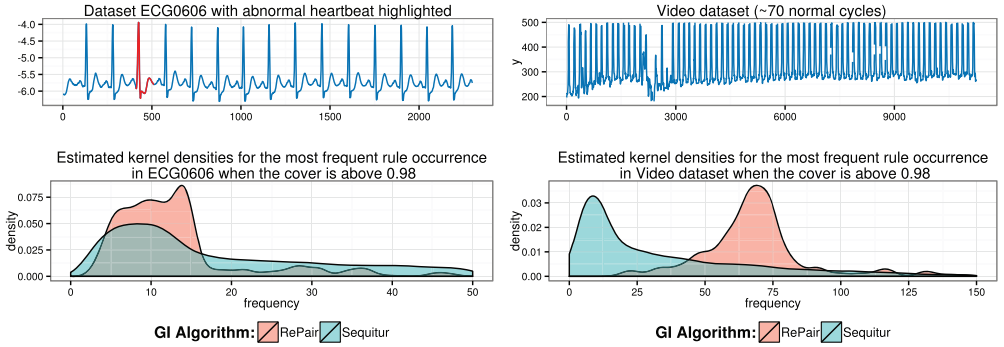

Fig. 13. The illustration of the used GI algorithms performance difference when using datasets with many recurrent subsequences and few anomalies. As shown, Re-Pair is much more likely to yield a grammar where the most frequent rule occurrence is equal to the occurrence of the most frequently observed structural phenomena, such as a heartbeat or a telemetry cycle.

duction with Re-Pair, which is of quadratic complexity $O(n^2)$ where n is the size of the input time series. The second step of the optimization procedure is the rule pruning step of following complexity: if m is the number of rules in the Re-Pair grammar, the pruning process is progressively reduces this number to the k irreducible rules—a stepwise process with (sorting) complexity $m * \log m + (m-1)\log(m-1) + \cdots + (m-k)\log(m-k)$, which is in the worst case (i.e., $k = 1$) has a complexity of $O(m^2 \log m)$. Since the both steps are run for each of the user-defined combination, the optimization algorithm's complexity is $W * P * A * (O(n^2) + O(m^2 \log m))$ or $O(C(n^2 + m^2 \log m))$, where the value of C depends on the user's input.

7 EMPIRICAL VALIDATION

From our experience with the target function design, the proposed parameter optimization process works best when the Re-Pair GI algorithm is used, in this case GrammarViz 3.0 parameter optimization workflow is able to find a discretization parameter combination that allows to capture the most frequently occurring structural anomaly. Consider Figure 13 that shows examples of two datasets, ECG0606 and Video telemetry, and compares kernel density estimates for the most frequent rule of Sequitur and Re-Pair grammars. The sampling shown was done over the parameters grid with the range of parameters for the sliding window size [30–460], PAA [2–30], and the Alphabet [2–16], while the minimal grammar cover threshold was set to 0.98. Obviously, the Re-Pair algorithm is much more likely than Sequitur to produce a grammar whose most frequent rule occurs the same amount of times as the most frequently observed time series structural phenomenon. Figures 15 and 16 also support our claim that the proposed automated parameters selection process based on the Re-Pair algorithm is likely to find a parameters set that allows to capture the time series structural phenomena that is the most frequent and that contributes to the cover value the most.

We did not formally evaluate our technique performance in time series motif discovery when compared with other techniques. The technical reason for that is the specificity of our symbolic discretization and grammar inference-based technique, which essentially reports correlation of symbolic sequences that we map back to the input time series subsequences. Due to this specificity we had to significantly relax the definition of time series motif by allowing variable length and overlap, which makes our technique incomparable with exact motif discovery techniques such as Mueen and Chavoshi (2015), Mohammad and Nishida (2014), or Mueen et al. (2009). In addition

Table 5. The Optimal Discretization Parameters Selected by Our Approach

	Optimal SAX			Grammar	Full grammar			Pruned grammar			Reduction
Dataset	Win	PAA	Alphbt	Approx. err.	Rules	Max. freq.	Cover	Rules	Max. freq.	Cover	coefficient
GPS Track	330	30	2	0.4656	393	23	0.9997	11	23	0.9997	0.0280
Dutch PD	460	30	2	0.5480	1397	46	0.9999	9	46	0.9999	0.0064
ECG0606	180	22	3	0.5383	152	14	0.9961	4	14	0.9961	0.0263
ECG15	220	22	2	0.6628	825	59	0.9988	10	59	0.9988	0.0121
ECG308	440	30	3	0.5113	246	13	0.9994	4	13	0.9994	0.0163
ECG108	220	4	12	0.5705	711	38	0.9989	25	35	0.9989	0.0352
ECG300	330	20	2	0.7088	1414	78	0.9999	8	78	0.9999	0.0057
ECG318	280	22	3	0.5233	1551	59	0.9999	14	58	0.9999	0.0090
Insect	160	4	9	0.6552	696	27	0.9993	36	26	0.9993	0.0517
NPRS43	30	30	2	0.3982	831	444	0.9998	18	444	0.9998	0.0217
NPRS44	30	26	2	0.4499	1158	620	0.9998	18	620	0.9998	0.0155
TEK14	360	24	8	0.3759	267	7	0.9860	7	5	0.9860	0.0262
TEK16	360	24	4	0.5092	190	17	0.9620	5	12	0.9620	0.0263
TEK17	340	20	8	0.4128	248	25	0.9884	6	6	0.9884	0.0242
Video	200	6	12	0.4203	621	45	0.9556	12	45	0.9556	0.0193
Winding	280	30	2	0.6617	456	3	0.9996	5	3	0.9996	0.0110

Note: these values computed for Re-Pair-generated grammars.

to that, with our approach, by simply adjusting the sliding window size and the discretization granularity (manually or in automated fashion) we can easily achieve the performance in frequent pattern discovery demonstrated by other approximate motif discovery techniques such as Grabocka et al. (2016) or Chiu et al. (2003).

We evaluated the automated parameter selection procedure performance in the task of time series anomaly selection. First, we run the sampler using the same parameters ranges and the minimal cover threshold as in the above experimentation except that we increased the upper bound for the Dutch Power Demand dataset up to 900 points. The selected parameters are shown in Table 5 along with quantitative characteristics of resulting grammars. Second, we run our RRA algorithm (Senin et al. 2015) implementation using these parameters with Re-Pair algorithm. The comparison of the discords discovered this way and HOT-SAX (Keogh et al. 2005) discords is shown in Table 6—our automated discord discovery approach was able to find the true discord in 14 out of 16 test cases. In two unsuccessful attempts, the second-ranked discord coincides with the true discord—an issue that is known to happen because we use normalized Euclidean distance (by subsequence length) for exact RRA discord discovery (Senin et al. 2015).

7.1 Sensitivity Analysis

In order to evaluate the proposed algorithm behavior, and specifically its sensitivity to the noise in input data, we conducted an experiment using a synthetic dataset. The dataset was generated by planting anomalies of various types into a sine curve and adding a random trend and a various levels of noise to the resulting time series. For each of the synthetic datasets, the experiments were conducted by using sampling intervals of various length for the optimal parameters discovery followed by the discord discovery procedure.

Figure 14 shows the synthetic dataset construction: the sine wave, five randomly planted anomalies (from left to right: (1) max signal level for an interval of length π, (2) signal compression for an interval of 3π, (3) signal expansion (delay) for an interval of 1.5π, (4) no signal for an interval of length π, and (5) min signal level for an interval of length π). The added trend curve was generated by a random walk, noise was generated using Gaussian distribution and scaled by 0.1 for 10% noise dataset, 0.2 for 20%, and so on. The planted discords are highlighted in each of the six resulting datasets used for evaluation.

Table 6. The Illustration of Automated Discretization Parameters Optimization Performance When Applied to the Problem of Time Series Discord Discovery Using RRA Algorithm

	Hand-picked discretization parameters			HOTSAX discord	Automatically discovered discretization parameters			RRA discord		Both discords
Dataset	Window	PAA	Alphabet	position	Window	PAA	Alphabet	Position	Length	overlap
GPS Track	350	15	4	4,657	330	30	2	4,666	331	94.6%
Dutch PD	750	6	3	33,980	460	30	2	33,967	461	59.7%
ECG0606	120	4	4	391	180	22	3	1,294	181	0.0%**
ECG15	300	4	4	2,266	220	22	2	2,120	221	25.0%
ECG308	300	4	4	2,278	440	30	3	2,225	441	100.0%
ECG108	300	4	4	10,699	220	4	12	10,840	221	53.0%
ECG300	300	4	4	54,859	330	20	2	54,809	331	93.7%
ECG318	300	4	4	307,102	280	22	3	307,113	281	97.3%
NPRS43	128	5	4	14,817	30	60	2	14,861	31	24.2%
NPRS44	128	5	4	20,458	30	26	2	20,533	31	24.2%
Video	150	5	3	2,302	200	6	12	2,305	201	98.0%
TEK14	128	4	4	1,091	360	24	8	72	361	0.0%**
TEK16	128	4	4	4,253	360	24	4	4,061	361	100.0%
TEK17	128	4	4	2,101	340	20	8	2,037	341	100.0%

**In these two cases, the second discord discovered with RRA coincides with the HOTSAX discord with the overlap of 91.7% for ECG0606 and 84.4% for TEK14.
Note: We consider HOTSAX discords to be true discords.

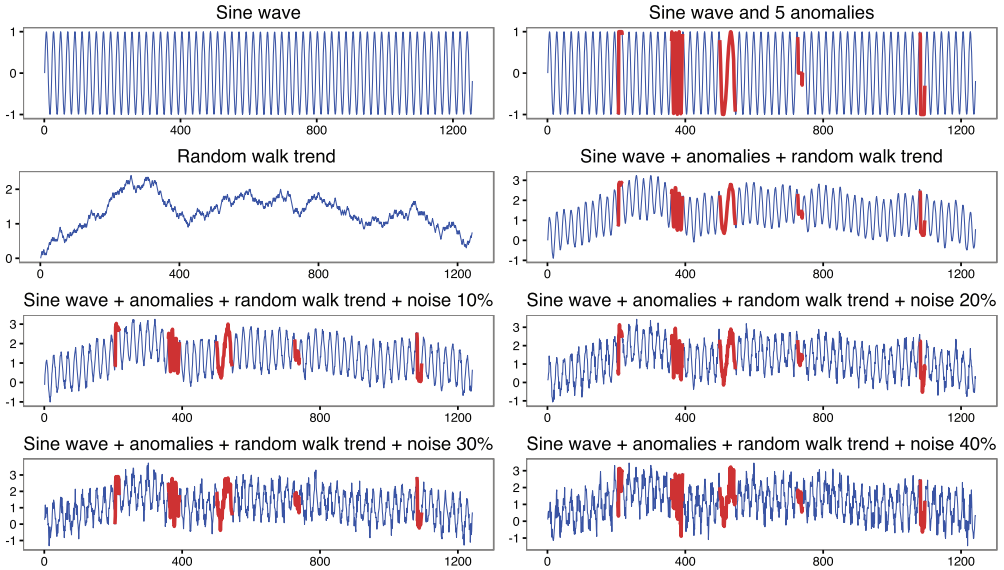

Fig. 14. Synthetic datasets used in experimental evaluation: five different anomalies were planted into a sine wave modified with a random walk-based trend and added Gaussian noise.

For each of the six datasets, the following experimental procedure was performed: (i) a sampling interval from the range [745–1070] (we used increment 5 for constructing 64 sampling intervals of length 10, 15, . . . , 325) was selected as the input for the parameters optimization procedure where sliding window was bounded by the range from 10 to the maximal possible window and PAA and alphabet values were bounded by the range of [2–12]; (ii) parameters optimization procedure discussed earlier was performed and the best parameters discovered; (iii) the best parameters set was used for the discord discovery with RRA algorithm. Five best discords reported by RRA were

Fig. 15. The illustration of the successful parameters selection with the proposed technique for four distinct in their generative process datasets: Respiration (NPRS, top left), Dutch Power Demand (top right), Video telemetry (bottom left), and ECG (bottom right). The selected parameters highlight the most repeated and explainable structural pattern in these datasets.

Table 7. The Results of Experimental Evaluation

	Discord id and its discovery success count (out of 64 tries)					Discord discovery rate		Mean discretization values (64 tries)		
Dataset	#1	#2	#3	#4	#5	μ	σ	Win	PAA	Alphb
Sine wave and anomalies	40	64	64	64	53	4.5	0.5	29.1	6.2	5.4
Sine wave, anomalies and trend	52	63	34	53	36	3.7	0.8	28.0	6.1	2.9
Sine wave, anomalies, trend + 10% noise	56	63	43	50	35	3.9	0.8	27.8	4.8	3.3
Sine wave, anomalies, trend + 20% noise	58	64	54	50	30	3.3	0.6	29.0	4.7	3.3
Sine wave, anomalies, trend + 30% noise	52	63	32	53	18	3.4	0.8	28.0	4.9	3.0
Sine wave, anomalies, trend + 40% noise	45	64	8	48	7	2.7	0.6	31.6	3.7	3.7

Note: For each of 6 dataset from Figure 14 we ran an automated parameters discovery followed by the discord discovery procedure (with RRA Algorithm). Count of GrammarViz 3.0 success in each of planted discords discovery is shown as a count. Also shown the discord discovery rate and the mean discretization parameters values.

evaluated for an overlap with the planted anomalies—in the case of an overlap we accounted for a successful planted anomaly discovery.

Table 7 shows the results of the experimentation. First, as it was expected, the discord discovery rate decreased with the noise level growth. Second, we noticed that the parameters optimization procedure was able to adequately adapt to the increasing noise levels by coarsening the discretization: the mean PAA value (the SAX word length) decreased from 6.2 to 3.7, the mean Alphabet size decreased from 5.4 to 3. Third, we noted the differences in discord discovery rates among different anomaly types. Specifically, the signal compression anomaly (#2) was discovered in almost all cases, followed by the signal loss anomaly (#4), maximum signal value (#1), signal expansion (#3), and the minimal signal value (#5) that was the most difficult to discover for our technique. Finally, we noted that the mean discretization parameters values computed for all 64 runs of GrammarViz 3.0 enable all five anomalies discovery, which is an additional evidence that in an average case our approach is able to successfully discover close to the optimal discretization parameters.

Fig. 16. The illustration of less explainable parameters selected by the proposed technique. Top row shows the Winding (left) and Insect (right) datasets—we unable to explain the nature of the discovered motifs without understanding of the generative processes. The bottom row shows patterns in TEK16 and GPS Track datasets—in TEK, the approach picked overlapping patterns due to the high amount of low-amplitude noise, whereas in the GPS track the most frequent pattern corresponds to the Hilbert transform artifact that results in a step as the trajectory travels through Hilbert cell boundaries.

8 CONCLUSION AND FUTURE WORK

In this work, we propose a grammar decomposition-based, interactive methodology to find variable length time series patterns. Specifically, our technique enables discovery of variable-length approximate time series motifs and variable-length exact and approximate time series anomalies.

In contrast with previous effort, this work proposes solutions for two problems associated with the grammar-based pattern mining framework—the candidate pattern numerosity and the optimal discretization parameters selection. Specifically, we have shown two complementary heuristics for pattern numerosity reduction: one of them is based on the pattern similarity and aids discovery of structurally similar non-overlapping time series subsequences, whereas another heuristics, based on the greedy minimum cardinality maximum cover principle, aids the discovery of the minimal set of frequent, overlapping patterns which explain the time series hierarchical structure at most. Our discretization parameters optimization technique is based on the latter numerosity reduction technique and guides the discretization parameters search toward a set of parameters that provides a maximal reduction of the full grammar rules amount while preserving the grammar cover, or, in other words, captures the most of structural recurrent patterns in the smallest amount of grammar rules.

We have implemented all proposed techniques in an interactive visual time series pattern mining tool called GrammarViz 3.0 that extends their capabilities by enabling the user to control the process and visually validate the results. With its use we obtained empirical results, which confirm that the grammar-based approach with automated parameters selection is capable to find important motifs and discords.

Many future directions are possible. First, we would like to explore other rule-pruning heuristics. Second, we would like to investigate the performance of other grammar inference algorithms, such as SEQUENTIAL (improved SEQUITUR), BISECTION, and GREEDY (Charikar et al. 2005). We also plan to

improve our parameters optimization process speed by utilizing the DIRECT algorithm (Jones et al. 1993). For the visualization tool, we would like to enhance the rule ranking and filtering feature.

APPENDIX

The GrammarViz 3.0 source code is open source and is available at https://github.com/GrammarViz2/grammarviz2_src; the online tutorials are available at http://grammarviz2.github.io/grammarviz2_site/.

REFERENCES

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *SIGMOD Record* 22, 2 (June 1993), 207–216. DOI: http://dx.doi.org/10.1145/170036.170072

Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*. IEEE Computer Society, Washington, DC, 3–14.

Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'08)*. Eurographics Association, Aire-la-Ville, Switzerland, 117–126.

Jeremy Buhler and Martin Tompa. 2002. Finding motifs using random projections. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology* 9, 2 (2002), 225–242. DOI: http://dx.doi.org/10.1089/10665270252935430

Nuno Castro and Paulo J. Azevedo. 2010. Multiresolution motif discovery in time series. In *Proceedings of the SDM*. SIAM, 665–676.

Varun Chandola. 2009. Detecting anomalies in a time series database Varun Chandola, Deepthi Cheboli, and Vipin Kumar. (2009).

Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. 2005. The smallest grammar problem. *IEEE Transactions on Information Theory* 51, 7 (2005), 2554–2576.

Bill Chiu, Eamonn Keogh, and Stefano Lonardi. 2003. Probabilistic discovery of time series motifs. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*. ACM, New York, NY, 493–498. DOI: http://dx.doi.org/10.1145/956750.956808

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids* (1st ed.). Cambridge University Press.

Aristides Gionis and Heikki Mannila. 2003. Finding recurrent sources in sequences. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB'03)*. ACM, New York, NY, 123–130. DOI: http://dx.doi.org/10.1145/640075.640091

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. 2000. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (2000), e215–e220.

Dina Q. Goldin and Paris C. Kanellakis. 1995. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*. Springer-Verlag, London, UK, 137–153. http://dl.acm.org/citation.cfm?id=647484.726176.

Josif Grabocka, Nicolas Schilling, and Lars Schmidt-Thieme. 2016. Latent time-series motifs. *ACM Transactions on Knowledge Discovery from Data* 11, 1 (2016), 6.

Peter D. Grünwald. 2007. *The Minimum Description Length Principle*. MIT press.

M. Gupta, Jing Gao, C. C. Aggarwal, and Jiawei Han. 2014. Outlier detection for temporal data: A survey. *Knowledge and Data Engineering, IEEE Transactions on* 26, 9 (Sep. 2014), 2250–2267. DOI: http://dx.doi.org/10.1109/TKDE.2013.184

Eran Halperin and Richard M. Karp. 2005. The minimum-entropy set cover problem. *Theoretical Computer Science* 348, 23 (2005), 240–250. DOI: http://dx.doi.org/10.1016/j.tcs.2005.09.015 *Automata, Languages and Programming: Algorithms and Complexity (ICALP-A'04)*.

Dan He. 2006. Using suffix tree to discover complex repetitive patterns in DNA sequences. *Conference Proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference* 1 (2006), 3474–3477.

Donald R. Jones, Cary D. Perttunen, and Bruce E. Stuckman. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Applications* 79, 1 (1993), 157–181.

Andreea Julea, Nicolas Méger, Philippe Bolon, Christophe Rigotti, Marie-Pierre Doin, Cécile Lasserre, Emmanuel Trouvé, and Vasile N. Lăzărescu. 2011. Unsupervised spatiotemporal mining of satellite image time series using grouped frequent sequential patterns. *IEEE Transactions on Geoscience and Remote Sensing* 49, 4 (April 2011), 1417–1430. DOI: http://dx.doi.org/10.1109/tgrs.2010.2081372

Yoshinobu Kawahara and Masashi Sugiyama. 2009. Change-point detection in time-series data by direct density-ratio estimation. In *Proceedings of the SDM*. Vol. 9. SIAM, 389–400.

E. Keogh, J. Lin, and A. Fu. 2005. HOT SAX: Efficiently finding the most unusual time series subsequence. In *Proceedings of the 5th IEEE International Conference on Data Mining.*. pp. 8. DOI: http://dx.doi.org/10.1109/ICDM.2005.79

Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. 2004. Towards parameter-free data mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM, New York, NY, 206–215. DOI: http://dx.doi.org/10.1145/1014052.1014077

Hoang Thanh Lam, Fabian Mörchen, Dmitriy Fradkin, and Toon Calders. 2014. Mining compressing sequential patterns. *Statistical Analysis and Data Mining* 7, 1 (2014), 34–52.

N. J. Larsson and A. Moffat. 1999. Offline dictionary-based compression. In *Proceedings Data Compression Conference (DCC'99)*. 296–305. DOI: http://dx.doi.org/10.1109/DCC.1999.755679

Ming Li and Paul Vitányi. 2013. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Science & Business Media.

Yuan Li, Jessica Lin, and Tim Oates. 2012. Visualizing variable-length time series motifs. In *Proceedings of the SDM*. SIAM, 895–906.

Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'03)*. ACM, New York, NY, 2–11. DOI: http://dx.doi.org/10.1145/882082.882086

Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. 2002. Finding motifs in time series. (2002), 53–68.

Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery* 15, 2 (1 Oct. 2007), 107–144. DOI: http://dx.doi.org/10.1007/s10618-007-0064-z

Per Martin-Löf. 1966. The definition of random sequences. *Information and Control* 9, 6 (1966), 602–619.

Jingjing Meng, Junsong Yuan, Mat Hans, and Ying Wu. 2008. Mining motifs from human motion. In *Proceedings of the EUROGRAPHICS*. Vol. 8.

David Minnen, Charles L. Isbell, Irfan Essa, and Thad Starner. 2007. Discovering multivariate motifs using subsequence density estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

David Minnen, Thad Starner, Irfan Essa, and Charles Isbell. 2006. Activity discovery: Sparse motifs from multivariate time series. In *The Learning Workshop (SNOWBIRD)*.

Yasser Mohammad and Toyoaki Nishida. 2014. Exact discovery of length-range motifs. In *Proceedings of the Asian Conference on Intelligent Information and Database Systems*. Springer, 23–32.

Abdullah Mueen and Nikan Chavoshi. 2015. Enumeration of time series motifs of all lengths. *Knowledge and Information Systems* 45, 1 (2015), 105–132.

Abdullah Mueen, Eamonn Keogh, Qiang Zhu, and Sydney Cash. 2009. Exact discovery of time series motifs. In *Proceedings of the SDM*.

Craig G. Nevill Manning and Ian H. Witten. 1997. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research* 7, 1 (Sep. 1997), 67–82.

Craig G. Nevill-Manning and Ian H. Witten. 1997. Linear-time, incremental hierarchy inference for compression. In *Proceedings of the Data Compression Conference (DCC'97)*. IEEE, 3–11.

P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana. 2011. Discovery of variable length time series motif. In *Proceedings of the 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON'11)*. 472–475. DOI: http://dx.doi.org/10.1109/ECTICON.2011.5947877

Tim Oates. 2002. PERUSE: An unsupervised algorithm for finding recurring patterns in time series. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*. 330–337. DOI: http://dx.doi.org/10.1109/icdm.2002.1183920

Pranav Patel, Eamonn Keogh, Jessica Lin, and Stefano Lonardi. 2002. Mining motifs in massive time series databases. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*. 370–377. DOI: http://dx.doi.org/10.1109/icdm.2002.1183925

C. Piciarelli, C. Micheloni, and G. L. Foresti. 2008. Trajectory-based anomalous event detection. *IEEE Transactions on Circuits and Systems for Video Technology* 18, 11 (Nov. 2008), 1544–1554. DOI: http://dx.doi.org/10.1109/TCSVT.2008.2005599

Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. 2015. Time series anomaly discovery with grammar-based compression. In *Proceedings of the International Conference on Extending Database Technology*.

Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, Susan Frankenstein, and Manfred Lerner. 2014. GrammarViz 2.0: A tool for grammar-based pattern discovery in time series. In *Proceedings of the Machine Learning and Knowledge Discovery in Databases*. Springer, 468–472.

R. Staden. 1989. Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in the Biosciences : CABIOS* 5, 4 (Oct. 1989), 293–298.

Yoshiki Tanaka, Kazuhisa Iwamoto, and Kuniaki Uehara. 2005. Discovery of time-series motif from multi-dimensional data based on MDL principle. *Machine Learning* 58, 2–3 (Feb. 2005), 269–300. DOI: http://dx.doi.org/10.1007/s10994-005-5829-2

Yoshiki Tanaka and Kuniaki Uehara. 2004. Motif discovery algorithm from motion data. In *Proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence*.

Heng Tang and Stephen S. Liao. 2008. Discovering original motifs with different lengths from time series. *Knowledge-Based Systems* 21, 7 (Oct. 2008), 666–671. DOI: http://dx.doi.org/10.1016/j.knosys.2008.03.022

Nikolaj Tatti and Jilles Vreeken. 2012. The long and the short of it: Summarising event sequences with serial episodes. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 462–470.

Matthijs Van Leeuwen and Jilles Vreeken. 2014. Mining and using sets of patterns through compression. In *Proceedings of the Frequent Pattern Mining*. Springer, 165–198.

Jarke J. Van Wijk and Edward R. Van Selow. 1999. Cluster and calendar based visualization of time series data. In *Proceedings of 1999 IEEE Symposium on Information Visualization (Info Vis'99)*. IEEE, 4–9.

M. S. Vitevitch, P. A. Luce, J. Charles-Luce, and D. Kemmerer. 1997. Phonotactics and syllable stress: Implications for the processing of spoken nonsense words. *Language and Speech* 40 (Pt. 1) (1997), 47–62.

Li Wei, Eamonn Keogh, and Xiaopeng Xi. 2006. SAXually explicit images: Finding unusual shapes. In *Proceedings 6th International Conference on Data Mining (ICDM'06)*. IEEE, 711–720.

Dragomir Yankov, Eamonn Keogh, and Umaa Rebbapragada. 2008. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. *Knowledge and Information Systems* 17, 2 (2008), 241–262.

Neal E. Young. 2008. Greedy set-cover algorithms. In *Encyclopedia of Algorithms*, Ming-Yang Kao (Ed.). Springer, 379–381.

Received December 2015; revised January 2017; accepted February 2017