Approximate Variable-Length Time Series Motif Discovery Using Grammar Inference

Yuan Li
Computer Science Department
George Mason University
Fairfax, VA
ylif@gmu.edu

Jessica Lin
Computer Science Department
George Mason University
Fairfax, VA
jessica@cs.gmu.edu

ABSTRACT

The problem of identifying frequently occurring patterns, or motifs, in time series data has received a lot of attention in the past few years. Most existing work on finding time series motifs require that the length of the patterns be known in advance. However, such information is not always available. In addition, motifs of different lengths may co-exist in a time series dataset. In this work, we propose a novel approach, based on grammar induction, for approximate variable-length time series motif discovery. Our algorithm offers the advantage of discovering hierarchical structure, regularity and grammar from the data. The preliminary results are promising. They show that the grammar-based approach is able to find some important motifs, and suggest that the new direction of using grammar-based algorithms for time series pattern discovery might be worth exploring.

Keywords

Time Series, Frequent Patterns, Grammar Induction

1. INTRODUCTION

The vast growth of disk technology in the past decade has enabled us to store large multimedia databases, such as audio, video, images, time series, etc. While storage is no longer an impediment, it has become increasingly apparent that efficient techniques are needed for humans to quickly browse through, understand, and discover interesting patterns from the data.

Like other multimedia data types, time series data are ubiquitous. They are prevalent in almost every aspect of human life. Some examples of such data include speech, electrocardiogram (ECG) signals, radar signals, seismic activities, etc. In addition to the conventional definition of time series, i.e., measurements taken over time, recently, it has been shown that certain other multimedia data, e.g., images and shapes [48, 49], and XML [19], can be converted to time series and mined with promising results.

Figure 1 shows an example of how shapes can be converted to time series.

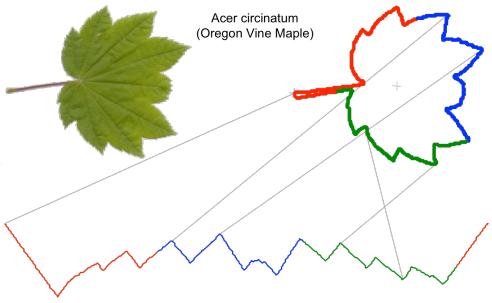

Figure 1. A shape is converted to time series. Image courtesy of Eamonn Keogh [24].

There has been a great amount of interest in mining time series data in the past decades. Most of the work in time series research has concentrated on solving classic data mining tasks such as similarity search, classification, and clustering [24]. There is relatively little work on learning hierarchy, structure, and patterns from time series data. At first glance, it seems a surprising oversight, as learning patterns and structures from data can provide valuable insights on the regularity and hidden semantics of the data and its underlying generative process. In addition, understanding the structure of data can potentially help solve the aforementioned data mining tasks. Such algorithms, e.g. grammar induction [25, 26, 28, 39, 40], have received decades of attention in the natural language and text processing communities. In this paper, we focus on one specific pattern discovery task that can benefit from learning hierarchy and structure from data, namely, frequent pattern mining [10, 12, 30, 32, 33, 35, 44, 45, 46].

The task of frequent pattern mining is an important problem that has many applications. In addition to its own

MDMKDD'10, July 25, 2010, Washington DC, U.S.A.

merit of summarizing and compressing data, it is also a precursor to association rule mining [2]. Furthermore, in bioinformatics, it is well understood that overrepresented DNA sequences often have biological significance [8, 16, 20, 23, 43, 47]. A substantial body of literature has been devoted to techniques to discover such patterns [2, 4], including sequential patterns on sequence data.

In a previous work, we defined the related concept of "time series motif" [30], which are frequently occurring patterns in time series data. Since then, a great deal of work has been proposed for the discovery of time series motifs [10, 12, 30, 32, 33, 35, 44, 45, 46]. Figure 2 shows an example of a time series motif in an insect behavior dataset[1].

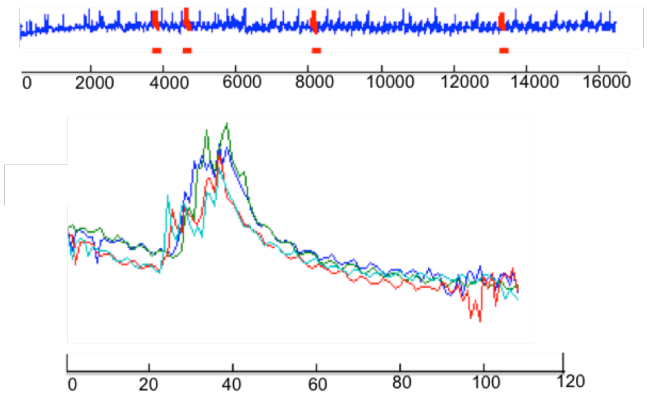

Figure 2. A motif of length 109 from an insect dataset. The top plots show the entire dataset to present a context for the motifs. The bars below the plots denote the locations of the motifs. The original dataset is 16,384 in length.

Identifying time series motifs is a much more complex problem than finding frequent sequential patterns, as the task of sequential pattern mining is often performed on discrete data. Existing algorithms on finding frequent itemsets in transaction data [2], and on finding sequential patterns in sequence data [4] are not suitable for time series motif discovery for one or more of the following reasons: (1) They focus on discrete data, whereas time series are continuous, real-valued data. (2) They are iterative methods and typically require multiple database scans. (3) They are designed for static databases [43].

Intuitively, one may try to convert a real-valued time series data into a data format for which existing off-the-shelf algorithms can be applied. One way to achieve this is to discretize each data value in the time series. However, the problem with this approach is that time series data are typically noisy, and considering every single point as an "event" would result in a noisy string that inaccurately reflects the noises as true patterns in the data. A better alternative is to consider subsequences instead. However, this approach poses another challenge: we simply cannot know in advance which subsequences are frequent. As a result, we need to consider *every* possible subsequence in the data. Considering all subsequences of any length, with overlaps, is undoubtedly a tedious task. To alleviate the complexity, most existing work [12, 30, 34, 35] thus require an input parameter: a pre-defined motif length n. This limits the search space for the algorithms; however, it also implies that the length of motifs (n) must be known in advance. In addition, frequent patterns of different lengths might co-exist within the same dataset. In order to find all significant patterns with unknown lengths, one would need to repeat the motif discovery algorithm several times—each time with a different window size. It is more desirable to have an algorithm that can automatically detect significant motifs of variable, previously unknown lengths, without exhaustively trying different subsequence lengths.

In this work, we propose to utilize a grammar-based compression algorithm, *Sequitur* [37], that can automatically and efficiently identify frequent patterns and hierarchical structure in data. We believe that a grammar-based approach [25, 28, 37, 50] is suitable and advantageous for our task due to several reasons. First, producing a (relatively) small set of interpretable rules from a massive dataset is a desirable goal of data mining. Clearly, a grammar-based method will allow a more natural mapping from data to rules [51], and can reveal the hidden hierarchical structures in the data. There has also been increasing interest in grammar-based methods for feature extraction, classification and forecasting of time series [17, 53]. Furthermore, as mentioned earlier, it is important to consider every subsequence in the time series when trying to identify motifs.

While this work is still at its early stage, the preliminary results are promising. Specifically, they show that the grammar-based approach has the potential to identify some important motifs in time series.

The rest of the paper is organized as follows. Section 2 discusses background and related work on time series motifs and grammar induction. Section 3 describes the grammar induction algorithm, *Sequitur*, that we adapt in this work. We describe our approach in Section 4. Section 5 presents some preliminary results using grammar-based approach to find variable-length motifs. We conclude in Section 6 and discuss future work.

2. BACKGROUND AND RELATED WORK

In this section, we briefly discuss background and related work on time series similarity search.

For concreteness, we begin with definitions of time series:

Definition 1. *Time Series*: A time series $T = t_1,\ldots,t_m$ is an ordered set of m real-valued variables.

Since we are interested in finding local patterns, we consider time series subsequences as the basic unit:

[1] Information on the insect dataset can be found in [35].

Definition 2. *Subsequence:* Given a time series T of length m, a subsequence C of T is a subsection of length $n \leq m$ of contiguous position from p, that is, $C = t_p..t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.

Since all subsequences may potentially be the candidates for motifs, any algorithm would have to extract and consider all of them. This can be achieved via the use of a sliding window:

Definition 4. *Sliding Window:* Given a time series T and a user-defined subsequence length n, all possible subsequences can be extracted by sliding a window of size n across T and considering each subsequence C_p, for $1 \leq p \leq m - n + 1$.

Figure 3 summarizes the definitions mentioned above.

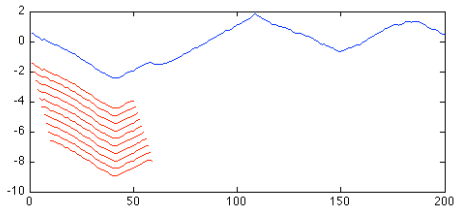

Figure 3. Subsequences of length 50 are extracted from a time series T via a sliding window.

Next, we briefly discuss related work on time series motif discovery and grammar induction techniques.

2.1 Related Work

2.1.1 Time Series Motifs

In [30], we defined time series motif C as the subsequence in T that has the highest count of non-trivial matches, that is, subsequences that are within ε units of distance away from C. We proposed a sub-quadratic algorithm to find exact motifs of a given length. Mueen et al proposed an algorithm named MK that is an improvement from the brute-force exact motif discovery algorithm [35]. The intuition behind MK is that the linear ordering of subsequence provides some useful heuristic information that guides motif search. The observation is that if two subsequences are close in the original space, they must also be close in the linear ordering.

In some applications, it may be sufficient or even desirable to have a fast algorithm that can find *approximate* motifs [12]. As an example, Chiu and Keogh proposed probabilistic motif discovery algorithm based on random projection [12]. The advantage of probabilistic motif discovery algorithm is its efficiency. Other approximate motifs algorithms exist [10, 12, 32, 42, 45, 52]; however, one common drawback for all these algorithms is that they require an input parameter for the motif length.

A few algorithms were proposed to discover motifs of variable lengths [33, 38, 45]; however, they either do so via post-processing, scale poorly, or quantize the whole data rather than considering overlapping subsequences, resulting in inaccurate and incomplete patterns found.

Some work has been proposed to find motifs from multivariate time series. Minnen and Isbell proposed a multivariate motifs discovery algorithm by using subsequence density estimation and greedy mixture learning [34]. They present motif discovery as the problem of locating regions of high density in the space of all time series subsequences. This algorithm locates an over-complete set of candidate motif seeds by identifying subsequences located near high density regions at first, then it adopts a greedy mixture learning framework to select the correct motifs from over-complete set of candidate motif seeds and to find the additional motif occurrences.

2.1.2 Grammar Induction

Grammatical induction, also known as grammatical inference, refers to the process of learning regular grammars or automata. Broadly speaking, a learner is required to induce a grammar from some data which are sequential or structured (strings, words, trees, terms or limited forms of graphs) [13]. Contrary to the tradition decision rule problem, which seeks a descriptive model, grammatical induction may be said to seek a generative model. There are a wide variety of algorithms for grammatical induction. RPNI (Regular Positive and Negative Grammatical Inference) [40], proposed by Oncina and Garcia, is a polynomial time algorithm to identify a DFA consistent with a given sample, but there are some open problems for this algorithm both on time complexity and definition of characteristic set. Angluin identifies the smallest k-reversible language, a subset of the class of regular languages, that contains any finite positive sample in $O(n^3)$, where n is the sum of length of the strings in the sample [6, 39]. *Active learning* is a learning model proposed by Angluin [7], where the learner can ask string membership and grammar equivalence queries to an oracle. The membership and equivalence queries form a *Minimal Adequate Teacher*. In this model of learning, the algorithm L* identifies regular languages with a polynomial number of queries. Another grammar inference algorithm, GRIDS [26], guides search through a space of context-free grammars by using a simplicity metric. Due to the use of Minimum Description Length (MDL) for scoring and selecting the most plausible grammars, GRIDS tend to find simple grammars [26].

3. SEQUITUR

We plan to investigate how to extract patterns using techniques that identify hierarchies and frequent sequences. Although aimed at compressing discrete sequences of data, there are algorithms that can be used as a proof-of-concept for our preliminary study. For instance, *Sequitur* [37] is a string compression algorithm that infers a context-free grammar from a sequence of discrete symbols [37]. It has

been adopted in various domains due to the many nice properties it offers: it has been used to find repeated DNA sequences [11, 50] and repeated function call sequences [27], and to segment time series [9]. The main premise is that repeated subsequences are replaced by a grammatical rule that generates the subsequence, thereby reducing the length of the original sequence, and producing a hierarchical representation that summarizes the structure of the data. Although simple in design, *Sequitur* has been shown to be competitive with the state-of-the-art compression algorithms [37], maintaining its scalability even for large sequences. Moreover, *Sequitur* offers a unique advantage—it utilizes and identifies the hidden structure (recurring subsequences) in the input data sequence, requiring relatively small memory footprint. Due to these reasons, we choose *Sequitur* in this work to demonstrate the utility of using grammar-based compression algorithms to find patterns in time series data.

Sequitur works by maintaining two properties: digram uniqueness and rule utility [37]. The first property governs that no pair of consecutive symbols (terminals or non-terminals) can appear more than once. When *Sequitur* reads a new symbol from the input sequence, the last two symbols of the sequence read so far—the new symbol and its predecessor symbol—form a digram [37]. A table that stores all existing digrams is maintained. If this new digram already exists in the digram table, i.e., it appears somewhere in the sequence already read, *Sequitur* uses a non-terminal to substitute these digrams, and, if such rule[2] has not yet existed, it forms a new grammar rule with the non-terminal on the left hand side. The second property, rule uniqueness, ensures that each grammar rule be used more than once except for the top-level rule, since a grammar rule that occurs just once is not meaningful and should be removed. As an example, the input string S1: "12131213412" can be converted to the following grammar:

Grammar rule	***Expanded Grammar rule***
S1 -> BB4A	*12131213412*
A -> 12	*12*
B -> A13	*1213*

The top-level grammar rule, *S1→ BB4A*, denotes the sequence seen so far. *Sequitur* is an online algorithm that generates the grammar incrementally as each symbol arrives. It is, therefore, ideal for the streaming scenarios. It is both time- and space-efficient, requiring O(m) time to compress a sequence of size m, and a compressed sequence is of size O(m) in the worst case (i.e., no compression), and O(log$m$) in the best case [37].

The main advantages of *Sequitur* (or many grammar-induction algorithms in general) are three-fold: (1) it identifies recurring patterns automatically, e.g., "1213" in the previous example, as well as hierarchical structure; (2) the recurring patterns found can be of any lengths; and (3) it is suitable for streaming data since it constructs the grammars in an incremental fashion. These benefits suggest that we may be able to adapt it to find variable-length motifs for time series. We describe how we achieve this in the next section.

4. FINDING APPROXIMATE VARIABLE-LENGTH MOTIFS BY SEQUITUR

We propose an algorithm that finds approximate variable-length motifs using *Sequitur*. Our approach consists of three steps: Pre-processing (discretization), Motif Discovery (*Sequitur*), and Post-processing. We describe each step in more details below.

4.1 Step 1: Discretization

Sequitur, or more generally, grammar induction algorithms, were originally designed for discrete data. However, time series are real-valued data, requiring a pre-processing step to allow the application of a grammar-based algorithm.

In a previous work, we introduced a time series symbolic representation called Symbolic Aggregate approXimation (SAX) [29, 31]. While there have been dozens of symbolic representations proposed for time series data, SAX has been shown to outperform existing methods. In addition, SAX has some unique, desirable properties such as dimensionality reduction, lower-bounding distance measures, and equiprobable symbols. For these reasons, we will utilize SAX for our pre-processing step.

Given a time series, SAX performs discretization by dividing the time series into w equal-sized segments. For each segment, their mean value is computed, and then mapped to a symbol according to a set of breakpoints that divide the distribution space into α equiprobable regions, where α is the alphabet size specified by the user. If the symbols were not equiprobable, some of the symbols would occur more frequently than others. As a consequence, we would inject a probabilistic bias in the process. It has been noted that some data structures such as suffix trees produce optimal results when the symbols are of equiprobability [14]. The discretization steps are summarized in Figure 3.

[2] Note the "rules" here should not be confused with the sequential or association rules. The "rules" here refer to those that are generated by the algorithm. It is equivalent to the concept of frequent itemsets in association rule mining.

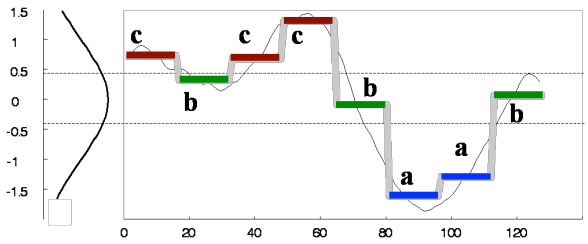

Figure 4. Example of SAX for a time series. The time series above is transformed to the string *cbccbaab*, and the dimensionality is reduced from 128 to 8.

There are two ways we may discretize a time series:

(a) Whole discretization: Convert each time series to one SAX word. Figure 4 shows an example of whole discretization. The drawback with this approach is that it often does not capture enough local details in data, which are essential for motif discovery.

(b) Subsequence discretization: Extract subsequences of length n from the time series, and convert each subsequence into a SAX word. The result is a bag of SAX words generated from (all or selected) subsequences in the original data.

For subsequence discretization, we need to determine which subsequences to consider. In general, we need to decide whether or not overlapping subsequences are allowed. Non-overlapping subsequences can be obtained through "chunking," which is the process where a time series is segmented into subsections by a specific period. This is very similar to whole discretization, and shares similar drawbacks as whole discretization. More specifically, unless the data is periodic, in which case we could segment the data by days, weeks, etc., segmenting the data into non-overlapping subsequences could break a pattern into two or more segments and/or shift the patterns.

Figure 5 shows an example. In the figure, an ECG signal is chunked into segments of length 500. Since each heartbeat has different length, the individual heartbeats eventually become misaligned.

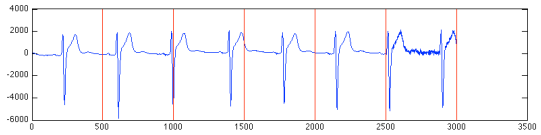

Figure 5. An ECG signal is chunked into non-overlapping segments of length 500. The individual heartbeats eventually become misaligned.

A better approach is to consider overlapping subsequences. Since each subsequence in the dataset can be a potential candidate for motif, we should consider all possible subsequences in order to ensure correct results. This can be achieved by using a sliding window of length n across the time series. Note that n is just the initial window length for our algorithm; the algorithm will grow the patterns automatically. Once we collect all the subsequences, we can then discretize each subsequence individually using SAX, and then concatenate them to form one single sequence. A transformed sequence might look something like this:

S = *1131-1132-1232-1223-1344-1131-1132-1232*...

where the '-' denotes the delimiter between consecutive, overlapping subsequences.

The reason we choose to discretize subsequences rather than individual points is that time series are typically very noisy. If we discretize each time point into a symbol, and then form a string from these symbols, then we would give equal weight to each time point, including the noises. On the other hand, the "aggregating" feature of SAX would smooth out the subsequences and essentially remove the noises.

4.2 Step 2: Sequitur on SAX Words

Once we transform the time series into a discrete sequence consisted of SAX strings, the application of *Sequitur* on the sequence is straight-forward. Each string delimited by '-' represents one subsequence, and is treated as a terminal symbol, an atomic unit for patterns. *Sequitur* embodies efficiency and accuracy in finding the repeated patterns of sequences in many cases. One possible grammar rule that can be generated from the above string is

$$A \rightarrow 1131\text{-}1132\text{-}1232.$$

We modify the original *Sequitur* algorithm and record the offsets of the subsequences that occur in each grammar rule.

4.3 Step 3: Post-Processing

Since we discretized the data before running the algorithm, we now need to map the rules and frequent strings back to the time series subsequences. We can simply record the starting offsets of all grammar rule instances. We will obtain approximate results if we do not perform any verification step. In some cases, it is acceptable to obtain approximate motifs, trading accuracy for speed. In other cases where the user desires to find the exact motifs, we can use the grammar rules found as our seeds and perform a range search using these motif seeds. In this preliminary work, we only examined the approximate motifs and will leave the exact motif discovery for future work.

The number of rules generated can be large and, similar to association rules mining [2], not all rules are interesting or important. Several refinement steps can be performed on the grammar rules, for example

(1) Eliminate trivial matches for a motif. For the purpose of this paper, the trivial match of a subsequence M is any subsequence that overlaps M. We need to eliminate those occurrences from a grammar rule because nearby subsequences are likely to be similar to

each other and may simultaneously contribute to a motif, causing the patterns to be over-counted.

(2) Rank the rules by their "interestingness." Several ranking criteria are possible:

   a. Frequency – Frequency of a pattern, equivalent to the *support* in association rule mining, is perhaps the most trivial criterion to consider; however, it often is not the most meaningful way to measure the significance of a pattern.
   b. Rule length – In many cases, long patterns are more interesting than short ones.
   c. Pattern variation - Patterns with variations are likely to be more interesting than the monotonically increasing or decreasing ones. Fortunately, since SAX preserves the general shape of the time series, the pattern variation can be approximated without much difficulty.
   d. Overlapping information – The amount of overlap between a new pattern to be examined and those that are already deemed significant, should also be taken into consideration.

5. EMPIRICAL EVALUATION

In this section we evaluate the potential of using *Sequitur* to find frequent patterns in time series data. While the experimental evaluation is brief, as this work is still at its early stage, the following examples show that *Sequitur* can find *some* time series motifs without knowing the exact lengths of the motifs in advance. For all examples shown below, we choose $\alpha = 4$ and $w = 4$, which are both arbitrary choices that have been shown to work well for most datasets [31].

As a sanity check, the first dataset we used is an ECG dataset. As Figure 6 shows, the obvious motif, i.e. the individual heartbeats, are indeed discovered. The length of the motif shown is 159, with the initial window length of 150.

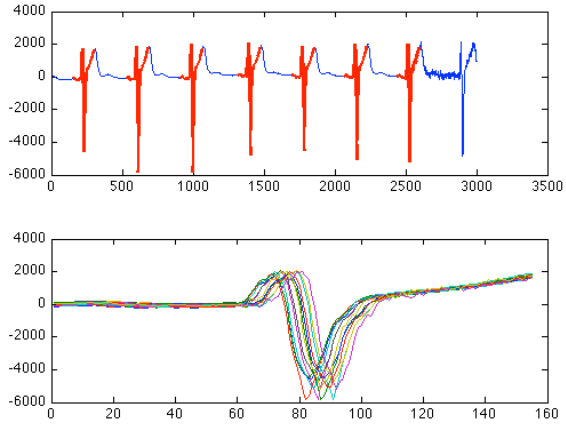

Figure 6. (Top) Original ECG time series. Matches for a motif of length 159 is highlighted. The initial length used is 150. (Bottom) The discovered motif instances are plotted.

The next dataset we use in our evaluation is an image of a leaf that was converted to time series. In Figure 7, a motif of length 54 is shown.

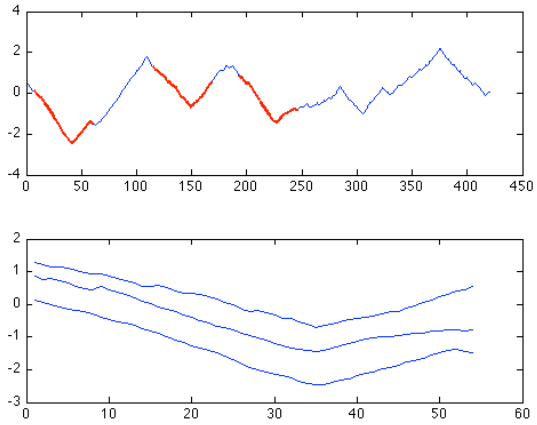

Figure 7. (Top) Original leaf time series. Matches for a motif of length 54 are highlighted. The initial length used is 50. (Bottom) The discovered motif instances are plotted.

Figure 8 shows the winding dataset from UCR Time Series Archive [54]. A motif of length 84 is found, when the initial window length is 50.

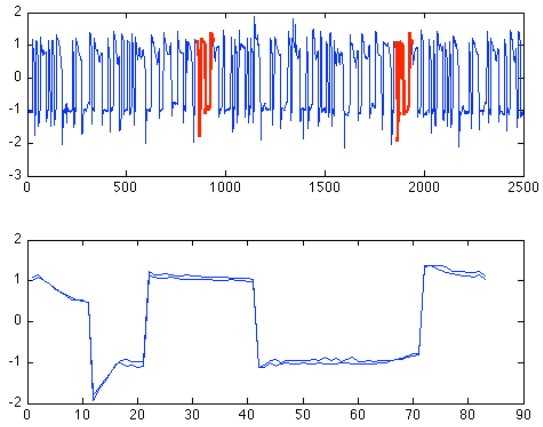

Figure 8. (Top) Original winding dataset. Matches for a motif of length 84 is highlighted. The initial length used is 50. (Bottom). The discovered motif instances are plotted.

5.1 Efficiency

The approximate, variable-length motifs discovery algorithm is highly efficient. It requires O(*m*) to pre-process the time series and convert the subsequences to SAX strings; and *Sequitur* takes O(*m*) to compress a sequence of length *m*. As for post-processing, for approximate motifs discovery, the time complexity is linear in the size of the grammar rules. However, with some refinement, the number of rules can be reduced significantly.

6. RESEARCH ISSUES, LIMITATIONS, AND FUTURE WORK

The quality of the motifs found is directly affected by both SAX and *Sequitur*. Both algorithms are extremely efficient, i.e. they both perform in linear time. While our approach offers a quick and easy way to learn the approximate motifs in time series data without the need of knowing the exact lengths, the motifs found are often not complete. This is partly due to the definition of SAX – even though SAX has the ability to smooth the data and remove noises from the data, in some occasions similar subsequences could be mapped to similar, but different strings. This scenario arises when a segment falls near a breakpoint, and slight variation could push the segment over to the other side of the breakpoint. For example, in Figure 4, the second segment, discretized to "b," falls right below a breakpoint. If there exists a similar subsequence B in which the second segment falls just *above* the breakpoint, then the resulting string for B will be different. Since *Sequitur* looks for exact matching of the symbols, this sensitivity on borderline segments partially explains why most motifs discovered by the algorithm are relatively short. Nevertheless, the quality of motifs found by our approach is reasonably good. In addition, our algorithm offers the opportunity to discover some motifs of variable lengths simultaneously with just one pass of the data.

Another limitation of the algorithm lies within *Sequitur*. *Sequitur* is able to find repeated patterns of sequences in many cases. Nevertheless in some cases, the inherent nature of *Sequitur* prevents it from meeting our expectations. Specifically, it is not surprising that the grammar inferred by *Sequitur* is not minimal for it is an on-line, greedy algorithm. Regardless of its shortcomings, *Sequitur* has been adopted in various domains due to the many nice properties it offers. We plan to further explore the feasibility of using *Sequitur* for motif discovery, and investigate whether it is possible to modify the algorithm without compromising its efficient time complexity.

Other future directions are possible. Due to the preliminary stage of this work, the literature comparison is non-existent. It is partly due to the fact that to the best of our knowledge, there is no known algorithm that can find variable-length frequent patterns with comparable efficiency. Nonetheless, we would like compare with some existing methods to further validate our findings. In addition, the post-processing step of the algorithm can be further refined for better exact motif extraction. Furthermore, we would like to investigate the utilities of finding hierarchical structures and grammars from time series data.

7. CONCLUSION

In this preliminary work, we propose a methodology to find approximate variable-length time series motif using an efficient grammar-based compression algorithm. Several algorithms were proposed to discover motifs of variable lengths; however, they either do so via post-processing, scale poorly, or quantize the whole data rather than considering overlapping subsequences, resulting in inaccurate and incomplete patterns found. Our algorithm mitigates the shortcomings; in addition, it offers the advantage of discovering hierarchical structure, regularity and grammar from the data. Even though there are some known issues with the algorithm, the preliminary results are promising. They show that the grammar-based approach is able to find some important motifs and suggest that the new direction of using grammar-based algorithms for time series pattern discovery might be worth exploring.

8. ACKNOWLEDGMENTS

Our thanks to Eamonn Keogh for the shapes datasets.

9. REFERENCES

1. SAX page: http://www.cs.gmu.edu/~jessica/sax.htm

2. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In proceedings of the 1993 ACM SIGMOD Int'l Conference on Management of Data. Washington, D.C. May 26-28, 1993. pp. 207-216
3. R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying Shapes of Histories. In proceedings of the 21st Int'l Conference on Very Large Databases. Zurich, Switzerland. Sept 11-15, 1995. pp. 502-514
4. R. Agrawal and Ramakrishnan Srikant. Mining Sequential Patterns. In Proc. of the 11th Int'l Conference on Data Engineering, Taipei, Taiwan, March 1995.
5. H. Andre-Jonsson and D. Badal. Using Signature Files for Querying Time-Series Data. In proceedings of Principles of Data Mining and Knowledge Discovery, 1st European Symposium. Trondheim, Norway. Jun 24-27, 1997. pp. 211-220
6. Angluin, D. Inference of reversible languages, Journal of the ACM (JACM), Vol.29, No. 3, pp. 741-765, 1982
7. Angluin, D. A note on the number of queries needed to identify regular languages, Inform. and Control 51 (1981) 76–87.
8. A. Apostolico, M. E. Bock, and S. Lonardi. Monotony of Surprise in Large-Scale Quest for Unusual Words. In proceedings of the 6th Int'l Conference on Research in Computational Molecular Biology. Washington, D.C. Apr 18-21, 2002. pp. 22-31
9. T. Armstrong and T .Oates. RIPTIDE: Segmenting Data Using Multiple Resolutions. In the Proceedings of the 6th IEEE International Conference on Development and Learning (ICDL), 2007.
10. P. Beaudoin, M. van de Panne, P. Poulin and S. Coros, *Motion-Motif Graphs*, Symposium on Computer Animation 2008.
11. N. Cherniavsky and R. Ladner. Grammar-based Compression of DNA Sequences. UW CSE Technical Report 2007-05-02.
12. Chiu, B. Keogh, E., & Lonardi, S. (2003). Probabilistic Discovery of Time Series Motifs. In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August 24 - 27, 2003. Washington, DC, USA. pp 493-498.
13. Colin de la Higuera. A bibliographical study of grammatical inference", Pattern Recognition 38 (2005): 1332--1348
14. M. Crochemore, A. Czumaj, L. Gasjeniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter. Speeding Up Two String-Matching Algorithms. Algorithmica. vol. 12. pp. 247-267. 1994.
15. C. S. Daw, C. E. A. Finney, and E. R. Tracy. Symbolic Analysis of Experimental Data. Review of Scientific Instruments. vol. 74. pp. 915-930. 2001.
16. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids: Cambridge University Press. 1998.
17. D. Eads, E. Rosten, D. Helmbold. "Grammar-guided Feature Extraction for Location-Based Object Detection." British Machine Vision Conference. Queen Mary, University of London. London, UK. September 11, 2009.
18. A. Ekambaram and E. Montagne, “An Alternative. Compressed Storage Format for Sparse Matrices”, ISCIS. XVIII - Eighteenth International Symposium on Computer and Information Sciences.
19. Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, Andrea Pugliese (2005). Fast Detection of XML Structural Similarity. IEEE Trans. Knowl. Data Eng. 17(2): 160-175.
20. A. Gionis and H. Mannila. Finding Recurrent Sources in Sequences. In proceedings of the 7th Int'l Conference on Research in Computational Molecular Biology. Berlin, Germany. 2003. pp. 123-130
21. E. C. Gonzalez, K. Figueroa and G. Navarro, *Effective Proximity Retrieval by Ordering Permutations*, IEEE Transactions on Pattern Analysis and Machine
22. D. Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.
23. D. He. Using Suffix Tree to Discover Complex Repetitive Patterns in DNA Sequences, The 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE EMBC 2006, New York City, New York, USA, August 30 - September 3, 2006
24. Keogh, E. Tutorial in SIGKDD 2004. *Data Mining and Machine Learning in Time Series Databases*
25. P. Langley. Simplicity and Representation Change in Grammar Induction. Technical Report. 1995.
26. P. Langley and S. Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In proceedings of the 11th International Conference on Machine Learning. Standord, CA.
27. J.R. Larus. Whole program paths. SIGPLAN Not. 34, 5, pp. 259-269, 1999.
28. E. Lehman. Approximation Algorithms for Grammar-Based Data Compression, PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.
29. J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A Symbolic Representation of Time Series, with Implications for Streaming Algorithms, Workshop on Research Issues in Data Mining and Knowledge Discovery, the 8th ACM SIGMOD. San Diego, CA, 2003.
30. J. Lin, E. Keogh, P. Patel, and S. Lonardi, Finding Motifs in Time Series, the 2nd Workshop on Temporal Data Mining, the 8th ACM Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, 2002, pp. 53-68.

31. J. Lin, E. Keogh, W. Li, and S. Lonardi. (2007). Experiencing SAX: A Novel Symbolic Representation of Time Series. Data Mining and Knowledge Discovery Journal.
32. J. Meng, J.Yuan, M. Hans and Y. Wu, *Mining Motifs from Human Motion*, Proc. of EUROGRAPHICS, 2008.
33. D. Minnen, T. Starner, I. Essa, C. Isbell. Activity Discovery: Sparse Motifs from Multivariate Time Series. Snowbird Learning Workshop, Snowbird, Utah, April 4-7, 2006.
34. D. Minnen, C.L. Isbell, I. Essa, and T. Starner. Discovering Multivariate Motifs using Subsequence Density Estimation and Greedy Mixture Learning. Twenty-Second Conf. on Artificial Intelligence (AAAI-07), Vancouver, B.C., July 22-26, 2007.
35. A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover. Exact Discovery of Time Series Motifs. In proceedings of the 2009 SIAM International Conference on Data Mining (SDM09). April 30-May 2, 2009. Sparks, NV.
36. J.C. Na, A. Apostolico, C.S. Iliopoulos, and K. Park: Truncated suffix trees and their application to data compression. Theor. Comput. Sci. 1-3(304): 87-101 (2003)
37. C.G. Nevill-Manning and I.H. Witten. Identifying Hierarchical Structure in Sequences: A linear-time algorithm. Journal of Artificial Intelligence Research, 7, 67-82.
38. T. Oates. PERUSE: An Unsupervised Algorithm for Finding Recurring Patterns in Time Series. In proceedings of the International Conference on Data Mining. Maebashi City, Japan. Dec 9-12. pp. 330-337.
39. Oates, T. Desai, D. Bhat, V. Learning k-Reversible Context-Free Grammars from Positive Structural Examples, Proceedings of the Nineteenth International Conference on Machine Learning. 2002. pp 459 – 465
40. Oncina, J. and García, P. Inferring Regular Languages in Polynomial Updated Time. In Pattern Recognition and Image Analysys. Pérez de la Blanca, Sanfeliú and Vidal (Eds.) World Scientific. 1992.
41. M. Riesenhuber and T. Poggio. Hierarchical Models of Object Recognition in Cortex. Nature Neuroscience 2: 1019:1025.
42. S. Rombo and G. Terracina, *Discovering representative models in large time series databases,* Proc. of the 6th International Conference on Flexible Query Answering Systems, pp. 84–97, 2004.
43. R. Staden. Methods for Discovering Novel Motifs in Nucleic Acid Sequences. Computer Applications in Biosciences. vol. 5. pp. 293-298. 1989.
44. Y. Tanaka and K. Uehara. Motif Discovery Algorithm from Motion Data. In proceedings of the 18th Annual Conference of the Japanese Society for Artificial Intelligence. Kanazawa, Japan. June 2-4, 2004.
45. Y. Tanaka, K. Iwamoto, and K. Uehara. Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle. Mach. Learn. 58, 2-3 (Feb. 2005), 269-300.
46. H. Tang and S.S. Liao. Discovering original motifs with different lengths from time series. Know.-Based Syst. 21, 7 (Oct. 2008), 666-671.
47. M. Tompa and J. Buhler. Finding Motifs Using Random Projections. In proceedings of the 5th Int'l Conference on Computational Molecular Biology. Montreal, Canada. Apr 22-25, 2001. pp. 67-74
48. Wei, L., Keogh, E., and Xi, X. 2006. SAXually Explicit Images: Finding Unusual Shapes. In *Proceedings of the Sixth international Conference on Data Mining* (December 18 - 22, 2006).
49. Ye, L. and Keogh, E. 2009. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining* (Paris, France, June 28 - July 01, 2009).
50. R. Ladner. Enhanced Sequitur for Finding Structure in Data. In Proceedings of the 2003 Data Compression Conference. March 25-27, Snowbird, UT. pp 425-.
51. M.L. Wong and K.S. Leung. Data mining using grammar based genetic programming and applications. In: Genetic programming, vol. 3. The Netherlands: Kluwer Academic Publishers; 2000.
52. T. Guyet, C. Garbay and M. Dojat, *Knowledge construction from time series data using a collaborative exploration system*, Journal of Biomedical Informatics 40(6): 672-687 (2007).
53. E. Keogh. Personal Communications.
54. E. Keogh The UCR Time Series Data Mining Archive. http://www.cs. ucr.edu/~eamonn/tsdma/index.html