Iterative Grammar-based Framework for Discovering Variable-Length Time Series Motifs

Yifeng Gao*, Jessica Lin*, Huzefa Rangwala*
*Department of Computer Science, George Mason University, Virginia, USA
{ygao12, jessica}@gmu.edu,{rangwala}@cs.gmu.edu

***Abstract*—In recent years, finding repetitive similar patterns in time series has become a popular problem. These patterns are called time series motifs. Recent studies show that using grammar compression algorithms to find repeating patterns from the symbolized time series holds promise in discovering approximate motifs with variable length. However, grammar compression algorithms are traditionally designed for string compression. Therefore, existing work on grammar induction has not fully utilized much available information that can be used to enhance the performance of the algorithms. In this work, an iterative framework based on grammar induction is proposed. In each iteration, a revision operator called Noise Reduction Operator is applied to revise the symbolized time series string based on the rules returned from a base grammar induction algorithm. In our experiments, we show that the proposed work can find motifs of the same quality, with much faster running time compared to the state-of-the-art variable-length exact motif discovery algorithm in real world time series data.**

I. Introduction

The task of finding repetitive similar patterns in time series data, known as motif discovery, has received a great amount of attention in the past decade. It has been used as an important subroutine in many time series data mining tasks and applications such as data visualization [1], classification [2] and anomaly detection [3].

Figures 1(a)-(c) illustrate two examples of different (planted) motif in a random walk time series. The two motifs are colored in red and green, respectively.

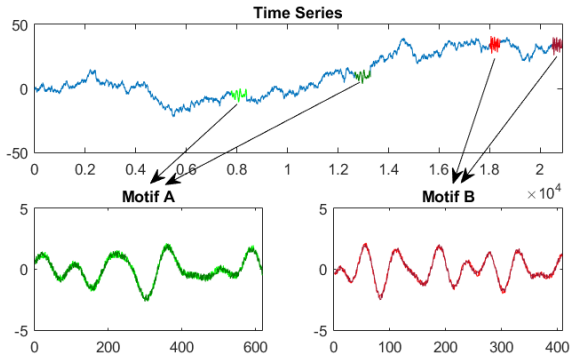


Fig. 1: **(a)** A random walk time series with 2 different (planted) pair-motifs. **(b)** Instances of the first motif. **(c)** Instances of the second motif.

However, most existing motif discovery algorithms [4][5] can only find motifs of a given length. In order to obtain significant patterns of different lengths, one would need to repeat the algorithm numerous times. In several recent studies, researchers have shown that enumerating motifs in a range of lengths is tractable [5]. However, most variable-length exact motif discovery algorithm quickly becomes intractable for large time series. For example, it takes 8 hours to enumerate motifs with lengths between 300 and 600 in an EEG time series with 160000 points.

Recent work shows that compression or grammar induction algorithms can be used to find approximate motifs at a reduced time complexity [1][3]. Such algorithms apply a grammar induction algorithm to build the hierarchical structure consisting of context-free grammar rules, each of which approximating a repeated pattern in the data, from discretized time series. Rules are ranked by various interestingness. Since different rules represent patterns of different lengths, the algorithm is naturally capable of discovering variable-length motifs without enumerating all lengths. However, these algorithms suffer from information loss caused by the symbolization process. The sensitivity to initial parameters thus impacts the quality of patterns detected.

In this paper, we focus on finding approximate motifs within a range of lengths efficiently. We propose a framework that iteratively uses a base grammar induction algorithm to address the limitations of existing grammar-based motif discovery algorithms. At each iteration, a revision operator called Noise Reduction Operator (NR-Operator) is applied to revise the symbolized time series based on the rules returned from a base grammar induction algorithm. We utilize Sequitur as our choice of the base grammar induction algorithm to illustrate the capability of the framework. We call our Sequitur-based iterative framework "**Iter**ative Sequitur" (ItrSequitur).

To summarize, we make the following contributions:

- We propose an approximate variable-length motif discovery algorithm that can find comparable motifs as an exact motif discovery algorithm, with much less computational cost.
- We address the limitations of Sequitur by allowing approximate matching via a Noise Reduction operator.

The rest of the paper is organized as follows. Sec 2 discusses background and related work on time series motif discovery. Sec 3 describes the original grammar-based motif discovery algorithm and our proposed iterative framework. Sec 4 presents experimental evaluation on our approach compared to the state-of-the-art algorithms. We conclude in Sec 5.

II. Related Work

We first briefly discuss the notions of time series motifs. While many definitions have been introduced in the literature, there are two main types of motifs that past work has considered. Some researchers refer to motif as the pair of subsequences that are the most similar to each other under some distance measure [6][5][7][8], whereas others consider motif as a set of non-overlapping subsequences that are similar to each other, or similar to some common seed subsequence [9][3]. While there is no agreed-upon distinct terminologies for the different types of motifs, for clarity we will refer to the first type of motifs as the "pair motifs", and the second type of motifs as the "set motifs." The interest measures used for the two types of motifs are different — for pair motif we may want to identify the *most similar* pair of subsequences, whereas for set motif we may want to identify the *most frequent* subsequences with similarity within certain threshold.

In [9], the authors introduced the first algorithm to find exact set motifs of a given length with sub-quadratic complexity. In [5], the authors introduced an algorithm called MK to find exact pair-motif of a given length. The general idea of both algorithms is similar. Both algorithms utilize a procedure to find approximate lower bound for the distance measure, which is then used for pruning unnecessary comparisons. Another category of motif discovery algorithms focus on finding approximate motifs [4][1][3]. One advantage for discovering approximate motifs as opposed to exact motifs is superior efficiency. However, like exact motif discovery, most approximate motif discovery algorithms only find motifs of a given length.

In recent years, researchers have attempted to find motifs in a range of lengths. In [10], the authors introduced an algorithm named VLMD to find exact K pair-motifs by calling MK with all possible lengths within a range. In [7], the authors proposed the MOEN algorithm to find exact K variable-length pair-motifs under correlation distance. MOEN incrementally enumerates motifs of all lengths and prunes unnecessary computations through a novel lower-bound measure. Following the same direction, MK++ [11] and sMD [12] are introduced for fast incremental enumeration of motifs under Euclidean distance and normalized Euclidean distance, respectively. A common drawback for these algorithms is that they all conduct an incremental enumerating process starting from the smallest length in a given range. As a consequence, it takes a significant amount of time to find longer motifs, which are said to contain more useful information [7].

The authors in [1][3] introduced a framework, GrammarViz, that uses grammar induction to find approximate set motifs of variable length. Sequitur [13], a linear-time greedy grammar-based compression algorithm, is used for identifying repeated patterns in the input sequence. While the algorithm is highly efficient, the results depend on the appropriate choices of the discretization parameters, which vary with the input data.

Other works such as [14][15] also focus on variable length motif discovery; however, these techniques have some shortcomings that prevent them to be useful in general cases. In [14], the subsequences are not normalized, which makes it difficult for the algorithms to find patterns that are similar but with different amplitudes. The work proposed in [15] also consists of a discretization step, but the subsequences are non-overlapping. So it does not consider every possible pattern candidate [1][3].

In this work we follow the same motif definition as in [6][5][7][8] and focus on discovering top-K most similar pairs of subsequences in a given time series.

III. Methodology

A. Grammar-based Motif Discovery

Grammar Induction is a process of learning grammars from a sequence of tokens. The intuition for using context-free grammar induction for motif discovery task is that the grammar reveals repetitive patterns in the input sequence. A grammar-based time series motif discovery algorithm consists of three steps. First, the input time-series is discretized to a sequence of tokens called *terminals*. A grammar induction algorithm is then applied to compress the token sequence by replacing repetitive tokens with a new non-terminal symbol (i.e. grammar rule). Each non-terminal symbol represents a motif in the symbolic space. Finally, these symbolic motifs are converted back to subsequences, identified by the previously recorded locations. The motifs are sorted using different criteria such as frequency of occurrence and length.

1) Discretization: Discretization of time series into discrete sequences is often a necessary pre-processing step for efficient pattern discovery [9][4][3]. Since our proposed framework is based on grammar induction, we discretize the time series into a symbolic sequence using Symbolic Aggregate approXimation (SAX) [16]. In this step, all possible z-normalized subsequences of length n are first extracted via a sliding window. These subsequences are converted to Piecewise Aggregate Approximation (PAA) representation of size w. Finally, every PAA segment is mapped to a set of symbols with alphabet size a, according to the breakpoints table, to form a *token sequence*.

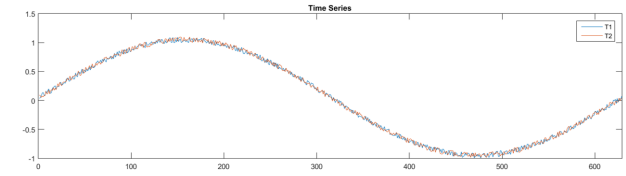
2) Numerosity Reduction: In practice, neighboring tokens (words) generated by SAX are often identical to each other, since they represent consecutive subsequences that are off by one point. To avoid over-counting a pattern, and to allow variable-length pattern discovery, Numeriosity Reduction is used to further compress the token sequence. More specifically, only the first occurrence of a consecutive repeating token is recorded along with its. For example, a token sequence $S = abc, abc, abc, bca, bca, aaa, acc$ can be compressed to $S_{NR} = abc_1, bca_4, aaa_6, acc_7$. It is easy to see that we can retrieve the original token sequence based on the offsets recorded in S_{NR}.

3) Sequitur: Sequitur is a grammar based compression algorithm designed for discovering context-free grammar from a sequence of tokens [13]. The algorithm can identify the hierarchical structure of input sequence in linear time. Sequitur has been used widely in many different applications due to its efficiency and efficacy [17][1]. Two principles, digram uniqueness and rule utility, are applied to constraint the rules during grammar construction.

For example, given a token sequence $S = \{aba, bca, aba, bca, aaa\}$, Sequitur forms two grammar rules $R_0 \rightarrow R_1, R_1, aaa$ and $R_1 \rightarrow aba, bca$. The top-level

rule, R_0, summarizes the input sequence using non-terminal symbol R_1, which identifies the *motif* $\{aba, bca\}$ in the sequence.

Drawbacks of Sequitur.: Although Sequitur has been successfully used in time series data mining problems, it has some drawbacks that can strongly affect the quality of patterns found. For example, a motif can be identified by Sequitur only if the instances have identical discrete representation. However, the large amount of noise that is typically present in time series makes this a difficult and unrealistic restriction, even with the discretization and numerosity reduction procedures. To demonstrate this, we generate two time series by adding random noise to a sine wave shown in Figure 2. If we concatenate the two time series, we can see that the longest repeating pattern should be a single sine wave. However, while the time series look very similar, their respective sequences of SAX words, as shown in the table in Figure 2, ($T1$ and $T2$) are slightly different (the differences are highlighted in bold). As a result, Sequitur only identifies fragments of the long pattern, reflected by three different rules, $R1$, $R2$ and $R3$. This problem can prevent Sequitur from discovering some interesting patterns that are longer, and often more informative than short patterns.



String Correspond to Time Series	
$T1$	$\leftarrow acd, acc, adc, adb, \mathbf{ddd}, bdb, bda, cda, cca, dca, dba, dbb, dab, dac, cac, \mathbf{aaa}, cad, bad, bbd, abd$
$T2$	$\leftarrow acd, acc, adc, adb, bdb, bda, cda, cca, dca, dba, dbb, dab, dac, cac, \mathbf{daa}, \mathbf{cac}, cad, bad, bbd, abd$
R_{T1}	$\leftarrow R1, \mathbf{ddd}, R2, \mathbf{aaa}, R3$
R_{T2}	$\leftarrow R1, R2, \mathbf{daa}, \mathbf{cac}, R3$

Fig. 2: ItrSequitur Pair Motif Performance Comparison

B. ItrSequitur: Proposed Iterative Framework

Our proposed method, ItrSequitur, alternates between two steps: the *Induction* and the *Revision* steps, until the terminating condition is satisfied.

- **Induction Step**: In this step, a grammar induction algorithm is applied to learn rules. The rules obtained in the current iteration are combined with the rules obtained from the previous iterations.
- **Revision Step**: In this step, a Revision Operator is used to revise the token sequence based on the rules found by the Induction Step. The Revision Operator can be applied to remove noise in the token sequence so that in the next iteration, better grammar may be induced in the Induction Step.

The intuition is that one pass of grammar induction may not be sufficient due to the noises in the data. ItrSequitur thus induces the initial grammar, re-constructs the input sequence using the learned rules (but eliminating the noises), and induces grammar again.

The terminating condition of the iterative framework depends on the problem that the algorithm attempts to solve. In this work, the terminating condition is satisfied when no new rules can be found.

Algorithm 1 shows the pseudo code of ItrSequitur. After discretization (Line 4), ItrSequitur starts by using Sequitur to obtain the initial set of rules (Line 6). Then a fast filtering process is applied to remove some obvious false positive motifs caused by the revision operator, i.e. by removing rules that have a large differential between the longest token sequence and the shortest token sequence in the same rule (Lines 7-11). The threshold is set to $3L$, which will be discussed in the next section. After the revision step which we discuss next, ItrSequitur calls Sequitur again on the revised token sequence, and the process continues until convergence. In our experiments, the number of iterations is between 3 and 8 in most cases, so the algorithm converges quickly.

Algorithm 1 ItrSequitur with NR-Operator

1: **Input**: Time Series T
2: **Output**: Set of Rules R
3: $R = \{\}$ {Universal rule set for collecting rules from every iteration}
4: $S^{(0)} \leftarrow$ Symbolization(T); $i = 1$;
5: **repeat**{Induction Step: running Sequitur to find rules.}
6: $Rules \leftarrow$ Sequitur$(S^{(i-1)})$
{Ignore obvious false positive motifs and add the rule to R}
7: **for** each rule r **in** $Rules$ **do**
8: **if** $MaxLengthDifference(r) < 3 \times L$ **then**
9: R.add(r)
10: **end if**
11: **end for**{Revision Step: applying NR-Operator for revising token sequence}
12: $S^{(i)} \leftarrow$ NR-Operator($Rules$,L)
13: $i++$
14: **until** $Rules.size == 1$
15: Return R

1) Noise Reduction Operator: In this subsection, we introduce a specific Revision Operator called the Noise Reduction Operator (NR-Operator) to enhance the robustness of the algorithm. NR-Operator is described in Algorithm 2.

In each iteration, NR-Operator consists of two steps to mitigate the drawbacks of Sequitur mentioned above. The first step expands the top-level rule by recursively unfolding each non-terminal symbol. We stop expanding a non-terminal symbol when it contains only terminal symbols (Lines 5). The motivation for this step is to "uncover" patterns that might have been hidden by previously generated rule. For example, consider the following grammar rules set $Rules$ learned from an input sequence:

$R0 \rightarrow R1, abc, R3, R3, R2, abc, R3$

$R1 \rightarrow R2, R2$

Suppose $R2$ and $R3$ are rules containing only terminal symbols. Note $R0$ is the top-level grammar rule, which summarizes the entire input string. In this step, the sequence is re-written to "$R2, R2, abc, R3, R3, R2, abc, R3$" by expanding/unfolding $R1$. After the revision, the pattern '$R2, abc, R3$' which was previously "covered" by $R1$ can now be discovered in the next iteration.

Algorithm 2 Noise Reduction Operator (NR-Operator)

1: **Input**: Grammar Rules Set $Rules$, Tolerate Length L
2: **Output**: Symbolic Sequence S'
3: R_0 =TopLevelGrammar($Rules$)
4: S' = DeepCopy(R_0)
5: ExpandingRule(S',$Rules$) {Expanding top-level rule}
6: RemoveConsecutiveSymbol(S',L){Remove small block of consecutive terminal symbols if the block size is L or smaller}

For the second step, a block of terminal symbols with block size equal to or less than L is removed from the sequence (Line 6). This step aims to remove the unwanted, and most likely noisy tokens, to prevent them from breaking a longer rule. The parameter L can be considered as the degree of "don't cares" for string matching.

Like the original Sequitur, the rules found during subsequent iterations after the first iteration may produce false positive results due to revision. However, these false positives can be filtered through the same post-processing as Sequitur.

For example, suppose we apply ItrSequitur with $L = 3$ to the example shown in Figure 2. In the first step (Induction), ItrSequitur will get the same rules set as the original Sequitur. After applying NR-Operator, the different tokens, "ddd" and "aaa" in R_{T1} and "daa cac" in R_{T2}, are removed because the length is less than 3. As a result, the sequence can be compressed into a new rule by Sequitur in the next iteration. Finally, the new generated rule is accepted because the length of T1 and T2 are the same, which is less than 9.

C. SAX Parameter Selection

We observe that the quality of patterns found by a grammar-based algorithm is highly dependent on the input parameters (e.g. SAX parameters). Here we introduce an random search approach to address the parameter selection problem.

Consider the problem of finding top-K variable-length pair-motifs with lengths between l_{min} and l_{max}. We apply ItrSequitur with randomly chosen parameter settings (w,a,n), and use the average intra-motif dsitance as the metric to select the best parameter setting. For all experiments, w and a are sampled from 2 to 6 in Algorithm 3. n is sampled from $\{n|n = \frac{i}{100}(l_{max} - l_{main}) \quad i = 1, 2, \dots 25\}$.

The process consists of two steps. First, pair-motifs are extracted by computing pairwise distances between subsequences belonging to the same rule. Then for each length between $[l_{min}, l_{max}]$, K pairs of subsequences with the smallest pairwise distances are stored as discovered pair-motif. In order to make the metric meaningful, for each pair of motif instances, we decreases or increases their length based on a search range ($\pm\frac{l_{max}-l_{min}}{4}$) to ensure that the algorithm can cover the whole interval $[l_{min}, l_{max}]$.

In our experiments, the algorithm terminates when the best-so-far average intra-motif distance has not changed in 20 iterations. Alternatively, we can also set the terminating condition based on running time.

IV. Experimental Evaluation

We perform a series of experiments evaluating our proposed approach compared with the baseline and the state-of-the-art exact motif discovery algorithms. All the experiments are conducted on a 8 GB RAM laptop with quad core processor at 2.7 GHz. For all experiments except for parameter sensitivity experiment, we set $L = 5$.

A. Comparing with Sequitur in Planted Motif Discovery

In this subsection, we show that ItrSequitur outperforms Sequitur in finding planted pair motifs without using random search.

We randomly generated the subsequences of motifs by $x = \sum_{i=1}^{5} A_i \sin \alpha_i x + \beta_i$ with random parameters $A_i \in [0, 5]$, $\alpha_i \in [-2, 2]$ and $\beta_i \in [-\pi, \pi]$. The length of motifs in both experiments are randomly chosen from range 300 to 600. We add 5% random noise to every instance of the motifs.

Two pair motifs are planted. We generated 100 different random walk time series with planted motifs in each experiment. We ran both Sequitur and ItrSequitur to find the planted motifs with parameters of all combinations: PAA size ranging from 3 to 6, alphabet size from 3 to 6, and sliding window length from $200, 210, \dots, 300$.

A motif instance is considered found if 80% of the subsequence overlaps with the subsequence found by the algorithm. For each rule returned by both algorithms, we check whether the subsequences in the rule match the ground truth locations. For each algorithm, we check whether the planted motifs are inside the rules found by the algorithm.

The performance metric we use is the number of pair motifs found by the algorithms. Since Sequitur's rules are a subset of ItrSequitur's rules, the performance of ItrSequitur must always exceed that of Sequitur. Therefore, we compute the percentage of time series datasets for which ItrSequitur outperforms Sequitur to demonstrate the improvement of the iterative framework. We also compute the percent increase on the distance calls to show the increased cost of the iterative framework.

Figure 3 shows the result for the pair motif experiment. The blue line shows the percentage of the test cases that ItrSequitur outperforms Sequitur. The red line indicates the percent increase of distance calls. ItrSequitur improves Sequitur's performance in 40% of test cases for pair motif discovery, and it requires additional 15% computation. Moreover, ItrSequitur has higher chance of improving the result when the sliding window length differs significantly from the true motif lengths. The reason is that when the sliding window is much smaller than the true motif length, more tokens are required to describe the instances. As a result, it is less likely for motif instances to have identical token sequences. Sequitur would fail to identify the whole motif in this case.

B. Scalability and Parameter (In)Sensitivity

The scalability of Sequitur and ItrSequitur with random search is tested on two datasets, ECG and EEG, and compared with MOEN [7], an exact motif discovery algorithm, on the same datasets. Other exact variable-length motif discovery algorithms have the same or higher complexity compared to

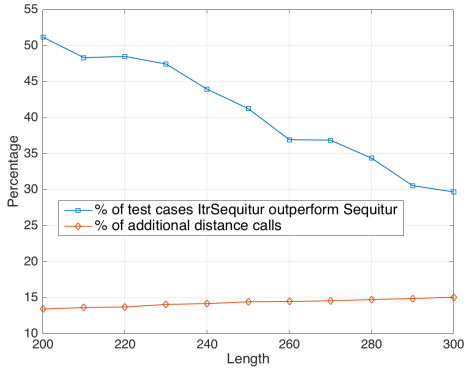


Fig. 3: ItrSequitur Pair Motif Performance Comparison

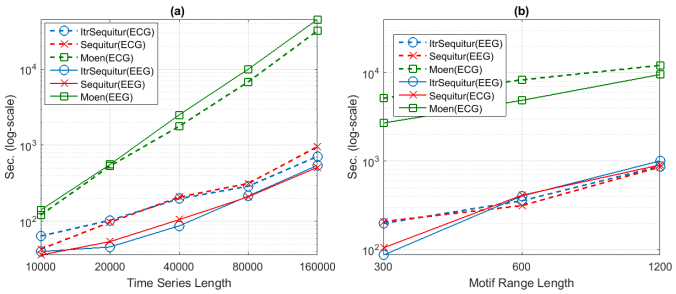


Fig. 4: **(a)** Execution time to find 1-motif in time series of different lengths for the motif length range 300 to 600 **(b)** Execution time to find 1-motif with different motif length ranges

MOEN. Each dataset has 160,000 points. As in [7], we extracted subsequences of lengths [10000, 20000, 40000, 80000, 160000]. Figure 4(a) shows the increasing execution time (in log scale) for all three algorithms as the length of time series grows. From the figure, we can see that the running times for Sequitur and ItrSequitur grow at a much slower rate than MOEN. Figure 4(b) shows the increase in execution time with the increasing motif length range from 300 to 1200 in ECG and EEG time series, each with length 40000. Both Sequitur and ItrSequitur's execution time are linearly increasing.

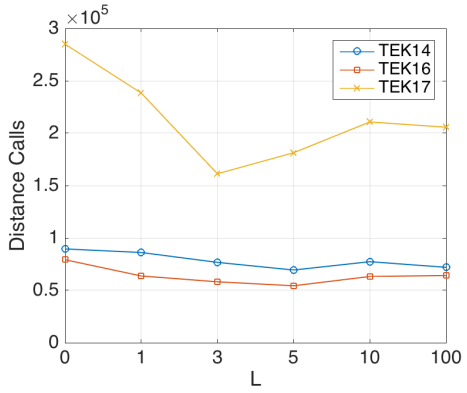


Fig. 5: Distance Calls vs. L in 3 datasets

Figure 5 shows the average execution time to reach the average intra-motif distance 10% greater than the state of art result for $L \in \{0, 1, 3, 5, 10, 100\}$ in 3 different datasets [3]. From the experiment, choosing L between 3 and 5 reaches better and stable execution time. This is because if L is too small, ItrSequitur does not differ much from Sequitur, which results in more random search iterations. On the other hand, setting L too large will make the algorithm unstable. This is because in some cases, larger L can reduce execution time by allowing more mismatches in each iteration; however, it also increases the chance of longer execution time by allowing too many false positives.

C. Case Studies

In this section, we show that ItrSequitur can find high quality pair-motifs in different real world time series.

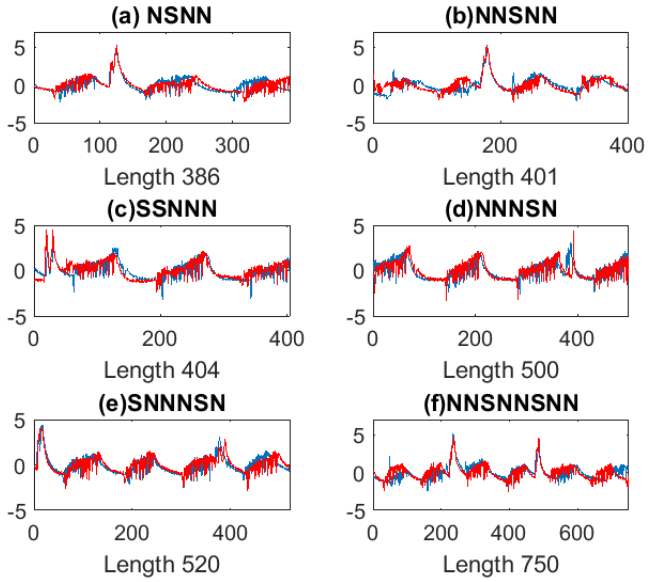


Fig. 6: Motifs found in EPG data, similar to the results in [7]

1) Repeated Pattern Discovery in EPG Data: The study in [7] shows that enumerating pair-wise motifs by state of art algorithm in Electrical Penetration Graph (EPG) time series [5] can help scientist understand hierachy of insect behavior patterns. We also use ItrSequitur to find motifs in EPG data to demonstrate that the result can carry the same valuable information.

Figure 6 shows the top pair-motifs found by ItrSequitur in the length range 300 to 800. We use the same sequence pattern identified in [7] to label the motifs. As shown in Figure 6, the spike shape is labeled as 'S' and the noise shape is labeled as 'N'. We found that for shorter motifs, we found patterns 'NSNN','NNSNN', 'SSNNN' with length 386, 401 and 404 respectively. As the length increases, we discover longer patterns 'SNNNSN' and 'NNSNNSNN'. Clearly, short patterns such as 'NNSNN' is a sub-pattern of the longer pattern 'NNSNNSNN'. Therefore, we conclude that ItrSequitur can discover the hierarchy of insect behavior patterns like [7]. The execution time of our algorithm for this experiment is 84 seconds, which is much faster than the exact algorithm which is around 15 minutes.

2) Electrocardiogram Time Series Data: Electrocardiogram (ECG) data measures heart-related activities, and identifying motifs can help medical practitioners understand different patient conditions. We use ItrSequitur to find motifs in ECG time series with 1 million sample points from the MIMIC

II dataset [18]. The variable length range parameter for our approach is set between 150 and 350.

From a qualitative perspective, Figures 7 shows two meaningful patterns found by ItrSequitur. Figure 7(a) shows a motif of length 226, which consists of two heartbeats. The first heartbeat contains a depression T-wave while the second one contains a normal T-wave. Figure 7(b) is a motif of length 320, which consists of three heartbeats. The first heartbeat contains an depression T-wave, then followed by a normal heartbeat. The shape of ST segment at the end of the third heart beat is unusual.

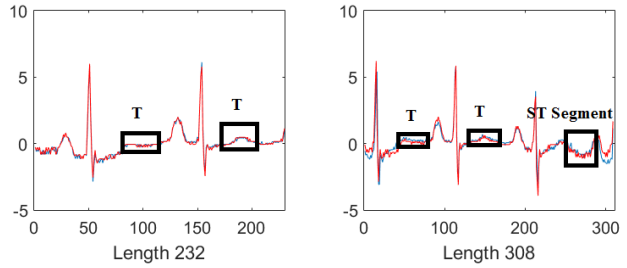


Fig. 7: 2 meaningful motifs found by ItrSequitur in an ECG time series with 1 million sample points

3) Electro-Oculogram Time Series Data: Electro-Oculogram (EOG) is used for recording eye movements and diagnosing eye diseases. We used ItrSequitur to find motifs in EOG data [18] with 2 million sample points. The range of length parameter was set from 300 to 600.

Figure 8 shows 6 motifs found by our approach. Existing methods compute the ratio of voltage change as a measure of eye-related diagnostic metric coupled with analysis of activity patterns. The motifs discovered from our approach show activities corresponding to the response of eye to light and have a high change in voltage levels.

V. Conclusion and Future Work

In this paper, we propose an iterative framework based on grammar induction for variable-length motif discovery. Our experiments show that the proposed work not only achieves competitive accuracy compared to the state-of-the-art variable-length exact motif discovery algorithm in real world time series, but is also scalable. In the future, we plan to design novel revision operators for different problems such as anomaly detection and classification.

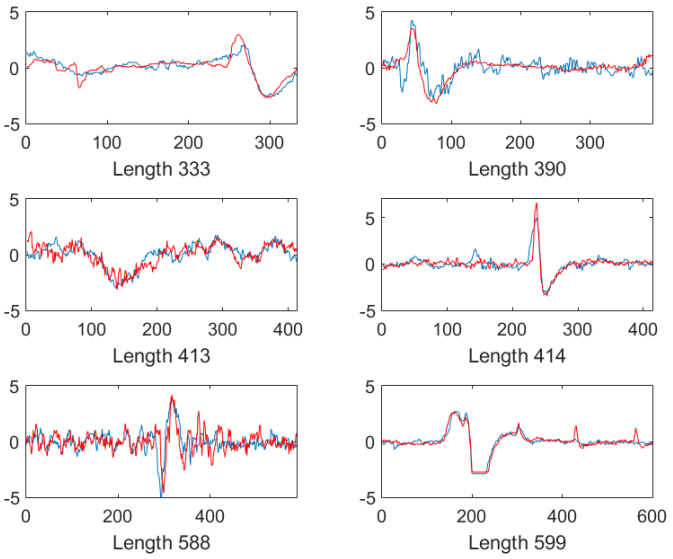


Fig. 8: Motifs found by ItrSequitur in EOG data containing 2 million points

References

[1] Y. Li, J. Lin, and T. Oates, "Visualizing variable-length time series motifs." in *SDM*. SIAM, 2012, pp. 895–906.

[2] X. Wang, J. Lin, P. Senin, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein, "Rpm: Representative pattern mining for efficient time series classification," pp. 185–196, 2016.

[3] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner, "Grammarviz 2.0: a tool for grammar-based pattern discovery in time series," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 468–472.

[4] B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic discovery of time series motifs," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 493–498.

[5] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover, "Exact discovery of time series motifs." in *SDM*. SIAM, 2009, pp. 473–484.

[6] A. Mueen and E. Keogh, "Online discovery and maintenance of time series motifs," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 1089–1098.

[7] A. Mueen, "Enumeration of time series motifs of all lengths," in *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 2013, pp. 547–556.

[8] M. Shokoohi-Yekta, Y. Chen, B. Campana, B. Hu, J. Zakaria, and E. Keogh, "Discovery of meaningful rules in time series," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1085–1094.

[9] J. L. E. K. S. Lonardi and P. Patel, "Finding motifs in time series," in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[10] P. Nunthanid, V. Niennattrakul, and C. A. Ratanamahatana, "Discovery of variable length time series motif," in *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on*. IEEE, 2011, pp. 472–475.

[11] Y. Mohammad and T. Nishida, "Exact discovery of length-range motifs," in *Intelligent Information and Database Systems*. Springer, 2014, pp. 23–32.

[12] ——, "Scale invariant multi-length motif discovery," in *Modern Advances in Applied Intelligence*. Springer, 2014, pp. 417–426.

[13] C. G. Nevill-Manning and I. H. Witten, "Identifying hierarchical strcture in sequences: A linear-time algorithm," *J. Artif. Intell. Res.(JAIR)*, vol. 7, pp. 67–82, 1997.

[14] H. Tang and S. S. Liao, "Discovering original motifs with different lengths from time series," *Knowledge-Based Systems*, vol. 21, no. 7, pp. 666–671, 2008.

[15] N. Castro and P. J. Azevedo, "Multiresolution motif discovery in time series." in *SDM*. SIAM, 2010, pp. 665–676.

[16] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[17] N. Cherniavsky and R. Ladner, "Grammar-based compression of dna sequences," *DIMACS Working Group on The Burrows-Wheeler Transform*, vol. 21, 2004.

[18] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.