Exposé Master's Thesis: A Synthetic Motif Generator

Rafael MOCZALLA[1]

[1]Humboldt University of Berlin

October 3, 2019

1 Introduction

In colloquial language a time series is a (mostly long) sequence of reals and a subsequence is a sequence of consecutive values in a time series. Two sequences of reals are matching if both have the same number of values and a very similar shape. A motif can be considered as a repetitive pattern in a time series and is represented by a sequence of reals that matches multiple subsequences in a time series. Motif discovery denotes the problem of finding a previously unknown motif in a time series.

In chronological order the motif discovery problem was first tackled by Lin, Keogh, Lonardi, and Patel, 2002. They search for the subsequence in a time series that matches the largest number of subsequences of that time series within a given similarity range to discover a repeating pattern in a time series. We call this problem *set motif* problem. Next Mueen et al., 2009, faced the motif discovery problem. They search for the most similar pair(s) of subsequences in a time series to discover a repeating pattern. We call this problem *pair motif* problem. After that Grabocka et al., 2015, dealed with the motif discovery problem. They search for a sequence of reals, not necessarily a subsequence of the time series, that matches the largest number of subseqences in a time series within a given similarity range. We call this problem *latent set motif* problem.

As the pair motif problem defines the motif as the most similar pair(s) of subsequences in a time series, the set motif defines potentially more subsequences in the same time series. The latent set motif matches potentially even more subsequences in the same time series than the set motif, since a repeating pattern in a time series occurs noisily under real conditions and may not be a subsequence of the time series. Therefore, the motif discovery problem is divided into the pair motif, set motif and latent set motif problem. Figure 1 illustrates all three motif discovery problems.

In our previous student research project "A Synthetic Motif Generator" we

1. categorized the approaches of evaluation methods on pair motif, set motif and latent set motif discovery algorithms,

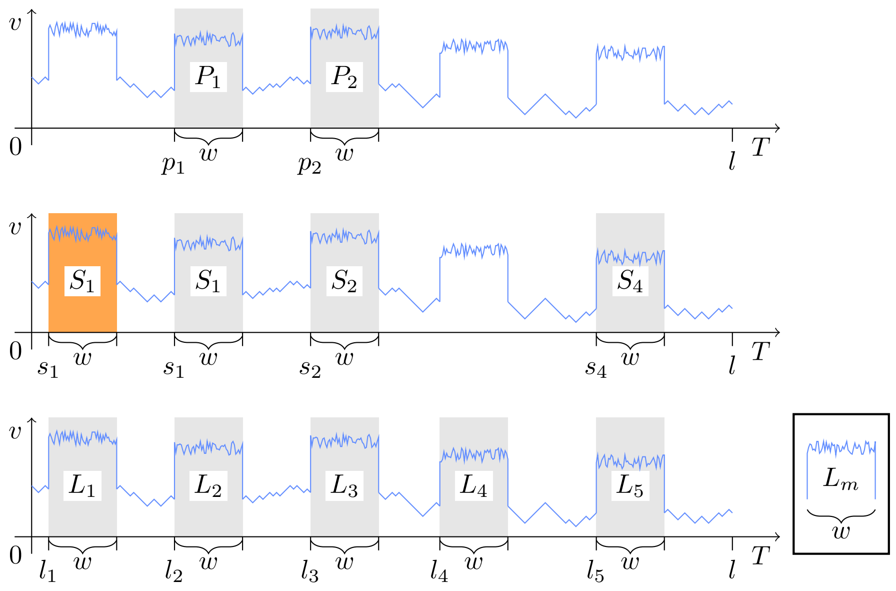


Figure 1: Comparison of pair motif, set motif and latent set motif in the same time series of length l and range $r = 1.38$. From top to bottom all subsequences of length w are marked that match the pair motif, set motif and latent set motif. The pair motif matches the subsequences P_1 and P_2 at the positions p_1 and p_2. The set motif S_1 matches the subsequences S_1, S_2, S_3 as well as S_4 within range r and the latent set motif L matches the subsequences L_1, L_2, L_3, L_4 as well as S_5 within range r.

2. implemented a synthetic latent set motif generator that calculates a synthetic time series with predefined sets of subsequences matched by a latent set motif in the same synthetic time series and

3. designed and performed a benchmark for motif discovery algorithms.

We noticed the following issues with our synthetic motif generator, the state-of-the-art motif discovery algorithms and our motif discovery benchmark as part of the student research project.

1. Our previous synthetic motif generator design has a two staged check for larger hidden motifs in a time series. This two staged check prohibits any set motif overlapping the injected set motif to guarantee that it is the largest set motif which makes searching the injected set motif easier. We change this approach by allowing overlapping set motifs whose size is smaller than the injected set motif to make the motif search harder for set motif discovery algorithms. See section 4.1.

2. The current synthetic motif generator design has no functionality for custom motif shape inputs and no graphical user interface. To overcome this issue we will design a graphical user interface for the synthetic motif generator and user inputs for custom motif shapes like drawing a shape in the graphical user interface, an interface accepting sequences of reals. See section 4.2.

3. (optional) The size of a top set motif with range $2r$ is an upper bound for the size of a top latent set motif with range r. The `EMMA` algorithm from Patel et al., 2002, is the standard algorithm for searching set motifs. We noticed that the `EMMA` algorithm searches for motifs by maximizing the number of overlapping subsequences instead of non overlapping subsequences in a time series which is problematic as overlapping subseqeuences in the same time series are very similar to each other and conflicts the set motif definition. We will redesign the `EMMA` algorithm to exclude sets of subsequences with overlapping subsequences in a time series. See section 4.3.

4. (optional) The `LM` algorithm from Grabocka et al., 2015, interprets latent set motifs as an optimization problem. Also, we noticed that the `LM` algorithm is based on random restarts to generate good global solutions of the latent set motif discovery problem in a time series. To reduce the number of random restarts we will design our own version of the `LM` algorithm which will use the results of our `EMMA` algorithm with range set to $2r$ as initial seeds. See section 4.3.

5. The previously designed benchmark for state-of-the-art motif discovery algorithms benchmarked unfairly pair motif, set motif and latent set motif discovery algorithms with injected latent set motifs in time series. We will split the motif discovery benchmark into a pair motif, set motif and latent set motif discovery algorithm. See section 4.4.

6. Also, the number of synthetic time series in the benchmark is too small to get significant result. We will scale up the number of synthetic time series per benchmark and perform the pair motif, set motif and latent set motif benchmarks on state-of-the-art motif discovery algorithms including our version of the `EMMA` algorithm and the `LM` algorithm. See section 4.4.

2 Background

To be precise, we present definitions of time series motif discovery problems in this section. We distinguish between

1. top pair motifs,
2. top set motifs and
3. latent top set motifs.

First we need to define the terms time series, subsequence, distance and non-self matching subsequences.

Definition 2.1 (Time Series)**.** *The time series $T = (t_1, t_2, \dots, t_n)$ of length n is an ordered sequence of reals with n values.*

n is called length of the time series.

Definition 2.2 (Subsequence)**.** *Let $T = (t_1, t_2, \dots, t_n)$ be a time series of length n and $w \in \mathbb{R}$ a window size. A subsequence in T is a time series $T_{i,w} = (t_i, t_{i+1}, \dots, t_{i+w-1})$ where $1 \leq i < i + w - 1 \leq n$.*

The offset i is called position of the subsequence $T_{i,w}$ in the time series $T = (t_1, t_2, \ldots, t_n)$.

Definition 2.3 (Distance)**.** *Let $T_{i,w}$ and $T_{j,w}$ be two subsequences of length w in the same time series and $d(\cdot,\cdot) \in \mathbb{R}$ is a metric on the w dimensional real space. The distance between $T_{i,w}$ and $T_{j,w}$ is the real value $d(T_{i,w}, T_{j,w})$.*

The distance can be interpreted as a measure for the similarity of two subsequences. High values indicate low similarity and low values indicate high similarity. We present the definition of matching subsequences as follows.

Definition 2.4 (Matching Subsequences)**.** *Let $T_{i,w}$ and $T_{j,w}$ be two subsequences of length w in the same time series, $d(\cdot,\cdot) \in \mathbb{R}$ is a metric on the w dimensional real space and r a real value called range. $T_{i,w}$ and $T_{j,w}$ are matching if their distance is at most r, i. e. $d(T_{i,w}, T_{j,w}) \leq r$.*

The scientific community uses the z-normalized real space to transform the sequences of reals into sequences whose mean is approximately 0 and the standard deviation is in a range close to 1 such that sequences with similar shape are matching regardless of there elongation or value shift. Furthermore, we define the notion of non-self matching subsequences to avoid trivial matches of overlapping subsequences in the same time series.

Definition 2.5 (Non-Self Matching Subsequences)**.** *Let $T_{i,w}$ and $T_{j,w}$ be two subsequences of length w in the same time series. $T_{i,w}$ and $T_{j,w}$ are non-self matching if they match but do not overlap in the time series, i. e. $i + w - 1 \leq j$ or $j + w \leq i$.*

We are finally in a position to define the motif discovery problems. We define the term of top pair motif discovery as follows.

Definition 2.6 (Top Pair Motif)**.** *Given a time series T, a window size w and a metric on the w dimensional real space $d(\cdot,\cdot) \in \mathbb{R}$. The top pair motif is an unordered non-self matching pair $\{T_{i,w}, T_{j,w}\}$ of subsequences in T with smallest distance $d(T_{i,w}, T_{j,w})$ among all non-self matching subsequence pairs in T.*

We move from top pair motifs to top set motifs. Therefore, we define the top set motif discovery problem as follows.

Definition 2.7 (Top Set Motif)**.** *Given a time series T, a window size w, a metric on the w dimensional real space $d(\cdot,\cdot) \in \mathbb{R}$ and radius r. The top set motif is a subsequence $T_{i,w}$ that has the largest count of non-self matching subsequences of length w in T within range r.*

The matching subsequences are non-self matching to each other and to the top set motif. The top set motif can be interpreted as the most frequent pattern of length w in a time series. Next we define the latent top set motif problem.

Definition 2.8 (Latent Top Set Motif)**.** *Given a time series T, a window size w, a metric on the w dimensional real space $d(\cdot,\cdot) \in \mathbb{R}$ and radius r. The latent top set motif is a real sequence L of w values, not necessarily a subsequence in T, that has the largest count of non-self matching subsequences of length w in T within range r.*

The matching subsequences are non-self matching to each other. As already mentioned for top set motifs, latent top set motifs can also be interpreted as the most frequent pattern in a time series.

From now on we use the Euclidean Distance on the z-normalized w dimensional space as distance measure $d(\cdot,\cdot)$. In the worst case we need $\mathcal{O}(w)$ time to calculate the Euclidean Distance.

The `BFNSPD` procedure describes a brute force algorithm to find a subsequence pair with smallest distance to each other in a time series The algorithm is presented in Algorithm 1. The algorithm compares the distance between any

Algorithm 1 Brute Force Nearest Subsequence Pair Discovery

```
procedure $\{P_1, P_2\} =$ BFNSPD($T, w$)
in $T = (t_1, t_2, \ldots, t_n), w \in \mathbb{N}^+$                      ▷ Time series, window size
out non-self matching subsequences $P_1, P_2$ in $T$          ▷ Top motif pair
1: best-so-far = inf
2: for $i = 1, 2, \cdots, n - w$ do
3:     for $j = i + 1, i + 2, \cdots, n - w + 1$ do
4:         if $T_{i,w}, T_{j,w}$ non-self matching then              ▷ Subsequences in $T$
5:             if $d(T_{i,w}, T_{j,w}) <$ best-so-far then           ▷ Distance $d(\cdot,\cdot)$
6:                 best-so-far = $d(T_{i,w}, T_{j,w})$
7:                 $P_1 = T_{i,w}, P_2 = T_{j,w}$
```

non-self matching subsequences $T_{i,w}$ and $T_{j,w}$ in a time series T and returns a subsequence pair with smallest distance to each other in T. Since there are $\mathcal{O}(n^2)$ many subsequence pairs of length w in T, line 4 is passed $\mathcal{O}(n^2)$ times. Additionally the calculation of the distance $d(\cdot,\cdot)$ in line 5 and line 6 takes $\mathcal{O}(w)$ time. All in all we need $\mathcal{O}(wn^2)$ time to determine a subsequence pair with smallest distance to each other in the time series with the `BFNSPD` algorithm.

3 Related Work

An extensive overview of the general field of time series data mining is given by Esling et al., 2012, including definitions, tasks, implementation components and a categorization of the existing literature until 2012.

SAX from Lin, Keogh, Lonardi, and Chiu, 2003 is a method for descretizing time series. SAX transforms a time series into a string of symbols with smaller length than the original time series. First SAX performs a z-normalization on the time series and next computes the means over segments of the time series with segments of equal size. Next SAX calculates the Gaussian curve for the time series and the mean for each segment. Then each mean is mapped to a symbol according to an equal-sized histogram under the Gaussian curve. The runtime of SAX is within $O(n)$ where n is the time series length.

3.1 Pair Motifs

As defined by Mueen et al., 2009, a top pair motif are two subsequences of length w with smallests distance to each other in a time series. The exact algorithm uses the z-normalized Euclidean distance and performs a special pruning technique. For that purpose the algorithm chooses R random subsequences in the time series (called *reference points*), writes the distance from any reference point to any subsequence in the time series into a table and maintains the *best-so-far distance*. The absolute difference between two subsequences in the time series and a reference point (estimated distance) is at least as large as the distance of both subsequences to each other. Therefore, the algorithm prunes subsequence pairs with at least one estimated distance larger than the best-so-far distance, reference point by reference point. The algorithm chooses reference points with high standard deviation first to prune subsequence pairs as early as possible. The runtime is within $O((R+w)n^2)$ where n is the length of the time series, w is the subsequence length and R is the number of reference points.

Yeh et al., 2016, revealed an approach for finding the top pair motif with fixed subsequence length w in a single time series. The algorithm calculates for any subsequence in the time series the subsequence with smallest distance to each other and stores the corresponding distance termed Matrix Profile. For that purpose the algorithm computes the distances (z-normalized Euclidean distances) of one subsequence to any subequence in the time series by calculating the dot product between one subsequence and the time series, the moving mean and the standard deviation. The dot product is calculated with Fast Fourier Transformations. The top pair motif is established by choosing a subsequence pair with smallest distance to each other. The runtime is within $O(n^2 \log n)$ where n is the time series length. Since the algorithm calculates the distances by using Fast Fourier Transformations, rolling mean and standard deviation the runtime is independet of the subsequence length w.

3.2 Set Motifs

An algorithm for approximate time series set motif discovery with fixed subsequence length w is proposed by Senin et al., 2014. First GrammarViz 2.0 discretizes the time series with SAX. The resulting string is parsed and decomposed into a context free grammar. For that, the string is scanned and repeating symbol pairs are replaced with a non-terminal symbol. The usage of any non-terminal symbol is counted and the set of subsequences represented by a non-terminal symbol with largest count is detected as top set motif in the time series. The overall runtime of GrammarViz 2.0 is within $O(n)$ where n is the time series length.

An algorithm for approximate time series set motif discovery with fixed subsequence length w and range r is proposed by Lin, Keogh, Lonardi, and Patel, 2002. The algorithm uses the z-normalized Euclidean distance. First the algorithm discretizes the time series with SAX and hashes subequences with equal word representation into the same bucket labeled with the word representation. In the second step the algorithm chooses a bucket with largest number of subsequences and any bucket with a label matching the label of the first bucket. Next the algorithm calculates the largest set motif among all subsequences in the selected buckets. For that purpose the algorithm prunes the number of distance calculations by pre-computing distances between arbitrary many subsequences and calculating the minimum distances for each unprocessed subsequence pair with the triangle inequality. The algorithm maintains the largest set motif so far and repeats with an unprocessed bucket of at least the size of the largest set motif so far. The runtime is within $O(\max(w^{a+2}, n^3))$ where n is the time series length, w is the subsequence length and a is the number of symbols.

A. J. Bagnall et al., 2014, introduce the approximate set motif discovery algorithm `ScanMK` with fixed subsequence length w and range r and the exact set motif discover algorithm `SetFinder` with fixed subsequence length w and range r. These are the `ScanMK` algorithm and the `SetFinder` algorithm. Both algorithms use the z-normalized Euclidean distance. The `ScanMK` algorithm repeatedly calculates set motifs as long as there is a pair of unprocessed subsequences in the time series with distance less than r to each other and maintains the largest set motif. The set motifs are calculated by

1. determining the top pair motif $\{T_{i,w}, T_{j,w}\}$,
2. removing all subsequences that are self matching $T_{i,w}$ or $T_{j,w}$,
3. keeping only subsequences that are matching both $T_{i,w}$ and $T_{j,w}$,
4. removing self matching subsequences and
5. removing subsequences with most subsequences out of range $2r$ first.

An arbitrary subsequence in the maintained set motif is chosen as the top set motif. The runtime is within $O(n^3 w)$ where n is the time series length and w is the subsequence length. The `SetFinder` algorithm compares each subsequence in the time series to every other subsequence and counts the non-self matching subequences for each subsequence. The top set motif is a subsequence with largest count. The runtime is within $O(n^2 w)$ where n is the time series length and w is the subsequence length.

3.3 Latent Set Motifs

A. J. Bagnall et al., 2014, introduce `ClusterMK`, an approxmate time series latent top set motif discovery algorithm with fixed subsequence length w and range r. The algorithm uses the z-normalized Euclidean distance. The `ClusterMK` algorithm first creates for any subsequence in the original time series a new time series of length w with same values as the subsequence. Next the algorithm replaces repeatedly two time series with smallest distance to each other by an averaged time series of both weighted by the number of time series they replaced. The last time series is the approximate latent top set motif. The runtime is within $O(n^3)$ where n is the time series length.

Grabocka et al., 2015, introduce Grabocka et al., 2015, an approximate time series latent set motif discovery approach with fixed subsequence length w and range r. The algorithm uses the z-normalized Euclidean distance. The main idea of the algorithm is to maximize an objective function equal to frequency minus violation, where frequency and violation are two smooth and differentiable functions. The frequency function calculates a scalar between 0 and 1 that corresponds to the number of subsequences non-self matching the latent set motifs, where 0 means that the latent set motifs do not non-self match any subsequence in the time series. The violation function calculates a scalar between 0 and 1 that corresponds to the distance between latent set motifs to each other, where 0 means that each latent set motif is out of range $2r$ to each other latent set motif and 1 means that all latent set motifs are identical sequences. First the algorithm chooses k random subsequences in the time series as starting values. Next the algorithm uses a high order Runge-Kutta method with partially derived frequency and violation function to permute the sequence of reals (latent set motif) based on the starting subsequences within i iteration steps. To get a global approximate latent set motif Grabocka et al., 2015, repeat the process with different random subsequences as starting values. As the approximate top latent set motif they choose the sequence of reals with largest number of non-self matching subsequences in the time series. The runtime is within $O(piknw)$ where p is the number of random restarts, i is the number of iterations, k is the number of latent set motifs, n is the time series length and w is the subsequence length.

3.4 Generator

A. Bagnall et al., 2017, introduce a generator producing time series for simulated time series classification problems. The time series classification is the problem of identifying to which set of categories a time series belongs. Since the time series do not contain ground truth annotated motifs, they cannot be used as a motif discovery benchmark.

Zhou et al., 2019, published an aggregator for time series anomalies. In this context an anomaly is a sequence of unpredicted values in a time series, e. g. if a time series is supposed to have real values reaching from 0 to 1 a sequence of reals with values smaller than 0 and larger than 1 is an anomaly. Similar to A. Bagnall et al., 2017, the generated time series are supposed for simulated time series anomaly detection problems. Since the time series do not contain ground truth annotated motifs, they cannot be used as a motif discovery benchmark.

4 Objectives

4.1 Previous Student Research Project

We focused on three main tasks in the previous student research project. First we worked out an overview of state-of-the-art motif discovery algorithms and analyzed the applied evaluation methods. Then we designed a synthetic motif generator and used this synthetic motif generator to create a motif discovery benchmark. Finally we applied our motif discovery benchmark to state-of-the-art motif discovery algorithms and analyzed the results.

4.2 Harden the Generator

The current synthetic motif generator has several options to manipulate the resulting synthetic time series and the injected synthetic motif. There are parameters to specify the length and value range of the time series, the length, height and shape of the injected subsequences, and the amount of noise added to the synthetic time series and the injected subsequences. The synthetic motif generator offers 8 predefined motif shapes. We plan to add a functionality for custom motif shapes, i. e. an option to specify a function or a sequence of reals that will be used as the motif shape.

Our synthetic motif generator injectes latent set motifs into a synthetic time series and outputs 3 different files:

1. a file containing the values of the synthetic time series,
2. a file containing the location of the injected subsequences, length of the injected subsequences and range r of the injected synthetic latent set motif, and
3. a script to plot the synthetic time series.

To check that the injected synthetic latent set motif is always the one with the largest number of non-self matching subsequence among all latent set motifs in the synthetic time series the synthetic motif generator

1. verifies that no subsequence is non-self matching subsequences overlapping the already injected subsequences and
2. verifies that each set of non-self matching subsequences, i. e. subsequences overlapping with the injected subsequences, is smaller than the injected set motif.

As this two verifications are applied each time when a subsequence is injected, we also prohibit motifs with a smaller number of non-self matching subsequences than the injected synthetic latent set motif within range r in the same synthetic time series. Figure 2 shows an example of a hidden synthetic latent set motif.

We plan to change this approach by memorizing the potentially harmfull subsequences detected in the first verification step and using them in the second verification step to detect larger passive (non-harmfull) set motifs in the synthetic time series. This will make the discovery of the injected synthetic set motif harder as the injected set motif is hidden between passive (smaller) set motifs.

Also we noticed that all state-of-the-art motif discovery algorithms except the `GrammarViz` algorithm from Senin et al., 2014, and the `LM` from Grabocka et al., 2015, cannot find any of our synthetic motifs when the motif shape was different than the box shape. We need to inverstigate the synthetic motif injection procedure more accurate.

There is no graphical interface in the synthetic motif generator design. We plan to add a graphical user interface based on the `GrammarViz` tool from Senin et al., 2014, (see Figure 3) or a local web server. The `GrammarViz` tool displays the time series and offers a list of discovered motifs. The user can zoom in and out in the plotted time series. The discovered motifs in the time series can be selected and the non-self matching subsequences are displayed in a seperate plot as well as marked with yellow background in the time series plot. We need to modify the `GrammarViz` tool or design an own graphical user interface to plot the generated synthetic time series, the injected latent set motif and mark the injected subsequences in the synthetic time series plot. Also we plan to add an interface for custom motif shapes. The graphical user interface will also have an optional field to draw a shape by hand or insert a custom function.

4.3 Modifying Motif Discovery Algorithms

Also we plan two motif discovery modifications if time permits. As Theorem 4.1 claims the number of subsequences non-self matched by a top set motif

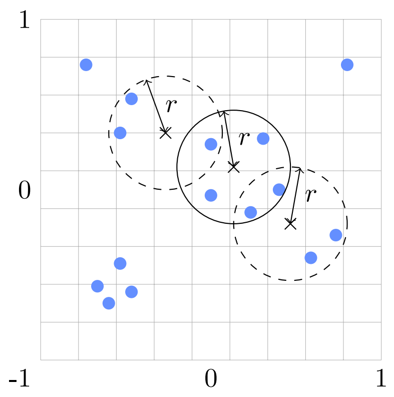


Figure 2: Example of a synthetic latent set motif with range r hidden between two passive latent set motifs with range r in the synthetic time series in the z-normalized 2 dimensional space of subsequences in the same time series. The injected latent set motif is represented by the cross within the solid circle and the passive latent set motifs are represented by the crosses within the dashed circles.

within a given range $2r$ in a time series is an upper bound for the number of subsequences non-self matched by a top latent set motif within range r in the same time series.

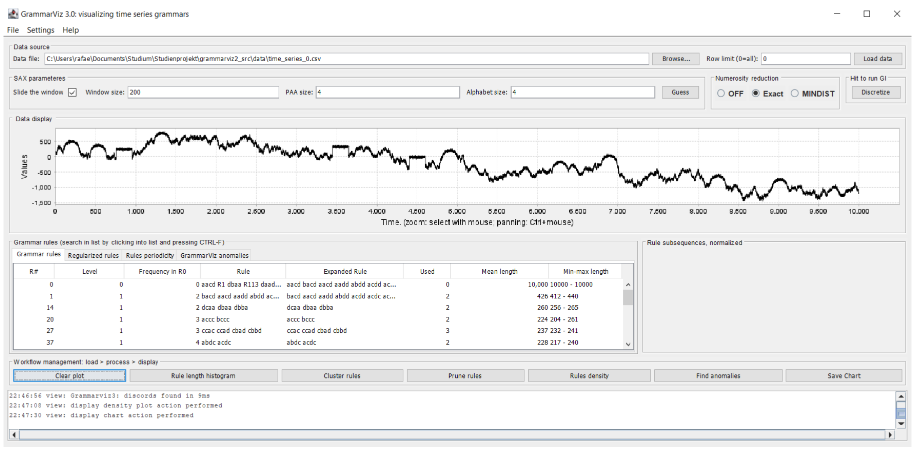
Theorem 4.1. *The number of subsequences non-self matched by the latent top set motif in a time series within range r is maximum as large as the number of subsequences non-self matched by the top set motif in the same time series within range $2r$.*

The original `EMMA` algorithm from Patel et al., 2002, searches for a subsequence in a time series with largest number of overlapping subsequences within a given range which conflicts the set motif definition. When we search for a set motif with range $2r$ we get an upper bound on the latent set motif size for the latent set motifs with range r in the same time series. Therefore, we plan to update the original `EMMA` algorithm from Patel et al., 2002, to exclude set motifs based on overlapping subsequences in a time series.

As the `LM` algorithm calculates a solution for an optimization problem the `LM` algorithm tends to get stuck in local optima. To get good global solutions for a given time series the original `LM` algorithm has to be repeated many times with random seeds. We plan to set the seed of the `LM` algorithm from Grabocka et al., 2015, to the subsequences non-self marched within range $2r$ by the subsequences discovered with our modified version of the `EMMA` algorithm.

4.4 Benchmark

Our motif discovery benchmark benchmarked unfairly the pair motf, set motif and latent set motif discovery algorithms on synthetic time series with injected latent set motifs. To investigate the results fairly we will split the benchmark into a pair motif, set motif and latent set motif benchmark.



Data source

Data file: C:\Users\rafae\Documents\Studium\Studienprojekt\grammarviz2_src\data\time_series_0.csv

Row limit (0=all): 0

SAX parameteres

Slide the window ✓ Window size: 200 PAA size: 4 Alphabet size: 4

Numerosity reduction: OFF [● Exact] MINDIST

Hit to run GI

Grammar rules (search in list by clicking into list and pressing CTRL-F)

R#	Level	Frequency in R0	Rule	Expanded Rule	Used	Mean length	Min-max length
0	0	0	aacd R1 dbaa R113 daad...	aacd bacd aacd aadd abdd acdd ac...	0	10,000	10000 - 10000
1	1	2	bacd aacd aadd abdd ac...	bacd aacd aadd abdd acdd acdc ac...	2	426	412 - 440
14	1	2	dcaa dbaa dbba	dcaa dbaa dbba	2	260	256 - 265
20	1	3	accc bccc	accc bccc	2	224	204 - 261
27	1	3	ccac ccad cbad cbbd	ccac ccad cbad cbbd	3	237	232 - 241
37	1	4	abdc acdc	abdc acdc	2	228	217 - 240

Workflow management: load > process > display

```
22:46:56 view: Grammarviz3: discords found in 9ms
22:47:08 view: display density plot action performed
22:47:30 view: display chart action performed
```

Figure 3: GrammarViz graphical user interface by Senin et al., 2014.

Also, the number of time series has to be larger to get significant results. We plan to use the updated synthetic motif generator to publish a new extended and harder version of the previously generated benchmark for motif discovery algorithms. Therefore, we generate separate benchmarks for pair motif, set motif and latent set motif discovery algorithms. To get more meaningful results for each discovery algorithm, we will scale up the number of synthetic time series in each benchmark. All in all, the benchmark will be performed on the pair motif discovery algorithms

1. `MK` from Mueen et al., 2009, and
2. `SCRIMP++` from Zhu et al., 2018,

the set motif discovery algorithms

1. `EMMA` from Patel et al., 2002,
2. our version of the `EMMA` algorithm,
3. `GV` from Senin et al., 2014,
4. `ScanMK` from A. J. Bagnall et al., 2014, and
5. `SetFinder` from A. J. Bagnall et al., 2014,

and the latent set motif algorithms

1. `ClusterMK` from A. J. Bagnall et al., 2014,
2. `LM` from Grabocka et al., 2015, and
3. our modified version of the `LM` algorithm.

We record the runtimes and the non-self matched subsequences of each time series motif discovery algorithm run. To evaluate the accuracy of each time

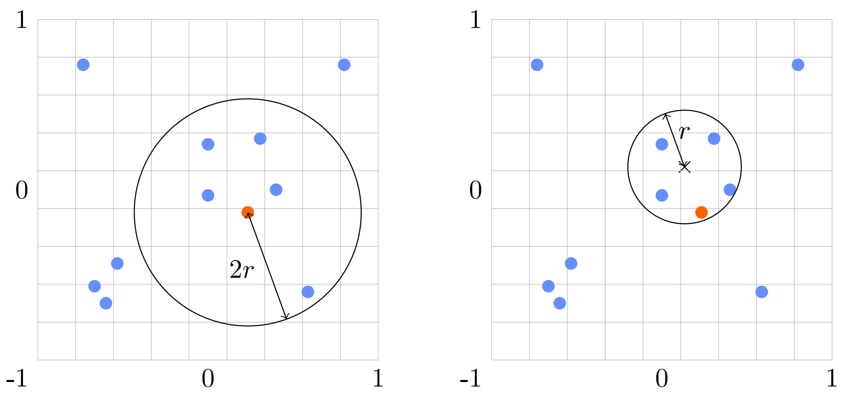


Figure 4: Illustration of Theorem 4.1. Both diagrams contain all non-self matching z-normalized subsequences of the same time series in the 2 dimensional space. The orange dot is a set motif that non-self matches the largest number of subsequences within range $2r$ in the time series. The cross on the right is a latent set motif with largest number of non-self matching subsequences within range r in the same time series.

series motif discovery algorithm more precisely we compute the point based precision

$$precision = \frac{TP}{TP + FP}$$

the point based recall

$$recall = \frac{TP}{TP + FN}$$

and the point based F_1 score

$$F_1 = 2\frac{precision \cdot recall}{precision + recall}$$

where TP is the number of all values of the subsequences, non-self matched by the calculated motif, overlapping with the injected subsequences, non-self matched by the injected latent top set motif; FP is the number of all values of the injected subsequences, non-self matched by the injected latent top set motif, which do not overlap with the subsequences, non-self matched by the calculated motif, and FN is the number of all values of the subsequences, non-self matched by the calculated motif, which do not overlap with the injected subsequences, non-self matched by the injected latent top set motif.

4.5 Unveiling

Our work will be presented in a scientific paper and we offer the source code as well as the updated benchmark on a git repository.

References

Bagnall, Anthony J., Jon Hills, and Jason Lines (July 2014). “Finding Motif Sets in Time Series”. In: *Computing Research Repository* 3685, pp. 1–6. URL: https://arxiv.org/pdf/1407.3685.

Bagnall, Anthony, Aaron Bostrom, James Large, and Jason Lines (Mar. 2017). “Simulated Data Experiments for Time Series Classification Part 1: Accuracy Comparison with Default Settings”. In: *arXiv* 1703.09480, pp. 1–28. URL: https://arxiv.org/pdf/1703.09480.

Esling, Philippe and Carlos Agon (Nov. 2012). “Time-Series Data Mining”. In: *ACM Computing Surveys (CSUR)* 45.1, 12:1–12:34. URL: https://hal.archives-ouvertes.fr/hal-01577883/document.

Grabocka, Josif, Nicolas Schilling, and Lars Schmidt-Thieme (May 2015). “Latent Time-Series Motifs”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11, 6:1–6:20. URL: https://www.ismll.uni-hildesheim.de/pub/pdfs/grabocka2016a-tkdd.pdf.

Lin, Jessica, Eamonn Keogh, Stefano Lonardi, and Bill Chiu (June 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*, pp. 2–11. URL: https://www.cs.ucr.edu/~eamonn/SAX.pdf.

Lin, Jessica, Eamonn Keogh, Stefano Lonardi, and Pranav Patel (Oct. 2002). “Finding Motifs in Time Series”. In: *CiteSeerX* 10.1.1.19.6629, pp. 1–11. URL: https://cs.gmu.edu/~jessica/Lin_motif.pdf.

Mueen, Abdullah, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover (Oct. 2009). “Exact Discovery of Time Series Motifs”. In: *Proceedings of the 2009 SIAM International Conference on Data Mining* 9781611972795.41, pp. 473–484. URL: http://alumni.cs.ucr.edu/~mueen/pdf/EM.pdf.

Patel, Pranav, Eamonn Keogh, Jessica Lin, and Stefano Lonardi (Dec. 2002). “Mining Motifs in Massive Time Series Databases”. In: *Proceedings of the 2002 IEEE International Conference on Data Mining*, pp. 370–377. URL: https://cs.gmu.edu/~jessica/publications/motif_icdm02.pdf.

Senin, Pavel, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, Susan Frankenstein, and Manfred Lerner (Sept. 2014). “GrammarViz 2.0: A Tool for Grammar-Based Pattern Discovery in Time Series”. In: *Machine Learning and Knowledge Discovery in Databases* 8726, pp. 468–472. URL: https://cs.gmu.edu/~xwang24/papers/grammarviz2.pdf.

Yeh, Chin-Chia Michael, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh (Dec. 2016). “Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)* 1, pp. 1317–1322. URL: http://www.cs.ucr.edu/~eamonn/PID4481997_extend_Matrix%20Profile_I.pdf.

Zhou, Shengtian, Mejbah Alam, and Justin Gottschlich (Apr. 2019). “DATSA Tutorial (Data Aggregator for Time Series Anomalies)”. In: *IntelLabs* 1, pp. 1–22. URL: https://github.com/IntelLabs/DATSA.

Zhu, Yan, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh (Nov. 2018). “Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds”. In: *2018 IEEE International Conference on Data Mining (ICDM)* 1, pp. 837–846. URL: https://www.cs.ucr.edu/~eamonn/SCRIMP_ICDM_camera_ready_updated.pdf.