Discovering frequent patterns on agrometeorological data with TrieMotif

Discovering Frequent Patterns on Agrometeorological Data with TrieMotif

Daniel Y.T. Chino[1](✉), Renata R.V. Goncalves[2], Luciana A.S. Romani[3], Caetano Traina Jr.[1], and Agma J.M. Traina[1]

[1] Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil
{chinodyt,caetano,agma}@icmc.usp.br
[2] Cepagri-Unicamp, Campinas, Brazil
renata@cpa.unicamp.br
[3] Embrapa Agriculture Informatics, Campinas, Brazil
luciana.romani@embrapa.br

Abstract. The "food safety" issue has concerned governments from several countries. The accurate monitoring of agriculture have become important specially due to climate change impacts. In this context, the development of new technologies for monitoring are crucial. Finding previously unknown patterns that frequently occur on time series, known as motifs, is a core task to mine the collected data. In this work we present a method that allows a fast and accurate time series motif discovery. From the experiments we can see that our approach is able to efficiently find motifs even when the size of the time series goes longer. We also evaluated our method using real data time series extracted from remote sensing images regarding sugarcane crops. Our proposed method was able to find relevant patterns, as sugarcane cycles and other land covers inside the same area, which are really useful for data analysis.

Keywords: Time series · Frequent motif · Remote sensing image

1 Introduction

One of the issues being pursued by the database researchers is how to take advantage of the large volume of data daily generated by the plethora of sensors placed in many environments and systems. The information gathered by the sensors are usually stored in time series databases, being a rich source for decision making for the owners of such data. One of the main tasks when mining time series is to find the motifs present therein, that is, to find patterns that frequently occurs in time series. That is, the motifs provide key information to mine association rules aimed at spotting patterns indicating that some events frequently lead to others.

Existing applications involving time series, such as stock market analysis, are not yet able to record all the factors that govern the data stored in the time series, such as political and technological factors. On the contrary, climate variations nowadays have most of its governing factors being recorded, using sensing

DOI: 10.1007/978-3-319-22348-3_6

equipments such as satellites and ground-based weather stations. However, the diversity of data available makes it hard to discover complex patterns that can support more robust analyzes. In this scenario, finding motifs has an even greater importance.

An example of time series, is the one extracted from remote sensing imagery containing Normalized Dierence Vegetation Index (NDVI) measurements. The NDVI time series present the vegetative strength of the plantation [14]. To follow the development of a crop is strategic for agribusiness practices in Brazil, since agriculture is the country's main asset. The accurate monitoring of agriculture in the whole world have become more and more important specially due to climate change impacts. The "food safety" issue has concerned governments from several countries and the development of new technologies for monitoring, as well as the proposition of mitigation and adaptation measures, are crucial. In this sense, remote sensing can be an important tool to improve the fast detection of changes in the land cover besides to aid at monitoring the crop cycle.

Since the volume of time series databases as well as the length of the series is growing at a very fast pace, it is mandatory to develop algorithms and methods that can deal with time series in the scenario of big data. In this paper we present the TrieMotif, a new method to extract motifs from time series and a new algorithm to index them in a trie data structure that performs well over large time series, which is up to 3 times faster than the state-of-the-art method. We evaluated TrieMotif over both synthetic and real data time series, obtaining very promising results.

This paper is organized as follows. Section 2 summarizes the main concepts used as basis to develop our work. Section 3 describes the TrieMotif algorithm and Sect. 4 discusses its evaluation. Section 5 shows the TrieMotif performance on real data obtained from remote sensing images. Section 6 concludes this paper.

2 Background and Related Works

A time series motif is a pattern that occur frequently. They were first defined in [11] and a generalized definition was given in [2]. In this section we recall these definitions and notations, as they will be used in this paper. First we begin with a definition 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 want to find patterns that frequently occur along a time series, we will not work with the whole time series, we are aiming only at parts of a time series, which are called subsequences and are defined as follows.

Definition 2. *Subsequence:* Given a time series T of length m, a subsequence S_p of T is a sampling of length $n < m$ of contiguous positions from T beginning at position p, that is, $S_p = t_p, \ldots, t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.

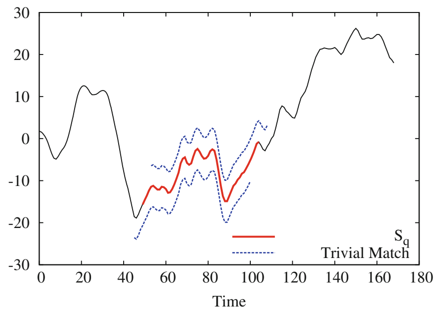


Fig. 1. The best matches of a subsequence S_q are probably the trivial matches that occur right before or after S_q.

In order to find frequent patterns, we need to define a matching between patterns.

Definition 3. *Match:* Given a distance function $D(S_p, S_q)$ between two subsequences, a positive real number R (*range*) and a time series T containing a subsequence S_p and a subsequence S_q, if $D(S_p, S_q) \leq R$ then S_q is called a *matching* subsequence of S_p.

On subsequences of the same time series, the best matches are probably subsequences that are slightly shifted. Matching between two overlapped subsequences is called a trivial match. Figure 1 illustrates the idea of a trivial match. The trivial match is defined as follows:

Definition 4. *Trivial Match:* Given a time series T, containing a subsequence S_p and a matching subsequence S_q, we say that S_q is a *trivial match* to S_p either if $p = q$ or if there is no subsequence $S_{q'}$ beginning at q' such that $D(S_p, S_{q'}) > R$, and either $q < q' < p$ or $p < q' < q$. That is, if two subsequences overlaps, there must exist a subsequence between them that is not a match.

These definitions allow defining the problem of finding Frequent K-Motif. First, all subsequences are extracted using a sliding window. Then, since we are interested in patterns, each subsequence is z-normalized to have zero mean and one standard deviation [6]. The K-Motif is defined as follows.

Definition 5. *Frequent K-Motifs:* Given a time series T, a subsequence of length n and a range R, the most significant motif in T (*1-Motif*) is the subsequence $F^{\{1\}}$ that has the highest count of non-trivial matches. The K^{th} most significant motif in T (*K-Motif*) is the subsequence $F^{\{K\}}$ that has the highest count of non-trivial matches, and satisfies $D(F^{\{K\}}, F^{\{i\}}) > 2R$, for all $1 \leq i < K$.

The Nearest Neighbor motif was defined by Yankov et al. [17], and it represents the closest pair of subsequences. In our proposed work, we focus on the Frequent K-Motif problem and we will be referring to them as K-Motif. Since the K-Motifs are unknown patterns, a brute-force approach would compare every

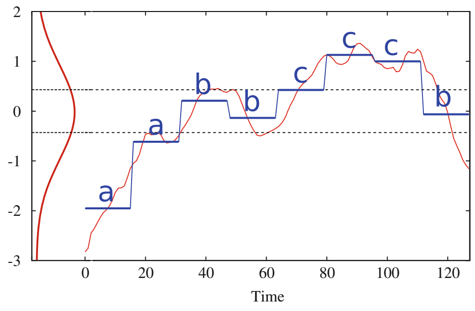


Fig. 2. A time series subsequence of size $n = 128$ is reduced to a PAA representation of size $w = 8$ and then is mapped to a string of $a = 3$ symbols *aabbcccb*.

subsequence with each other. It has quadratic computational cost on the time series size, since it requires $O(m^2)$ distance function evaluations.

An approach to reduce the complexity of this problem employs dimensionality reduction and discretization of the time series [11]. The SAX (Symbolic Aggregate approXimation) technique allows time series of size n to be represented by strings of arbitrary size w ($w < l$) [10]. For a given time series, SAX consists of the following steps. Firstly, the time series is z-normalized, so that the data follow normal distribution [4]. Next, the normalized time series is converted into the Piecewise Aggregate Approximation (PAA) representation, decreasing the time series dimensionality [5]. The time series is then replaced with w values corresponding to the average of the respective segment. Thus, in the PAA representation, the time series is divided into w continuous segments of equal length. Finally, the PAA representation is discretized into a string with an alphabet of size $a > 2$. Figure 2 shows an example of a time series subsequence of size $n = 128$ discretized using SAX with $w = 8$ and $a = 3$. It is also possible to compare two SAX time series using the MINDIST function. The MINDIST lower bounds the Euclidean distance [12], warranting no false dismissals [3].

Chiu et al. proposed a fast algorithm based on Random Projection [2]. Each time series subsequence is discretized using SAX. The discretized subsequences are mapped into a matrix, where each row points back to the original position of the subsequence on the time series. Then, the algorithm uses the random projection to compute a collision matrix, which counts the frequency of subsequence pairs. Through the collision matrix, the subsequences are checked on the original domain seeking for motifs. Although fast, the collision matrix is quadratic on the time series length, requiring a large amount of memory. Also using the SAX discretization, Li and Lin [8,9] proposed a variable length motif discovery based on grammar induction. Catalano et al. [1] proposed a method that works on the original domain of the data. The motifs are discovered in linear time and constant memory costs using random sampling. However, this approach can lead to poor performance for long time series with infrequent motifs [13].

Several works proposed to solve the motif discovery problem taking advantage of tree data structures. Udechukwu et al. proposed an algorithm that uses a suffix tree to find the Frequent Motif [15]. The time series are discretized considering the slopes between two consecutive measures. The symbols are chosen according to the angle between the line joining the two points and the time axis. Although this algorithm do not require to set the length of the motif, the algorithm is affected by noise. A suffix tree was also used to find motifs in multivariate time series [16]. Keogh et al. solved a problem similar to the motif discovery using a trie structure to find the most unusual subsequence in a time series [7].

The TrieMotif algorithm is up to 3 times faster and requires up to 10 times less memory than the current state of the art, the Random Projection approach, because TrieMotif selects only the candidates that are most probably a match to a motif. Using this approach, TrieMotif reduces the number of unnecessary distance calculations.

3 The TrieMotif Algorithm

In this section we present the TrieMotif algorithm. Since a motif is an unknown pattern, one of the problem of finding them is the need to compare every subsequence with every other. On this context, we intend to reduce the number of subsequence comparisons by discarding subsequences that are discrepant from each other. Let S_q be a possible motif, matching subsequences might have values similar to S_q and therefore they might be on a region near S_q, as shown in Fig. 3. The TrieMotif algorithm defines this area by setting an upper and a lower limit to S_q, this way, we only compare possible matching subsequences. The TrieMotif algorithm consists of three stages:

- First, all subsequences are extracted from the time series and converted into a symbolic representation;
- The subsequences are indexed using a Trie and a list of possible non trivial matches (candidates) are generated for each subsequence using the Trie index;
- The distances between the motif candidates on the original time series are calculated.

On the first stage, all subsequences S_i of size n are extracted using a sliding window and are z-normalized. These subsequences pass through a dimensionality reduction process to reduce the computational cost on the next stage. A subsequence of size n can be represented as a sequence of size w, via the PAA algorithm. On the next step of this stage, subsequences are converted into a symbolic representation. This representation is obtained by dividing the interval of the subsequence values into a equal size bins, where each bin receives a symbol. Each value in the subsequence is converted into the symbol of the corresponding bin. Figure 4 shows an example that converts a subsequence S of size $m = 128$ and values between $[-2.83, 1.36]$ into a string of size $w = 8$ using an alphabet a of 3 symbols. Initially the subsequence passes through a dimensionality reduction via PAA with $w = 8$. Then, assuming $a = 3$, three bins are

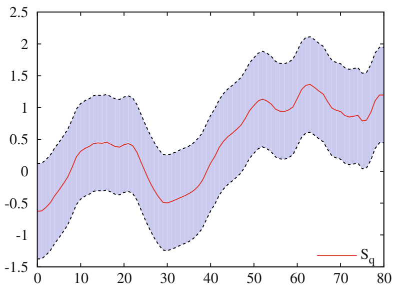


Fig. 3. Subsequences S_i that satisfy $D(S_q, S_i) \leq R$ are possibly in the highlighted area.

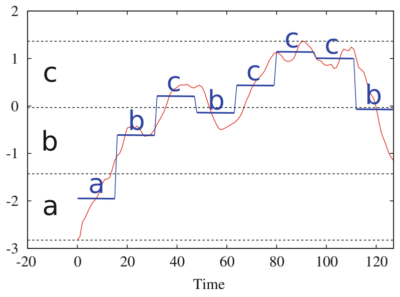


Fig. 4. A time series subsequence of size $m = 128$ is converted into a string of $a = 3$ symbols and size $w = 8$.

created: $a = [-2.83, -1.43)$, $b = [-1.43, -0.03)$ and $c = [-0.03, 1.36]$. Notice, that we kept the zero mean and the standard deviation requirements. Finally, the symbolic representation of S is $\hat{S} = abcbcccb$.

To find the top K-Motifs, the brute force algorithm calculates the distance of each subsequence S_q to every other subsequence. Our proposal reduces the number of distance calculations by selecting only candidates S_i that may be a match – the trivial matches are discarded on the next stage. To select the candidates we index all subsequences (already represented as a string) in a trie. For example, consider the subsequences S_1, S_2, S_3 and S_4, $w = 4$ and $a = 4$, as shown in Fig. 5. After processed in the first stage, they become $\hat{S}_1 = aabd$, $\hat{S}_2 = babd$, $\hat{S}_3 = abda$ and $\hat{S}_4 = dcca$ respectively. Figure 6(a) shows how the trie is built.

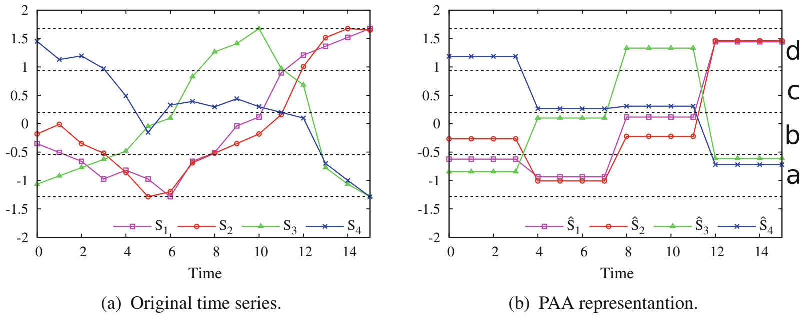
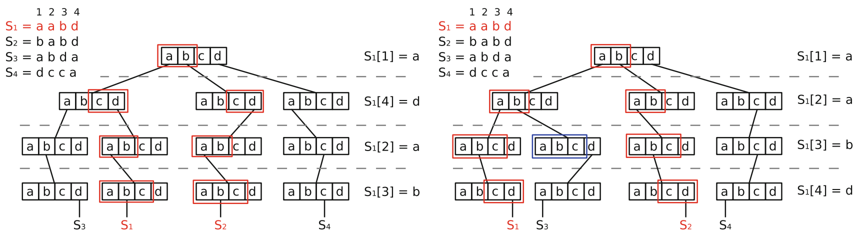


Fig. 5. Symbolic conversion process of the subsequences S_1, S_2, S_3 and S_4 to the strings $\hat{S}_1 = aabd$, $\hat{S}_2 = babd$, $\hat{S}_3 = abda$ and $\hat{S}_4 = dcca$.

An exact search on the trie would return candidates faster, but some candidates S_i that are a match could be discarded. If we search for candidates of $\hat{S}_1$, although $\hat{S}_2$ is probably a match, it would not be selected. To solve this problem, we modified the exact search on the trie to a range-like search. On the exact search, when the algorithm is processing the j^{th} element of $\hat{S}_q$ ($\hat{S}_q[j]$), it only visits the path of the trie where $Node_{symbol} = \hat{S}_q[j]$. In our proposed approach, we associate numerical values to the symbols. For example, A is equal to 1, B is equal to 2, and so on. Therefore, we can take advantage of closer values to compute the similarity when comparing the symbols. On our modified search, the algorithm also visits paths where $|\hat{S}_q[j] - Node_{symbol}| \leq \delta$. That is, if $\hat{S}_q[j] = b$ and $\delta = 1$, then the algorithm visits the paths of a, b and c (one up or one down). As backtracking all possible paths might be computationally expensive, we exploit pruning of unwanted paths by indexing the elements of the strings in a non-sequential order. This approach is interesting, because time series measures over a short period tend to have similar values. For example, if in a given time there is a b symbol, probably the next symbol might be a, b or c (even in large alphabets). Taking that into account, we interleave elements from the beginning and the end of the string, i.e., the first symbol, then the last symbol, then the second and so on.

Figure 6(a) shows how the modified search behave when searching for candidates to $\hat{S}_1$. On the first level, $\hat{S}_1[1] = A$ and the algorithm will visit the paths of the symbols a and b. On the next level, $\hat{S}_1[4] = D$ and it will visit the paths of c and d. This process is repeated until it reaches a leaf node. In this example, it will return the candidates $\hat{S}_1$ and $\hat{S}_2$, but $\hat{S}_3$ and $\hat{S}_4$ will not be checked. Figures 6(a) and 6(b) also show that changing the order of the elements can reduce the backtracking. If the subsequences were indexed using the sequential order, our modified search would also visit the path of $\hat{S}_3$ for at least one more level (marked in blue), while changing the order, this path is never visited.



(a) Non-sequential order indexing. (b) Sequential order indexing.

Fig. 6. By selecting candidates using our modified search, the algorithm backtracks only on nodes of the strings $\hat{S}_1$ and $\hat{S}_2$ paths. Creating the trie index using normal order increases the backtracking, since the algorithm visits part of the string $\hat{S}_3$ path (Color figure online).

On the last stage, after obtaining a list of candidates for each S_q, we calculate the distance between S_q and S_i on the original time series domain, discarding trivial matches of S_q. We also make sure that the K-Motifs satisfy Definition 5.

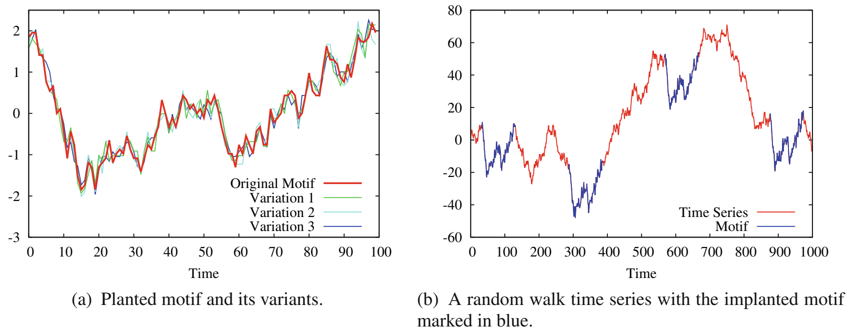
Algorithm 1 shows how to use the TrieMotif algorithm to locate the top K-Motifs. First, all subsequences S_i are converted into a symbolic representation $\hat{S}_i$ and every $\hat{S}_i$ is indexed into a trie index. Through the Trie index, we can reduce the number of non-trivial match calculations by generating a set C of the possible candidates for every $\hat{S}_i$. Then, we check on the original time series domain if the subsequences C_j in C satisfies $D(S_i, C_j) \leq R$ and it is not a trivial match. Since it is not possible to know the top K most frequent motifs before computing every subsequence, we store the motif on a list ($ListOfMotif$). As the last step of the algorithm, it is needed to check if the top K-Motifs satisfies Definition 5. To do so, we sort $ListOfMotif$ by the decreasing number of non-trivial matches. Thus, the most frequent motifs appear at the list head. Thereafter we iterate through the sorted $ListOfMotif$ and whenever $F^{\{i\}}$ satisfies Definition 5, we insert $F^{\{i\}}$ in the result set $TopKMotif$, otherwise $F^{\{i\}}$ is discarded.

To improve the efficiency of the method, we also calculate every distance using the "early abandonment" approach. Note that, although we presented a method to symbolic represent time series subsequences, it is possible to use others symbolic representations, such as the SAX algorithm. In those cases, provided that the distance function is the Euclidean one, it is possible to take advantage of the low complexity of the $MINDIST$ and discard calculations on the original time series whenever $MINDIST(\hat{S}_q, \hat{S}_i) > R$, since the $MINDIST$ is a lower bound to the Euclidean distance.

4 Synthetic Data Analysis

To validate our method, we performed tests on a synthetic time series generated by random walk of size $m = 1,000$. We also embedded a motif of size 100 in four different positions of the time series. Figure 7(a) shows the planted motif and its variants. The motifs were planted on positions 32, 287, 568 and 875, as shown in Fig. 7(b). We ran the TrieMotif using the bin discretization method with $n = 100$, $R = 2.5$, $w = 16$, $a = 4$ and $\delta = 1$. As expected, the TrieMotif was able to successfully find the motif on the planted positions.

Knowing that our method is able to find motifs, we made a series of experiments varying the length of the time series to evaluate the method efficiency. We compared our method with the brute force algorithm and with a Random Projection method, since it used extensively and is the basis of others algorithms



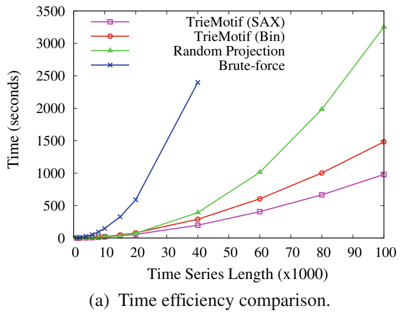
(a) Planted motif and its variants.
(b) A random walk time series with the implanted motif marked in blue.

Fig. 7. Planted motifs into a longer dataset for evaluation tests.

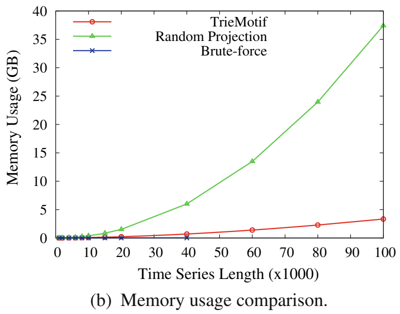
Algorithm 1. K-TrieMotif Algorithm.

```
Input: $T$, $n$, $R$, $w$, $a$, $\delta$, $K$
Output: List of the top $K$-Motif
1: $ListOfMotif \leftarrow \emptyset$
2: for all Subsequence $S_i$ with size $n$ of $T$ do
3:     $\hat{S}_i \leftarrow$ ConvertIntoSymbol($S_i$, $w$, $a$)
4:     Insert $\hat{S}_i$ in the Trie index
5: end for
6: for all Subsequence $\hat{S}_i$ of $T$ do
7:     $C \leftarrow$ GetCandidatesFromTrie($\hat{S}_i$, $\delta$)
8:     $MotifS \leftarrow \emptyset$
9:     for all $C_j \in C$ do
10:        if $NonTrivialMatch(S_i, C_j, R)$ then
11:            Add $C_j$ to $MotifS$
12:        end if
13:    end for
14:    Insert $MotifS$ in the $ListOfMotif$
15: end for
16: Sort $ListOfMotif$ by size in decreasing order
17: $TopKMotifs \leftarrow \emptyset$
18: $k \leftarrow 0$
19: for all ($F^{\{i\}} \in ListOfMotif$) and ($k < K$) do
20:    if ($Distance(F^{\{i\}}, F^{\{l\}}) \leq 2R$), $\forall F^{\{l\}} \in TopKMotif$ then
21:        Discard $F^{\{i\}}$
22:    else
23:        Insert $F^{\{i\}}$ in $TopKMotif$
24:        $k$++
25:    end if
26: end for
27: return $TopKMotif$
```

in the literature. We searched for motifs of size $n = 100$ and used the Euclidean distance. For both Random Projection and TrieMotif we used the same parameters for the discretization, $a = 4$ and $w = 16$. We ran the Random Projection with 100 iterations. For the TrieMotif, we used both bin and SAX representations with $\delta = 1$. We varied the time series length from $1,000$ to $100,000$. Each set of parameters were tested using 5 different seeds for the random walk. All tests were performed 5 times, totalizing 25 executions. The wall clock time and memory usage measurements were taken from these 25 runs of the algorithm. The presented values correspond to the average of the 25 executions. As it is too time consuming, the brute-force algorithm was not executed for time series with lengths above $40,000$. The experiments were performed in an HP server with 2 Intel Xeon 5600 quad-core processors with 96 GB of main memory, under CentOS Linux 6.2. All methods (TrieMotif, Random Projection and Brute-force) were implemented using the C++ programming language. The time spent comparison is shown in Fig. 8(a) and the memory usage is shown on Fig. 8(b).



(a) Time efficiency comparison.



(b) Memory usage comparison.

Fig. 8. Comparison of the TrieMotif with brute force and random projection.

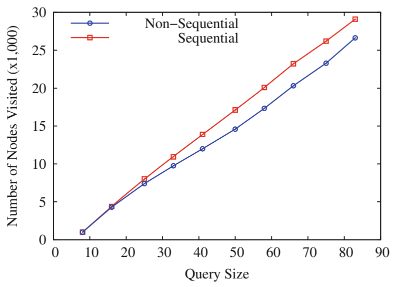


Fig. 9. Comparison of the number of visited nodes on the trie index using sequential and non-sequential order.

As expected, both Random Projection and TrieMotif performed better than the brute-force approach. For time series of length below $10,000$, both Random Projection and the TrieMotif had similar performance. However, as the time series length grows, the TrieMotif presented a increasingly better performance. This result is due to the fact that TrieMotif selects only the candidates that are probable matches for a motif, reducing the number of unnecessary distance calculations. TrieMotif also presents better performance using SAX because the interval of values is different for each symbol. The symbols corresponding to values near 0 are smaller and therefore, less candidates are selected. The TrieMotif also consumes less memory than the Random Projection, since the Random Projection algorithm needs at least m^2 memory for the collision matrix. From Fig. 8(b), we can see that the memory requirements of our method is significantly less demanding than Random Projection, requiring 10 times less memory than the state of the art (the Random Projection approach).

We also checked the differences between indexing the subsequences in sequential and non-sequential order. For this test we counted the number of nodes visited for each query on the trie index varying the query size. We executed the queries for every subsequence of the random walk time series. The presented values correspond to the average these executions. Figure 9 shows that the TrieMotif algorithm visited less nodes when using non-sequential order indexing.

5 Real Data Analysis

We also performed experiments using data from real applications. For this evaluation, we extracted time series from remote sensing images with two different spatial resolution. The spatial resolution of a remote sensing image specifies the size of the area that the pixel represents on the earth surface. On the first experiment, we extracted data from AVHRR/NOAA[1] images, a low spatial resolution image with a spatial resolution of 1 kilometer, i.e., each pixel represents an area of 1km x 1km on the ground. The AVHRR/NOAA images correspond to monthly measures of the Normalized Difference Vegetation Index (NDVI), which indicates the soil vegetative vigor represented in the pixels of the images and is strongly correlated with biomass. We used images of the São Paulo state, Brazil (Fig. 10), corresponding to the period between April 2001 and March 2010. From these images, we extracted 174,034 time series of size 108. Each time series corresponds to a geographical point in São Paulo state, excluding the coastal region.

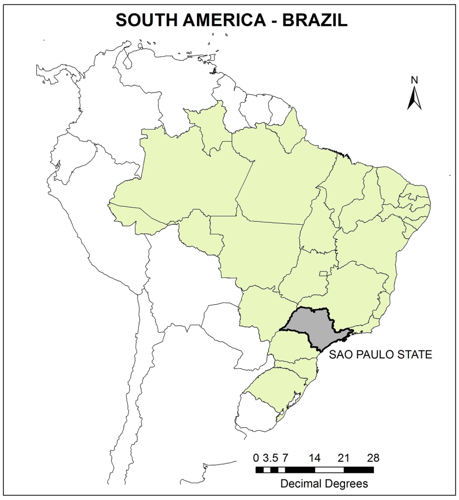


Fig. 10. State of São Paulo, Brazil.

In order to find the motifs in the time series database, we submitted all the time series to the TrieMotif algorithm, generating a single index. The São Paulo state is one of the greatest producers of sugarcane in Brazil. In this work, we

[1] Advanced Very High Resolution Radiometer/National Oceanic and Atmospheric Administration.

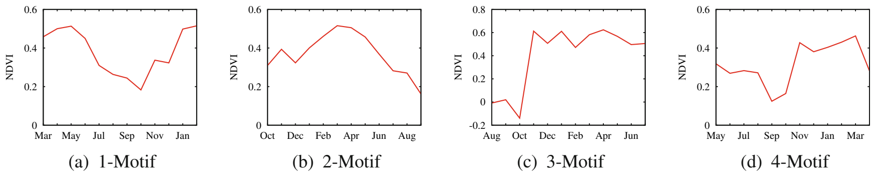


Fig. 11. Top 4-Motif for the São Paulo state area.

consider a complete sugarcane cycle of approximately one year, each starting in April and ending in March. We looked for the top 4-Motifs of size $n = 12$ (annual measures). We ran the TrieMotif algorithm using bin discretization with $R = 1.5$, $w = 4$, $a = 4$ and $\delta = 1$. The experiments were performed using the same computational infrastructure of the synthetic data.

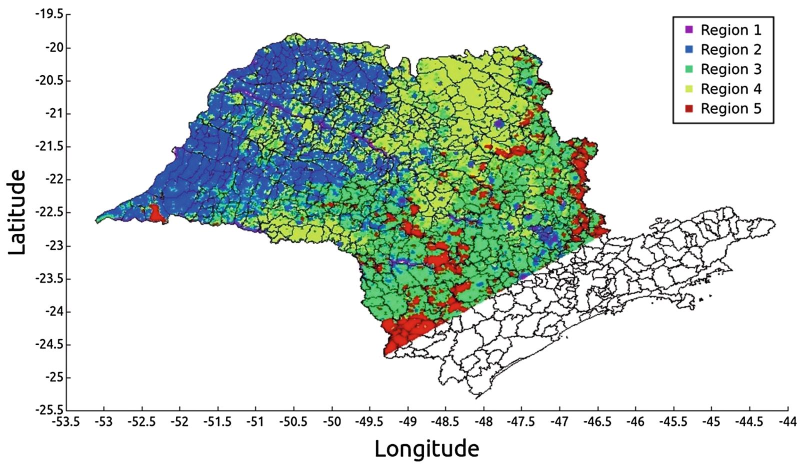


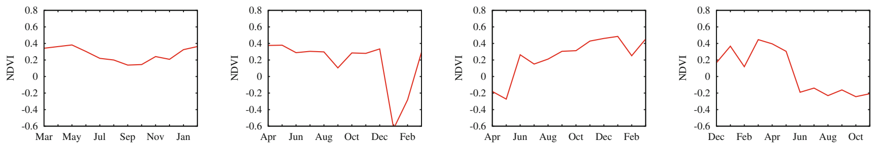
Fig. 12. The São Paulo state was divided into 5 regions of interest. Region 1 (purple) corresponds to water/cities, Region 2 and 3 (blue and green respectively) to a mixture of crops and grassland, Region 4 (yellow) is sugarcane crop and Region 5 (red) corresponds to forest. The black lines represent the political counties of the state. The south of the state is a forest nature federal reserve (Color figure online).

Figure 11 shows the top 4-Motifs for the São Paulo state. The plots are not z-normalized due to the experts restrictions for analyzing them. As expected, the two most frequent patterns (Figs. 11(a) and 11(b)) have a behavior similar to a sugarcane cycle, with high values of NDVI on April and low values on October. This behavior is due to the fact that sugarcane harvesting begins in April, when the NDVI slowly starts to decrease. On October, the harvest finishes and the soil remains exposed, when the NDVI has the lowest value. From October to March, the sugarcane grows and the NDVI increases again. This pattern

Region	Color	# of Series	Represents
1	Purple	2,804	Water and Cities
2	Blue	55,815	Grassland and Perennial Crops
3	Green	50,785	
4	Yellow	48,301	Sugarcane crops
5	Red	16,329	Forest

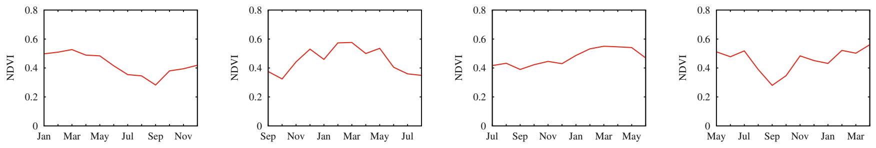
Fig. 13. Number of time series in each region of interest.

was found on almost every area of the São Paulo state and according to the experts was not a coherent result. To overcome this problem, we divided the São Paulo state into 5 regions of interest as shown in Fig. 12. The regions were obtained through a clustering algorithm and the results were analyzed by experts in the agrometeorology and remote sensing fields. According to the experts, the region 1 found by the algorithm corresponds to water and urban areas, regions 2 and 3 correspond to a mixture of crops (excluding sugarcane) and grassland, region 4 corresponds to sugarcane crops and region 5 to forest. Figure 13 shows a summary of each region of interest and the number of time series in each region.



(a) 1-Motif (b) 2-Motif (c) 3-Motif (d) 4-Motif

Fig. 14. Top 4-Motif for region of interest 1.



(a) 1-Motif (b) 2-Motif (c) 3-Motif (d) 4-Motif

Fig. 15. Top 4-Motif for region of interest 2.

Figures 14, 15, 16, 17 and 18 show the top 4-Motif for each region of interest. Once again, the plots are not z-normalized due to the experts restrictions for analyzing them. The 1-Motif found on region 1 (Fig. 14(a)) has a pattern with low values and variation, which experts say correspond to urban regions. The other Motifs found in region 1 (Figs. 14(b), (c) and (d)) have a pattern that corresponds to water regions. Figures 15 and 16 show patterns with a higher value of NDVI

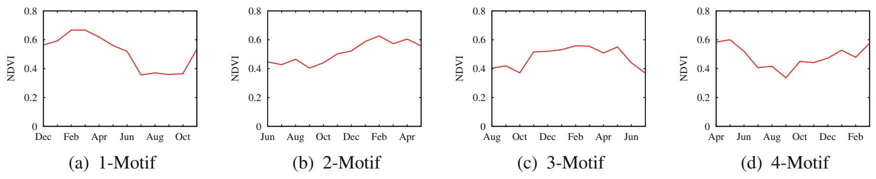


Fig. 16. Top 4-Motif for region of interest 3.

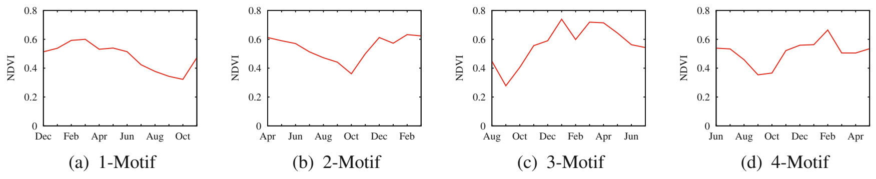


Fig. 17. Top 4-Motif for region of interest 4.

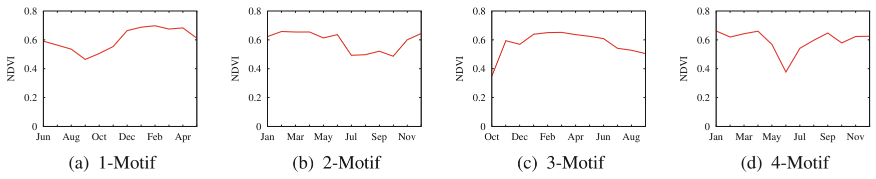


Fig. 18. Top 4-Motif for region of interest 5.

and with a high variation, which experts attribute to agricultural areas. The top 3-Motifs found on region 4 (see Figs. 17(a), (b) and (c)) is clearly recognized by agrometeorologists as a sugarcane cycle. Figure 18 shows a pattern with high values of NDVI and low variations, corresponding to the expected forest behavior, where the NDVI follows the local temperature variation.

The results obtained by the TrieMotif algorithm on the most prominent regions confirmed the correctness of the algorithm. However, the most interesting results correspond to patterns that were not expected by the experts. According to them, for some regions, the 4-Motifs found do not resemble the previously known patterns. The 4-Motifs found on both regions 2 and 3 (Figs. 15(d) and 16(d) respectively) indicates the presence of sugarcane and the 4-Motif found on region 4 (Fig. 17(d)) does not resemble a sugarcane cycle. This result indicates the need for a further inspection in the areas where these patterns occur.

On the second experiment, we used data extracted from MODIS[2] images. Each pixel in a MODIS image represents an area of 250 m x 250m, thus a higher spatial resolution. MODIS images correspond to biweekly NDVI measures. We used images of the São Paulo state, Brazil, corresponding to the period between

[2] Moderate-Resolution Imaging Spectroradiometer.

April 2004 and March 2005. Since MODIS images have a higher spatial resolution than AVHRR/NOAA images, they allow a better analysis of certain regions. We focused our analysis only on sugarcane crops regions and we extracted 40,133 time series of size 24. We ran the TrieMotif algorithm using bin discretization with $R = 1.5$, $w = 4$, $a = 4$ and $\delta = 1$. This time, we looked for the top 8-Motifs of size $n = 24$, annual measures for the MODIS images time series. The experiments were performed using the same computational infrastructure of the synthetic data.

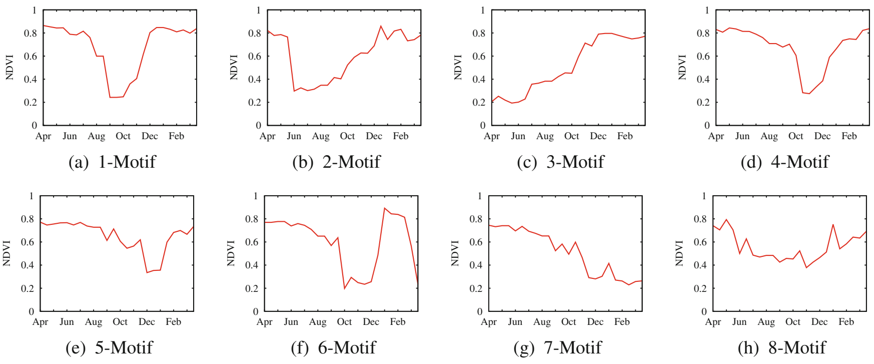


Fig. 19. Top 4-Motif for region of interest 1.

The results are shown on Fig. 19. As expected, the most frequent motif was a sugarcane cycle pattern (Fig. 19(a)) with high values of NDVI on April and low values on October. This time, the experts were also able to identify regions with different harvest periods (Figs. 19(b), (d) and (e)). And once again, they were able to spot regions with unexpected frequent patterns. Figure 19(f) shows a region where there was a problem with the sugarcane crop on March and Figs. 19(c), (g) and (h) show regions that might not be sugarcane regions.

6 Conclusions

Finding patterns in time series is highly relevant for applications where both the antecedent and the consequent events are recorded as time series. This is the case of climate and agrometeorological data, where it is known that the next state of the atmosphere depends in large amount of its previous state. Today, a large network of sensing devices, such as satellites recording both earth and solar activities, as well as ground-based whether monitoring stations keep track of climate evolution. However, the large amount of data and their diversity makes it hard to discover complex patterns able to support more robust analyzes. In this scenario, is very important to have powerful and fast algorithms to help analyzing that data.

In this paper we presented the TrieMotif, a technique that provides, in an integrated framework, an automated technique to extract frequent patterns in time series as K-Motifs. It reduces the resolution of the data, speeding up the sub-sequence comparison operations, indexes them in a trie structure and adopts heuristics commonly employed by the meteorologists to prune from the similarity search operations those branches that do not represent interesting answers. In this way, our technique is able to select only candidate subsequences that have high probability to match a motif, thus reducing the number of comparisons performed.

Experiments performed over data from both synthetic and real applications showed that our technique indeed perform in average 3 times faster them the state of the art approach (Random Projection), including the best methods previously available. It also requires less memory, and the experiments revealed that it requires up to 10 times less memory than the competitor methods. For a qualitative analysis, we presented the results to experts in the field (meteorologists), whom confirmed that the results are indeed correct and useful for their day-to-day activities to process climate data. For future works, we intend to explore data from larger regions, as the whole Brazil and South America. We also intend to explore data from different sensors in order to evaluate improvements that may be needed on sugarcane crop regions.

Acknowledgements. The authors are grateful for the financial support granted by FAPESP, CNPq, CAPES, SticAmsud and Embrapa Agricultural Informatics, Cepagri/Unicamp and Agritempo for data.

References

1. Catalano, J., Armstrong, T., Oates, T.: Discovering patterns in real-valued time series. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 462–469. Springer, Heidelberg (2006)
2. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 493–498, New York, NY, USA. ACM (2003)
3. Faloutsos, C., Ranganathan, M., Manolopoulos, Y., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 419–429. Minneapolis, USA (1994)
4. Goldin, D.Q., Kanellakis, P.C., Kanellakis, P.C.: On similarity queries for time-series data: Constraint specification and implementation. In: Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming, pp. 137–153. Cassis, France (1995)
5. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. **3**, 263–286 (2001)
6. Keogh, E., Kasetty, S.: On the need for time series data mining benchmarks: a survey and empirical demonstration. Data Min. Knowl. Disc. **7**, 349–371 (2003). Springer

7. Keogh, E., Lin, J., Lee, S.-H., Herle, H.: Finding the most unusual time series subsequence: algorithms and applications. Knowl. Inf. Syst. **11**(1), 1–27 (2007)
8. Li, Y., Lin, J.: Approximate variable-length time series motif discovery using grammar inference. In: Proceedings of the Tenth International Workshop on Multimedia Data Mining, MDMKDD 2010, pp. 10:1–10:9, New York, NY, USA. ACM (2010)
9. Li, Y., Lin, J., Oates, T.: Visualizing variable-length time series motifs. In: SDM, pp. 895–906. SIAM / Omnipress (2012)
10. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2003, pp. 2–11, New York, NY, USA. ACM (2003)
11. Lin, J., Keogh, E., Patel, P., Lonardi, S.: Finding motifs in time series. In: The 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada. ACM (2002)
12. Lin, J., Keogh, E.J., Wei, L., Lonardi, S.: Experiencing sax: a novel symbolic representation of time series. Data Min. Knowl. Disc. **15**, 107–144 (2007)
13. Mohammad, Y., Nishida, T.: Constrained motif discovery in time series. New Gener. Comput. **27**(4), 319–346 (2009)
14. Rouse, J.W., Haas, R.H., Schell, J.A., Deering, D.W.: Monitoring vegetation systems in the great plains with ERTS. In: Proceedings of the Third ERTS Symposium, Washington, DC, USA, pp. 309–317 (1973)
15. Udechukwu, A., Barker, K., Alhajj, R.: Discovering all frequent trends in time series. In: Proceedings of the winter international synposium on Information and communication technologies, WISICT 2004, pp. 1–6. Trinity College Dublin (2004)
16. Wang, L., Chng, E.S., Li, H.: A tree-construction search approach for multivariate time series motifs discovery. Pattern Recogn. Lett. **31**(9), 869–875 (2010)
17. Yankov, D., Keogh, E., Medina, J., Chiu, B., Zordan, V.: Detecting time series motifs under uniform scaling. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 844–853. ACM, New York, NY, USA (2007)